Optimización del Tráfico P2P-TV mediante el uso de Técnicas de Compresión y Multiplexión

Idelkys Quintana-Ramirez, Jose Saldana, Jose Ruiz-Mas, Luis Sequeira, Julian Fernandez-Navajas, Luis Casadesus
Grupo de Tecnologías de las Comunicaciones (GTC) - Instituto de Investigación en Ingeniería de Aragón (I3A)
Dpt. IEC, Escuela de Ingeniería y Arquitectura, Edif. Ada Byron, Universidad de Zaragoza
50018, Zaragoza, España
Email: {idelkysq, jsaldana, jruiz, sequeira, navajas, luis.casadesus}@unizar.es

***Resumen*—En este trabajo se presenta un estudio sobre la optimización del tráfico P2P-TV, específicamente de SOPCast, una de las aplicaciones más utilizadas por los usuarios a día de hoy. En primer lugar se presenta una caracterización del tráfico, observándose que genera altas tasas de paquetes UDP de pequeño tamaño entre diferentes *peer*, presentando por tanto una eficiencia baja. Posteriormente se propone el uso de un método de optimización de ancho de banda basado en la compresión de las cabeceras y en la multiplexión de los paquetes originales. Se emplean dos políticas de multiplexión, la primera se basa en un *período* fijo y en la segunda se define un *umbral* para el tiempo entre paquetes. En las simulaciones se emplea una traza real del tráfico SOPCast, mostrándose para ambas políticas, una mejora en la eficiencia y valores significativos de ahorro de ancho de banda para el enlace de subida de un usuario (aproximadamente entre un 26% y un 35%). La cantidad de paquetes por segundo se reduce en un factor de 10 en ambos casos. Como contrapartida, se añade un retardo a los paquetes nativos, pero las pruebas muestran que no empeora la experiencia del usuario ni la calidad del vídeo percibido.**

***Palabras Clave*—P2P-TV, SOPCast, video-streaming, compresión, multiplexión, optimización del tráfico.**

I. INTRODUCCIÓN

En la actualidad, el creciente éxito de los servicios en tiempo real (ej. VoIP, juegos *online* y *video streaming*) junto con el incremento del número de usuarios que los emplean, están modificando el *"traffic mix"* presente en Internet [1], debido a la gran cantidad de paquetes pequeños que generan. P2P-TV es uno de los servicios multimedia más extendidos y prácticos en la distribución de contenidos de vídeo y TV en la red, permitiendo el intercambio entre un gran número de usuarios de manera simultánea. La filosofía del modelo *Peer-to-Peer* (P2P) es fomentar la cooperación entre usuarios (también conocidos como *peer*), que pueden actuar como clientes y servidores al mismo tiempo. De esta manera, los costes requeridos tanto en equipos como en ancho de banda se comparten entre ellos. Un *peer* puede convertirse en un emisor de contenidos sin la necesidad de un potente servidor o de un gran ancho de banda. De este modo, P2P presenta una mejor escalabilidad que el modelo cliente-servidor, usado por ejemplo en las *Content Delivery Network* (CDN) [2].

Sin embargo, las redes P2P se han convertido en uno de los más grandes desafíos para la ingeniería del tráfico y los Proveedores de Servicio de Internet (ISPs), al ser una importante fuente de tráfico en la red [3]. Las diferentes aplicaciones P2P-TV son ampliamente empleadas por usuarios residenciales con un limitado ancho de banda de subida. Al emplear *router* de gama media o baja, pueden convertirse en cuellos de botella [4]. De ahí que sea necesario estudiar con más detalle el comportamiento del tráfico P2P y evaluar su impacto en la red. Una de las aplicaciones P2P-TV más populares es SOPCast [5]. En [6] y [7] los autores estudiaron el tráfico de esta aplicación y concluyeron que los paquetes intercambiados se dividen fundamentalmente en dos tipos: $\approx 40\%$ son paquetes grandes (de vídeo) y $\approx 60\%$ son paquetes UDP pequeños (de señalización). Los paquetes pequeños son necesarios para monitorizar, controlar y reorganizar a los paquetes de vídeo (o *chunk*) desde la capa de aplicación. Como se ha mencionado, estas aplicaciones P2P-TV producen altas tasas de paquetes UDP pequeños entre *peer*, los cuales presentan una baja eficiencia debido al significativo *overhead* causado por las cabeceras IP/UDP.

En este sentido, la Fig.1 muestra un histograma del tamaño de los paquetes UDP enviados y recibidos por un *peer*, a partir de una traza SOPCast obtenida durante la transmisión de un partido de fútbol de la *Champions League* 2013, con una duración de $30\ min$ y un total de $940,000$ paquetes. Se aprecia una clara distinción en los tamaños de los paquetes, tanto en el enlace de subida como en el de bajada (*uplink* y *downlink*, respectivamente). En el primer caso, aproximadamente el 90% de los paquetes son pequeños, correspondiéndose a la señalización del nivel de aplicación, utilizados para realizar el acuse de recibo de los paquetes de vídeo. Sin embargo, se puede observar un conjunto de paquetes con un comportamiento totalmente opuesto: paquetes de gran tamaño que incluyen la información del vídeo. Esta distribución se corresponde al comportamiento típico de un *peer* que sólo recibe vídeo; en las redes P2P la mayoría de los *peer* asumen el papel de sólo consumidores de contenidos durante la mayor parte del tiempo [2].

La distribución del tamaño de los paquetes mostrada en la Fig.1, hace muy conveniente una optimización del tráfico. Si se comprimen las cabeceras y se utiliza la multiplexión para los paquetes de señalización, se pueden obtener ahorros

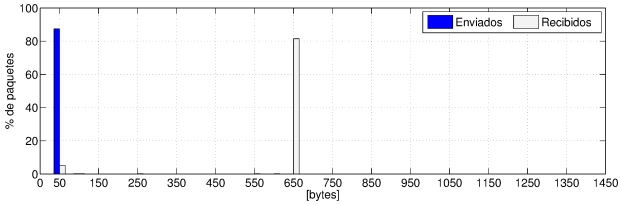


Fig. 1. Histograma del tamaño de los paquetes SOPCast intercambiados entre dos *peer*.

de ancho de banda significativos, ya que el método es especialmente interesante para flujos de paquetes pequeños. Las técnicas de multiplexión ya se han empleado en otros servicios de tiempo de real como por ejemplo VoIP [8] y juegos *online* [9], lográndose un notable ahorro de ancho de banda. Este mecanismo de optimización se podría implementar en varios lugares de la comunicación: en la propia aplicación SOPCast, en los *router* presentes entre dos *peer* o podría ser implementado por los propios proveedores de la red. Estas técnicas introducen un retardo adicional provocado por el tiempo de retención de los paquetes en el multiplexor.

En resumen, este trabajo presenta el análisis de las ventajas de optimizar el tráfico UDP de aplicaciones P2P-TV. Para ello ha sido necesario definir dos políticas de multiplexión adecuadas. El estudio cuantifica la reducción del ancho de banda y del número de paquetes por segundo, mejorando el uso de los recursos de la red. Por otro lado, se asegura que el retardo añadido no afecta a la percepción final del usuario de SOPCast.

El resto del trabajo se organiza de la siguente manera: en la Sección II se discuten los trabajos relacionados. En la Sección III se describen los métodos de compresión y multiplexión propuestos. Las pruebas realizadas y los resultados obtenidos se muestran en la Sección IV y el trabajo se cierra con las conclusiones.

II. TRABAJOS RELACIONADOS

A. Aplicaciones P2P-TV

En los últimos años ha habido un creciente interés en el análisis y caracterización del tráfico P2P-TV. Muchos investigadores se han centrado en el impacto de diferentes aplicaciones de *video streaming* en las redes de comunicaciones [10], como por ejemplo: *PPLive, TVAnts, Coolstreaming, SOPCast y PPStream*. Las mejoras que aportan a los operadores de redes IPTV [11], la calidad de experiencia aportada (*QoE*) [7], [12], los diferentes algoritmos de distribución [13] y el creciente número de usuarios que hacen uso de ellas [5], son varios de los temas fundamentales abordados en relación con los servicios P2P.

En la literatura se pueden encontrar diferentes trabajos centrados en la caracterización del tráfico, incluyendo las velocidades de *upload* y *download* de contenidos, el tipo de paquetes intercambiados y los protocolos empleados [6], así como los mecanismos empleados en algunos programas P2P-TV. Otros trabajos se centran en las características y propiedades de las estructuras de las redes P2P [14] y las implicaciones del tráfico P2P-TV para los ISPs [3], [15]. Sin embargo, en ninguno de los trabajos citados centran su atención en la optimización del tráfico y su efecto en otros servicios que comparten el mismo enlace de acceso a Internet.

B. Comportamiento del SOPCast

Entre las aplicaciones mencionadas, SOPCast se ha convertido en una de las más estudiadas debido a la gran cantidad de personas que la utilizan [5]. El estudio de su funcionamiento, el ánalisis de su rendimiento y las mediciones de la *QoE* aportada son importates temas para los investigadores, operadores y usuarios finales. Esta aplicación funciona sobre un protocolo de comunicación propietario denominado **sop** o **SoP technology** [16]. En [2] y [17] se puede encontrar una caracterización detallada de SOPCast, enfocada especialmente en el tiempo entre paquetes, el número de *peer* con los que se intercambian datos, la duración de la comunicación, entre otros. En [6] y [7] se estudian los mecanismos básicos de SOPCast y se muestra que el tráfico de señalización y vídeo es transportado fundamentalmente sobre UDP.

El sistema de *streaming* desarrollado por SOPCast se sustenta en una arquitectura de distribución no estructurada de tipo *mesh-based*, donde se posibilita que cada *peer* descargue y distribuya los contenidos, llegando a existir múltiples proveedores y consumidores de los mismos contenidos. Este mecanismo es tolerante a fallos, pues los *peer* pueden incorporarse y abandonar la red P2P en cualquier momento y sin previo aviso, asegurando que los errores sean los menos posibles en la visualización del vídeo. Se precisa un mecanismo de control del tráfico para establecer y mantener esta estructura de *peer*; siendo necesario un tráfico de señalización compuesto por paquetes UDP pequeños (menos de $100\ bytes$) [7]. Este control de flujo es necesario para gobernar el intercambio de los diferentes segmentos *(chunk)* en los que se divide el vídeo. En la bibliografía se proponen tres niveles de clasificación para los *peer* de la aplicación; una pequeña cantidad de *super peer* (aproximadamente 5) son los responsables de proveer la mayor cantidad de vídeo (casi el 90% del total) a un *peer*; los *ordinary peer* envían algún contenido y finalmente los *supplementary peer* solo intercambian paquetes de señalización [2].

Al inicio de la aplicación se necesita de un período de tiempo para realizar la búsqueda de *peer* activos, de los cuales se pueda descargar vídeo, en la estructura de distribución. Una vez que se obtiene la lista de *peer* activos y se selecciona el canal deseado, el *video stream* es almacenado en dos *buffer* consecutivos antes de comenzar su visualización. El primero se corresponde al *buffer* de la propia aplicación SOPCast, y añade un retardo desde el momento en que un canal se selecciona hasta que comienza el *streaming* del vídeo; el segundo es el *buffer* del reproductor del cliente. Como consecuencia, el tiempo total que el cliente requiere para disfrutar del *live streaming* oscila entre 30 y $40\ seg$ [7].

C. Métodos de Optimización del Tráfico

Como se ha mencionado, la gran cantidad de paquetes pequeños generados por SOPCast produce un *overhead* significativo en la red, de la misma manera que ocurre con otros servicios multimedia (VoIP, videoconferencia y juegos *online*), ya que sus requerimientos de tiempo real hacen que envíen una gran cantidad de paquetes por segundo, con lo que tienen poca eficiencia. Las técnicas de multiplexado y compresión se utilizan hace ya algún tiempo, e incluso se han estandarizado para escenarios donde varios tráficos de tiempo real comparten la misma ruta [8]. En [9] y [18], se emplea un método denominado TCM (*Tunneling, Compressing and Multiplexing*), con el que se logra ahorrar ancho de banda en el tráfico UDP de los juegos *online*. Se comprimen las cabeceras de los paquetes mediante algoritmos estándar; se multiplexa un número de paquetes en uno de mayor tamaño, y finalmente se realiza el envío extremo a extremo empleando un túnel L2TP. Los resultados muestran un ahorro de ancho de banda de hasta un 38% para juegos que emplean IPv4/UDP,

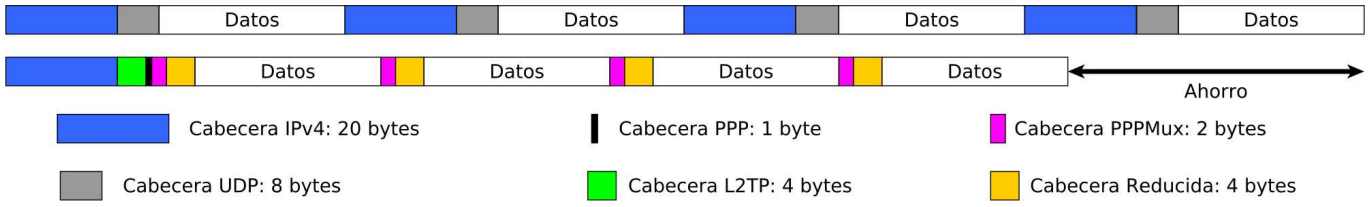


Fig. 2. Tráfico Original y Multiplexado de SOPCast (*con tamaño de paquetes a escala real*).

añadiendo un retardo a causa de la retención de los paquetes originales en el multiplexor.

Respecto al algoritmo de compresión de cabeceras, en este trabajo se precisa uno capaz de comprimir cabeceras IP/UDP, por lo que se podría utilizar IPHC [19] o ROHC [20], siendo este último más reciente y con un mejor comportamiento en redes inalámbricas, aunque conlleva más complejidad en su implementación. Estos métodos utilizan la redundancia de los campos de las cabeceras IP y UDP, para evitar el envío repetido de algunos de ellos. Utilizan también compresión *delta* para reducir el número de bits de los campos con un comportamiento incremental (ej. el número de secuencia). Se necesita por tanto un *contexto*, que se transmite inicialmente con las primeras cabeceras y que almacena los valores de los campos que no se envían, y debe estar sincronizado entre el emisor y el receptor.

Considerando lo anteriormente expuesto, las técnicas de optimización presentadas en [9] pueden ser muy útiles si se aplican al tráfico P2P-TV, caracterizado por generar altas tasas de paquetes pequeños. Se podrían multiplexar los paquetes que se envíen a mismo *peer*, por ejemplo los paquetes de acuse de recibo del nivel de aplicación. Incrementar la eficiencia de estos flujos podría ser beneficioso para las redes residenciales y de agregación, puesto que el ahorro puede llevar a una mejor utilización de los recursos de la red, y la optimización de los recursos permite que más *peer* puedan participar sin trabas en la distribución de contenidos dentro de la red P2P [21].

III. COMPRESIÓN Y MULTIPLEXIÓN

En primer lugar, necesitamos un protocolo capaz de comprimir cabeceras IP/UDP. En este caso se ha seleccionado IPHC, por ser suficiente para los própositos de este trabajo y por presentar una implementación más sencilla que la de ROHC. IPHC es capaz de comprimir las cabeceras UDP a 2 $bytes$, empleando sólo 8 $bits$ para el *Context Identifier* (CID) y evitando el campo de control opcional (*checksum*). La cabecera IPv4 puede comprimirse también a 2 $bytes$, por lo que se considerará una media de 4 $bytes$ para todas las cabeceras comprimidas, excepto para las cabeceras completas que serán de 28 $bytes$ y que, de acuerdo con la especificación de IPHC se envían cada 5 seg [22].

Para ilustrar el *overhead* que puede generar el tráfico original de SOPCast y el ahorro de ancho de banda que es posible obtener gracias a la compresión de cabeceras y la multiplexión de los paquetes, la Fig.2 muestra la reducción del tráfico alcanzado cuando cuatro paquetes P2P-TV se multiplexan en uno más grande.

En este trabajo, se utilizarán dos políticas diferentes para seleccionar qué paquetes se multiplexan juntos. Están basadas en el uso de un *período* fijo o un *umbral* de tiempo entre paquetes respectivamente (Fig.3(a) y Fig.3(b)). Los paquetes generados por la aplicación SOPCast se denominarán *nativos*, para diferenciarlos de los multiplexados o *mux*.

Ambas políticas tratan de mantener los valores del retardo añadido por debajo de una cota superior, para evitar afectar la *QoE* de los usuarios de SOPCast. Algunos servicios multimedia, como VoIP o los juegos *online* presentan restricciones muy rigurosas para el retardo añadido. Sin embargo, en el caso de P2P-TV este problema es menos severo, pues los contenidos descargados se almacenan en el *buffer* de la aplicación antes de reproducirse. En SOPCast, el *buffer* puede almacenar aproximadamente un minuto de vídeo [5], por lo que la mayor limitación en el número de paquetes a multiplexar vendrá dada por la Unidad Máxima de Transferencia (MTU) de la red.

A continuación se explican en detalle las dos políticas de multiplexión propuestas.

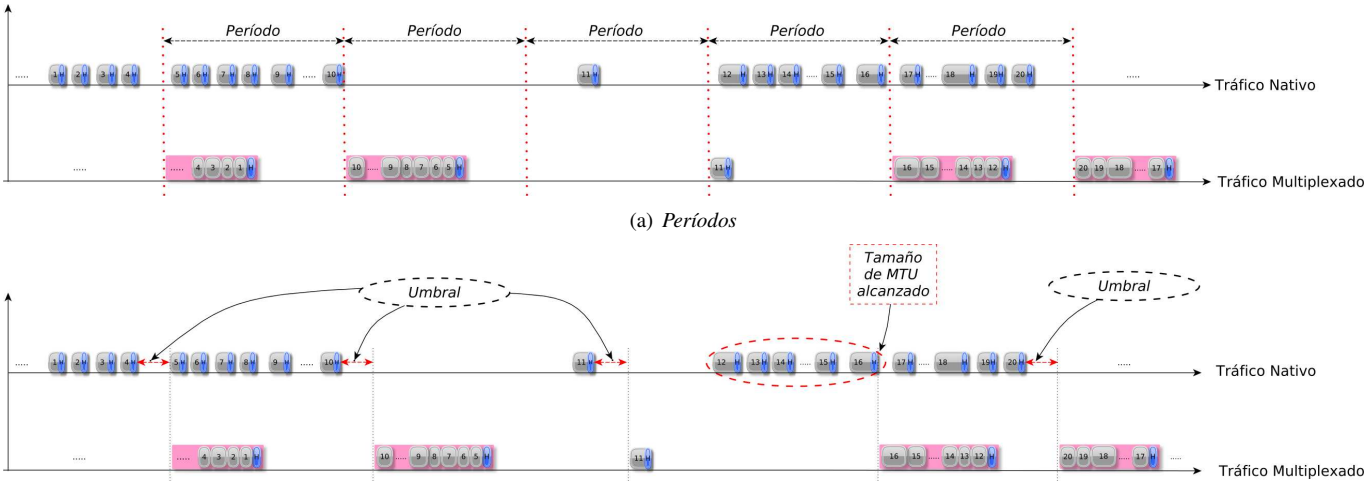
A. Multiplexión basada en un Período

En este caso se define un *período*, de forma que se envía un paquete multiplexado, incluyendo los que han llegado hasta ese momento (Fig.3(a)), al final de cada intervalo de tiempo. Hay tres excepciones: si no ha llegado ningún paquete, no se envía nada; si sólo hay un paquete, se envía en su forma original; finalmente, si se alcanza el tamaño de la MTU, se envía el paquete multiplexado y se comienza un nuevo *período*. Si el valor del *período* aumenta, el ahorro de ancho de banda (*BWS*) mejorará, pues los paquetes multiplexados serán más grandes y el *overhead* total disminuirá. Los valores seleccionados para el *período* no pueden incrementarse indefinidamente, ya que se perdería el contacto con los *peer* proveedores del vídeo.

B. Multiplexión basada en un umbral del tiempo entre paquetes

La caracterización del tráfico generado por SOPCast en [17] sugiere que el envío de paquetes entre dos *peer* sigue un patrón a ráfagas, observándose grandes grupos de paquetes consecutivos cada ciertos intervalos de tiempo. Se puede notar que los *peer* reciben la información de vídeo concentrada en bloques, es decir, en un intervalo determinado de tiempo se recibe una cantidad notable de paquetes de vídeo consecutivamente; posteriormente sólo se reciben paquetes de señalización hasta que comienza nuevamente la transmisión de información. De ahí que se puede concluir que el tráfico analizado sugiere un patrón a ráfagas (Fig.4).

Teniendo esto en cuenta, se ha definido una polítca de multiplexión capaz de adaptarse a este comportamiento (Fig.3(b)). Se define un diagrama de estados (Fig.5), que comienza en un estado de *espera* hasta que llega el primer paquete de la ráfaga. Una vez que se recibe un paquete, el sistema pasa al estado



(a) *Períodos*

(b) *Umbrales*

Fig. 3. Políticas de Multiplexión.

de *almacenamiento* (*transición A*), en el cual se acumulan los paquetes que van llegando y que presentan un tiempo entre paquetes menor o igual al *umbral* seleccionado (*transición B*). Sin embargo, si el tiempo entre dos paquetes supera el valor de dicho *umbral* o si se alcanza el valor de la MTU (*transición C*), el sistema considera que la ráfaga concluye, entonces se multiplexan los paquetes que han llegado hasta ese momento y se envían; el sistema retorna al estado de *espera*.

Para lograr que esta política se adapte al tráfico, se debe seleccionar un valor adecuado del *umbral* (tiempo entre paquetes), que nos permita discriminar claramente qué paquetes forman parte de cada ráfaga.

IV. PRUEBAS Y RESULTADOS

En esta sección se presentan algunos resultados de las simulaciones realizadas. En primer lugar, se obtuvieron trazas de tráfico de la aplicación con la herramienta Tshark, capturándose alrededor de 30 min de un partido de fútbol de la *Champions League* 2013. El cliente SOPCast se encuentra ubicado en nuestro campus universitario, con una dirección IP pública; trabajando sobre Linux (kernel $2.6.38-7$), y con un procesador $Intel® \; Core^{TM} \; i3 \; CPU \; 2.4 \; GHz$.

Para las pruebas se utiliza el tráfico intercambiado con el *peer* que nos provee con más del 90% del vídeo durante toda la comunicación y con el cual, por tanto, existe la mayor cantidad de paquetes intercambiados [6]. Analizando este tráfico, se nota que por cada paquete de vídeo aparece un paquete de confirmación del nivel de aplicación (ACK) de 28 $bytes$ de *payload* (Fig.6); esto explica la gran cantidad de paquetes pequeños generados, que presentan una gran redundancia en los campos de la cabecera, hecho que es de interés para la compresión de las mismas. Se seleccionan valores entre 10 y 50 ms para definir el *período* y los *umbrales* utilizados en las simulaciones, pues más del 84% de los paquetes presentan un intervalo de llegada en este rango.

En primer lugar, para comprobar que los retardos añadidos (en el orden de las decenas o centenas de milisegundos) por las técnicas de optimización empleadas no afectan a la visualización del vídeo, hemos realizado una prueba utilizando Netem para filtrar los paquetes ACK enviados desde la aplicación local al resto de los *peer* durante la visualización de un vídeo. A medida que los *peer* dejan de recibir las confirmaciones, asumen que se ha desconectado la sesión y por tanto dejan de enviar contenidos. Aún así, el *video streaming* se sigue reproduciendo sin problemas en el ordenador local durante casi 1 min. El comportamiento observado cuadra con [5], donde se llega a la conclusión de que el tamaño del *buffer* de SOPCast tiene esa misma duración.

Una vez comprobado que las técnicas de optimización no empeoran la experiencia del usuario, se ha utilizado MATLAB para realizar simulaciones con el fin de obtener tráfico comprimido y multiplexado para ambas políticas. En primer lugar, se separa la traza obtenida del SOPCast en dos: el tráfico de subida generado por nuestro peer (*uplink*) y el tráfico de bajada (*downlink*); y se elimina el tráfico generado por la fase de inicialización de esta aplicación P2P-TV. Los experimentos se han centrado en el tráfico de subida (*uplink*), pues es donde se presenta la mayor limitación de las redes residenciales. Para generar el nuevo tráfico se tiene en cuenta el instante de generación del paquete *nativo* y su tamaño. Posteriormente, se aplica la compresión de las cabeceras del flujo *nativo*; y finalmente se calcula el tiempo y tamaño de cada paquete multiplexado para cada política.

La Fig.7 ilustra el ahorro de ancho de banda (*BWS*) obtenido en ambas políticas. Primeramente se observa que se obtienen valores significativos de ahorro, entre el 25% y el 35% del total enviado por el *peer* local. Cuando se emplea la política basada en un *umbral* de tiempo entre paquetes, se obtiene un *BWS* entre un 33% y un 35%. Como el tiempo entre los paquetes pertenecientes a una misma ráfaga es aproximadamente 10 ms en la mayor parte de los casos, si se selecciona dicho valor como *umbral*, no se obtienen valores óptimos de *BWS*, pues se multiplexan muy pocos paquetes. Sin embargo, para *umbrales* superiores a 12.5 ms, el *BWS* presenta un mejor comportamiento y se aprecia que se mantiene prácticamente constante para el resto de los *umbrales*. Para la política basada en un *período*, el *BWS* alcanzado varía entre el 26% y el 33%. De manera similar a los resultados obtenidos en [18], los valores de *BWS* presentan un comportamiento asintótico.

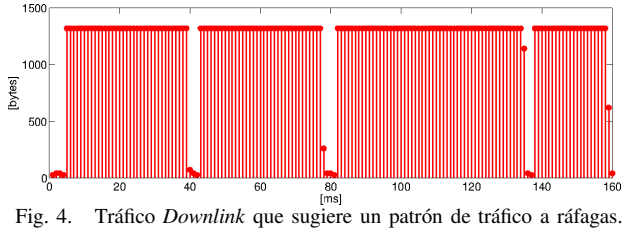


Fig. 4. Tráfico *Downlink* que sugiere un patrón de tráfico a ráfagas.

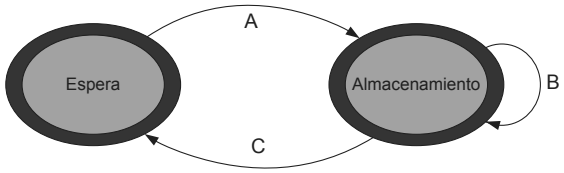


Fig. 5. Diagrama de Estado empleado en la política de *Umbrales*.

A continuación se presentan los histogramas del tamaño de los paquetes multiplexados en ambas políticas, usando un *período* y un *umbral* de 20 ms (Fig.8). Comparando los resultados obtenidos, se observa que con el uso del *período* se genera mayor cantidad de paquetes pequeños, y muy pocos paquetes grandes, con respecto a la política basada en un *umbral*. En el segundo caso, se consigue multiplexar una mayor cantidad de paquetes, al adaptarse mejor al tráfico generado. Este resultado avala los mejores resultados de *BWS* para esta política.

En la Fig.9 se presenta el número de paquetes por segundo (pps) generados en el caso del tráfico *nativo* y cuando se utiliza cada una de las políticas de multiplexión. Se observa una reducción significativa de este parámetro, que baja desde casi 50 hasta unos 5 pps al multiplexar. Como se ha explicado, esta disminución resulta interesante para reducir la carga en los *router* y al mismo tiempo, se muestra que con el aumento del *período* o el *umbral* aumenta también el número de paquetes multiplexados. Sin embargo, hay una diferencia: mientras aumenta el valor del *período*, la cantidad de pps disminuye; en el caso del *umbral*, llega un momento en que su incremento no produce ninguna mejora y la cantidad de pps

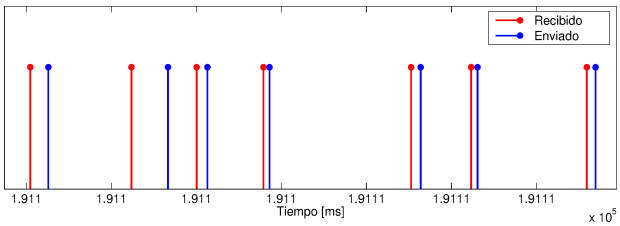


Fig. 6. Tráfico durante una comunicación entre dos peers (paquetes de vídeo y de confirmación).

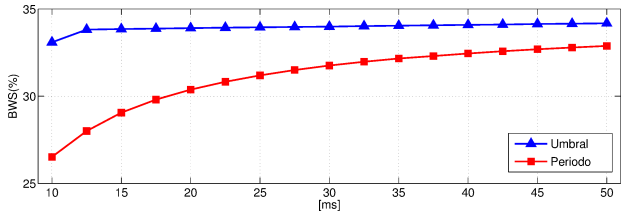
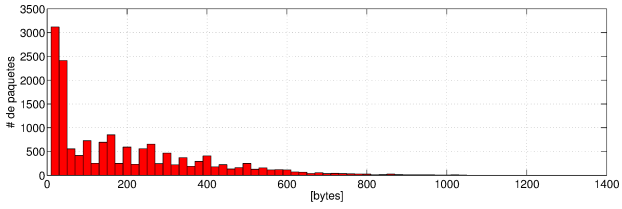
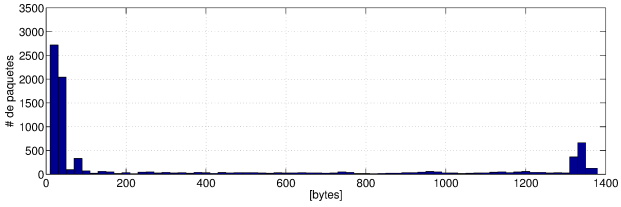


Fig. 7. *BWS* usando las dos políticas de multiplexión.



(a) *Períodos*



(b) *Umbrales*

Fig. 8. Histograma del tamaño de los paquetes *mux* para las dos políticas.

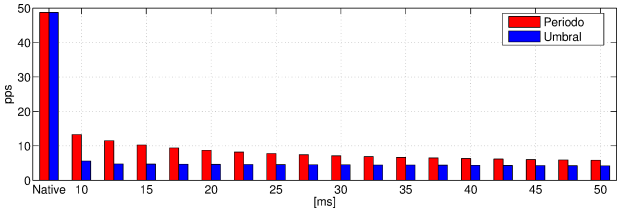


Fig. 9. Paquetes por segundo.

permanece prácticamente constante para valores superiores a 12.5 ms.

Con los resultados de las simulaciones y los valores de *BWS* y pps obtenidos, se ha mostrado que multiplexando el tráfico P2P-TV, aún cuando se considera la comunicación con un único *peer*, se logra un buen ahorro de tráfico y de paquetes por segundo, reduciendo también las necesidades de procesamiento. Estos resultados son muy prometedores para las redes residenciales, donde el *uplink* es limitado y se comparte con otros servicios. Si consideramos además que este tipo de usuarios utilizan normalmente *router* de gama media y baja con capacidad de procesamiento limitada, la reducción de pps conseguida adquiere mayor relevancia.

V. CONCLUSIONES

En este trabajo se aplica un método de compresión de cabeceras y dos políticas de multiplexión al tráfico generado por una popular aplicación P2P-TV basada en UDP. Teniendo en cuenta la cantidad de paquetes generados, se ha mostrado que se pueden obtener valores importantes de ahorro de ancho de banda en redes residenciales.

Se han desarrollado simulaciones con el propósito de estudiar el ahorro de ancho de banda para las distintas políticas de multiplexión propuestas. La primera se basa en la selección de un *período*, de forma que todos los paquetes que llegan durante ese intervalo de tiempo son multiplexados y enviados juntos. La segunda, basada en el hecho de que el tráfico SOPCast responde a un patrón de ráfagas, define un *umbral* para el tiempo de llegada entre paquetes, con el fin de multiplexar los que correspondan a la misma ráfaga. Los resultados muestran que con el empleo de las políticas propuestas se alcanzan ahorros significativos: el *uplink* puede reducirse entre un 26% y un 33% para la política basada en un *período* y entre un 33% y un 35% si se emplea un *umbral* para

el tiempo entre paquetes. Se consigue además, una importante reducción del número de paquetes por segundo. De igual manera, se ha comprobado que los retardos añadidos por el proceso de multiplexión no perjudican la experiencia del usuario con la aplicación.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por la Fundación Social Europea en colaboración con el Gobierno de Aragón, el Proyecto CPUFLIPI (MICINN TIN2010-17298), Ibercaja Obra Social, el Proyecto de la Cátedra Telefónica, la Universidad de Zaragoza, el Banco Santander y la Fundación Carolina.

REFERENCIAS

[1] C. Park, F. Hernandez-Campos, J. Marron, and F. D. Smith, "Long-range dependence in a changing internet traffic mix," *Computer Networks*, vol. 48, no. 3, pp. 401 – 422, 2005.

[2] P. Eittenberger, U. R. Krieger, and N. M. Markovich, "Measurement and analysis of live-streamed p2ptv traffic," in *Performance Modelling and Evaluation of Heterogeneous Networks in HET-NETs 2010*, January 2010, pp. 195–212.

[3] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08, 2008, pp. 363–374.

[4] L. Sequeira, J. Fernandez-Navajas, J. Saldana, and L. Casadesus, "Empirically characterizing the buffer behaviour of real devices," in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2012 International Symposium on*, 2012, pp. 1–6.

[5] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari, "Will iptv ride the peer-to-peer stream?" *Communications Magazine, IEEE*, vol. 45, no. 6, pp. 86–92, 2007.

[6] T. Silverston and O. Fourmaux, "Measuring p2p iptv systems," in *Proc. of NOSSDAV-07, International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2007.

[7] B. Fallica, Y. Lu, F. Kuipers, R. Kooij, and P. V. Mieghem, "On the quality of experience of sopcast," in *Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST'08. The Second International Conference on*, September 2008, pp. 501–506.

[8] B. Thompson, T. Koren, and D. Wing, "Tunneling Multiplexed Compressed RTP (TCRTP)," RFC 4170, November 2005.

[9] J. M. Saldana, J. Fernandez-Navajas, J. Ruiz-Mas, J. I. Aznar, E. Viruete, and L. Casadesus, "First person shooters: can a smarter network save bandwidth without annoying the players?" *IEEE Communications Magazine*, vol. 49, no. 11, pp. 190–198, 2011.

[10] S. Tang, Y. Lu, J. M. Hernandez, F. A. Kuipers, and P. V. Mieghem, "Topology dynamics in a p2ptv network." in *Networking*, ser. Lecture Notes in Computer Science, vol. 5550. Springer, 2009, pp. 326–337.

[11] M. Cha, P. Rodriguez, S. Moon, and J. Crowcroft, "On next-generation telco-managed P2P TV architectures," in *IPTPS '08*, 2008.

[12] U. R. Krieger and R. Schwessinger, "Analysis and quality assessment of peer-to-peer iptv systems," *Consumer Electronics 2008 ISCE 2008 IEEE International Symposium on*, pp. 1–4, 2008.

[13] Y. Liu, Y. Guo, and C. Liang, *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, March 2008.

[14] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Measurement and modeling of a large-scale overlay for multimedia streaming," in *The Fourth International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness; Workshops*, ser. QSHINE-07. ACM, 2007, pp. 3:1–3:7.

[15] D. Moltchanov, Y. Koucheryavy, and B. Moltchanov, "The effect of biased choice of peers on quality provided by p2p file sharing." in *CCNC*. IEEE, 2012, pp. 608–613.

[16] http://www.sopcast.com/.

[17] A. B. Vieira, P. Gomes, J. A. M. Nacif, R. Mantini, J. M. Almeida, and S. V. A. Campos, "Characterizing sopcast client behavior." *Computer Communications*, vol. 35, no. 8, pp. 1004–1016, 2012.

[18] J. Saldana, L. Sequeira, J. Fernandez-Navajas, and J. Ruiz-Mas, "Traffic optimization for tcp-based massive multiplayer online games," in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2012 International Symposium on*, July, pp. 1–8.

[19] M. Degermark, B. Nordgren, and S. Pink, "IP Header Compression," RFC 2507, February 1999.

[20] G. Pelletier and K. Sandlund, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite," RFC 5225, April 2008.

[21] L. Sequeira, I. Quintana, J. Saldana, L. Casadesus, J. Fernández-Navajas, and J. Ruiz-Mas, "The utility of characterizing the buffer of network devices in order to improve real-time interactive services," in *Proceedings of the 7th Latin American Networking Conference*, ser. LANC '12. ACM, 2012, pp. 19–27.

[22] V. Jacobson. (1990, Feb.) RFC1144: Compressing TCP/IP headers for low-speed serial links.