# Estimación proactiva de la calidad de recepción de video streaming en WiFi

E. M. Macías y A. Suárez
Grupo de Arquitectura y Concurrencia
Departamento de Ingeniería Telemática - Universidad de Las Palmas de Gran Canaria
emacias@dit.ulpgc.es, asuarez@dit.ulpgc.es

***Abstract* — En una red WiFi (*Wireless Fidelity*) no siempre se puede garantizar la QoS (*Quality of Service*) aún cuando los dispositivos vayan equipados con tarjetas WiFi 802.11e ya que en estas redes son comunes las desconexiones intermitentes, interferencias debidas a otros equipos operando en la misma banda de frecuencias, etc. que reducen notablemente la QoS demandada por las aplicaciones multimedia. De ahí la necesidad de un software como el que se presenta en este artículo que evalúe si se están dando las condiciones apropiadas para recibir los flujos con una calidad buena de presentación estimando no sólo la congestión sino también otros parámetros como el nivel de cobertura, el *throughput*, la tasa de paquetes recibidos y perdidos, etc. Nuestro software realiza una estimación proactiva, anticipándose a conocer si en el futuro inmediato se van a dejar de dar las condiciones apropiadas para recibir el video streaming con la calidad deseada, usando una técnica Cross-Layer con sentido ascendente que estima parámetros del nivel físico, el Round Trip Time (RTT) y obtiene estadísticas del tráfico global de la red y particular de cada sesión *Real Time Streaming Protocol/Real Time Protocol* (RTSP/RTP).**

## I. Introducción

Los servicios multimedia exhiben requisitos estrictos de QoS que en la actualidad no pueden ser garantizados en las redes WiFi, a pesar de los esfuerzos realizados por el grupo de trabajo 802.11e, ya que en estas redes son comunes el retraso para ganar el acceso al medio inalámbrico, las interferencias con otros canales que limitan el ancho de banda disponible y las desconexiones intermitentes (por ejemplo debidas a movimientos del terminal inalámbrico). Esto justifica el uso de un software como el que se presenta en este artículo que evalúe si se están dando las condiciones apropiadas para recibir los flujos con una calidad buena de presentación estimando no sólo la congestión sino también el nivel de cobertura, el *throughput*, la tasa de paquetes recibidos y perdidos, etc.

La solución óptima a estos problemas exige una estrategia cooperativa o técnica *Cross-Layer* entre niveles [1] que básicamente consiste en obtener los valores de varios parámetros a diferentes niveles y realizar una estimación sobre la calidad de recepción de video *streaming*. En [2] evalúan la proporción de: paquetes perdidos y tiempo de ida y vuelta a nivel de aplicación, datagramas perdidos a nivel de red, y la potencia de la señal y la capacidad a nivel físico para predecir las prestaciones de video *streaming*. En [3] se presenta una aplicación software que detecta cuando un dispositivo inalámbrico entra en un área donde las prestaciones de la red inalámbrica disminuyen. El software clasifica esta área en base a la combinación de parámetros tales como la potencia de la señal, la pérdida de paquetes y la latencia, demostrando que el porcentaje menor de falsas alarmas se obtiene al combinar todos los parámetros para realizar la clasificación. En [4] consideran una metodología *Cross-Layer* a tres niveles: físico, enlace y aplicación pasando información en los dos sentidos (hacia arriba y abajo) obteniendo resultados simulados para una sola sesión multicast.

En este artículo presentamos una aplicación software de estimación proactiva de la calidad de recepción de video *streaming* en WiFi percibido por un cliente en movimiento que recibe vídeo bajo demanda (*VoD - Video on Demand*) o vídeo en directo desde un servidor localizado en un computador fijo de Internet próximo a un *Punto de Acceso* (*PA*) WiFi. Nuestro software se implanta como intermediario (*proxy*) de un cliente de RTSP/RTP usando *Cross-Layer* con sentido ascendente. Nosotros obtenemos una mejor estimación de las condiciones actuales que [2] y [3] al considerar más parámetros y al diferenciar a los flujos multimedia afectados individualmente, y logramos resultados reales buenos y no simulados como [4].

El resto del artículo está estructurado de la siguiente manera. En el apartado 2 se presentan los parámetros que se contemplan para hacer estimaciones. En el apartado 3 se describe el software. En el apartado 4 se presentan algunos resultados experimentales que evalúan el tráfico agregado por el software de detección, el consumo de CPU y batería y su eficiencia a la hora de determinar las alertas evitando falsas alarmas. Finalmente se resumen las conclusiones y se enumeran algunas líneas de trabajo futuras.

## II. Descripción de la técnica cross-layer

Es evidente que si el dispositivo WiFi se encuentra en un área de cobertura deficiente o si la red inalámbrica está congestionada, no se estarán dando las condiciones óptimas para recibir el video con la calidad deseada (y los demás flujos que conforman la sesión, aunque por simplicidad nos referiremos siempre al video). Por ello, es deseable anticiparse a conocer si en el futuro inmediato el usuario percibirá o no el video correctamente para realizar medidas correctoras según

corresponda. Por ejemplo, si el dispositivo está en un área de baja cobertura, se le puede dar indicaciones acerca del movimiento que debe adoptar para recibir mejor la señal de radio. Si la red está congestionada, se podría modificar la codificación del video para disminuir el *throughput* o pausar temporalmente al servidor de *streaming*, etc. Un parámetro que estima correctamente ambas situaciones descritas es el RTT. Nosotros lo evaluamos a nivel de aplicación entre el dispositivo WiFi y el PA, intercambiando paquetes de control *User Datagram Protocol* (UDP) entre ellos usando una arquitectura *manager/agent* (el *manager* instalado en el PA y el agente en cada dispositivo WiFi). No usamos el protocolo *Internet Control Message Protocol* (ICMP) para obtener el RTT ya que en caso de existir un *firewall* en el PA, éste no permitirá generar respuesta a los paquetes ICMP por motivos de seguridad, en cuyo caso obtendríamos siempre un valor alto para el RTT por lo que siempre supondríamos que no se van a dar las condiciones apropiadas para recibir el video con buena calidad y en consecuencia las medidas correctoras que adoptásemos serían innecesarias e incorrectas.

Un único valor de RTT no se puede ser tenido en cuenta para hacer una correcta estimación de si se están dando o no las condiciones apropiadas para recibir el video. Por ejemplo, un valor aislado para el RTT elevado no es indicativo de que haya congestión o poca señal de radio. Por ello, es necesario tener un histórico de las últimos valores de RTT obtenidos y realizar una estimación en base a ello, logrando así reducir el porcentaje de falsas alarmas, esto es, el porcentaje de veces que se estima que no se reciben los flujos correctamente cuando en realidad sí se están recibiendo apropiadamente o viceversa. Los valores de RTT obtenidos se introducen en una ventana deslizante de tamaño fijo, obteniéndose la media de los últimos valores. Los valores en la ventana están discretizados a: *0* (RTT menor a 10 ms; no se detecta ningún problema), *1* (RTT entre 10 y 100 ms; indicativo de que el dispositivo WiFi podría estar entrando o saliendo de un área de baja cobertura o la red está congestionada) y *2* (valores de RTT altos igual o superior a 100 ms). Los umbrales de 10 y 100 ms han sido obtenidos mediante experimentación en un entorno real. Si la media del RTT es igual o superior a *Urtt* (0.5) se incrementa un contador. Si este contador llega a 10 transcurrido 1 segundo, el dispositivo pasa al estado de prealerta (*Pa*) que significa que en breve el video podría dejar de recibirse con la calidad deseada. La cancelación de la *Pa* tiene lugar cuando el contador llega a 0 (este contador se decrementa cada vez que la media del RTT sea inferior a *Urtt*). El tamaño de ventana, el valor máximo del contador y la frecuencia de actualización han sido establecidos mediante experimentación.

Estando en *Pa* hacemos una segunda estimación midiendo parámetros de otros niveles para determinar si efectivamente el usuario dejará de recibir correctamente el video, en cuyo caso lo notificaremos indicando que el dispositivo está en un estado de alerta (*Al*) dejando que sea el usuario, un software de localización, etc. el que realice la medida correctora más apropiada. Por tanto, estando en *Pa* tenemos que averiguar si se debe a su ubicación física (se recibe poca señal de radio) o por la congestión en el canal. Para estimar el nivel de cobertura, evaluamos a nivel físico el valor de señal, de ruido y calidad del enlace (los proporciona el *driver* de la tarjeta inalámbrica) teniendo en cuenta que hay poca consistencia entre los valores devueltos por diferentes fabricantes lo que hace que cada *driver* pueda o no, proporcionar medidas fiables. Para evitar inconsistencias, y siempre y cuando el *driver* que soporta la API de las *Wireless Extensions* (WE) [5] de Linux permita obtener el valor máximo, medio o mínimo que la tarjeta es capaz de detectar para cada parámetro, se facilitan unos límites que nos sirven para calibrar los distintos parámetros de una manera general, y así nos abstraemos de las inconsistencias entre los distintos fabricantes.

Para estimar la congestión, evaluamos a nivel de red el *throughput* global (*Tg*) de la red, así como el *throughput* (*Tf*), la tasa de paquetes recibidos (*Tpr*) y perdidos (*Tpd*) por flujo. La evaluación de *Tg* permite distinguir el tráfico utilizado en las aplicaciones de *streaming* que se analizan, del tráfico utilizado por el resto de las aplicaciones y que influyen en la calidad de recepción del video ya que estas contribuyen a la congestión de la red. Estando en *Pa* y antes de pasar a un estado de *Al*, se tienen en cuenta los valores calculados para: *Tg*, *Tf*, *Tpd* y *Tpr* para las sesiones abiertas en el dispositivo, y se comparan con los umbrales fijados para estos parámetros (los denotaremos por *Up* donde *p* es el parámetro correspondiente). Hay que tener en cuenta que no se conoce a priori estos umbrales por lo que se resuelve esta dificultad optimizando su valor en tiempo de ejecución, calculando la media de los últimos valores obtenidos para *Tg*, *Tf*, *Tpd* y *Tpr* cada segundo y actualizando los umbrales cada 3 minutos. Lógicamente los umbrales no se pueden obtener cuando el dispositivo está en *Pa* o *Al* ya que pueden variar considerablemente al cambiar las condiciones de la red (congestión) o la ubicación del dispositivo (cobertura deficiente). Otro problema añadido es si se usa un *codec Variable Bit Rate* (*VBR*) dado que el *throughput* variará con frecuencia, de tal forma que en esa variación puede que tome valores momentáneos por debajo del valor umbral fijado. En este caso *Tpd* y *Tpr* indican si los flujos se reciben correctamente. Si para un flujo determinado hay valor umbral fijado, la media de los últimos valores obtenidos se compara con ese valor. Si se dan las siguientes circunstancias se activa la *Al*: a) *Tf* < % *Utf* (95%). b) *Tpd* > %*Utpd* (3%). c) *Tpr* < % *Utpr* (95%). En el caso de que, para ese flujo, no haya umbral fijado, los únicos parámetros que sirven de métricas para la toma de decisiones son *Tpr* y *Tpd*. Si *Tpd* > %*Tpr* (20%) entonces se están perdiendo muchos paquetes y la recepción de contenidos multimedia para ese flujo de datos no está siendo correcta.

## III. IMPLANTACIÓN DE LA TÉCNICA CROSS-LAYER EN UN ESCENARIO REAL

En la figura 1 (parte izquierda) se presentan los diferentes programas que conforman nuestro software y su ubicación en las distintas máquinas. El programa *detector* se implanta en los dispositivos asociados al PA que inician al menos una sesión con el servidor de *streaming*. Es un programa multihebra escrito en el lenguaje C encargado de evaluar si para la sesión actual se

dan las condiciones apropiadas para recibir los flujos con una calidad de presentación buena, estimando el RTT, consultando al *driver* de la tarjeta inalámbrica los parámetros de nivel físico a través de las WE y obteniendo las estadísticas de tráfico. Cuando el reproductor de vídeo se activa, se abre un canal de comunicación bidireccional entre éste y el programa *detector* (en nuestra implementación práctica la comunicación se hace a través de un *proxy* para no modificar el código de ningún reproductor). El programa *detector* queda a la escucha de las órdenes RTSP enviadas por el usuario al servidor de *streaming* y comunicadas por el *proxy*, realizando para cada una de ellas las acciones siguientes: a) *SETUP*: se debe almacenar información de la sesión que se acaba de iniciar con el objetivo de poder capturar sus paquetes RTP para calcular el *Tg*, *Tf*, *Tpd* y *Tp;* b) *PLAY*: se debe comenzar a capturar el tráfico global y de la sesión en particular, además de evaluar los parámetros del nivel físico. Si la sesión estaba parada, se reinicia la captura y lectura de parámetros; c) *PAUSE*: se detiene la captura del tráfico y la lectura de parámetros del nivel físico; d) *TEARDOWN*: se finaliza la evaluación de parámetros.

El programa *centralización* se implanta en un dispositivo asociado al PA en un área de buena cobertura. Obtiene información de los dispositivos que ejecutan el programa *detector*: número de sesiones RTSP activas que tienen la *Al* activa por problemas de cobertura o por congestión en la red, etc. Esta información puede ser consultada a través del programa *consulta* por cualquier usuario (por ejemplo podría ser consultada por el PA para realizar un control de admisión de los dispositivos WiFi que quieren asociarse a la red permitiendo o negando su petición en función de cómo se encuentre la red para soportar más usuarios).

## IV. RESULTADOS EXPERIMENTALES

Se realizaron diferentes pruebas encaminadas a evaluar las características del programa *detector*: la sobrecarga extra que introduce en la red, el consumo de CPU y batería en el dispositivo WiFi, su eficiencia a la hora de determinar las alertas y su escalabilidad para gestionar varias sesiones de *streaming*. Por limitaciones de espacio sólo se muestran una parte de los resultados obtenidos (por ejemplo, se han omitido los resultados obtenidos para la recepción de flujos en tiempo real presentado únicamente algunos de los obtenidos para VoD).

### *A. Sobrecarga*

Para este análisis se usó un archivo poco “pesado” que corresponde al caso más desfavorable, aunque menos realista, para comparar con el tráfico introducido por el programa *detector* (en una situación más general, el tráfico multimedia sería mayor). La figura 1 (parte derecha) presenta el tráfico correspondiente a los flujos de los primeros 66 segundos de una sesión de streaming, y el tráfico extra generado por el programa *detector* (se puede considerar baja la sobrecarga introducida). Obviamente, el tráfico extra introducido por el programa *detector* se incrementa linealmente con el número de programas de este tipo en funcionamiento.

### *B. Consumo de CPU y batería*

El incremento de CPU promedio está por debajo del 1% al reproducir 1, 2 y 3 archivos en el mismo dispositivo WiFi. Para evaluar el consumo de batería, el dispositivo empezó con la batería cargada al 100% de su capacidad. La carga remanente en la batería después de los 20 minutos de la reproducción fue la misma tanto al usar el programa *detector* como al desactivarlo, siendo esta carga del 80% para la reproducción de un archivo, del 79% para dos y del 77% para tres (en este último caso, fue del 76% usando el programa *detector*). No se aprecia variación en el consumo de batería cuando se ejecuta el programa *detector*.

### *C. Fiabilidad*

Para evaluar la precisión del programa *detector* a la hora de detectar prealertas y alertas, así como la causa que origina la *Al*, durante el periodo de evaluación movimos el dispositivo a un área de baja cobertura en algunas ocasiones, y en otras saturamos el canal con el envío de tráfico no correspondiente con la sesión de *streaming*. En total, se forzaron que hubieran 5 *Pa* y 3 *Al*. La primera *Al* fue causada por congestión y las otras dos por problemas de cobertura. Esto se corroboró en la práctica pues durante la primera *Al* el programa *detector* obtuvo valores de los parámetros del nivel físico que rondaban los valores medios proporcionados por la API WE. Sin embargo, en las otras dos alertas, estos valores estaban muy por debajo del valor medio lo que significa que el dispositivo WiFi estaba localizado en un área de cobertura reducida. Al evaluar el *Tf*, *Tpr* y *Tpd* de cada flujo, se observó cómo disminuye el *Tf* y *Tpr*, y cómo aumenta *Tpd* durante las *Al*. En la primera y tercera *Al* se detectó que sólo el flujo de vídeo se vio afectado por la congestión. Sin embargo, en la segunda *Al* detectada, el valor del *Tf* (de ambos flujos) disminuyen. Las *Al* se cancelaron automáticamente debido a la finalización de la *Pa* antes de que el programa *detector* observara que todos los flujos se recibían correctamente. En la figura 2 se muestra la media de los valores de RTT normalizados. Se muestra en el eje horizontal la situación temporal de las alertas y prealertas durante el periodo de evaluación con la notación siguiente: *Pai* (prealerta *i-esima*), ~~*Pai*~~ (cancelación de la prealerta *i-esima*) y *Alixy* (alerta *i-esima*, flujos afectados: video (sufijo x con valor *v*) y/o audio (sufijo y con valor *a*)). Se observa como en los instantes en los que se inicia las situaciones de *Pa* los valores de RTT crecen muy rápidamente. Para los casos en los que se activa la *Al*, ésta se cancela cuando el valor medio del parámetro RTT normalizado está por debajo de *Urtt* durante 1 segundo. Obsérvese también que para cualquier *Al* notificada, el valor del RTT aumenta.

## V. CONCLUSIONES

En este artículo se presentó un software que estima si se están dando las condiciones apropiadas para la reproducción óptima de datos correspondientes a una o más sesiones de *streaming* usando el protocolo RTSP/RTP desde un terminal asociado a una red con infraestructura *WiFi*. Las pruebas experimentales demostraron que el software es fiable, poco pesado y consume pocos recursos. Las líneas de trabajo futuras van encaminadas no a mejorar el software en sí, sino a usar la información que de éste se obtiene, de manera que se está trabajando en el diseño e implantación de un mecanismo de control de admisión a la red, y en el desarrollo de medidas correctoras una vez que el programa *detector* señalice una alerta y el motivo que la causa.


## AGRADECIMIENTOS

Este trabajo ha sido subvencionado en parte por el Ministerio de Educación y Ciencia, CICYT y el Fondo Europeo de Desarrollo Regional (FEDER) (TSI2005-07764-C02-01), y la Consejería de Educación, Cultura y Deporte del Gobierno de Canarias y FEDER (PI042004/164).


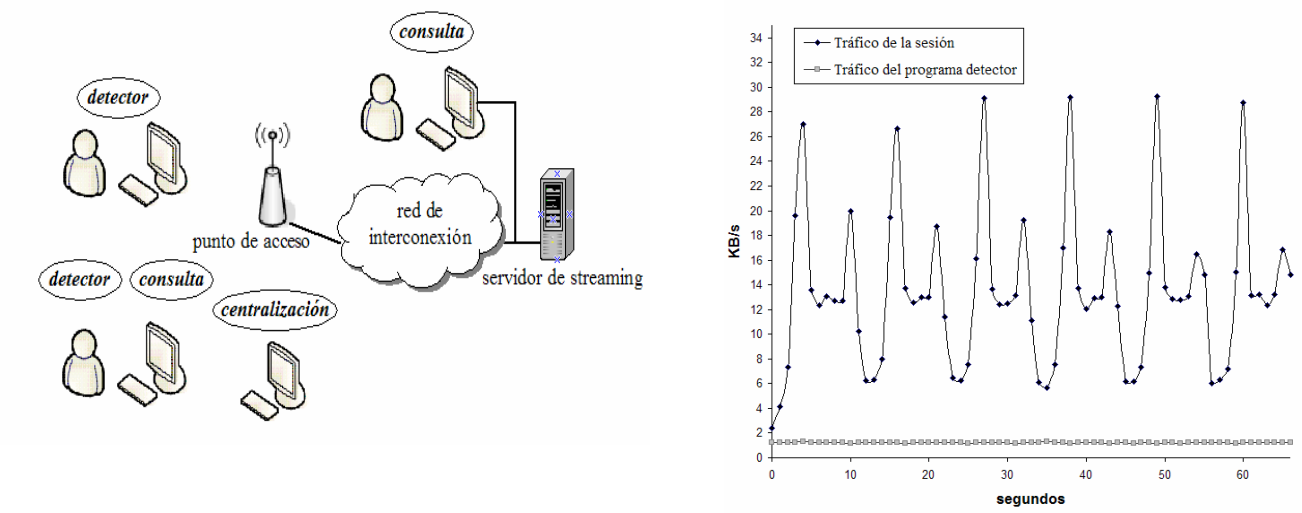


Fig. 1. Visión general de dispositivos e implantación de los programas detector, centralización y consulta (parte izquierda de la figura). Tráfico de la sesión y del programa detector (derecha).

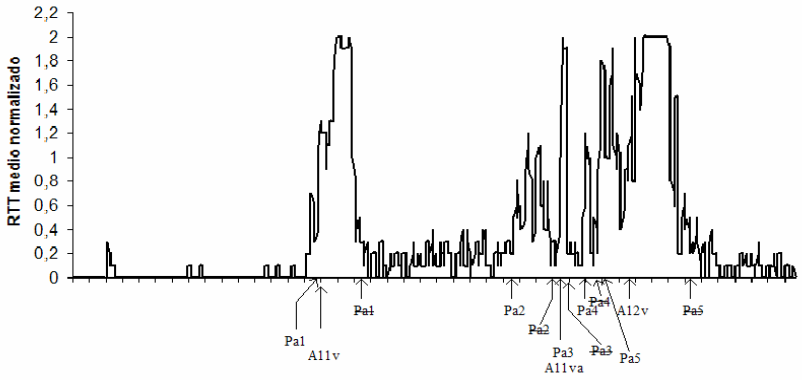


Fig. 2. Valor medio del RTT normalizado.

## REFERENCIAS


[1] W. Kumwilaisak et al, “A cross-layer quality-of-service mapping architecture for video delivery in wireless networks”, *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 10, pp. 1685-1698, 2003.

[2] M. Li, F. Li, M. Claypool, and R. Kinicki, “Weather forecasting: predicting performance for streaming video over wireless LANs“, *Proc. International Workshop on Network and Operating Systems Support For Digital Audio and Video*, pp. 33-38, 2005.

[3] G. Tonev, V. Sunderam, R. Loader, and J. Pascoe, “Location and network quality issues in local area wireless networks”, *International Conference on Architecture of Computing Systems: Trends in Network and Pervasive Computing*, pp. 131-148, 2002.

[4] J. Villalón, P. Cuenca, L. Orozco-Barbosa, Y. Seok, and T. Turletti, “Cross-layer architecture for adaptive video multicast streaming over multirate wireless LANs”, *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 4, 2007.

[5] “Wireless LAN Resources for Linux”. Disponible: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Wireless.html. Última consulta: abril de 2008.