Nuevas heurísticas para la detección de nodos y flujos eDonkey

Rafael A. Rodríguez-Gómez, Gabriel Maciá-Fernández, Pedro García-Teodoro
Departamento de Teoría de la Señal, Telemática y Comunicaciones,
CITIC - Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación, Universidad de Granada
C/Periodista Daniel Saucedo Aranda s/n E-18071
rodgom@ugr.es, gmacia@ugr.es, pgteodor@ugr.es

***Resumen*—El uso de aplicaciones basadas en redes peer-to-peer (P2P) ha experimentado un incremento exponencial, lo que ha provocado que el volumen de tráfico generado por éstas llegue a suponer alrededor de un 80% de todo el ancho de banda de la red. Por este motivo, el interés de los proveedores de servicio de Internet (ISPs) por clasificar este tráfico ha aumentado también de forma considerable.**

En este contexto, en el presente artículo se describen dos algoritmos de detección del protocolo eDonkey. El primero de ellos tiene como objeto de detección los flujos del protocolo y está basado en la hipótesis de que son flujos del protocolo eDonkey aquellos en los que el cliente que inicia la conexión envía sustancialmente más información que la que recibe. El segundo algoritmo ha sido desarrollado para detectar nodos generadores de tráfico eDonkey, basándose en la hipótesis de que son nodos generadores de tráfico eDonkey aquellos cuya tasa de subida es constante conectándose a múltiples IPs.

Ambos algoritmos de detección han sido probados en tres conjuntos de trazas. Como resultado, se ha comprobado que las hipótesis utilizadas en los algoritmos de detección son ciertas en el protocolo eDonkey. Adicionalemente, los experimentos muestran que los algoritmos propuestos tienen una elevada tasa de reconocimiento y una baja tasa de falsos positivos.

***Palabras Clave*—Clasificación de tráfico, detección de flujos, detección de nodos, P2P, eDonkey**

I. Introducción

El desarrollo y el uso de aplicaciones que utilizan redes P2P para comunicarse han experimentado un crecimiento exponencial en los últimos años. Encontramos en la actualidad múltiples ejemplos de éstas como pueden ser: eMule o uTorrent, como aplicaciones de compartición de ficheros, Skype, como aplicación de voz sobre IP, y Spotify, para la compartición de flujos de audio.

El tráfico generado por las aplicaciones P2P supone un enorme consumo del ancho de banda de la red; de hecho, Callado et al. [1] aseguran que el volumen de tráfico generado por aplicaciones P2P supone un 80% de todo el ancho de banda de red. Este consumo de ancho de banda conlleva la aparición de retardos en las comunicaciones y, en suma, una disminución en la calidad de los servicios proporcionados.

La capacidad de clasificar el tráfico P2P es una tarea de altísimo valor para los proveedores de servicio de Internet (IPSs), que ven perdido el control de su tráfico y que se ven forzados a incrementar las operaciones de mantenimiento debido a este crecimiento en el uso de las redes P2P [2]. La disponibilidad de mecanismos eficientes y eficaces de clasificación de tráfico permitirá a los ISPs definir políticas de gestión de tráfico, ofreciendo diferentes clases de servicio y aplicando a cada una de ellas una conformación del tráfico que dependa, por ejemplo, de las aplicaciones utilizadas por el usuario.

Los métodos de clasificación de tráfico pueden dividirse actualmente en tres grupos: *(i)* basados en el puerto, *(ii)* basados en el contenido de los paquetes, y *(iii)* basados en las características estadísticas de los flujos. Las aplicaciones P2P pueden utilizar cualquier puerto para comunicarse y encriptar el contenido de sus mensajes, lo que dificulta enormemente su clasificación mediante los dos primeros tipos de técnicas.

El presente trabajo se centrará, pues, en la detección basada en las características de los flujos. El protocolo a detectar es eDonkey, protocolo de comunicación para redes P2P que sigue representando, hoy por hoy, un porcentaje considerable del volumen de tráfico de Internet, y que es utilizado principalmente en aplicaciones de compartición de ficheros como eMule o aMule.

Se proponen aquí dos heurísticas para la detección de tráfico eDonkey, que son el resultado de un estudio detallado del comportamiento de este protocolo: detección de nodos basada en la tasa de subida y detección de flujos basada en la inversión del sentido de la descarga.

La detección de nodos basada en la tasa de subida se fundamenta principalmente en dos asunciones: (*i*) los usuarios limitan la tasa de subida de las aplicaciones que utilizan eDonkey, y (*ii*) la tasa de subida tiene un comportamiento constante en ciertos periodos temporales alrededor de este límite establecido por el usuario. De esta forma, si la tasa de subida de un nodo tiene un comportamiento constante en diferentes periodos temporales alrededor de un mismo valor, este nodo será detectado como nodo generador de tráfico eDonkey.

La detección de flujos propuesta se basa en la particularidad de que, en las aplicaciones de compartición de archivos que utilizan el protocolo eDonkey para comunicarse, el nodo que inicia la conexión es el nodo que envía el grueso de la información. Esto es radicalmente opuesto a lo que sucede en el paradigma de cliente-servidor, donde es el cliente quien inicia la conexión y el servidor quien envía la información requerida. Por tanto, si el sentido del envío de la información en un flujo es el contrario al común, es decir, la información viaja del nodo que inicia la conexión al que la recibe, este flujo será detectado como perteneciente al protocolo eDonkey.

El resto del artículo se divide como sigue: En la Sección II se presentan los trabajos relacionados con la clasificación de tráfico P2P, indicando la aportación aquí pretendida frente a ellos. En la Sección III se presentan conceptos generales de las redes P2P, junto a una descripción del funcionamiento

ISBN: 978-84-694-5948-5

general del protocolo eDonkey. Las heurísticas propuestas para la detección del protocolo eDonkey se detallan en la Sección IV, mostrándose en la Sección V el entorno de experimentación considerado. Por su parte, en la Sección VI se presentan y discuten los resultados experimentales obtenidos con las propuestas realizadas. Finalmente, en la Sección VII se exponen las principales conclusiones que se pueden extraer de este trabajo.

II. Trabajos relacionados

Los métodos de clasificación de tráfico existentes en la literatura pueden dividirse en tres: basados en el puerto, basados en el contenido de los paquetes y basados en las características de los flujos. Como aseguran los autores de [3], los métodos de clasificación de tráfico basados en puertos conocidos no resultan válidos actualmente en la detección de tráfico P2P y en cuanto a los basados en el contenido de los paquetes, su uso incurre en problemas legales relativos a la privacidad, lo que reduce su ámbito de aplicación enormemente.

Existe una gran cantidad de trabajos en los que se propone una clasificación basada en las características de los flujos. Por ejemplo, el trabajo de Moore y Zuev [4] utiliza análisis bayesiano para este fin. Las características utilizadas para esta clasificación son: puertos TCP, duración de los flujos, estadísticas del tiempo entre paquetes, estadísticas del tamaño de los paquetes y la transformada de Fourier del tiempo entre paquetes. Una particularidad de este trabajo es que solamente se utilizan flujos TCP completos, es decir, que contienen el establecimiento y cierre de conexión propios de este protocolo. En esta línea se puede destacar también BLINC [5], una herramienta de clasificación que asocia un equipo final con la aplicación que genera la mayor parte de su tráfico. Es decir, asocia los equipos finales con los servicios que ofrecen o utilizan, en lugar de analizar cada flujo individualmente. De forma similar a BLINC, una de las dos heurísticas presentadas en este trabajo se basa en la detección de nodos generadores de tráfico P2P.

Otros trabajos, en lugar de utilizar una única metodología para clasificar todos los protocolos existentes, clasifican únicamente un subconjunto de protocolos. Esta es la aproximación más frecuente para clasificar flujos de los protocolos utilizados en redes P2P. En esta línea se encuentra [3], el primer trabajo que intenta clasificar tráfico proveniente de aplicaciones P2P en puertos arbitrarios sin inspeccionar el cuerpo de los paquetes. Para esto se utilizan dos heurísticas: *(i)* En la primera se seleccionan las parejas IP origen y destino que se comunican utilizando tanto TCP como UDP, y *(ii)* la segunda se basa en los patrones de conexión de las parejas IP-puerto de las aplicaciones P2P. Básicamente, cuando un nodo P2P inicia una conexión con un nodo A, el puerto de destino será el puerto definido por el usuario del nodo A para atender las peticiones de la aplicación P2P. Las dos heurísticas de detección propuestas en nuestro trabajo también son capaces de clasificar el tráfico P2P encriptado en puertos arbitrarios.

Xu et al. [6] proponen un método para identificar tráfico P2P basándose en el comportamiento de la transferencia de datos de las aplicaciones P2P. Los autores aseguran que los datos descargados por un nodo de la red en las aplicaciones P2P serán subidos a otro nodo de la red con posterioridad. Así, dividen los datos descargados y subidos por los nodos en bloques de datos e intentan detectar aquellos flujos que comparten bloques de datos; éstos serán identificados como flujos P2P. De esta misma forma, las heurísticas propuestas en el presente artículo también se basan en el comportamiento de la transferencia de datos de las aplicaciones P2P, aunque, como se verá en la Sección IV, presentan grandes diferencias con respecto al trabajo de Xu et al.

Por último, concretando aún más, existen aportaciones destinadas a la clasificación de un único protocolo. Como ejemplo de detección de tráfico Skype se puede citar [7], en el que los autores proponen un marco basado en dos técnicas complementarias para detectar en tiempo real el tráfico perteneciente a dicho protocolo. La primera aproximación se basa en el test Chi-Cuadrado de Pearson y en las características del tráfico de VoIP para detectar, en la estructura de los paquetes, huellas fundamentales del tráfico de Skype, valiéndose de la aleatoriedad introducida a nivel de bit en el proceso de encriptación. La segunda se basa en la caracterización estadística del tráfico de Skype en términos de tasa de llegada y tamaño de paquetes. El presente trabajo se basa también en la detección de un protocolo concreto, en este caso, en la detección del protocolo eDonkey, cuyas características principales se describen en la siguiente sección.

III. Conceptos generales del protocolo eDonkey

Las redes P2P se pueden dividir en tres tipos, según su arquitectura: centralizadas, distribuidas e híbridas. En las redes centralizadas existe un servidor encargado de indexar los recursos de la red, asociando recursos y clientes que los comparten. En el caso de las redes distribuidas no se requiere ninguna gestión centralizada. Los nodos son los encargados de almacenar la distribución de los contenidos en la red. Por último, las redes híbridas son una combinación de las dos anteriores. En éstas los clientes pueden ser clientes y supernodos (o servidores), formando los clientes una red distribuida en torno a los supernodos, los cuales realizan las tareas de los servidores en las redes centralizadas.

En este contexto de redes P2P, el protocolo eDonkey resulta de muy amplio uso. Por ello es el objetivo central de este trabajo. Con el fin de comprender el alcance del estudio aquí realizado, el presente apartado lleva a cabo una breve discusión acerca de las principales características y operativas de eDonkey.

El protocolo eDonkey se diseñó para la comunicación de nodos en una red P2P híbrida formada por servidores y clientes. Por un lado, los servidores dan acceso a la red, se encargan de manejar la distribución de la información en estructuras similares a diccionarios que almacenan la relación entre los recursos y los nodos que los comparten. Por otro lado, los clientes son los únicos nodos que comparten datos, y son los encargados de almacenar los recursos de la red.

A continuación se presenta una breve descripción de las comunicaciones del protocolo eDonkey, de especial interés para el presente trabajo, divididas en comunicaciones cliente-servidor y comunicaciones cliente-cliente.

A. Comunicaciones cliente-servidor

Para poder acceder a los recursos de la red, los clientes deben conectarse a un servidor eDonkey. Esta conexión puede

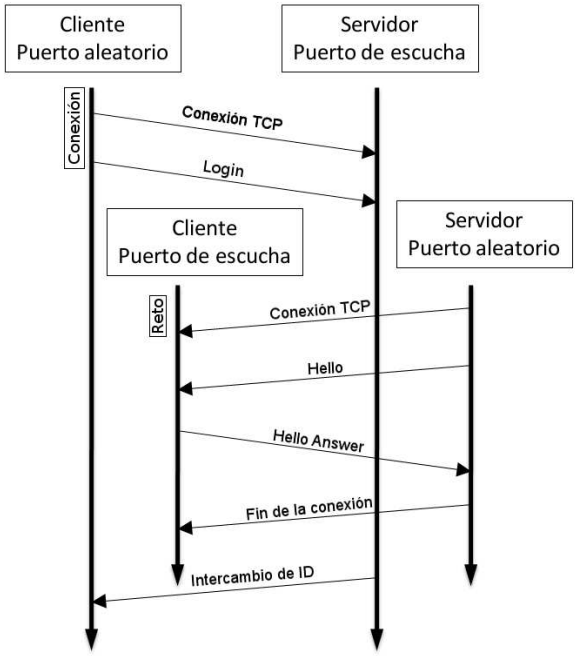

Fig. 1. Proceso de conexión cliente-servidor del protocolo eDonkey.

dividirse en dos fases (Figura 1): *(i)* Solicitud de conexión por parte del cliente, y *(ii)* reto por parte del servidor. En la solicitud de conexión, el cliente inicia una conexión TCP con el servidor y, posteriormente, envía el mensaje de registro del protocolo eDonkey, denominado `login`. El servidor responde al cliente con la segunda parte de la conexión, el reto. Para esto, se realiza una segunda conexión TCP, en este caso iniciada por el servidor y cuyo destino es el puerto definido por el cliente para la recepción de las conexiones de otros clientes de la red eDonkey. Este reto determina si el cliente es capaz o no de recibir conexiones de otros clientes. El proceso de conexión finaliza con el envío, por parte del servidor, de un identificador único. En caso de que el reto haya sido superado con éxito el cliente es identificado con *ID alta*, en otro caso con *ID baja*. En conclusión, una ID baja implica que el cliente no es capaz de aceptar conexiones de otros clientes de la red, y una ID alta que sí lo es.

Después de una conexión exitosa, tanto servidor como cliente intercambian información acerca de su estado actual. El cliente comienza enviando la lista de recursos que comparte, y el servidor envía, entre otra información, su versión y la lista de servidores que conoce.

Una vez que el cliente tiene acceso a un servidor eDonkey puede realizar búsquedas mediante palabras clave, a las que el servidor responderá con una lista de recursos relacionados. Posteriormente, el cliente decide descargar uno o más archivos de la lista obtenida enviando un mensaje al servidor en el que pregunta por el recurso concreto que desee, y al que el servidor responde con una lista de clientes que poseen dicho recurso.

B. Comunicaciones cliente-cliente

Para descargar el recurso o recursos por los que el cliente ha preguntado al servidor, es necesario realizar conexiones cliente-cliente. Estas conexiones se realizan mediante una

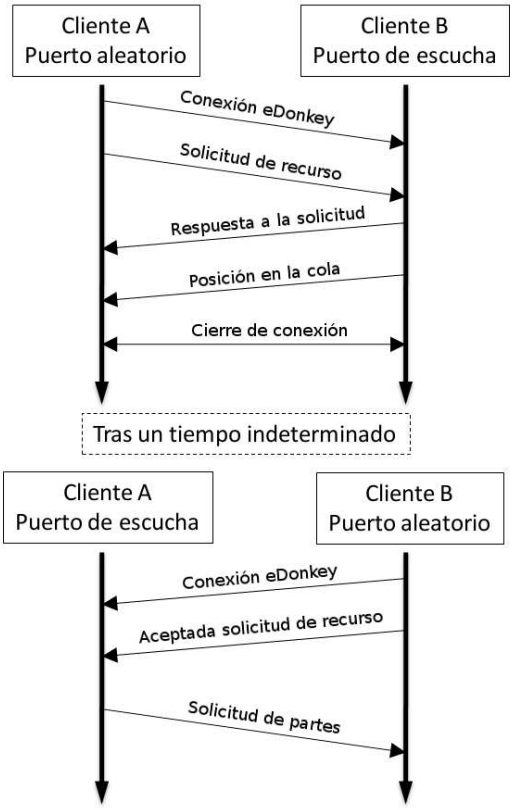

Fig. 2. Entrada en la cola de peticiones de un cliente e inicio de la descarga de un recurso.

conexión TCP inicial, seguida por un intercambio de mensajes del protocolo eDonkey `hello` y `hello answer`. Estos mensajes portan información del nodo que lo envía, principalmente el identificador de usuario y el puerto TCP en el que se reciben conexiones eDonkey.

Existe un caso especial en las conexiones cliente-cliente: aquel en el que el cliente al que se intenta conectar tiene un identificador de ID baja. Por definición, un cliente con ID baja no es capaz de aceptar conexiones, por lo que esta conexión resulta imposible mediante el mecanismo convencional. Para solventar este problema, el protocolo eDonkey dispone de un mensaje llamado `callback`. Aprovechando que el cliente con baja ID tendrá una conexión con el servidor, el cliente origen envía un mensaje `callback` al servidor indicando su intención de conectarse con el cliente destino. El servidor, como respuesta, envía otro mensaje `callback` al cliente destino, a través de la conexión establecida previamente con él (que fue generada por el cliente). Por último, el cliente destino, al recibir el mensaje del servidor, inicia una conexión con el cliente origen. Este mecanismo, sin embargo, no solventa el caso en el que cliente origen y destino posean ID baja, ya que ninguno de los dos es capaz de aceptar conexiones externas. En este caso, la conexión entre ellos no está soportada en el protocolo eDonkey.

Una vez establecida la conexión entre clientes de la red, es posible iniciar el proceso de descarga de un recurso. La situación más común en este proceso se muestra en la Figura 2. El cliente A (cliente origen) envía un mensaje solicitando el recurso buscado y el cliente B (cliente destino) responde a esta solicitud indicando que lo posee. Posteriormente, el

cliente B indica la posición que ocupa la solicitud del cliente A en su cola de peticiones a servir y cierra la conexión. Cuando la petición del cliente A alcanza una posición en la cola que le permite ser servida, el cliente B inicia una conexión con el cliente A y le envía un mensaje indicando que acepta su solicitud. Finalmente, el cliente A solicita las partes concretas del recurso buscado para que el cliente B inicie su envío. Tal y como se detalla en la Sección IV, este comportamiento se utilizará como heurística de detección de patrones de generacón de tráfico eDonkey.

Otra posibilidad, aunque bastante menos frecuente, es que la solicitud del cliente A sea aceptada sin necesidad de cerrar la conexión iniciada por éste. Esto se puede deber a que la cola de servicio del cliente B no esté completamente llena, o a que, por algún motivo, B decida servir las peticiones del cliente A con mayor prioridad.

IV. Heurísticas de detección

Planteada como se ha hecho con anterioridad nuestra intención de desarrollar mecanismos para la detección de comunicaciones P2P, concretamente referidas al protocolo eDonkey, en lo que sigue se presentan a nivel teórico las heurísticas propuestas con tal fin.

Dos son las heurísticas de detección ideadas: detección de nodos basada en la tasa de subida y detección de flujos basada en la inversión del sentido de la descarga. A través de la detección de nodos P2P se persigue determinar qué dispositivos generan en un momento determinado tráfico de este tipo, lo cual puede resultar de interés para los ISPs. Sustentada en esta primera detección, la determinación de qué flujos concretos son de tipo P2P puede resultar de interés de cara, por ejemplo, a la caracterización de los mismos orientada, por ejemplo, a un filtrado más específico.

A. Detección de flujos

La primera heurística tiene como objetivo la detección de flujos del protocolo eDonkey basándose en la hipótesis de que el cliente que inicia la conexión es el que envía los datos. Esto se debe a que el proceso más común de descarga de un archivo mediante el protocolo eDonkey (Figura 2) está formado por dos conexiones TCP. En la primera, el cliente A indica su interés por un recurso del cliente B entrando en su cola de peticiones a servir, y en la segunda el cliente B inicia la conexión con el cliente A para indicarle que su petición puede ser servida y comenzar a enviarle datos, de modo que el cliente que inicia la conexión (cliente B) es el que envía los datos.

En la mayoría de las aplicaciones cliente-servidor es el servidor el encargado de enviar los datos tras una conexión iniciada por el cliente. Este comportamiento es inverso por tanto al que se ha descrito en el protocolo eDonkey y, por este motivo, los autores proponen la siguiente hipótesis para la detección de flujos eDonkey:

Hipótesis 1. *Son flujos del protocolo eDonkey aquellos en los que el cliente que inicia la conexión envía sustancialmente más información que la que recibe.*

Esta heurística se cumple únicamente en flujos del protocolo eDonkey utilizados para transferir archivos y no en flujos de señalización. Aún así, por ejemplo para la clasificación de tráfico por parte de los ISPs, los flujos de transferencia de archivos son los especialmente relevantes, al ser los que ocupan el mayor porcentaje del ancho de banda de la red.

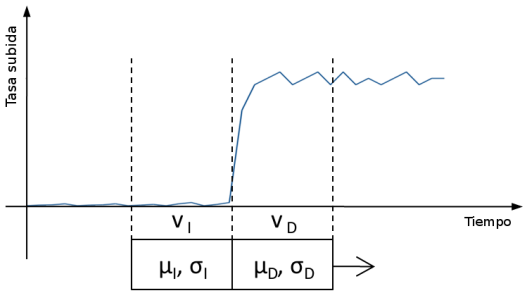

Fig. 3. Avance a lo largo del tiempo del cálculo de la divergencia de Kullback-Leibler.

Basado en la hipótesis de inversión del sentido de la descarga, se ha desarrollado el algoritmo de detección de flujos de transferencia de archivos del protocolo eDonkey mostrado más abajo como Algoritmo 1. Este algoritmo de detección ha sido desarrollado para ejecutarse en capturas de tráfico analizadas no en tiempo real. Se recorren uno a uno los flujos que conforman la captura y se extraen aquellos cuyo número de *bytes* enviados desde el cliente que inicia la conexión al que la recibe es mayor que la información recibida en sentido contrario más un umbral, Umb_B. Este umbral se debe determinar experimentalmente en base a un estudio de la distribución de tamaños de los flujos de transferencia de archivos del protocolo eDonkey. Finalmente, los flujos extraídos son los que el algoritmo detecta como flujos de compartición de archivos del protocolo eDonkey ($flujos\_eD$).

Algoritmo 1 Detección de flujos

1: **para** $i = 0$ mientras $i < num\_flujos$ **hacer**
2: **si** $bytes\_env[i] > bytes\_rec[i] + Umb_B$ **entonces**
3: $flujos\_eD \Leftarrow flujos[i]$
4: **fin si**
5: $i \Leftarrow i + 1$
6: **fin para**
7: **devolver** $flujos\_eD$

La elección del umbral utilizado en este algoritmo de detección, como se verá en la Sección VI, no resulta crítica, ya que la diferencia de tamaños en los flujos de compartición de archivos del protocolo eDonkey es muy elevada.

B. Detección de nodos

Una alternativa distinta, aunque complementaria como se ha apuntado anteriormente, a la detección de los flujos perteneciente al protocolo eDonkey es (como sucede en BLINC [5]) la detección de los nodos generadores de tráfico eDonkey. Conociendo los nodos que generan tráfico de tipo eDonkey, se podrían aplicar políticas de conformación del tráfico proveniente de los nodos que sean detectados.

Esta heurística de detección se basa en la asunción de que los usuarios limitan la tasa de subida de las aplicaciones P2P que utilizan el protocolo eDonkey como eMule o aMule, entre otras. Esto es así porque sin dicha limitación estas

aplicaciones saturarían la capacidad de subida de las conexiones a Internet de los usuarios, provocando una disminución considerable en la velocidad de la navegación Web.

La otra hipótesis en la que se basa la presente heurística es que la limitación de la tasa de subida implica, la mayor parte del tiempo, una tasa de subida constante igual al límite establecido por el usuario. Esta tasa constante representa un comportamiento extremadamente característico a nivel de nodo: un nodo se conecta con multitud de IPs diferentes a lo largo del tiempo, manteniendo un nivel aproximadamente constante de su tasa de subida. En conclusión, la hipótesis de detección queda como:

Hipótesis 2. *Son nodos generadores de tráfico eDonkey aquéllos cuya tasa de subida es constante conectándose a múltiples IPs.*

La cuestión a abordar es qué se debe considerar como un nivel constante de tasa de subida. La respuesta a esta cuestión no es trivial y para abordarla se toma como referencia el trabajo [8], en el que los autores utilizan la divergencia de Kullback-Leibler (KL) para detectar la actividad de voz en señales de audio. Esta detección consiste en determinar el instante temporal en el que la señal de audio evaluada pasa de ser únicamente ruido a contener también voz. Este cambio viene caracterizado por un cambio en la media y la varianza de la señal de audio, que se detecta gracias a la divergencia de KL. La detección de la presente heurística representa el caso contrario: se debe observar ausencia de cambios significativos en la tasa de subida (tasa constante) para determinar que un nodo está generando tráfico eDonkey.

Se puede definir la divergencia de KL como un indicador de la similitud entre dos funciones de distribución, y en el caso de dos distribuciones gaussianas p_I y p_D se representa así:

$$H(p_I||p_D) = \frac{1}{2}[log(\frac{\sigma_D^2}{\sigma_I^2}) - 1 + \frac{\sigma_I^2}{\sigma_D^2} + \frac{(\mu_I - \mu_D)^2}{\sigma_D^2}] \quad (1)$$

donde σ_D y σ_I representan las desviaciones típicas de p_I y p_D, y μ_I y μ_D las medias.

La divergencia de KL no es simétrica, lo que quiere decir que $H(p_I||p_D)$ puede ser diferente de $H(p_D||p_I)$. En la Expresión 2 se muestra la divergencia de KL de dos distribuciones gaussianas p_I y p_D en su forma simétrica:

$$\rho_{I,D} = \frac{1}{2}\left[\frac{\sigma_I^2}{\sigma_D^2} + \frac{\sigma_D^2}{\sigma_I^2} - 2 + (\mu_I - \mu_D)^2(\frac{1}{\sigma_I^2} + \frac{1}{\sigma_D^2})\right] \quad (2)$$

El algoritmo de detección derivado de la heurística propuesta puede ser descrito como sigue (véase Algoritmo 2 más adelante). Primero, los valores de la tasa de subida de un nodo son calculados en intervalos de t segundos. Los valores resultantes son filtrados mediante un filtro de mediana [9] de tamaño N. Este filtro toma N valores de tasa de subida (una ventana de tamaño N) y los ordena de menor a mayor, quedándose con el valor que ocupa el centro del intervalo.

El resultado del filtrado anterior es recorrido por dos ventanas consecutivas (v_I y v_D), cada una de tamaño N (Figura 3). En cada ventana se halla de forma independiente la media y la varianza de los valores en ellas comprendidos, asumiendo que pueden ser modelados por distribuciones gaussianas, y se calcula la divergencia de KL simétrica (Expresión 2) de estas distribuciones.

En el caso de [8] se utilizan los máximos de la divergencia de KL para encontrar los instantes en los que la señal de audio pasa de ser únicamente ruido a contener voz, o viceversa. Esto se debe a que estos valores representan los instantes en los que las distribuciones gaussianas que modelan los valores en v_I y v_D difieren en mayor medida.

La principal aportación del mecanismo de detección propuesto con respecto a [8] radica en que en la presente detección se busca que las distribuciones gaussianas que modelan los valores comprendidos en las ventanas sean lo más similares posible, lo que quiere decir que la divergencia de KL es cercana a cero. En el algoritmo propuesto esto implica que la divergencia de KL no debe superar un umbral determinado experimentalemente (Umb_{KL}) como se verá en la Sección VI.

Algoritmo 2 Detección de nodos

para $nodo = 0$ mientras $nodo < num\_nodos$ **hacer**
  para $i = 0$ mientras $i < len(tasa\_sub_{nodo})$ **hacer**
    $tasa\_filt_{nodo} \Leftarrow filt\_mediana(tasa\_sub_{nodo}, N)$
    $v_I \Leftarrow tasa\_filt_{nodo}[i : i + N]$
    $v_D \Leftarrow tasa\_filt_{nodo}[i + N + 1 : i + 2N + 1]$
    $\rho_{I,D}[i] \Leftarrow \frac{1}{2}\left[\frac{\sigma_I^2}{\sigma_D^2} + \frac{\sigma_D^2}{\sigma_I^2} - 2 + (\mu_I - \mu_D)^2(\frac{1}{\sigma_I^2} + \frac{1}{\sigma_D^2})\right]$
    si $\rho_{I,D}[i] < Umb_{KL}$ AND μ_D distinto a 0 **entonces**
      devolver Sí eDonkey
    si no
      devolver No eDonkey
    fin si
    $i \Leftarrow i + 1$
  fin para
  $nodo \Leftarrow nodo + 1$
fin para

V. Descripción del entorno experimental

Se han utilizado tres conjuntos de capturas de tráfico para realizar la experimentación relativa a las heurísticas presentadas en la Sección IV. A continuación se describen las características de estas trazas.

- *Trazas en entorno controlado (EC).* Un total de 5 usuarios aceptaron voluntariamente someterse a la monitorización de su tráfico de red durante 72 horas ininterrumpidas. En este período compartieron, mediante el programa aMule en su versión 2.2.6, una carpeta de archivos con idéntico contenido. Todos ellos se conectaron al servidor de eDonkey *se-Master Server 1* y limitaron la tasa de subida de aMule a 30kB/s. Todos los usuarios utilizaron sus PCs y su conexión a Internet sin ninguna restricción. En media cada usuario generó alrededor de 19000 conexiones del protocolo eDonkey y más de 7000 de otros protocolos, entre los que cabe destacar DNS, HTTP, SSH y SMTP.
- *Trazas de un servidor HTTP (SH).* Este conjunto de trazas contiene el tráfico generado por un servidor HTTP de una universidad europea durante 7 días. Es un servidor

Tabla I
TASA DE DETECCIÓN DEL ALGORITMO DE DETECCIÓN DE FLUJOS EN EL CONJUNTO DE TRAZAS TU.

	Tasa detección	Flujos detectados	Flujos totales
BitTorrent	0,0256	854	33304
HTTP	0,01691	50795	3003161
FTP	0,01423	35	2460
SSL	0,01244	2808	225685
IRC	0,00213	7	3281
Oscar	0,00079	2	2528
DNS	0,00001	8	1508413
Mail_POP	0,00000	0	5208
Todos	0,01139	54509	4784040

Apache versión 2.2.0 que recibe una media de 8971 conexiones por día.

- *Trazas de un troncal de universidad (TU).* Se ha almacenado todo el tráfico saliente del troncal de una universidad de Oriente Medio durante 48 horas. En este conjunto de trazas se encuentran alrededor de 73000 IPs, se han transmitido cerca de 300 millones de paquetes y los principales protocolos utilizados son: Bittorrent, HTTP, DNS, SSL y FTP entre otros varios. Tras el análisis de toda la base de datos mediante una inspección de *payload* con `OpenDPI` [10], no se ha detectado ningún paquete eDonkey. Esto se debe a que las aplicaciones P2P de compartición de archivos utilizadas en esta universidad de Medio Oriente se basan en Bittorrent en lugar de en eDonkey.

VI. RESULTADOS EXPERIMENTALES

La experimentación realizada para ambos algoritmos presentados en la Sección IV ha ido enfocada a dos objetivos principales: *(i)* Estudiar la validez de las hipótesis presentadas en las heurísticas, en base a la tasa de aciertos y de falsos negativos en las trazas de entorno controlado y, *(ii)* analizar si estas hipótesis se cumplen en otros protocolos, extrayendo para ello, en las trazas SH y TU el porcentaje de falsos positivos para protocolos diferentes a eDonkey.

A continuación se analizan separadamente los resultados obtenidos para los algoritmos de detección de flujos y de nodos.

A. Detección de flujos

En la experimentación realizada se ha supuesto que, por la propia filosofía de las redes P2P, un nodo de la red eDonkey utiliza para su funcionamiento normal más de un flujo del protocolo de forma simultánea. De modo que sólo se consideran flujos eDonkey si, aparte de ser detectados mediante la heurística propuesta, su actividad ha coincidido temporalmente con al menos otro flujo detectado en el mismo nodo.

Para aplicar la heurística de detección de flujos es necesario determinar el umbral de diferencia de *bytes* enviados frente a recibidos, Umb_B. Para esto se ha realizado un estudio del porcentaje de flujos detectados en los tres conjuntos de trazas en función del valor de Umb_B. Los resultados de este análisis indican que existe un amplio rango para la elección de este umbral dentro del cual la eficacia de este algoritmo de detección es considerable. Concretamente el valor utilizado para la experimentación de detección de flujos es de 10kB.

Las trazas EC contienen 37089 flujos de compartición de archivos eDonkey. De éstos, 28016 han sido detectados correctamente, lo que supone una tasa de acierto del 77,53%. En este conjunto de trazas no se ha producido ningún falso positivo. El porcentaje de los flujos de compartición de archivos de eDonkey que no han sido detectados es de un 22,47%. Esto se debe principalmente a dos motivos:

1) *ID baja en alguno de los extremos.* Los nodos con ID baja no pueden aceptar conexiones de la red eDonkey y, por este motivo, siempre inician la conexión, independientemente de si han de enviar los datos ellos o no. Esta situación queda, por tanto, fuera de la hipótesis del mecanismo propuesto de detección de flujos.
2) *Servicio sin cierre intermedio de conexión.* Aunque es menos frecuente, es posible que una solicitud de un recurso llegue a ser servida sin necesidad de cerrar la conexión inicial. Esto se produce cuando la petición inicial empieza a ser servida sin entrar en la cola de espera (vease Sección III). Si esto es así, la Hipótesis 1 de detección no es válida.

También se ha ejecutado el algoritmo de detección de flujos eDonkey en las trazas SH para explorar los falsos positivos que se producen en el protocolo HTTP. Los resultados de la detección indican que ninguno de los 62798 flujos totales ha sido detectado como eDonkey.

Para obtener el porcentaje de falsos positivos en las trazas de navegación real sin tráfico eDonkey (trazas SH y TU), se han etiquetado, en primer lugar, los protocolos a los que corresponden los diferentes flujos. Para ello, se ha modificado la aplicación `OpenDPI` para transformarla en una aplicación de clasificación de flujos en lugar de clasificación de paquetes.

Gracias a la modificación de `OpenDPI` se han podido extraer los resultados de detección que se muestran en la Tabla I. BitTorrent es el protocolo que mayor tasa de falsos positivos presenta y esto se debe a que al ser también un protocolo P2P para la compartición de archivos puede ser coherente con la hipótesis presentada. A diferencia de eDonkey, los flujos BitTorrent son bidireccionales, es decir, a través del mismo flujo pueden enviar partes de un recurso tanto un cliente como el otro. Por este motivo, el algoritmo no detecta un porcentaje mayor de flujos BitTorrent. Es lógico que los flujos FTP sean detectados porque es muy frecuente que el cliente envíe por FTP más que el servidor (caso de subida de ficheros al servidor). Un caso particular es el protocolo HTTP, el cual, aunque no fue detectado ningún flujo en las trazas SH, en éstas presenta un 1,69% de falsos positivos. Estudiados en detalle estos falsos positivos se descubrió que son provocados principalmente por tres motivos: (i) Campos de Cookie y URL muy extensos en el mensaje HTTP GET, (ii) respuestas del servidor con código 304 (Recurso no modificado), que son muy reducidas en tamaño y, por tanto, menores que la petición y, (iii) peticiones HTTP POST, en las que el envío de datos de formulario al servidor hacen que la petición pueda ser más grande que la respuesta.

B. Detección de nodos

Se ha verificado que la Hipótesis 2, para la detección de nodos, es coherente con la tasa de subida de cada uno de los nodos del experimento controlado, como se muestra en la

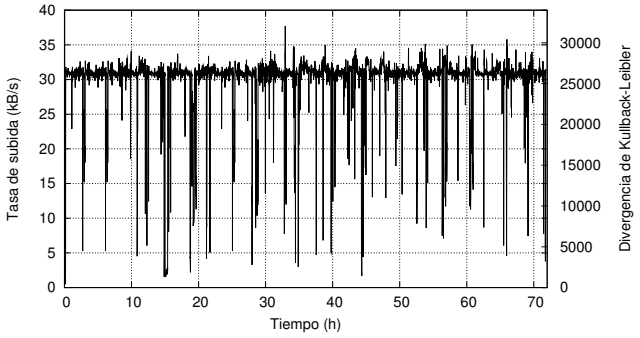

Fig. 4. Tasa de subida de uno de los nodos monitorizados en la traza EC.

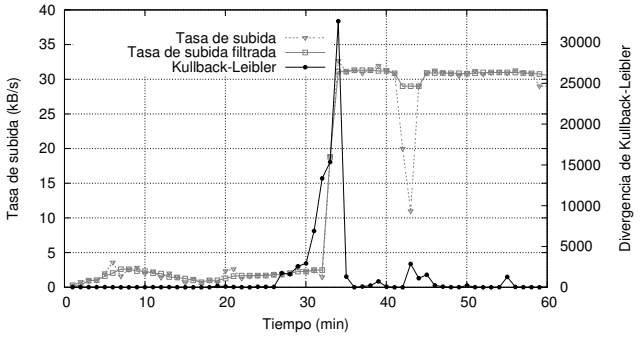

Fig. 5. Evolución de la divergencia de Kullback-Leibler en el análisis de una hora de tráfico (traza EC).

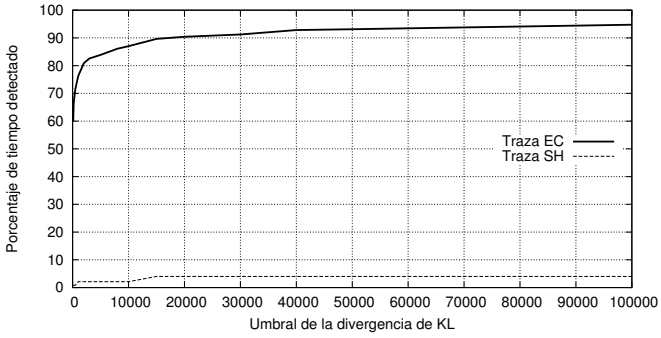

Fig. 6. Tasas de detección de la heurística de detección de nodos en las trazas EC y SH variando el valor de Umb_{KL}.

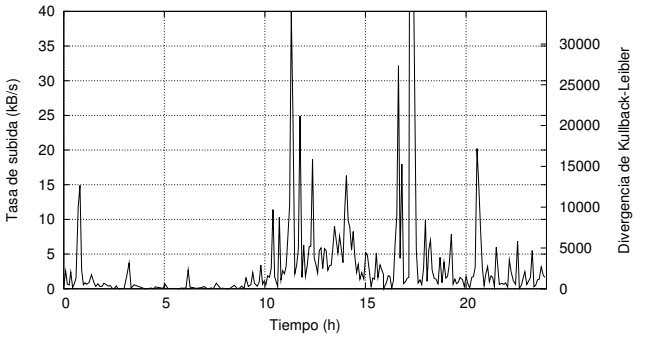

Fig. 7. Tasa de subida del servidor HTTP (traza SH) durante 24 horas.

Figura 4, en la que se representa dicha tasa para uno de los nodos. Durante las 72 horas de duración de la traza se puede apreciar claramente un comportamiento constante alrededor de 30kB/s, que es el límite superior impuesto por el usuario en el experimento.

También se observan en esta figura decaimientos de la tasa de subida. Éstos son de duración reducida y corresponden a instantes en los que se deja de enviar datos a un nodo para iniciar la transferencia con otro. El *churn* de las redes P2P (tasa de entrada y salida de nodos a la red) es elevado y es la principal razón por la que se producen decaimientos en la tasa de subida.

Los instantes en los que se supera el límite establecido por el usuario son otra característica a resaltar de la Figura 4. Éstos corresponden a actividad de red adicional al tráfico eDonkey, como puede ser: navegación HTTP, subida de algún archivo mediante SSH, envío de correo, etc. En estas trazas, un 38,7% de los flujos pertenecen a protocolos diferentes a eDonkey, entre los que destacan DNS, HTTP, SSH y SMTP.

En la Figura 5 se muestra la tasa de subida de uno de los usuarios de las trazas EC durante la primera hora del experimento. Durante los primeros 30 minutos apenas hay tasa de subida porque el servidor aún no ha dado a conocer a suficientes usuarios la existencia de este nuevo nodo en la red. A partir de esa primera media hora se aprecia que el mencionado comportamiento es constante alrededor de la tasa de 30kB/s. También se puede apreciar la reducción del efecto de los valores de tasa de subida que se desvían excesivamente de los esperados gracias a la aplicación del filtro de mediana. Por último, en esa misma representación se superpone la divergencia de KL de los valores filtrados de la tasa de subida. La media del primer tramo de la divergencia de KL es cercana a cero y la varianza reducida. Esto se debe a que la tasa de subida en este intervalo es muy similar. En el segundo tramo se puede observar un incremento en la divergencia de KL, debido al abrupto cambio de media y varianza de la tasa de subida. Por último, la media y varianza de la divergencia vuelven a ser reducidas en el tercer tramo de la representación. Este último tramo se detecta como tráfico generado por un nodo de la red eDonkey.

Adicionalemente, se ha realizado un estudio de los tres conjuntos de trazas para escoger el valor del umbral, Umb_{KL}, y el tamaño de la ventana, N, del filtro de mediana y del cálculo de la divergencia de KL. Se ha buscado un valor de Umb_{KL} que maximice la detección de los nodos generadores eDonkey y minimice la detección en los conjuntos de trazas TU y SH. Para esto se ha ejecutado el algoritmo de detección sobre estos tres conjuntos de trazas con valores del umbral en el rango de 10^2 a 10^5. Los resultados de detección se muestran en la Figura 6, en la que se puede observar que existe un amplio rango de Umb_{KL} sin grandes variaciones en las tasas de detección y de falsos positivos. Este resultado permite a los autores asumir que la elección de Umb_{KL} dentro de un rango lógico $[10^3, 10^4]$ no es crítica. Para los resultados expuestos a continuación se ha seleccionado un valor de Umb_{KL} igual a 8000.

Respecto a la elección del tamaño de ventana, N, del filtrado de mediana y del cálculo de la divergencia de KL, se ha fijado a 5 minutos por ser ésta la mínima ráfaga de tráfico constante que se pretende detectar. Un tamaño de ventana mayor dificultaría enormemente la detección de ráfagas de este tamaño. Las ráfagas de un tamaño menor no son objeto de este trabajo ya que se asume que un nodo generador de tráfico eDonkey interesante por su elevado consumo de ancho

de banda permanece conectado a la red un tiempo prolongado.

El porcentaje de tiempo durante el cual, según el algoritmo propuesto, los nodos han estado generando tráfico eDonkey ha sido de un 86,10%. Los intervalos de la tasa de subida que no se han detectado como constantes se relacionan con los decaimientos provocados por el cambio de usuario con el que se comparte.

Por último, también se ha comprobado, como puede verse en la Figura 7, que la tasa de subida del servidor HTTP no parece tener un comportamiento constante con media superior a cero. Este mismo comportamiento, aparentemente aleatorio, se observa en los nodos monitorizados en la traza TU. Solamente un 1,678% de las 168 horas en las que se monitorizó el servidor HTTP (traza SH) se clasificó a este nodo como generador eDonkey (falsos positivos).

VII. Conclusiones

En el presente artículo se aborda el problema de detectar el tráfico perteneciente al protocolo eDonkey sin inspeccionar el *payload*. Para este fin, se proponen dos algoritmos de detección. El primero para la detección de flujos y el segundo para la detección de nodos. La heurística de detección de flujos se basa en la afirmación de que los flujos del protocolo eDonkey cumplen que el número de *bytes* enviados desde el cliente que inicia la conexión al que la recibe es sustancialmente mayor que en el sentido inverso. La segunda heurística afirma que son nodos generadores de tráfico eDonkey aquellos cuya tasa de subida es constante conectándose a múltiples IPs. En base a la experimentación realizada con los tres conjuntos de trazas se puede concluir que las hipótesis de detección propuestas se cumplen para el protocolo eDonkey, presentando una buena tasa de detección y un reducido número de falsos positivos. La experimentación también ha permitido especificar que la heurística de detección de flujos es válida únicamente en flujos de compartición de archivos para nodos eDonkey con ID alta.

Actualmente se está trabajando para ampliar la experimentación a trazas reales, fuera de un entorno controlado, que contengan de forma simultánea tráfico eDonkey y de otros protocolos. Adicionalmente, se pretenden abordar en un futuro próximo dos líneas de trabajo principales:

- La combinación de la información de ambas heurísticas para la mejora de la tasa de detección y la reducción de los falsos positivos.
- Explorar la posibilidad de detectar otros protocolos P2P de compartición de archivos mediante la ejecución de la segunda heurística. Es asumible que esta detección pueda ampliarse porque los protocolos de compartición de archivos saturan la tasa de subida del usuario y esto implica la necesidad de una limitación en la misma. Esta limitación es aprovechada en la detección propuesta y por ello se cree que puede ser ampliada a otros protocolos como Kademlia o BitTorrent.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el MICINN mediante el proyecto TEC2008-06663-C03-02.

Referencias

[1] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A Survey on Internet Traffic Identification," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 37–52, Aug. 2009.

[2] A. Feldmann, "A possibility for isp and p2p collaboration," in *Broadband Communications, Networks and Systems, 2008. BROADNETS 2008. 5th International Conference on*, 2008, p. 239.

[3] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ser. IMC '04. New York, NY, USA: ACM, 2004, pp. 121–134.

[4] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '05, vol. 33, no. 1. New York, NY, USA: ACM, Jun. 2005, pp. 50–60.

[5] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: multilevel traffic classification in the dark," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '05, vol. 35, no. 4. New York, NY, USA: ACM, 2005, pp. 229–240.

[6] K. Xu, M. Zhang, M. Ye, D. M. Chiu, and J. Wu, "Identify P2P traffic by inspecting data transfer behavior," *Computer Communications*, vol. 33, no. 10, pp. 1141–1150, Jun. 2010.

[7] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing skype traffic: when randomness plays with you," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '07. New York, NY, USA: ACM, 2007, pp. 37–48.

[8] J. Ramirez, J. Segura, C. Benitez, A. de la Torre, and A. Rubio, "A new kullback-leibler vad for speech recognition in noise," *Signal Processing Letters, IEEE*, vol. 11, no. 2, pp. 266–269, 2004.

[9] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proceedings of the IEEE*, vol. 78, no. 4, pp. 678–689, Apr. 1990.

[10] Opendpi. [Online]. Available: http://www.opendpi.org/