# APLICACIÓN DE MÉTODOS DE COMPRESIÓN DE CABECERAS TCP/IP COMO OPTIMIZACIÓN DE LAS COMUNICACIONES MÓVILES

*Anna Calveras Augé, María Luisa Catalán Cid*

*Grupo de Comunicaciones Móviles*
*Departamento de Ingeniería Telemática (ENTEL)*
*Universidad Politécnica de Cataluña (UPC)*
*Barcelona, España*

**Abstract- El protocolo TCP (Transmission Control Protocol) está implementado para actuar en redes fijas, donde las incidencias en la comunicación pueden interpretarse como problemas de congestión. En entornos móviles, en cambio, los problemas son debidos a una alta tasa de error en bit (BER); la utilización del protocolo TCP en este tipo de entornos afecta negativamente a la eficiencia de la transmisión. En este artículo se presenta una solución para optimizar la comunicación TCP: la compresión de cabeceras; concretamente se expone la necesidad de un algoritmo específico para comunicaciones móviles y se valora la propuesta de compresión de cabeceras de la IETF.**

## 1.INTRODUCCIÓN

Con el auge de las tecnologías móviles en las últimas décadas, ha aparecido también la necesidad de que el usuario pueda comunicarse a cualquier hora y desde cualquier parte, ya sea mediante servicios de voz o de transferencia de datos (e-mail, acceso a Internet, MMS). La aparición de los servicios de 2.5G y 3G, tales como GPRS o UMTS, ofrecen la posibilidad de proporcionar al usuario acceso a todos estos recursos con las ventajas de la tecnología inalámbrica y con el único requerimiento de que exista cobertura.

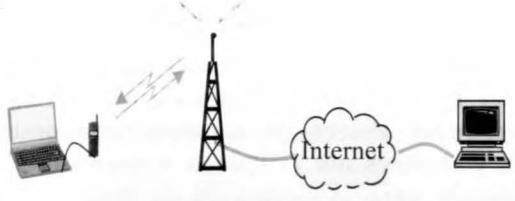


**Figura 1.** *Esquema de una conexión a Internet mediante un terminal móvil*

Internet está basada en la arquitectura de protocolos TCP/IP. Estos están diseñados para actuar de forma eficiente en redes fijas, donde la tasa de error en bit (BER) es baja y los principales problemas son causados por la congestión de la red. Las comunicaciones móviles, en cambio, se realizan a través de enlaces inalámbricos, caracterizados por una alta BER. Especialmente, en condiciones de movilidad de terminales y reselección de celdas, pueden producirse ráfagas de error pronunciadas que afectan negativamente al rendimiento de los protocolos TCP/IP [3].

Por otra parte, los retardos extremo-extremo (RTT)[1] en una comunicación móvil toman valores muy superiores a los obtenidos en redes fijas (entre centenares de milisegundos en sistemas 3G y entre 1 y 3 segundos en comunicaciones GPRS) [15]. Además, los sistemas móviles permiten combatir la alta probabilidad en bit mediante la actuación de mecanismos de recuperación de errores a nivel de enlace RLC/LLC (por ejemplo, después de que se haya llevado a cabo una reselección de celda o después de una interrupción en la comunicación); esto, unido al hecho de que durante la transmisión pueden limitarse la cantidad de recursos (slots) disponibles, puede provocar variaciones importantes del retardo entre paquetes consecutivos. Por lo tanto, se provocará un funcionamiento poco eficiente del mecanismo habitual de TCP [14] para el cálculo del temporizador de retransmisión (RTO).

La Figura 2 ejemplifica ambos aspectos. Se muestran los resultados de la captura (en el terminal móvil) de una transmisión realizada en una situación de movilidad. Tal como puede observarse, debido al paso de los paquetes a través del enlace inalámbrico, se producen dos fenómenos importantes a destacar. Por una parte cortes o desconexiones en los que se pierden ráfagas de paquetes, por otra, retardos variables y de una duración considerable.

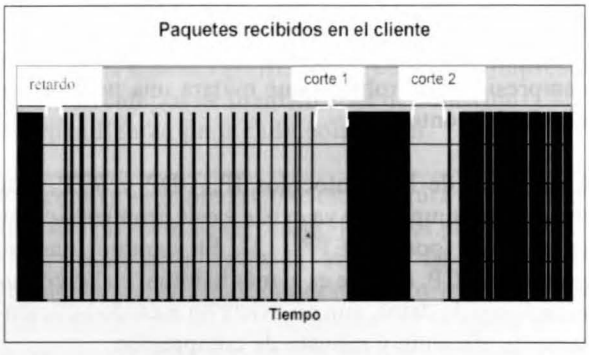


**Figura 2.** *Fragmento de una transmisión. Escenario móvil*

[1] *Se denomina retardo extremo-extremo ó Round Trip Time al tiempo transcurrido entre que se envía un paquete y se recibe la confirmación.*

Desde el punto de vista del usuario, otra característica negativa de este tipo de comunicaciones es que se llevan a cabo en enlaces de baja velocidad. Actualmente en Internet y redes IP se utiliza la pila de protocolos TCP/IP en su versión 4. En estos sistemas se envía en cada paquete un mínimo de 40 bytes de cabecera a nivel TCP/IP. Sin embargo, para poder dar servicio a más usuarios y facilitar la movilidad,en un futuro será necesaria la utilización del protocolo IPv6 [9] que entre otras características dispone de un campo de direcciones mayor (128 bits frente a los 32 de IPv4), por lo que la cabecera TCP/IP se verá incrementada. Este hecho reducirá el tamaño del campo disponible para enviar información y se hará aun más plausible la sensación de baja velocidad y la consecuente insatisfacción del usuario. Por todo esto se justifica la necesidad de métodos adicionales que permitan optimizar el protocolo TCP para obtener un mejor rendimiento en entornos móviles.

Si se analizan los campos de las cabeceras de los principales protocolos de nivel de transporte (TCP/IP, UDP, RTP), puede observarse que para una transmisión concreta, gran parte de los campos de las cabeceras se mantienen constantes, son deducibles o varían siguiendo un cierto patrón. Una manera de reducir la influencia de la cabecera en el tamaño total del paquete sería la utilización de algoritmos de compresión: el compresor únicamente envía los campos de cabecera necesarios para que el descompresor pueda reconstruir la cabecera original a partir de ese paquete y de un cierto contexto[2]. Así se aumentaría el throughput de la transmisión. Este hecho mejoraría la apreciación del usuario sobre el rendimiento de las comunicaciones móviles de baja velocidad.

No obstante, la mayoría de los algoritmos de compresión existentes [7, 10], aunque fueron ideados para enlaces de baja velocidad no son directamente aplicables a las comunicaciones móviles. Esto es debido, sobretodo, a que no están suficientemente adecuados para soportar altas BER. En el caso de pérdida de un solo paquete, se pierde el contexto y se descartan muchos paquetes en el descompresor antes de volver a recuperarlo. Por lo tanto, sería necesario un método de compresión más robusto que evitara una pérdida tan rápida del contexto.

En el caso de los protocolos IP, UDP y RTP, este método de compresión ya existe y está propuesto como un estándar por el IETF [12]. En cambio, para el protocolo TCP, aunque es el más habitual en las comunicaciones de datos, aun no existe un método suficientemente eficiente y robusto de compresión.

*[2] Se denomina contexto a toda la información necesaria (normalmente se trata de los campos de cabecera de un paquete anterior) para recuperar la cabecera original de un paquete comprimido*

El grupo de trabajo ROHC de la IETF [18] en el área de transporte está elaborando una nueva propuesta de compresión, el [17].

En el artículo se analiza la viabilidad de optimizar las comunicaciones móviles mediante métodos de compresión de cabeceras. En concreto, se detallarán los problemas de los mecanismos de compresión de la cabecera TCP/IP actuales [7, 10] y se presentarán posibles soluciones. En particular nos centraremos en la propuesta que actualmente desarrolla el grupo ROHC de IETF [17], basada en el protocolo de compresión robusta ROHC [12].

El artículo se estructura de la siguiente forma. En las secciones 2 y 3 analizaremos el funcionamiento y los problemas que presentan los métodos actuales de compresión de cabeceras TCP: Van Jacobson e IPHC, en enlaces móviles. En la sección 4 presentaremos las características y las mejoras respecto a los protocolos de otras propuestas anteriores, en concreto la del grupo de trabajo ROHC del IETF para un protocolo de compresión robusta [17]. Finalmente, en la sección 5 concluiremos porqué creemos que la compresión de cabeceras, y en concreto la propuesta [17], podría ser una buena opción para la optimización de las comunicaciones móviles de 2.5G y 3G.

## 2. COMPRESIÓN VAN JACOBSON

El protocolo de compresión Van Jacobson [7] fue ideado para comunicaciones unidireccionales mediante el protocolo TCP/IPv4 en enlaces fijos de baja velocidad (PPP, SLIP). Basa su funcionamiento en mantener, para cada conexión, una cabecera base (o contexto) común en compresor y descompresor. En el compresor la cabecera base se actualiza con la información de cabecera de cada nuevo paquete enviado; en el descompresor ésta se inicializa cuando se envía un paquete sin comprimir y se actualiza mediante la información que se transmite en cada paquete comprimido.

La cabecera comprimida incluye como mínimo un primer byte a modo de máscara que indica los campos de la cabecera que se transmiten a continuación, y el checksum TCP de la cabecera original, necesario para la detección de errores en la descompresión[3]. El resto de los campos mostrados en la Figura 3 se transmiten sólo si varían respecto a los campos de la cabecera base. Para reducir al máximo el tamaño de la cabecera comprimida, en vez del valor real del campo del paquete a comprimir, se transmite la diferencia entre éste y el almacenado en la cabecera base del compresor.

*[3] Se considera que un paquete se ha descomprimido correctamente si el CRC de la cabecera descomprimida coincide con el Checksum TCP de la cabecera original.*

En el caso de que varíen otros campos no contemplados en la estructura de la cabecera comprimida o que se produzca una retransmisión, debe volver a enviarse un paquete sin comprimir para actualizar la cabecera base del descompresor.

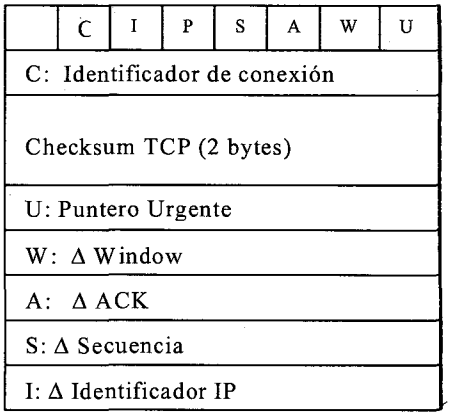


**Figura 3** *Cabecera comprimida (VJacob)*

Al tratarse de una comunicación unidireccional, la compresión Van Jacobson hace uso del mecanismo de reconocimientos de TCP para sincronizar la cabecera base en caso de pérdida o error de los paquetes. Si se pierde un paquete o se descarta por ser erróneo, la cabecera base en el descompresor no podrá actualizarse y, por tanto, diferirá de la del compresor que sí se habrá actualizado. Al estar los campos codificados de forma incremental, la cabecera del próximo paquete comprimido no podrá reconstruirse correctamente, ya que el Checksum TCP enviado será diferente del calculado en el descompresor. Por lo tanto, se descartará el paquete y el destino no enviará ningún ACK al emisor. Así sucederá con todos los paquetes que lleguen al descompresor, hasta que expire el temporizador en el emisor y se reenvíe el paquete erróneo o perdido (sin comprimir) y se vuelva a sincronizar el contexto del descompresor.

Esta política de recuperación de errores presenta tres grandes problemas en las comunicaciones móviles:

- En primer lugar, como los RTTs pueden llegar a ser muy altos, aunque el ancho de banda sea reducido, el tamaño de la ventana de transmisión puede ser elevado. Por lo tanto, el descompresor ya habrá descartado gran cantidad de paquetes, antes de que se haya logrado actualizar el contexto, y deberá retransmitirse toda la ventana de transmisión y la eficiencia de la transmisión se verá afectada.

- En segundo lugar, cada vez que expire el RTO del emisor, éste, suponiendo que se trata de un problema de congestión, utilizará el mecanismo de Slow Start y consecuentemente cerrará la ventana de transmisión. Teniendo en cuenta que, a diferencia de las redes fijas, el problema principal de las comunicaciones inalámbricas es la alta tasa de error y no la congestión, se estaría aplicando una política de recuperación inadecuada que muy lejos de ayudar, ralentizaría aun más la transmisión.

- Por último, debe destacarse también que el protocolo de compresión Van Jacobson no contempla la utilización de opciones TCP, como por ejemplo SACK [6] y Timestamp [8]. En comunicaciones móviles es recomendable la utilización de la opción SACK [13], por lo tanto parece adecuado que el protocolo de compresión contemple la utilización, como mínimo de esta opción, y que sea capaz de comprimirla de forma eficaz.

En [3, 1] se muestra que, en un entorno inalámbrico, el throughput obtenido en conexiones que utilizan compresión Van Jacobson es menor que en conexiones en las que no se aplica compresión y por lo tanto no se considera como una buena opción para comunicaciones móviles. [4] presenta posibles soluciones para adaptar el protocolo Van Jacobson a entornos con altas BER. Para ello propone la implementación de algoritmos que aceleren la recuperación en caso de pérdida de sincronismo entre el compresor y el descompresor; además expone la necesidad de adaptar el protocolo de Van Jacobson a comunicaciones IPv6. Ambos puntos se incluyen en el método de compresión IPHC [10].

## 3. COMPRESIÓN IPHC

IPHC [10] es un método de compresión pensado no sólo como posible solución a los inconvenientes indicados en el apartado anterior para comunicaciones inalámbricas, sino también como alternativa para la compresión de cabeceras que no estén basadas en TCP.[4] Como método pensado para las comunicaciones móviles multiacceso, en el descompresor existirán campos de Identificador de conexión (CID) independientes para cada compresor.

Tal como puede observarse en la Figura 4, la estructura del paquete comprimido IPHC es prácticamente idéntica a la definida en [7]. El campo del identificador de conexión es obligatorio para evitar confusiones en el descompresor. El paquete comprimido contempla la utilización de opciones TCP,

*[4] El protocolo IPHC puede comprimir cabeceras UDP y cabeceras IP fragmentadas, por ejemplo. No contempla, en cambio, la compresión de cabeceras RTP, cosa que ha originado la aparición de otros protocolos de compresión [] que no trataremos aquí*

que se incluyen sin comprimir en el campo O de la cabecera comprimida. Por otra parte, también pueden transmitirse los posibles cambios en los bits reservados (TCP) o en el campo de tipo de servicio (IPv4/IPv6). En el campo Random, en el caso de comprimir subcabeceras , se incluyen los campos variables.

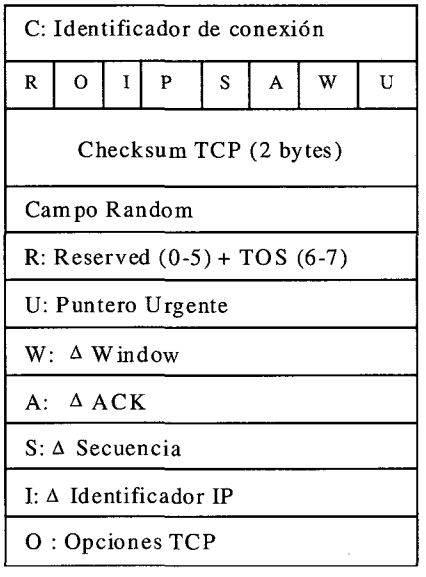


**Figura 4** *Cabecera comprimida (IPHC)*

En lo referente a la recuperación después de errores, se proponen dos formas de recuperar el contexto más rápidamente [4]:

a) Algoritmo TWICE: En las comunicaciones «bulk data», por ejemplo el ftp, la variación de los campos de la cabecera comprimida (Delta Seq, Delta ACK) suele seguir un mismo patrón. En caso de error el descompresor podrá aplicar a la cabecera base hasta dos veces el incremento indicado en la cabecera del paquete recibido para intentar recuperar el contexto localmente. Si aun así la descompresión es incorrecta, se recurrirá a b) o se esperará a que el contexto se actualice tal como en [7].
b) Petición de cabeceras: Si la descompresión del paquete recibido es incorrecta. el descompresor, puede solicitar al compresor que le envíe un paquete sin comprimir o comprimido de manera no diferencial para solventarlos.

Aunque estas soluciones representan una mejora del throughput respecto a la compresión Van Jacobson [4], no son totalmente eficientes en presencia de errores. El algoritmo TWICE presenta inconvenientes a la hora de comprimir los reconocimientos propios del algoritmo TCP (debido a la existencia de «delayed ACKs») y tampoco funcionará en servicios habituales como la descarga de páginas web o correo electrónico, debido a que en ellos es frecuente la variación del tamaño de los segmentos de datos. Además, tampoco será posible la recuperación si se producen errores en más de un paquete consecutivo. Por otra parte, mediante la petición de cabecera, como mínimo se tardará un RTT en corregir el contexto en el descompresor, y por lo tanto aun se perderán muchos paquetes antes de que esto suceda.

Por lo tanto, parece necesario un método que evite las situaciones de inconsistencia[5] en el descompresor. En la siguiente sección, se exponen diferentes propuestas ideadas con este fin.

## 4. OTRAS PROPUESTAS

Entre las posibles opciones de compresión que se centran en evitar la pérdida de contexto se encuentran [5, 2]. Estas propuestas basan su funcionamiento en variaciones del protocolo Van Jacobson ; además el grupo de trabajo ROHC de la IETF en el área de transporte desarrolla [17] como una nueva alternativa a [7] .

En [7] los campos de cada paquete se comprimen tomando como referencia el paquete anterior, por lo que si se pierde un paquete debe retransmitirse toda la ventana de transmisión.

En [5], en cambio, se mantiene un contexto al que se refieren todas las cabeceras comprimidas (cabecera base). De esta manera, aunque se pierda un paquete comprimido, no peligrará la consistencia del descompresor y sólo deberá retransmitirse el paquete perdido. Esta cabecera base sólo variará si se envía un paquete descomprimido; por ejemplo una retransmisión. La pérdida de uno de estos paquetes supondría la inconsistencia del descompresor. Por lo tanto, aunque la posibilidad de pérdida de contexto se ha minimizado, aun podría producirse.

El principal inconveniente de [5] es que si en la conexión no se producen errores durante un periodo de tiempo elevado, no se actualiza el contexto. Entonces se necesitará una cabecera cada vez mayor para poder codificar todos los campos y se desaprovechará ancho de banda.

La eficiencia mejora en enlaces sin errores si se introduce un valor umbral para los campos de las cabeceras comprimidas [2]. Si cualquier campo del paquete comprimido sobrepasa este umbral, se deberá actualizar el contexto. Esta actualización se indicará activando el primer byte de la máscara del paquete comprimido [7] en el que se ha sobrepasado el valor umbral y se realizará a partir de la información del paquete anterior que debe poder almacenarse en compresor y descompresor.

[5] *Consideramos que hay inconsistencia en el descompresor cuando su contexto no coincide con el del compresor y por lo tanto, se descomprimen erróneamente los paquetes y, por lo tanto, se descartan.*

Destacamos que, debido a que se utiliza el primer byte de la máscara, no es posible la adaptación inmediata de este método de compresión para poder utilizar el protocolo IPv6.

La propuesta actual del IETF, [17], pretende elaborar un método de compresión transparente, eficiente y robusto indicado para las comunicaciones móviles y , en general, para cualquier tipo de enlaces caracterizados por altos RTTs y altas tasas de pérdidas. Para ello, adopta un funcionamiento y una estructura de paquetes basados en el protocolo ROHC [12] definido para IP/UDP y RTP:

a) Los campos de cabecera se clasifican en tres tipos según la probabilidad de que varíen durante una conexión:

   - Información estática: Presenta un valor constante (direcciones IP, puertos).
   - Información dinámica: Campos que suelen ser constantes durante la conexión pero que puedencambiar de valor circunstancialmente.
   - Información comprimible: Campos cuyo valor puede variar en cada paquete y que pueden incluir se en la cabecera comprimida.

b) Tanto el compresor como el descompresor van pasando por distintos estados[6] de funcionamiento durante la comunicación. La transición entre estados se producirá debido a información de realimentación en caso de enlaces bidireccionales, o a la expiración de temporizadores. Los estados del compresor indican la cantidad de información de la cabecera base de la que dispone el descompresor; sólo en el último estado el compresor puede enviar paquetes comprimidos. En el resto de estados, se enviará información de refresco del contexto.

   Al tratarse de un método de compresión robusta, el descompresor trabajará mayoritariamente en el estado superior (que indica que el contexto está sincronizado); únicamente volverá a estados inferiores en caso de que se sobrepase un límite de paquetes erróneamente descomprimidos.

   Un ejemplo de funcionamiento, suponiendo una estructura de 2 estados para el compresor (INICIO y COMPRESION) y 3 para el descompresor (NO CONTEXTO, CONTEXTO ESTÁTICO, CONTEXTO COMPLETO), sería el siguiente:

   Primeramente, el compresor enviará los campos estáticos (direcciones IP, puertos TCP...) y esperará a recibir un reconoci miento del receptor. A partir de este reconocimiento, el compresor interpretará que la cabecera base del descompresor contiene la información estática correcta (el esta do del compresor será CONTEXTO ESTÁTICO) y enviará un nuevo paquete con información dinámica. Una vez que el descompresor haya confirmado que dispone de toda la información el contexto, tanto dinámica como estática, y que por lo tanto, se encuentra en el estado CONTEXTO COMPLETO, el compresor cambiará al estado COMPRESION y podrán enviarse paquetes con los campos de cabecera comprimidos.

c) La estructura de los paquetes utilizados para iniciar/refrescar el contexto es la misma que la definida en [12], por lo tanto se utilizarán paquetes especiales dependiendo de si se trata de una actualización del contexto estático o dinámico.

d) La estructura del paquete comprimido está aun por definir, pero especialmente de cara a la aplicación del protocolo a enlaces móviles debe contemplar la compresión de:

   - Las opciones TPC: SACK y Timestamp
   - Los campos especificados en IPHC
   - El protocolo IPv6 (incluyendo opciones y extensiones )
   - Cabeceras que contengan subcabeceras IPSEC

En la Tabla 1 se muestra que la eficiencia conseguida en cuanto al tamaño de la cabecera ROHC-TCP es comparable al resto de propuestas de compresión.

| Método de compresión | Sin compresión | Van Jacobson, [5],[2] | IPHC | ROHC |
|---|---|---|---|---|
| Tamaño mínimo de la cabecera comprimida | 40 bytes (TCP/IPv4) 60 bytes (TCP/IPv6) | 3 – 5 bytes (TCP/IPv4) | 4 –5 bytes (TCP/IPv4/6) | < 4 bytes[7] (TCP/IPv4/6) |

**Tabla 1.** *Comparación de los tamaños de cabecera para los distintos métodos de compresión*

Para evitar la pérdida de contexto y proporcionar un método de compresión robusto, se propone que el compresor guarde una ventana de *m* posibles contextos que podría haber en el descompresor. Se comprimirá los paquetes de tal manera que el paquete resultante pueda descomprimirse siempre en el destino sea cual sea el contexto del descompresor.

[7] *El formato del paquete comprimido no está definido aun. Según [19], el tamaño de la cabecera ROHC-TCP debe ser menor o igual a la de IPHC*

Cada vez que se reciba comprobación de que un paquete ha sido descomprimido en el destino, ya sea por realimentación o por el mecanismo implícito de TCP, los contextos asociados a ese paquete y a los anteriores serán borrados del compresor.

Esta técnica, aunque robusta, provoca un aumento del tamaño indeseado del paquete comprimido. Para solventar este efecto, se utilizará la codificación «Window LSB» para los campos de cabecera: se parte del hecho de que los campos de cabeceras variarán entre unos límites establecidos de tal manera que sólo es necesario enviar los *k* últimos bits significativos para poder recuperar el valor original. En el caso de comunicaciones en las que el tamaño del campo de datos se mantiene constante (SEQ [ACK] = m*Payload + n ); se minimizará el tamaño de cabecera enviando solamente *m* y aumentando así la eficiencia de la compresión.

Por último, cabe destacar que la propuesta [17] contempla un método de optimización de conexiones breves. Debido a que es necesario un tiempo considerable para inicializar el contexto en el descompresor, la mayoría de los paquetes enviados en conexiones de corta duración no estarán comprimidos y por lo tanto no se aprovecharán las ventajas del método de compresión de cabeceras. La descarga http de una página web, por ejemplo, se lleva a cabo mediante la apertura de distintas conexiones por las que se transmite una pequeña cantidad de datos. Estas conexiones difieren en el puerto TCP de origen pero coinciden en el valor del resto de los campos estáticos de cabecera (direcciones IP, puerto de destino...). [17] permite replicar en el descompresor [16] los valores de los campos comunes del contexto de este tipo de conexiones, de manera que sólo sea necesario inicializar los campos diferentes. De esta manera, puede minimizarse el número de paquetes no comprimidos que deben enviarse para iniciar el contexto y se mejora así la eficiencia de las transmisiones.

## 5. CONCLUSIONES

En los apartados anteriores se han analizado los inconvenientes de los protocolos de compresión existentes. Aunque existen diferentes propuestas para mejorar la eficiencia obtenida con el método de compresión Van Jacobson en enlaces de baja velocidad con altas BER y RTTs; en la práctica, si se utiliza compresión, el algoritmo implementado sigue siendo [7]. Este hecho, probablemente, se debe a que ninguna de las propuestas puede considerarse como una solución suficientemente global a los problemas de este tipo de enlaces.

El principal problema que presentan [7] y IPHC es que el tiempo de recuperación en el caso de pérdida de contexto es demasiado grande y no puede evitarse, mientras tanto, el descarte de paquetes en el descompresor. Aunque las propuestas expuestas en la sección anterior no solucionan que el contexto pueda recuperarse con más rapidez en caso de pérdidas de sincronismo, minimizan el riesgo de que esto suceda. Sin embargo, la robustez conseguida con [5, 2] presenta otros inconvenientes. En [5] la eficiencia de la transmisión se ve afectada en términos de ancho de banda debido al incremento del tamaño de las cabeceras comprimidas. Por otra parte, se considera necesario que un método de compresión orientado a comunicaciones móviles pueda comprimir cabeceras IPv6; sin embargo [2] no contempla esta opción por defecto.

La propuesta [17] se presenta como una solución eficiente, tanto en robustez (mediante la utilización de una ventana de contextos) como en términos de ancho de banda, ya que se requiere un tamaño de cabeceras igual o menor que el definido en IPHC. Por otra parte, aunque no se contempla la compresión de todas las opciones TCP, se especifica que las opciones SACK y Timestamp, recomendadas en [13], sí que deben poder comprimirse; por lo tanto, considerando que el resto de las opciones no se utilizarán con tanta frecuencia, se concluye que la propuesta [17] es suficientemente adecuada en este aspecto.

Por último, aunque la propuesta nos parece acertada y ,concretamente, una buena solución para las comunicaciones móviles, como nota negativa, destacamos que el hecho de utilizar este método basado en estados podría implicar un coste de procesado destacable, tanto en recursos como en tiempo. Los terminales móviles presentan limitaciones de proceso, por lo que la realización de un número elevado de instrucciones a la hora de comprimir/descomprimir supondría un aumento no deseable del retardo extremo-extremo. Por lo tanto, es un aspecto que debería tenerse en cuenta a la hora de implementar la solución [17].

## 6. AGRADECIMIENTOS

Parte de este trabajo ha estado financiado por el proyecto TIC2000-1041-C03-01.

## 7. REFERENCIAS

[1] A. Calveras. J. Paradells. "Performance Optimisation Evaluation of TCP/IPOver Wireless Networks". IEEE International Performance, Computerting and communications Conference.Phoenix/Tempe (Arizona). USA. Febrero1998.

[2] Calveras. M. Arnau. J. Paradells. "A controlled Overhead for TCP/IP Header Compression Algorithm

over Wireless Links". The 11th International Conference on Wireless Communications (Wireless'99). Calgary. Alberta. Canada. Julio 1999.

[3] A. Calveras. R. Rebollo. J. Paradells. "La Arquitectura TCP/IP en Entornos Móviles". Telecom I+D '97. Madrid. Octubre 1997

(4] Degermark, M et al, "Low Los TCP/IP Header Compression for Wireless Networks". Mobicom96

[5] Perkins, Stephen J., Mutka, Matt W., "Dependency Removal for Transport Protocol Header Compression over Noisy Channels", IEEE, 1997

[6] M. Mathis et al, "TCP Selective Acknowledgement options", Octubre 1996, RCF2018

[7] Jacobson, V., "Compressing TCP/IP Headers for Low-Speed Serial Links", Febrero 1990. RFC1144

[8] V. Jacobson et al, "TCP Extensions for High Performance", Mayo 1992 RFC1323

[9] S. Deering, R. Hinden ,"Internet Protocol, Version 6 (IPv6). Specification",Diciembre 1998 RFC2460

[10] Degermark, M., Nordgren, B. and S. Pink, "IP Header Compression", Febrero 1999. RFC2507

[11] Casner, S. y V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", Febrero 1999. RFC2508

[12] Carsten Bormann, et. al., "Robust Header Compression (ROHC)", Julio 2001. RFC3095

[13] H.Inamura y G. Montenegro et al, "TCPo v e r second (2.5G) and Third (3G) Generation Wireless Networks", Febrero 2002. RFC3481.

[14] Joan Postel,"Transport Control Protocol", Septiem bre 1981 RFC793

[15] RIU253 IST-2001-36510 D3.1 "Protocol options and their relation with the wireless packet protocols" http://www-riu253.upc.es/

[16] Pelletier, G., "Robust Header Compression (ROHC): Context replication for ROHC profiles", Internet Draft (work in progress), <draft-pelletier-rohc-context-replication-00.txt>, Mayo 2003

[17] Ghyslain Pelletier et al. "RObust Header Compression (ROHC):TCP/IP Profile (ROHC-TCP)", Internet Draft (work in progress), <draft-ietf-rohc-tcp-04.txt>, Marzo 2003

[18] http://www.ietf.org/html/rohc-charter.html

[19] M.West y S. McLann "TCP/IP Field behaviour", Internet Draft (work in progress), <draft-ietf-rohc-tcp-field-behaviour-02.txt>, Marzo 2003

[20] Lars-Erik Jonsson, "Requirements on ROHC TCP/IP Header Compression", Internet Draft (work in progress), <draft-ietf-rohc-tcp-requirements-06.txt>, Junio 2003

## 8. ACRÓNIMOS

| | |
|---|---|
| BER | Bit Error Rate |
| GPRS | General Packet Radio System |
| IETF | Internet Engineering Task Force |
| IP | Internet Protocol |
| IPHC | IP Header Compression |
| IPv4 | Internet Protocol version 4 |
| IPv6 | Internet Protocol version 6 |
| MMS | Multimedia Messaging Service |
| RFC | Request For Comment |
| RLC | Radio Link Control |
| ROHC | Robust Header Compression |
| RTO | Retransmission TimeOut |
| RTP | Real Time Protocol |
| RTT | Round Trip Time |
| SACK | Selective ACKnowledgement |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| UMTS | Universal Mobile Telecommunications System |

## AUTORES

***María Luisa Catalán Cid.*** *Ingeniera de Telecomunicaciones por la ETSETB desde el año 2003. Realizó el proyecto sobre la aplicación de mecanismos de mejora del comportamiento del protocolo TCP en entornos GPRS. Actualmente realiza el doctorado en el Departamento de Telemática de la UPC y participa en el proyecto de promoción tecnológica de UMTS* Servicios UMTS y técnicas avanzadas de localización.

***Anna Calveras-Augé.*** *Dr. Ingeniero de Telecomunicación por la Universidad Politécnica de Catalunya (UPC). Actualmente es profesor Titular de Universidad en dicha universidad, en el Departamento de Ingeniería Telemática. Es experta en optimización de los protocolos de Internet en entornos de red heterogéneos, especialmente el protocolo TCP. Ha estado involucrada en diversos proyectos. Actualmente está trabajando en el proyecto europeo ELIN (The Electronic Newspaper Initiative), el proyecto europeo RIU253 (Recommendations for Internet Usage over 2.5 and 3G networks), y el proyecto español Estudio de las Alternativas para ofrecer servicios multimedia con calidad de servicio extremo a extremo en una red IP integrada con servicios móviles (CICYT TIC2000-1041-C03-01). Ha publicado diversos trabajos en conferencias nacionales e internacionales.*