JITEL 2017
XIII Jornadas de Ingeniería Telemática
27-29 Septiembre 2017, Valencia (España)

Actas de las XIII Jornadas de Ingeniería Telemática (JITEL 2017), Valencia (España), 27-29 de Septiembre de 2017.

ISBN: 978-84-9048-595-8
DOI: http://dx.doi.org/10.4995/JITEL2017.2017.6611

Evaluación de la calidad de experiencia de YouTube Live en redes inalámbricas

Luis Jiménez, Marta Solera, Matías Toril, Pablo Oliver.
Dpto. de Ingeniería de Comunicaciones,
ETSI Telecomunicación, Universidad de Málaga,
Campus Universitario de Teatinos, s/n E-29071 Málaga (España).
lrjp@ic.uma.es, msolera@ic.uma.es,mtoril@ic.uma.es,pob@ic.uma.es

***Resumen*—YouTube Live is one of the most popular services on the Internet, enabling an easy streaming of a live video with acceptable video quality. Thus, understanding user´s perception of this service is of the utmost importance for network operators. As in other videostreaming services, YouTube Live traffic is sometimes affected by delays due to unfavourable network conditions, which translate into unacceptable initial reproduction times or image freezes as a result of client´s buffer underrun. Detecting these events is key to ensure an adequate Quality of Experience (QoE). Unfortunately, data encryption makes it very difficult for operators to monitor QoE from packet-level data collected in network interfaces. In this paper, an analytical model to estimate the QoE for encrypted YouTube Live service from packet-level data collected in the interfaces of a wireless network is presented. The inputs to the model are Transport Control Protocol (TCP)/Internet Protocol (IP) metrics, from which three Service Key Performance Indicators (S-KPIs) are estimated, namely initial video play start time, video interruption duration and video interruption. The model is developed with an experimental platform, consisting of a user terminal agent, a WiFi wireless network, a network-level emulator and a probe software. Model assessment is carried out by comparing S-KPI estimates with measurements from the terminal agent under different network conditions introduced by the network emulator.**

***Palabras Clave*—YouTube Live, Streaming, QoE, S-KPIs, Modeling, Pocket, Netem.**

I. INTRODUCCIÓN

En la última década, el incremento exponencial de usuarios y la aparición de nuevos servicios ha traído consigo una completa revolución en las redes de comunicaciones móviles. Se estima que, para el año 2021, existirán 31.750 billones de *smartphones* activos, conectados a diferentes redes [1].

En la actualidad, los operadores se han visto obligados a cambiar sus métodos de gestión de la red, pasando de utilizar indicadores objetivos enfocados en el rendimiento de la red y la calidad de servicio *(Quality of Service,* QoS), a indicadores más modernos y centrados en la opinión del usuario y la calidad de experiencia *(Quality of Experience,* QoE). La gestión de calidad de experiencia tomará, si cabe, una mayor importancia con la incorporación de la quinta generación de tecnologías de telefonía móvil (5G), cuyo lanzamiento se prevé para 2020, y que estará claramente dominada por los servicios de vídeo que representarán el 70% de la demanda de tráfico total [1][2]. En un entorno en el que la oferta de redes y servicios es similar en todos los operadores, la calidad de experiencia se convierte en uno de los principales factores que diferencia a unos operadores de otros y que permitirá fidelizar a los usuarios.

Entre todos los servicios, la reproducción de vídeos a través de la descarga progresiva (*video streaming*) es la aplicación que más tráfico genera en Internet en la actualidad [3]. De las diferentes técnicas, la más popular y la que proporciona una mayor accesibilidad desde cualquier punto de la red es la que utiliza los protocolos HTTP/HTTPS (*HyperText Transfer Protocol/ Secure*). En ella, los vídeos se almacenan en segmentos de diferentes longitudes (de 2 a 10 segundos normalmente), que se codifican con distintas resoluciones (regímenes binarios). En este tipo de *streaming*, el usuario es quien solicita a través de un mensaje HTTP el contenido multimedia, y la descarga del vídeo comienza como respuesta a esta solicitud. Durante la descarga, se cede el control de la descarga al cliente, que va solicitando los segmentos de vídeo mediante mensajes HTTP adaptándose a las fluctuaciones de las condiciones de la red. Las principales plataformas de servicio, tales como *YouTube, Hulu* y *Netflix*, emplean este tipo de *streaming*, siendo el primero el líder indiscutible del mercado.

YouTube ha añadido recientemente una nueva funcionalidad consistente en ofrecer secuencias de vídeo de alta calidad en directo (*Live video streaming*), causando un aumento exponencial en la generación de contenido por parte de los usuarios. Por esta razón, es necesario comprender las características del tráfico generado por este nuevo servicio, para poder monitorizar

y controlar la QoE percibida por sus usuarios [4].

Tradicionalmente, la QoE se ha evaluado mediante pruebas subjetivas realizadas con observadores reales, que reflejan su grado de satisfacción mediante un indicador de puntuación medio de opinión (*Mean Opinion Score,* MOS) [5]. Este tipo de enfoque es complejo, requiere tiempo, y no es válido para monitorización a gran escala. Por ello, en los últimos años se han estudiado nuevos métodos para estimar la calidad de experiencia a partir de indicadores de rendimiento construidos con datos recolectados en los equipos e interfaces de la red. Especialmente prometedores son los métodos basados en la información recolectada a nivel de paquete por sondas de nivel de red (habitualmente Internet Protocol, IP), ubicadas en las interfaces de la red. Dichos métodos son posibles gracias a que, en los servicios que utilizan como protocolo de transporte a TCP (*Transport Control Protocol*), los mecanismos de control de congestión y flujo hacen que los indicadores de rendimiento de nivel de red (p. ej., caudal, tasa de pérdidas, retardo medio) sean un reflejo de la calidad de servicio extremo a extremo. La principal dificultad estriba en identificar la relación existente entre los indicadores de rendimiento específicos del servicio (*Service Key Performance*, S-KPI), (en el caso del *videostreaming*, p. ej., tiempo inicial de espera para la reproducción, número y duración de las interrupciones de la reproducción, ..) y las métricas TCP/IP.

Hasta la fecha, los S-KPIs para el servicio de *videostreaming* se han venido obteniendo mediante la identificación de las distintas fases de la sesión a partir del análisis de los mensajes del protocolo HTTP. Sin embargo, dicho análisis ya no es posible, ya que, desde 2016, el 97% por ciento del tráfico de *YouTube* (video convencional y *Live streaming*) se cifra mediante conexiones HTTPS con *Transport Layer Security* (TLS) y *Secure Sockets Layer* (SSL). La situación se ha complicado aún más con la inclusión de técnicas de *streaming adaptativo* (*Dynamic Adaptive Streaming over HTTP*, DASH). Ambos procesos dificultan enormemente la estimación de los S-KPIs de *YouTube*. En [6], se presenta un modelo de QoE para el servicio de *YouTube* convencional, basado en la estimación del nivel de buffer del cliente. Sin embargo, hasta donde se sabe, ningún trabajo ha propuesto un modelo analítico sencillo para estimar los S-KPIs del servicio de *YouTube Live* a partir de métricas TCP/IP recolectadas en las interfaces de una red inalámbrica.

En este artículo, se presenta un modelo de regresión para estimar los principales indicadores de rendimiento del servicio de *YouTube Live* a partir de medidas obtenidas del análisis de paquetes capturados en las interfaces de una red inalámbrica. Las entradas del modelo son métricas TCP/IP comunes (p. ej., caudal, tasa de pérdida de paquetes o tiempo de ida y vuelta), a partir de las que se estiman tres de los S-KPIs más importantes del servicio, como son el retardo inicial de reproducción, el número total de interrupciones y el tiempo total de interrupción. El modelo se desarrolla utilizando técnicas de regresión sobre datos tomados con una plataforma experimental, consistente en un terminal de usuario, una red de acceso inalámbrica WiFi y un emulador de red. Esta plataforma permite: a) automatizar la creación y emisión de un *Live video streaming*, b) emular la interacción del usuario con el *Live video streaming* a través de un *smartphone*, y c) modificar las condiciones de la red mediante el emulador de red. La validación del modelo se lleva a cabo comparando las estimas de los S-KPIs con las medidas de los mismos realizadas por el agente de usuario bajo diferentes condiciones de red establecidas con el emulador de red. El resto del artículo se estructura de la siguiente manera. La sección II describe la plataforma experimental. La sección III explica el modelo de rendimiento del servicio de *YouTube Live*. La sección IV muestra las curvas de regresión con las que estimar los S-KPIs del servicio con métricas TCP-IP, que es la principal contribución. Por último, la sección V, expone las conclusiones del trabajo.

II. PLATAFORMA EXPERIMENTAL DE PRUEBAS

La Fig.1 muestra un esquema de la plataforma utilizada para automatizar la toma de medidas necesarias para la construcción de los modelos de QoE. La plataforma consta de 2 módulos: un primer módulo (izquierda de la figura) encargado de la emisión en directo de una secuencia de vídeo (*Live video streaming*) de *YouTube*, y b) otro módulo (derecha de la figura) encargado de la recopilación y análisis de las medidas. El módulo de emisión consta únicamente de un PC ejecutando la aplicación *Wirecast*. Por su parte, el módulo de medidas se compone de un terminal móvil conectado por WiFi a un PC con salida directa a Internet. En el terminal móvil, se ejecuta la aplicación que funciona como agente de usuario (*TEMS Pocket*), que modela las interacciones del usuario durante la sesión de *videostreaming*. Esta aplicación permite además recoger las medidas reales de los S-KPIs, al tener acceso a uno de los extremos de la comunicación. En el PC, se ejecuta un emulador de red (*NetEm*) para modificar de forma controlada las condiciones de red (p. ej., ancho de banda disponible, retardo medio y/o tasa de pérdidas de paquetes). Las medidas de red a nivel de paquetes se toman de la interfaz del emulador de red hacia Internet. Esta información se procesa en tiempo diferido con una aplicación de análisis y monitorización de tráfico (*Network Probe*), con la que se obtienen las estimas de los S-KPIs. Dichas estimas construidas con el modelo propuesto se contrastan con las medidas tomadas por el agente de usuario. A continuación, se describen cada uno de estos elementos de la plataforma, detallando el proceso de generación, modificación y captura del tráfico en la plataforma.

A. *Emisión de vídeo Live Streaming*

El proceso de emisión permite la creación de un *Live video streaming* utilizando la plataforma de *YouTube*. Como punto de partida, se requiere un equipo que genere el contenido multimedia a distribuir en tiempo real. En

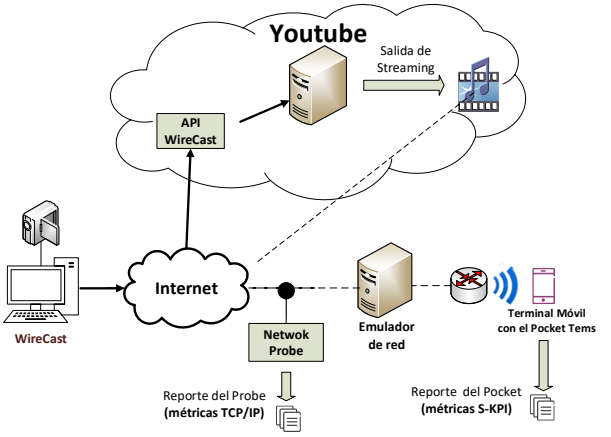

Fig. 1. Plataforma experimental de pruebas.

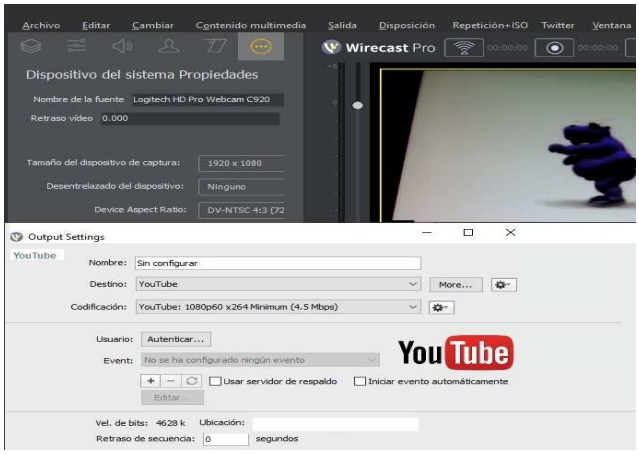

Fig. 2. Configuración *Wirecast Pro* 7.3

la plataforma, se utiliza un servidor de *Live Streaming* consistente en un PC *DELL PowerEdge* T430 con dos procesadores Intel(R) Xeon (R) E5-2630 V3 @ 2.40GHz, cada uno con 8 núcleos. Posee un disco duro STD de 1.4 TB, sistema operativo *Microsoft Windows Server 2016 Datacenter* ver. 10.0.14933 con 64GB de memoria RAM y tarjeta multimedia Matrox® G200 integrada con iDRAC8. El servidor está conectado a un punto de red aislado para evitar variaciones indeseadas del ancho de banda disponible (velocidad de subida de 94 Mbps) para el envío del flujo multimedia a *YouTube*. En el servidor se instala el software *Telestream Wirecast* y la cámara web. *Wirecast* es un software empleado para la emisión del *videostreaming*. Permite producir y transmitir eventos multimedia en tiempo real hacia *YouTube* desde una estación de trabajo [7]. La versión de *Wirecast* utilizada en la plataforma experimenta es *Wirecast* Pro 7.3 de 64 bits, con capacidad de emisión de vídeo en alta definición (HD). Para la captura del vídeo en directo, *Wirecast* emplea un dispositivo externo, que en la plataforma utilizada es una *webcam Logitech HD Pro* C920, que permite realizar grabaciones de alta definición (HD 1080p). Para este dispositivo, debe configurarse el formato de vídeo, el compresor de vídeo y el régimen binario de salida. En este trabajo, se selecciona la resolución 1920x1080, el formato progresivo, el codificador H.264 AVC y el régimen binario de 3-6 Mbps (perfil 1080p). Tras configurar el flujo de vídeo de subida, se debe crear un evento de emisión en la web de *YouTube* iniciando sesión en el servicio *Creator Studio de YouTube*. Este servicio permite organizar el canal donde se efectúa la transmisión, la creación del evento que proporciona el identificador del vídeo (ID) y la gestión de algunas características del flujo de vídeo en tiempo real (p. ej., codificador de audio y vídeo) [8]. Al crear el evento, se deben definir diversos parámetros, como su nombre, la hora de inicio de emisión, las características del flujo de vídeo de bajada y las características tipo de evento (Público, Privado, Oculto). El evento Público distribuye el ID del *Livestreaming* en las listas en directo de *YouTube* a nivel global permitiendo que cualquier usuario de la red pueda acceder al *Live streaming*, en el Privado y el Oculto se necesita obligatoriamente el ID del evento para acceder al *Live streaming*. En las pruebas realizadas, utilizamos un evento oculto dado que nos proporciona un menor nivel de encriptación manteniendo reservado el *Live streaming*.

La Fig.2 muestra el software *Wirecast* con la configuración seleccionada para el flujo de vídeo de bajada, igual al de subida (1920x1080p, H.264, 3 Mbps mínimo). Dicho evento se enlaza a la captura en directo. Posteriormente, *YouTube* transcodifica el contenido, recibido por el enlace de subida a una tasa de bits determinada, creando un flujo principal con el régimen binario establecido al seleccionar el codificador en la configuración del evento. No obstante, *YouTube* hace varias réplicas del flujo con diferentes regímenes binarios para que la emisión del *Live video streaming* esté disponible para todo tipo de usuarios, regulados por la capacidad de sus terminales y las condiciones de la red que utilicen para el acceso al contenido multimedia del *Live Streaming* [8].

B. Recepción de vídeo Live Streaming

El tráfico de *Live video streaming* se genera creando peticiones de visualización al evento público de emisión creado anteriormente. Para ello, se utiliza un *smartphone* con la aplicación *Tems Pocket* ver. 16.3, que se encarga de emular la interacción del usuario con el *Live video streaming*. Dicha aplicación se emplea también para la toma de medidas de los S-KPIs.

El terminal se conecta vía WiFi a Internet. La red inalámbrica está formada por un punto de acceso inalámbrico, configurado en modo puente, para interconectar los dispositivos WiFi a la subred de medición. La conexión entre el punto de acceso inalámbrico y la subred se realiza mediante un cable de par trenzado, conectado desde el punto de acceso hacia una de las tarjetas eth0 del PC que contiene el emulador de red.

C. Modificación de las condiciones de red

Para modificar las condiciones de red, se utiliza el emulador de red *Netem* [9], incluido en el kernel de

Linux desde la versión 2.6. Con él, se pueden introducir efectos controlados sobre la subred, tales como retardo, pérdida, duplicación y reordenamiento de paquetes. En *Netem*, el retardo de paquetes y el *jitter* se describen por el valor medio, la desviación estándar y el coeficiente de correlación. Por defecto, se utiliza una distribución uniforme para el retardo, que puede ser sustituida por otras funciones, tales como Pareto, Pareto-normal, normal o distribuciones personalizadas creados a partir de datos experimentales o de simulación. En la plataforma, se instala *Netem* sobre un PC con un procesador i5-750 a 3 GHz, 8 GB de RAM y sistema operativo Ubuntu 16.04 LTS 64 bits. Dicho PC incluye dos tarjetas de red enlazadas mediante una tabla de enrutamiento, para proporcionar el acceso a Internet a los dispositivos conectados a la subred inalámbrica.

Para recrear las distintas condiciones de red, se configuran los parámetros de *Netem* con el comando "tc" de acuerdo a la siguiente sintaxis [10][11]:

tc qdisc ... dev DEVICE] add netem OPTIONS

OPTIONS := [LIMIT] [DELAY] [LOSS] [RATE]
LIMIT := limit packets
DELAY := delay TIME [JITTER [CORRELATION]]]
LOSS := loss PERCENT
RATE := rate RATE ,

donde *qdisc*(*queuing discipline*) es la abreviatura de la cola asociada al dispositivo de interfaz a través del cual se envían los paquetes. Con respecto a los parámetros OPTIONS, LIMIT acota el efecto de otras opciones seleccionadas al número indicado de paquetes siguientes, DELAY añade el retardo medio elegido en milisegundos a los paquetes que salen a la interfaz de red elegida, JITTER se utiliza para cuantificar la variación de retardo en milisegundos, CORRELATION es un porcentaje que controla cuánto depende el valor de retardo actual del anterior, LOSS es la probabilidad de pérdida de paquetes (expresada en porcentaje) y RATE limita la tasa binaria de transmisión mediante un proceso de *throttling* (en kbps) [12]. En las pruebas realizadas, sólo se ajustan los parámetros *delay, loss* y *throttling* por simplicidad.

D. Recopilación de medidas

Tal como se refleja en la Fig.1, en la plataforma se habilitan dos puntos principales de medida: terminal y salida de Internet. El primero se dedica a la recopilación de estadísticas de rendimiento de la transmisión de paquetes a nivel de red (métricas TCP/IP), mientras que el segundo se dedica las medidas de los S-KPIs.

Medidas de nivel de red

Para la obtención de las métricas TCP/IP, se utiliza la herramienta *Network Probe*, que es un software propietario destinado al análisis y monitorización del tráfico. Este proceso se realiza mediante el análisis en tiempo diferido de archivos ".pcap" con las trazas de tráfico capturadas en la red. Dichos archivos de trazas se generan con la herramienta de código abierto *Tcpdump* [13].

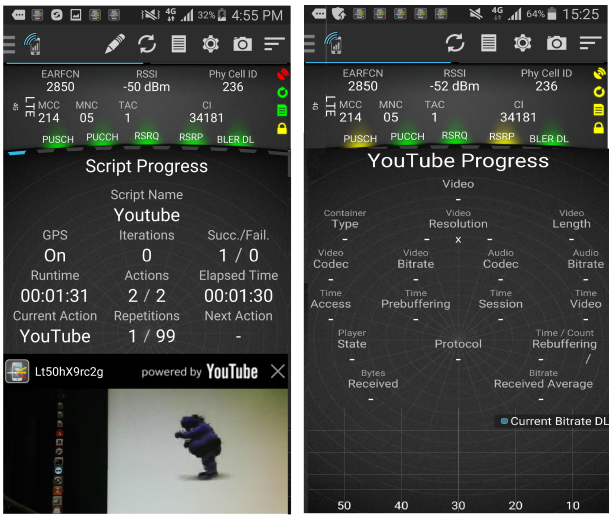

Fig. 3. Pantalla *Tems Pocket* mientras se ejecuta el *YouTube script.*

A partir de los ficheros de trazas, la herramienta *Network Probe* genera archivos de medidas en formato *.csv*, que incluyen las principales métricas de la transmisión de paquetes, desglosadas por usuario, conexión y ráfaga de paquetes.

Medidas de los S-KPIs

Para la toma de medidas de los S-KPIs, se utiliza el software *TEMS Pocket* versión 16.3. Dicha herramienta es un agente de usuario utilizado para la verificación, el mantenimiento y la resolución de problemas de redes móviles [14]. *TEMS Pocket* permite la creación de diferentes programas para automatizar pruebas de servicios tales como Facebook, Instagram, Twitter, WhatsApp, YouTube, etc. En el caso de la evaluación del servicio de *YouTube Live Streaming*, se utiliza la opción de *YouTube* convenientemente configurada para obtener un informe con las estadísticas de los diferentes S-KPIs. La Fig.3 muestra dos pantallas disponibles en el terminal cuando se efectúa una medición. En ella, se observa el tiempo de progreso del *script*, una pequeña pantalla que muestra el *Live video* a evaluar, los valores de los S-KPIs para cada segmento de *streaming* en curso.

E. Automatización

El proceso de automatización debe incluir la configuración automática del agente de usuario, el emulador de red y la captura de tráfico a nivel de paquetes. Dichos procesos deben realizarse de forma sincronizada para permitir posteriormente el procesado de las medidas.

Agente de usuario

La Fig.4 presenta las distintas pantallas de configuración del *TEMS Pocket*. La primera de ellas (*Script settings*) se utiliza para la creación del *script* encargado de descargar y reproducir el flujo multimedia de la sesión de *Live video streaming*. Sus principales opciones son: a) Video indica el identificador (ID) del canal de vídeo streaming a reproducir y analizar; b) *Streaming duration* (SD) indica la

duración de un periodo de medición, es decir, por cuánto tiempo se van a recoger estadísticas de los S-KPIs; c) *Preguard* (PG), *Postguard* (PTD) indican periodos de guarda que se insertan automáticamente antes y después de la medición, respectivamente; el propósito de los periodos de guarda es asegurar que el establecimiento y la liberación de la sesión de vídeo se grabe en el archivo de registro. Para *YouTube*, el valor recomendado para ambos es de 10 seg [14]; d) *Repeat action* (RA) indica el número total de veces que se ejecuta todas las opciones del *script*.

La segunda pantalla (*Actions*) indica las acciones que realiza un *script* determinado. En este trabajo, se seleccionan dos acciones: *YouTube script*, descrito anteriormente, y *Log file recordings*, que permite grabar en un solo archivo las medidas realizadas separadas por cada lazo de repetición, para su posterior post-procesamiento.

La tercera pantalla 3 (*YouTube settings*) es el *script* general conformado por *script setting* y *actions*. Además, posee la opción de establecer la cantidad total de veces que se ejecuta el *script* (*Max interation*s, MI). El total de mediciones obtenidas depende de las veces que se repita el *script setting* (RA) multiplicadas por las veces de ejecución del *script* general (MI).

$$E = MI \cdot RA \cdot (SD + PG + PTD) \qquad (1)$$

Emulador de red

La Fig.5 muestra el *script* utilizado para la configuración automática de *NetEm*. En el *script*, se definen como parámetros de entrada el retardo promedio y el *jitter* en milisegundos ($2, $3), la tasa de pérdida de paquetes expresada en % ($4) y la tasa de transmisión máxima del enlace descendente en kbps (*throughput*, $5). El comentario de la línea inicial indica el tipo de *shell* utilizado para interpretar el *script* (/bin/bash). La línea 3 se emplea para eliminar cualquier regla establecida anteriormente, ejecutando el comando *tc* con la opción "del". Posteriormente, la línea 6 establece los valores de retardo, *jitter* y tasa de pérdidas a los designados en los parámetros de entrada $2 - $4 para el próximo millón de paquetes. Las líneas 9-11 posteriores fuerzan la máxima tasa de transmisión (*throttling*) al valor del parámetro $5. Adicionalmente, el *script* contiene algunas líneas cuyo propósito es facilitar la monitorización, reflejando por pantalla cada cambio de las condiciones de red (Líneas 5 y 8) y almacenando en un fichero de salida la configuración de *NetEm* correspondiente junto a su fecha y tiempo de inicio (*timestamp*) (Línea 13). Conviene aclarar que la interfaz controlada por *NetEm*, denominada eth0, corresponde en la Fig.1 a la interfaz del emulador de red hacia el punto de acceso inalámbrico.

La Fig.6 muestra el *script* empleado para automatizar los cambios en la configuración de *NetEm* mientras se realiza la medición con el terminal móvil. Básicamente, consta de una matriz que contiene todas las combinaciones de los valores deseados para cada parámetro de entrada (THROUGHPUT, PACKET LOSS, DELAY). Para reducir el número de configuraciones simuladas, el parámetro

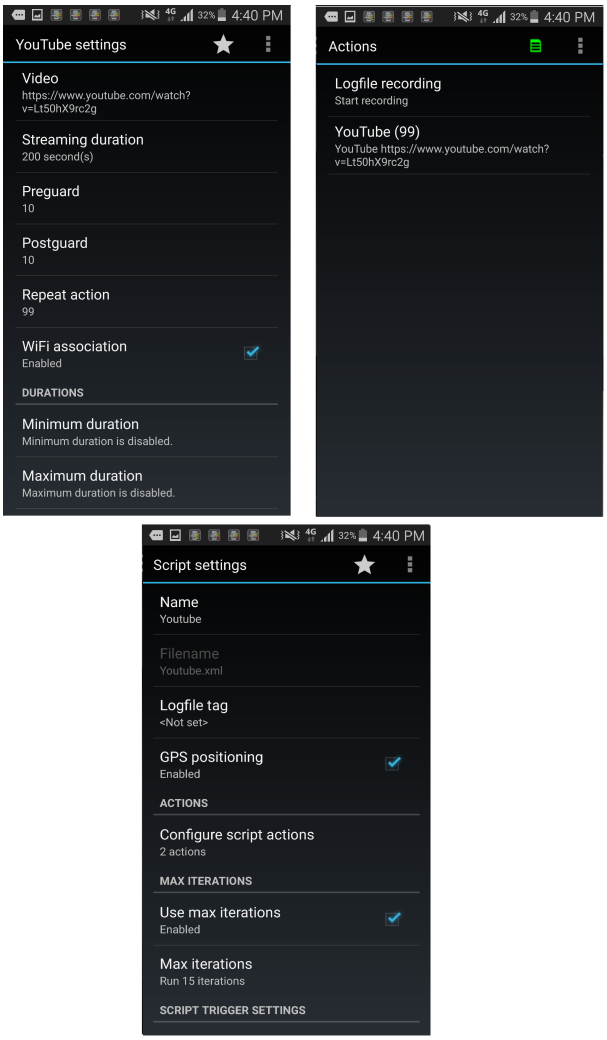

Fig. 4. Configuración del *Pocket script*.

JITTER se fija a 0. Así, cada efecto producido por el *script* de configuración de *Netem*, mostrado en la Fig.1, corresponde a una combinación de los tres parámetros indicados en la matriz (Líneas 7-12). A continuación, se establece el tiempo duración de cada efecto (1980 segundos, 33 minutos) y se realiza la captura de tráfico por este intervalo de tiempo usando *tcpdump* (línea 14). Con la sintaxis utilizada, los archivos de captura ".pcap" generados por cada efecto se almacenan con un nombre de acuerdo al siguiente formato: "Nombre dado al *script* (ej., 1.Csv)" _ "Combinación del efecto (DELAY, PACKET_LOSS, THROUGHPUT) "_ "%F (Fecha)"_ "%T (Hora)". La interfaz de captura, denominada eth1, corresponde a la interfaz entre el emulador de red e Internet. La opción **-G** se usa para indicar el tiempo de duración del proceso de captura, **-W** limitar el número de archivos creados. Al inicio y al final del *script* de automatización, se eliminan las reglas establecidas para garantizar que cada medida obtenida corresponde a un efecto deseado (líneas 3 y 20).

III. METODOLOGÍA EXPERIMENTAL

Esta sección se describe el proceso llevado a cabo para capturar las trazas de datos necesarias para estimar los S-KPIs de un servicio encriptado de *Live video streaming* de *YouTube Live* a partir de las métricas TCP/IP. En primer lugar, se describen los S-KPI más relevantes para este servicio. En segundo lugar, se explica cómo se llevó a cabo la batería de pruebas.

A. Definición de los S-KPI

La experiencia del usuario en *YouTube* (convencional y *Live streaming*) se caracteriza por tres problemas básicos: el retardo inicial de reproducción del video (*Initial buffer time*), la detección de la reproducción del video (*rebuffering /stalling event*) y la duración de este estancamiento [15][16]. Para analizar estos problemas se seleccionan tres S-KPI que nos ofrece la herramienta *TEMS Pocket*: *Streaming Video Play Start Time* (*Initial Buffering Time*), *Streaming Video Interruption Duration* y *Streaming Video Interruption Count* [17].

- Retardo inicial de reproducción (*Streaming Video Play Start Time* , SPT): Es el tiempo desde que el usuario envía la petición para iniciar el *streaming* (click en el ID) hasta que aparece la primera imagen de vídeo en la pantalla.
- Frecuencia de interrupción de reproducción (*Streaming* Video Interruption Frequency, IF): Es la frecuencia con la que se interrumpe la reproducción de vídeo en una sesión de *streaming* por razones de *rebuffering*, calculada a partir del número total de interrupciones en la sesión (*Interruption Count*, IC) y la duración del vídeo reproducido, SD (en minutos), como

$$IF = IC/SD \ [1/min]. \tag{2}$$

- Ratio de duración de interrupción de reproducción (*Streaming Video Interruption Duration Ratio*, IDR): Se define como el cociente entre el tiempo total de interrupciones en una sesión de *streaming* por razones de *rebuffering* (*Interruption Duration*, ID) respecto al tiempo total de la sesión, calculado a partir de la duración del vídeo reproducido (*Streaming Duration*, SD), como

$$IDR = ID/(ID + SD). \tag{3}$$

B. Batería de pruebas

Para identificar qué métricas impactan sobre el rendimiento de un servicio *Live video streaming* de *YouTube*, se realizaron baterías de pruebas durante el periodo comprendido del 7 al 15 de abril del 2017. Estas fechas coinciden con una época de vacaciones y se eligieron para tener las mayores prestaciones de red disponibles.

La batería de pruebas consiste en la emisión de un *live streaming* con una resolución de 1080p (la máxima permitida por la aplicación) desde el servidor local, que tiene instalado el software *Wirecast* y usa la plataforma de *YouTube* como pasarela para la masificación del *Live video*. La configuración exacta se detalla en la sección 2 . El servidor está conectado a Internet por un enlace de 94 Mbps. Eso garantiza un ancho de banda suficiente para transmitir sin interrupciones el *video live streaming*.

```
#!/bin/bash

sudo tc qdisc del dev eth0 root

echo "Changing parameters: Delay=$2ms +- $3ms, loss = $4%"
sudo tc qdisc change dev eth0 root netem delay "$2"ms "$3"ms
 loss "$4"% limit 1000000

echo Setting throughput limit to "$5"Kbps
sudo tc qdisc add dev eth0 handle 1: root htb default 11
sudo tc class add dev eth0 parent 1: classid 1:1 htb rate 1000Mbps
sudo tc class add dev eth0 parent 1:1 classid 1:11 htb rate "$5"Kbit

echo `date --utc +%s`",$2,$3,$4,$5" >> $1.csv
```

Fig. 5. *Script* de configuración de *Netem*.

```
#!/bin/bash

sudo tc qdisc del dev eth0 root

echo "TIMESTAMP,DELAY,VARIABILITY,LOSS,THROUGHPUT" > $1.csv

for THROUGHPUT in 250 500 1000 2000 4000
  do
  for PACKET_LOSS in  0 0.75 1.5 3
      do
      for DELAY in 0 50 100 200 400
        do
            ./netem_and_log_alldl.sh $1 $DELAY 0 $PACKET_LOSS $THROUGHPUT
           sudo tcpdump -w "$1"_"$DELAY"_"$PACKET_LOSS"_"$THROUGHPUT"_%F
                        _%T.pcap -i eth1 -G 1980 -W 1 'not port 22'

      done
  done
done

#Back to normal
sudo tc qdisc del dev eth0 root
```

Fig. 6. *Script* de automatización de *Netem*.

Una vez lanzado el *Live video streaming*, se ejecuta de forma sincronizada el emulador de red y el cliente. La configuración de red se modifica de la siguiente manera:

- Tasa de pérdida de paquetes [%]: 0, 0.75, 1.5, 3.
- Retardo medio de paquete [ms]: 0, 50, 100, 200, 400.
- Máximo ancho de banda disponible (*throughput*) [kbps]: 250, 500, 1000, 2000, 4000.

Dada la matriz anterior, se obtiene un total de 100 configuraciones de *Netem* (efectos). Cada configuración se mantiene durante 33 min (1980 s). El tiempo total de la batería de pruebas es de 55 horas (3.300 min). Durante todo este tiempo, se capturan los datos agrupados en dos clases de métricas:

- Reporte del *Pocket*: Es el archivo generado por el terminal móvil que contiene los S-KPIs medidos y organizados por efecto de *Netem* generado.
- Datos recogidos en el *Network Probe*: Es el archivo que contiene métricas TCP/IP para cada una de las configuraciones de red. Este archivo es generado por el procesado *offline* de los todo los ".pcaps" capturados en la interfaz eth1. El intervalo de tiempo de captura de los ".pcaps" se determina por el *script* de automatización de *Netem*.

IV. RESULTADOS

A continuación, se desglosan los resultados del análisis de regresión entre las medidas de *throughput* medio (THRU) de sesión obtenidas con la sonda de red (*Network Probe*) y las medidas de los distintos S-KPIs

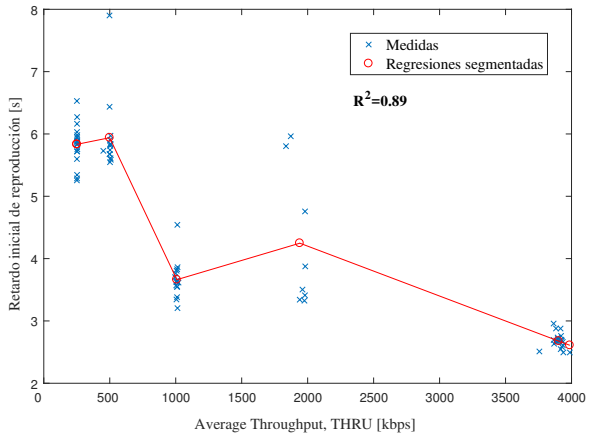

Fig. 7. Gráfica regresión SPT frente tasa de datos disponible.

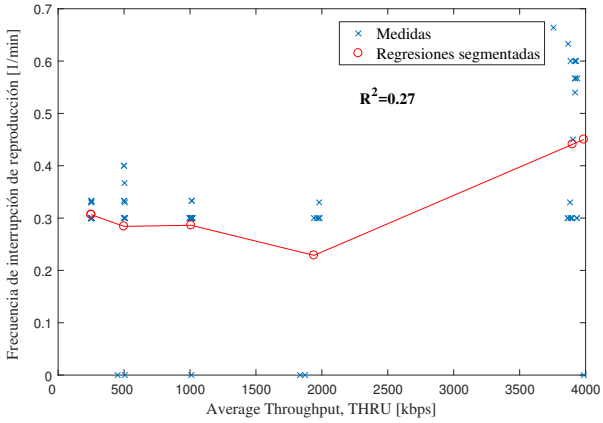

Fig. 8. Gráfica de regresión IF frente tasa de datos disponible.

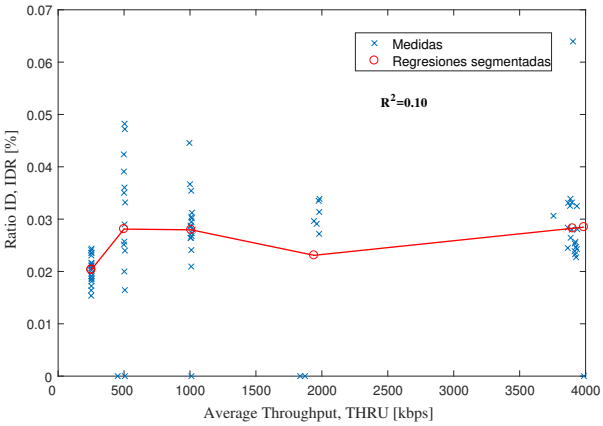

Fig. 9. Gráfica de regresión de IDR frente tasa de datos disponible.

recogidas por el agente de usuario (*Tems Pocket*). Las curvas de regresión que aquí se presentan son específicas del servicio de *Live Streaming*, del que no se han publicado resultados similares.

A. *Retardo inicial de reproducción*

En la Fig.7 se muestra un grafo de dispersión que relaciona el *throughput* medio con el retardo inicial de reproducción (*Streaming Video Play Start Time*, SPT). Al mismo tiempo, se superpone la curva de regresión segmentada que mejor ajusta los datos. En la nube de puntos, puede observarse cómo los puntos se agrupan por columnas, debido a la limitación de la velocidad disponible del canal establecido con el emulador de red (250, 500, 1000, 2000, 4000 kbps). La curva de regresión muestra cómo, en general, el SPT tiende a aumentar cuando se reduce el *throughput* medio. El principal incremento se produce al reducir el THRU por debajo de 1 Mbps. Aun así, llama la atención de que el SPT no se incremente de forma gradual, e incluso se decremente cuando pase el THRU de 2 a 1 Mbps. También llama la atención la gran dispersión de valores de SPT con THRU = 2 Mbps. Este resultado anómalo puede justificarse porque el *Tems Pocket* seleccione una calidad (resolución) del *Live video* al inicio de cada sesión de medida (efecto) en función del ancho de banda disponible para el usuario, lo que disminuiría el impacto de la reducción del THRU producida con el emulador de red. En cualquier caso, el modelo de regresión segmentado construido con las medidas de THRU de la sonda se ajusta razonablemente bien a las medidas del SPT realizadas con el TEMS, con un coeficiente de determinación de $R^2 = 0.89$.

B. *Frecuencia de interrupción de reproducción*

En la Fig.8 se muestra el grafo de dispersión entre el *throughput* medio y la frecuencia de interrupción (*Streaming Video Interruption Frequency*, IF). Este S-KPI contabiliza el número de veces que la reproducción del video se detiene. Siguiendo el mismo procedimiento anterior, se construye la curva de regresión a partir de la nube de puntos. A primera vista, se aprecia que, inesperadamente, cuanto mayor es el THRU, mayor es la frecuencia de interrupción. Este comportamiento anómalo se justifica por el mecanismo de adaptación de la resolución de vídeo introducido por DASH en *YouTube*. Cuando las condiciones de red son buenas, el agente de usuario selecciona una resolución de vídeo mejor, que conlleva una mayor tasa de transmisión. Con esta selección, si existe algún problema de congestión en la red, el *buffer* de reproducción se consume rápidamente, debido a la elevada tasa de codificación de vídeo, antes de que se descarguen los siguientes segmentos. Eso explica el por qué, aunque la capacidad del canal sea mayor, el número de interrupciones es mayor que en el caso de un ancho de banda más pequeño. Con peores condiciones de canal, los datos almacenados en el *buffer* se consumen más lentamente, disminuyendo la probabilidad de que el *buffer* de reproducción se vacíe. En cualquier caso, los valores de IF son bajos, inferiores a 0.7 interrupciones por minuto, lo que demuestra la capacidad de DASH para eliminar los problemas de interrupciones. Como se aprecia en la Fig.8, el modelo de regresión construido recoge la relación directa entre IF y THRU, aunque el coeficiente de correlación entre las estimas y las medidas de IF es sólo de $R^2 = 0.27$.

C. Ratio de duración de interrupción de reproducción

Por último, en la Fig.9 se muestra el grafo de dispersión que relaciona el *throughput* medio con el ratio de duración de interrupción (*Streaming Video Interruption Duration Ratio*, IDR). Este indicador refleja la duración total de las interrupciones debidas a que el *buffer* del reproductor se vacía. El análisis de regresión muestra una relación inversa entre las variables. Se observa cómo la tendencia es que a mayor THRU, menor es el tiempo de duración de las interrupciones. Este resultado, que de nuevo parece contraintuitivo, tiene también su explicación en los mecanismos de adaptación de la resolución del vídeo introducidos por DASH. Cuando empeoran las condiciones de canal, como resultado del aumento del retardo de transmisión, las pérdidas de paquetes o el *throttling*, el agente de usuario selecciona una resolución menor con un régimen binario que garantice la recepción de paquetes. Con ello, se reducen significativamente la duración de las interrupciones. Por el contrario, cuando el THRU es más alto, se elige una resolución de vídeo mayor, lo que aumenta la sensibilidad del reproductor a cambios bruscos de la tasa de transmisión. En cualquier caso, los valores de IDR son bajos, inferiores a 0.07 % del tiempo total de la sesión. Un análisis exhaustivo de las medidas (no mostrado aquí) refleja que la duración máxima de la interrupción es de 160 ms (equivalente a sólo 4 fotogramas con una frecuencia de cuadro de 25 Hz). El modelo de regresión construido recoge la relación directa entre IDR y THRU, aunque el coeficiente de correlación entre las estimas y las medidas de IF es sólo de $R^2 = 0.10$.

V. CONCLUSIONES

En este artículo se ha presentado un modelo de regresión para estimar los principales indicadores de rendimiento del servicio de *Live streaming* cifrado de *YouTube* en una red inalámbrica. El modelo se ha construido utilizando una plataforma experimental, compuesta por una estación de trabajo emisora, un terminal móvil reproductor, un emulador de nivel de red y una sonda de nivel de red. Con esta plataforma, se han capturado trazas de nivel de red y de usuario tras una batería de pruebas con más de 3000 minutos de reproducción de vídeo y 100 configuraciones diferentes del emulador de red. El análisis de los datos recogidos muestra la correlación existente entre los indicadores de rendimiento S-KPIs seleccionados y las métricas TCP/IP de la red. A partir de las curvas de regresión mostradas, pueden construirse modelos de QoE para el servicio de *Live Streaming* basados únicamente en métricas TCP/IP. Estos modelos de caja negra son la única opción de los operadores de red para monitorizar la QoE de servicios multimedia cifrados a gran escala.

Los resultados demuestran de manera clara el efecto beneficioso del DASH sobre el rendimiento del *Live streaming*, gracias a la disminución de las interrupciones. Estos resultados son coherentes con las estadísticas de QoE del servicio de *YouTube* convencional con DASH sobre móviles presentadas en [6]. De los resultados, también se deduce que se hacen necesarios nuevos indicadores de rendimiento del servicio que estimen el nivel medio de calidad de imagen ofrecida al usuario durante la sesión. En este sentido, conviene precisar que el agente de usuario utilizado en la campaña de medidas no recoge actualmente estadísticas del formato de vídeo reproducido. Actualmente, se trabaja en el descifrado y análisis del tráfico de aplicación para obtener dicha información.

AGRADECIMIENTOS

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad (Proyecto TEC2015-69982-R, UNMA13-1E-1864), y FEDER.

REFERENCIAS

[1] A. Ericsson, "Ericsson mobility report: On the pulse of the networked society", *Ericsson, Sweden, Tech. Rep. EAB-14*, vol. 61078, 2015.

[2] N. Alliance, "Next generation mobile networks recommendation on son and o&m requirements", *Req. Spec. v1*, vol. 23, 2008.

[3] I. Cisco, "Cisco visual networking index: Forecast and methodology, 2011–2016", *CISCO White paper*, pp. 2011–2016, 2012.

[4] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Analysis and modelling of youtube traffic", *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 4, pp. 360–377, 2012.

[5] A. Diaz, P. Merino, and F. J. Rivas, "Customer-centric measurements on mobile phones", in : IEEE *Int. Symp. on Consumer Electronics*, 2008, pp. 1–4.

[6] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hossfeld, "Modeling the youtube stack: From packets to quality of experience", *Computer Networks*, vol. 109, 2016.

[7] Telestream, "Wirecast user manual", 2017. [Online]. Available: www.telestream.net/application-content/wirecast/help/7-3/win/Wirecast-User-Guide-Windows.pdf

[8] Y. Corporation, "Creator studio", 2017. [Online]. Available: https://support.google.com/youtube/answer/6060318

[9] T. L. Foundation, "Netem", 2017. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem

[10] S. Salsano, F. Ludovici, A. Ordine, and D. Giannuzzi, "Definition of a general and intuitive loss model for packet networks and its implementation in the netem module in the linux kernel", *University of Rome*, vol. 3, 2012.

[11] U. M. Repository, "Ubuntu 16.10", 2017. [Online]. Available: http://manpages.ubuntu.com/manpages/xenial/en/man8/tc-netem.8.html

[12] S. Hemminger *et al.*, "Network emulation with netem", in: *Linux conf au*, 2005, pp. 18–23.

[13] Tcpdump-workers, "Tcpdump", 2017. [Online]. Available: http://www.tcpdump.org/

[14] Ascom, "Tems pocket specifications", 2017. [Online]. Available: http://www.tems.com/products-for-radio-and-core-networks/radio-network-engineering/portable-testing-for-wireless-networks

[15] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube QoE via crowdsourcing", in: IEEE *Multimedia Int. Symp. on Multimedia Quality of Experience*, 2011, pp. 494–499.

[16] P. Casas, A. Sackl, S. Egger, and R. Schatz, "Youtube & Facebook quality of experience in mobile broadband networks", in: IEEE *Globecom Workshops* (GC Wkshps), 2012, pp. 1269–1274.

[17] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the Quality of Experience of HTTP Video Streaming", in: IFIP/IEEE *International Symposium on Integrated Network Management* (IM), 2011, pp. 485–492.