Metodología de Empleo de Herramientas *Software* para el Análisis y Evaluación de Protocolos de Transporte *Multicast*

Juan Carlos Sánchez Aarnoutse Pilar Manzanares López Josemaría Malgosa Sanahuja Joan García Haro
Departamento de Tecnologías de la Información y las Comunicaciones
Universidad Politécnica de Cartagena

{juanc.sanchez, pilar.manzanares, josem.malgosa, joang.haro}@upct.es

***Abstract*-In this paper, we present a set of software tools (network analyzers and data processing language applications) which allow the study of multicast transport protocols in real environments. To evaluate the proposed tools' efficiency we used the MTP (MUst Transport Protocol) developed by us in a recent work and also briefly outlined in this paper. This transport protocol offers multicast transfer service to the application layer to replicate efficiently files or hard disk partitions among server and clients. There are multiple application scenarios where this protocol can be used (peer to peer, instant messaging, games, etc.). The main statistical performance parameters obtained by test labs, are also presented and discussed.**

I. INTRODUCCIÓN

En este artículo se presenta una serie de herramientas *software* utilizadas para la captura de tráfico y su posterior análisis estadístico. Dichas herramientas se basan en programas de libre distribución al alcance de cualquier usuario y nuestro objetivo ha consistido principalmente en reprogramarlas adecuadamente de tal forma que fueran capaces de analizar el protocolo de transporte *multicast* MTP (*MUST Transport Protocol*) propuesto y descrito exhaustivamente en [1][2].

Por lo general, los protocolos de transporte *multicast* se encapsulan dentro de una trama UDP (puesto que TCP no es capaz de soportar comunicaciones multipunto). Por ello, las herramientas de análisis que se exponen en este artículo pueden utilizarse (con las convenientes modificaciones) para analizar cualquier tipo de protocolo *multicast*.

Junto con las herramientas anteriores, se ha programado una aplicación *multicast* denominada MUST (*MUlticast Synchronous Transfer*) que permite la réplica masiva de archivos y particiones de disco duro hacia múltiples destinatarios. Dicha aplicación utiliza a nivel de transporte el protocolo MTP encargado de solventar todas las necesidades de transporte de la aplicación MUST.

Las principales características de MTP son: simplicidad, capacidad de trabajo en entornos *intra-campus*, simetría dentro del grupo *multicast* (todos los clientes participan de igual forma en las tareas relacionadas con el control de flujo) y por último, alta fiabilidad mediante una técnica de control de errores que simultáneamente consigue un caudal de transferencia elevado.

El resto del articulo se organiza de la siguiente manera: en la segunda sección se describe, a modo de resumen, el funcionamiento del protocolo MTP. Posteriormente, se exponen tanto las herramientas como la metodología de trabajo que ha permitido realizar las distintas pruebas de laboratorio. En la cuarta sección se presentan las figuras de mérito más representativas para la evaluación, tanto del protocolo como de las herramientas de análisis. Finalmente, se enuncian las conclusiones más relevantes y se enumeran algunas de las futuras líneas de trabajo.

II. DESCRIPCIÓN DEL PROTOCOLO

El protocolo MTP fue descrito de forma detallada en [1][2]. Sin embargo, para comprender con claridad el análisis matemático y las figuras de mérito expuestas en las secciones posteriores, conviene resumir brevemente el funcionamiento básico del protocolo.

Para conseguir un caudal de transferencia elevado y un tiempo de transferencia mínimo, el protocolo MTP divide el proceso de transferencia en dos fases. En una primera fase, el servidor envía paquetes de información en modo *multicast* hacia todos los clientes. Dichos clientes deben almacenarlos (en disco duro o en memoria) y confirmar la recepción del paquete mediante un solo paquete de reconocimiento (ACK). La generación de un único ACK por cada paquete de datos se consigue gracias a un algoritmo de competición en el que intervienen todos los clientes. Cuando esta fase termina, comienza una segunda fase en la que cada uno de los clientes solicita la retransmisión de aquellos paquetes perdidos. En esta segunda fase, todos los paquetes son transmitidos en modo *unicast*.

El protocolo propuesto es también capaz de trabajar en un entorno intra-campus (islas LAN interconectadas mediante un pequeño grupo de encaminadores). Por ello, junto con la corrección de errores y el algoritmo de reducción de ACK, MTP también implementa un algoritmo de control de flujo basado en la redefinición de los temporizadores del protocolo (*timers* y sus correspondientes valores de *time-out*) en función del estado de congestión de la red.

III. DESCRIPCIÓN DE LAS HERRAMIENTAS

En este artículo se analizan y se evalúan las prestaciones del protocolo MTP en diferentes pruebas de laboratorio. Para ello ha sido necesario en primer lugar capturar el tráfico generado por MTP, para posteriormente poder analizar estadísticamente las prestaciones de dicho protocolo. En los siguientes apartados se estudiará con mayor detalle estos dos aspectos.

A. Analizadores de red

Existen diversas soluciones que permiten la captura del tráfico de paquetes en red. La primera de ellas consiste en utilizar un analizador de protocolos *hardware* (como por ejemplo el equipo Domino-LAN de la firma *Wandel & Goltermann*). Estos equipos son capaces de capturar tramas y calcular algunas estadísticas de tráfico en tiempo real. Los resultados pueden posteriormente guardarse en un fichero de texto con el formato adecuado para su posterior procesado estadístico, ampliando de esta manera el abanico de los parámetros estadísticos de interés. Sin lugar a dudas, el inconveniente más notable de esta solución es el elevado precio del equipo (alrededor de los 6.000 euros). Además, es estrictamente necesario utilizar también un ordenador (por lo general portátil, puesto que en muchas situaciones deben hacerse medidas en distintos puntos de la red). Otro inconveniente es su limitada capacidad de adaptación a nuevas tecnologías de red (y cuando ello es posible, siempre es a costa de una inversión elevada).

La segunda solución consiste en emplear analizadores de red (*sniffers*) *software*, que emplean un ordenador compatible como plataforma *hardware*. Existe una gran diversidad de este tipo de analizadores, comercializados por diversas firmas de prestigio (Cisco, Hp, Nortel Networks, etc.). Esta opción, como en el caso anterior, también necesita una inversión considerable. Además, por lo general, este tipo de herramientas tienen una limitada capacidad de análisis de tráfico, centrando toda su potencialidad en la gestión de los equipos que conforman la red.

Otros analizadores *software* son de libre distribución, como por ejemplo *ethereal* [3] y *tcpdump* [4] (el primero de ellos trabaja en modo gráfico mientras que el segundo lo hace en modo cónsola). Para la mayoría de las situaciones esta solución es la más idónea puesto que sus prestaciones son comparables a las obtenidas con analizadores *hardware* y requiere una inversión económica mínima. Asimismo, esta opción se adapta fácilmente a cualquier tecnología de red, puesto que en la actualidad existen multitud de interfaces de red (X.25, *Frame Relay*, RDSI, ATM, etc.) para ordenadores compatibles, a precios muy competitivos. Ésta es la solución que se ha optado para la evaluación de las prestaciones del protocolo MTP.

En ambos casos, el hecho de trabajar con analizadores de código abierto ha permitido, con relativa facilidad, ampliar su capacidad de análisis de paquetes, de forma que también fueran capaces de extraer la cabecera del protocolo MTP y de mostrar las capturas en un formato adecuado.

Ethereal es un *software* multiplataforma codificado en lenguaje C. Emplea, entre otras, las librerías *libpcap* (*Packet CAPture LIBrary*, para la captura y filtrado de los paquetes de nivel de enlace), *gtk+* (para la creación de interfaces de usuario en modo gráfico) y *glib* (que permite construir un *stack* de protocolos de forma más o menos similar al recomendado por el modelo en capas OSI). La introducción de un nuevo protocolo a la amplia lista de los ya existentes se lleva a cabo mediante la codificación de un nuevo disector (que no es más que un fichero escrito en código C cuyo nombre debe ser *packet_mtp.c*, es decir, la palabra reservada *packet_* seguida por un acrónimo del nuevo protocolo). En dicho disector se codifica la definición del nuevo protocolo, de los campos que componen la cabecera de cada uno de los paquetes, sus nombres y el formato en los que deben ser mostrados en las ventanas de resultados. Además, este nuevo disector debe registrarse en el disector del protocolo del nivel inferior (de forma que la librería *glib* pueda insertar MTP dentro del *stack* global de protocolos). En nuestro caso, y puesto que MTP se programa utilizando un *socket* UDP, el disector padre debe ser el específico de UDP.

Tcpdump es un *software* que, al igual que *ethereal*, es de libre distribución, está desarrollado en C y utiliza la librería *libpcap*. Sin embargo, en este caso la información se presenta por pantalla en modo texto. Ello tiene la ventaja de poder capturar tráfico empleando ordenadores que no posean entorno gráfico (servidor X en el argot de Linux). Otra ventaja de este analizador es que emplea menos recursos del sistema, un parámetro importante en sistemas operativos multitarea que no trabajan en tiempo real.

En este caso la modificación del código se ha enfocado a dos niveles. El primero ha consistido en una redefinición completa del formato que emplea el programa para presentar las capturas; ahora en vez de presentar la información de cada paquete de forma estructurada en un árbol específico para cada protocolo, se presenta de forma compacta en una única línea.

El segundo nivel, como en el caso de *ethereal*, ha consistido en la programación de los módulos necesarios para que *tcpdump* sea capaz de entender la cabecera del protocolo MTP. En este caso hay que indicar en el fichero que procesa el protocolo UDP (*print_udp.c*) que, cuando el puerto UDP sea el correspondiente al protocolo MTP, llame a una función encargada de analizar y presentar su cabecera (*print_overudp*). Esta función se ubica en un nuevo archivo que hemos llamado *over_udp.c*. Una ventaja adicional de esta metodología es que puede emplearse para analizar cualquier protocolo encapsulado sobre UDP de una manera sencilla. Para ello basta con modificar el código de la función *print_overudp*.

Mencionar por último que la adaptación de *tcpdump* ha sido algo más compleja que la de *ethereal*, debido a que en el primero se ha tenido que redefinir la forma de presentar por pantalla las capturas para todos los protocolos. *Ethereal*, en cambio, pose una ventana de resumen propia.

B. Procesado estadístico

Para procesar los datos capturados se ha utilizado el lenguaje de libre distribución *awk* [5]. Como ya es sabido, este lenguaje permite procesar información representada en

formato de líneas (también denominadas registros). A su vez, la información contenida en cada registro debe representarse en sucesivas columnas (denominadas campos). Este esquema de representación ha impuesto el formato de salida de las capturas de los analizadores de red expuestos en el apartado anterior. La elección de este *software* ha venido impuesta por su capacidad de procesado de datos a gran velocidad, incluso con archivos compuestos por miles de registros.

La programación y las funciones asociadas de *awk* son muy parecidas al lenguaje C. Además, *awk* permite hacer multitud de operaciones aritméticas con los campos de cada uno de los registros de una forma extremadamente sencilla. Ello permite calcular todo tipo de figuras de mérito (como la utilización, el *throughoput*, etc.) de una forma rápida y eficaz.

IV. RESULTADOS EXPERIMENTALES

En esta sección se presentan y analizan los resultados de las pruebas realizadas con el protocolo MTP. Para la realización de las pruebas se implementó la aplicación MUST (*MUlticast Synchronous Transfer*), que permite la transferencia de un archivo desde un servidor a un grupo de clientes empleando el protocolo MTP a nivel de transporte. Las pruebas se realizaron en un laboratorio con una red local *Ethernet* compuesta por doce PCs PIII a 1GHz con 128 Mbytes de RAM conectados a través de hubs 3Comm de 10 Mbps. El sistema operativo de las máquinas es Linux SuSE versión 7.2.

En primer lugar se va a evaluar el comportamiento del algoritmo propuesto para la reducción de paquetes de reconocimiento (ACK), que fue exhaustivamente descrito en [1]. De forma resumida diremos que todos los clientes compiten por enviar el paquete de reconocimiento. En esta competición, cada cliente espera un tiempo aleatorio antes de generar su propio ACK (obtenido de una distribución aleatoria uniforme comprendida entre 0 y un valor máximo ARTPmax). El cliente que obtenga el tiempo aleatorio menor será el encargado de transmitir en modo *multicast* el ACK, que por tanto, será recibido por los demás clientes, abortando de inmediato el envío de su propia trama de reconocimiento.

Realizando un análisis matemático [6] llegamos a la conclusión de que la función de distribución de probabilidad que modela el tiempo transcurrido entre la recepción de un paquete de datos y la generación de su correspondiente reconocimiento sigue la siguiente ley:

$$F_n(t) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} \left(\frac{t}{k}\right)^i \qquad (1)$$

Para verificar dicho resultado, se ha realizado una captura de tráfico correspondiente al envío de un archivo de 20 Mbytes hacia un grupo *multicast* formado por 8 clientes (ver figura 1). Para obtener la función de distribución de probabilidad de este experimento se procesó esta captura mediante el código *awk* mostrado en la figura 2.

Las figuras 3 y 4 muestran la representación de la función de densidad de probabilidad analítica (ecuación (1)) y experimental. Podemos observar la similitud entre ambas

No.	Time	Source	Dest	Prot	SN	Size	PacketType
26	25.43632	192.168.2.8	224.0.1.1	MTP	0	512	1
27	25.43793	192.168.2.3	224.0.1.1	MTP	0	9	2
28	25.43852	192.168.2.8	224.0.1.1	MTP	1	512	1
29	25.44042	192.168.2.6	224.0.1.1	MTP	1	9	2
31	25.44420	192.168.2.5	224.0.1.1	MTP	2	9	2
32	25.44474	192.168.2.8	224.0.1.1	MTP	3	512	1
34	25.44533	192.168.2.4	224.0.1.1	MTP	3	9	2
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Fig. 1. Extracto de una captura de tráfico mediante la versión *tcpdump* modificada.

```
BEGIN {for (i=0; i<19; i++) fdp[i]=0; datos=0;
total=0; }
{
if($5=="MTP")
 {
    if (($8=="1")||($8=="3"))
       {gendatos[$6]=$2; datos++; genack[$6]=0;}
    if (($8=="2")&&(generacionack[$6]==0))
       {generacionack[$6]=$2;}
 }
}
END {
for (i in gendatos)
 {
  artp[i]=genack[i]-gendatos[i];
  total=total+artp[i];
  fdp[int(artp[i]/0.05)]++;
 }
for (i=0; i<19; i++) print fdp[i]/datos

}
```

Fig. 2. Programa en *awk* para la obtención de la función de distribución de probabilidad.

gráficas, lo que confirma la validez tanto de las herramientas de análisis de red propuestas como del comportamiento del protocolo MTP. Además puede observarse que, a pesar de que para la generación de su propio paquete de reconocimiento cada cliente usa una distribución uniforme, la función de densidad de probabilidad del grupo *multicast* pierde este comportamiento, siendo ahora la probabilidad de los valores pequeños mayor que la de los valores grandes.

En segundo lugar se va a evaluar el comportamiento temporal de la aplicación (utilizando también las herramientas descritas en la sección anterior). Para ello, se ha realizado un conjunto de experimentos que consisten en enviar archivos de diferentes tamaños (entre 1 y 20 Mbytes) hacia un mismo grupo *multicast* formado por 2, 4, 6, 8 y 10 clientes.

La figura 5 representa el tiempo de transferencia de cada uno de estos experimentos. En ella se observan dos tendencias. Por un lado, el tiempo de transferencia aumenta de forma proporcional con el tamaño del archivo, ya que cuanto mayor es el fichero, mayor es el número de paquetes a enviar. Por otro, para el mismo tamaño de archivo, el tiempo de transferencia decrece cuando el número de clientes aumenta. Ello es debido a que a medida que el número de clientes aumenta, el tiempo transcurrido entre la generación de un paquete de datos y la generación de su correspondiente trama de reconocimiento decrece (es decir, la ecuación (1) tiene un comportamiento decreciente en función del número de clientes).

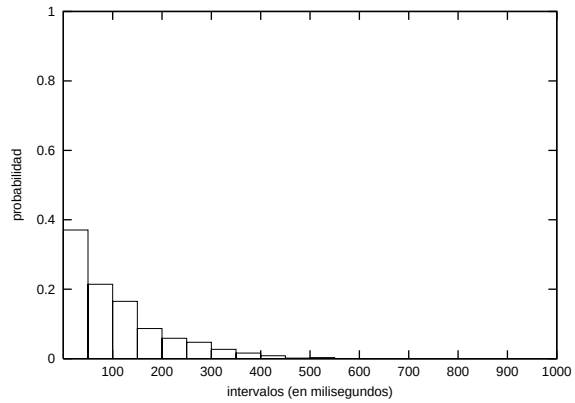

Fig. 3. Función de densidad de probabilidad experimental para 8 clientes con un ARTP máximo de 1 seg.

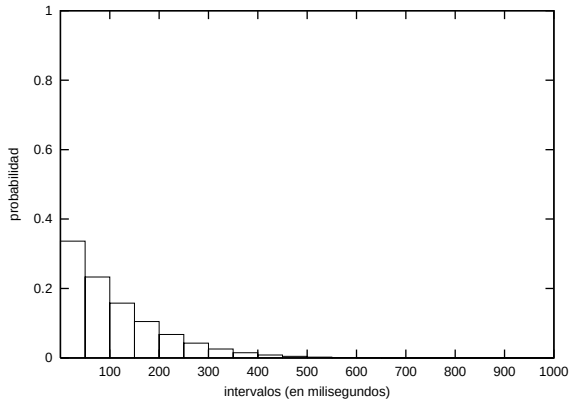

Fig. 4. Función de densidad de probabilidad analítica para 8 clientes con un ARTP máximo de 1 seg.

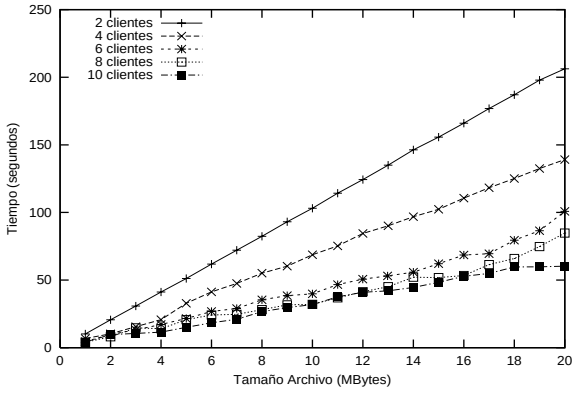

Fig. 5. Tiempos de transferencia en la transmisión de un archivo usando MUST para 2, 4, 6, 8 y 10 clientes, en función del tamaño del archivo.

V. CONCLUSIONES

En este artículo se han presentado varias herramientas *software* de análisis de tráfico de red para la evaluación del protocolo MTP. Tras estudiar las distintas opciones disponibles en el mercado (analizadores *hardware*, analizadores *software* comerciales y analizadores *software* de libre distribución), nos decantamos por el uso de éstos últimos. Este tipo de *sniffers* presenta una serie de ventajas frente al resto de opciones: bajo coste, fácil adaptación a las nuevas tecnologías y relativa sencillez en su adaptación a nuevos protocolos.

Para procesar la información proporcionada por los analizadores se optó por la utilización de *awk*, un lenguaje de procesado de registros que permite un rápido y eficaz cómputo de grandes cantidades de información.

Gracias a estas herramientas ha sido posible valorar una aplicación para la transferencia masiva de información hacia un conjunto de clientes, basada en el protocolo *multicast* MTP. En este artículo se ha comprobado el funcionamiento del algoritmo de generación de paquetes de reconocimiento, así como su implicación en el comportamiento temporal de la aplicación.

VI. LÍNEAS FUTURAS

El protocolo de transporte *multicast* propuesto en [1], cuyas prestaciones han sido parcialmente evaluadas en este artículo es susceptible de ser mejorado (por ejemplo mediante la inclusión de paquetes de confirmación negativa (NACK), la transmisión de bloques de datos, etc.). Todas estas modificaciones se reflejarán, de alguna forma, en una nueva estructura de cabecera del paquete. Por ello, se pretende construir un analizador *software* de tráfico de libre distribución escrito en un lenguaje orientado a objetos (C++ o Java si se desea también portabilidad) cuyas propiedades de herencia y polimorfismo permitirán la inclusión de nuevos protocolos a nivel de usuario, en lugar de tener que hacerlo a nivel de programador.

VII. RECONOCIMIENTOS

Este trabajo ha sido subvencionado por los proyectos nacionales FAR-IP (TIC2000-1734-C03-03) y MTCES (TIC2001-3339-C02-02).

VIII. REFERENCIAS

[1] P. Manzanares López, J. Malgosa Sanahuja, J. García Haro, J.C. Sánchez Aarnoutse, “Diseño e implementación de un protocolo de transporte *multicast* para la réplica masiva de información”, URSI’02, pp. 297-298, Alcalá de Henares (España), Septiembre 2002.

[2] Josemaria Malgosa-Sanahuja, Joan Garcia-Haro, Fernando Cerdan, Francesc Burrull-Mestres, “MUST, a Multicast Synchronous Transfer Application for Fast Intra-Campus Replication”, IEEE Melecon’02, pp. 100-104, El Cairo (Egipto), Mayo 2002.

[3] La información, descripción y código del *sniffer* ethereal puede encontrarse en www.ethereal.com

[4] La información, descripción y código del *sniffer* tcpdump y la librería *libpcap* puede encontrarse en www.tcpdump.org

[5] A. V. Aho, B. W. Kerninghan, P.J. Weinberger, “Awk - A Pattern Scaning and Preprocessing Language”. AT&T Bell Laboratories, Murray Hill, New Jersey.

[6] Pilar Manzanares-Lopez, Juan Carlos Sanchez-Aarnoutse, Josemaria Malgosa-Sanahuja, Joan Garcia-Haro, “Modelling and Performance Evaluation of a Multicast Synchronous Transfer Protocol”, (en proceso de revisión) IEEE PACRIM’03, Victoria (Canadá), Agosto 2003.