Multicast en redes Intra-campus heterogéneas.

P. Manzanares López, J.C. Sánchez Aarnoutse, J. Malgosa Sanahuja, J. García Haro
Departamento de Tecnologías de la Información y las Comunicaciones.
Escuela Técnica Superior de Ingeniería de Telecomunicación. Universidad Politécnica de Cartagena
Campus Muralla del Mar. Edificio Antiguo Hospital de Marina
30202 Cartagena
Teléfono: 968 32 6534 Fax: 968 32 5973
E-mail: {pilar.manzanares, juanc.sanchez, josem.malgosa, joang.haro}@upct.es

Resumen. *En este artículo, se propone y se analiza una aplicación multicast para la transferencia de información en entornos intra-campus (con la posibilidad de que algunas redes sean inalámbricas) que hemos denominado SOMA (SynchOnous Multicast Application). Dada la incapacidad de TCP para manejar comunicaciones multipunto, se ha desarrollado un protocolo de transporte que aporta mecanismos de control de flujo y de errores para conseguir una elevada tasa de transferencia y un retardo bajo, especialmente en entornos formados con redes heterogéneas.*

1 Introducción

Aunque la tecnología inalámbrica LAN IEEE 802.11 empezó a utilizarse en 1997, es hoy día cuando está alcanzando una notable difusión. Ello es debido, por un lado, a la reducción de los precios de los dispositivos inalámbricos, y por otro lado, a la creciente popularidad de los equipos portátiles (ordenadores, PDAs, tabletPCs, etc.).

Son bien conocidas las ventajas que aporta el uso de la tecnología *multicast*: reduce los costes de comunicación, minimiza el consumo del ancho de banda en los enlaces, reduce el procesado y el retardo tanto en el emisor como en los routers. Debido a que las redes inalámbricas generalmente operan con tasas de transmisión menores que las cableadas, las redes en las que coexisten ambas tecnologías pueden presentar congestión. Por lo tanto, es muy recomendable emplear (siempre que sea posible) tecnología *multicast* en estos escenarios [1].

En líneas generales, podemos afirmar que las redes *Ethernet*, cableadas e inalámbricas, y la mayoría de las tecnologías de acceso a Internet como *Cablemodem*, *ADSL*, o *Ehternet-at-home*, son inherentemente *multicast*. Además, muchas de las redes actuales están formadas por islas LAN interconectadas a través de varios routers. En este escenario (al que denominamos intra-campus), las aplicaciones tradicionales coexisten con nuevas aplicaciones de elevado consumo de ancho de banda. En todos estos casos, la tecnología *multicast* puede ayudar a optimizar el uso del ancho de banda [2].

A pesar de que existe alguna implementación experimental de TCP con *multicast*, la incapacidad de TCP para proporcionar conexiones multipunto sigue siendo una realidad. Por esta razón, en [3] se propuso un protocolo de transporte *multicast* que proporciona a estas nuevas aplicaciones una forma fácil y sencilla de replicar archivos y particiones de disco duro en un entorno intra-campus. En este artículo, se propone analizar, implementar y experimentar con una versión mejorada de dicho protocolo. Éste ha sido escrito empleando las librerías estándar del *kernel* de *linux*, por lo que es posible su implementación en cualquier tipo de aplicación sin incurrir en conflictos legales.

El resto del artículo se organiza de la siguiente manera: en la segunda sección se describe brevemente el funcionamiento básico del protocolo de transporte propuesto y más detalladamente las mejoras introducidas. En la tercera sección presentamos los resultados de los experimentos realizados en entornos intra-campus heterogéneos. Para finalizar se enuncian las conclusiones más relevantes del trabajo.

2 Descripción de SOMA

SOMA es una aplicación *multicast* diseñada para transmitir síncronamente archivos de elevado tamaño (incluso particiones de disco duro) a un conjunto de clientes. Dada la incapacidad de TCP para proporcionar soporte *multicast*, SOMA implementa su propio protocolo de transporte.

SOMA es una evolución de [3] especialmente diseñado para operar en entornos intra-campus compuestos por redes cableadas e inalámbricas. En este tipo de entornos heterogéneos se manifiestan tres características que afectan directamente al rendimiento del protocolo: (1) en un canal de radio, las variaciones de la relación señal ruido causan oscilaciones importantes en la capacidad del canal (medida en bps). (2) La tasa de error de bit medida en un periodo de tiempo breve no puede ser despreciada. Y (3), muchos puntos de acceso comerciales, empleados para conectar una red inalámbrica de 11 Mbps con una cableada a 100 Mbps, tienen una capacidad de buffer muy limitada.

Teniendo en cuenta estas características, se han realizado tres modificaciones en el protocolo original:

- El servidor envía un conjunto de paquetes consecutivos (una ventana) y los clientes responden con un único ACK para confirmar toda la ventana.

- La cabecera del protocolo tiene un nuevo campo (LWSN, Last Window Sequence Number) que indica el número de secuencia del último paquete de la ventana que se está transmitiendo.

- Los clientes envían paquetes NACKs cuando detectan errores o pérdidas de paquetes de datos. El control de flujo modificará el tamaño de la ventana de transmisión en función de la información proporcionada por los paquetes ACK y NACK y el *timeout*.

Con estas modificaciones se mantienen las mismas características del protocolo original (sincronismo y bajo consumo de ancho de banda) consiguiendo una tasa de transferencia mayor (en ocasiones máxima), pero a la vez, adaptándose rápidamente a situaciones de congestión.

A. Descripción general del protocolo

El protocolo SOMA divide el proceso de transferencia en dos fases. En una primera fase, el emisor (servidor) envía un conjunto de paquetes de datos (lo que denominamos una ventana) en modo *multicast* a todos los receptores (clientes). Dichos clientes almacenan los datos y confirman el último paquete de datos de la ventana con un único ACK por LAN conectada. La generación de un único ACK se consigue empleando el mismo algoritmo de competición que en el protocolo original. Si un cliente detecta un paquete erróneo o perdido, almacena una marca de error y envía un NACK al servidor. En esta fase no se retransmiten los paquetes perdidos o erróneos. La secuencia expuesta se repite hasta que se ha transmitido todo el archivo. Entonces, comienza una segunda fase en la que cada uno de los clientes solicita la retransmisión de aquellos paquetes perdidos. En esta última fase de retransmisión todos los paquetes son enviados en modo *unicast*.

B. Algoritmo de control de flujo

Durante la primera fase, cada vez que el servidor finaliza la transmisión de una ventana espera, durante un tiempo definido por un temporizador (*timeout*), una confirmación del último paquete por cada LAN conectada. Si el temporizador expira antes de confirmar la ventana, se aumenta su valor multiplicándolo por un factor α (con α>1). Por contra, si los paquetes de confirmación llegan antes de que expire, el nuevo valor del *timeout* se reduce de la siguiente manera:

$$T_{out} = max\{T_{out} / \beta, default\_T_{out}\}$$

Donde β es mayor que la unidad y $default\_T_{out}$ es el valor límite inferior del *timeout*.

Por otro lado, el servidor modificará el tamaño de la ventana justo antes de transmitir la siguiente ventana de transmisión dependiendo del temporizador y de los ACKs y NACKS que haya recibido:

- Si la ventana es confirmada antes de que expire el *timeout*, el servidor incrementará la ventana en una unidad.

- Si el temporizador expira antes de recibir todas las confirmaciones, el servidor decrementará la ventana en una unidad.

- El servidor decrementará la ventana en una unidad por cada NACK recibido durante esa ventana de transmisión.

Por su parte, cada cliente espera la llegada de los paquetes de datos y sólo iniciará el algoritmo de competición para generar un ACK cuando el número de secuencia coincida con el campo LWSN.

Fig. 1 muestra un esquema del funcionamiento habitual del protocolo. El servidor envía un conjunto de paquetes de datos, incrementando cada vez la ventana hasta que se alcanza el tamaño W. En este punto, el temporizador expira justo antes de recibir el ACK.

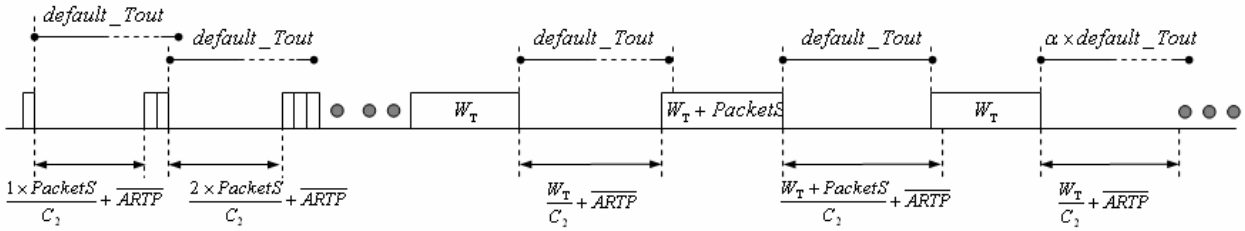

Fig. 1. Evolución y estabilización del tamaño de la ventana cuando una de las redes (LAN inalámbrica a 2 Mbps) es más lenta que la otra (*Ethernet* cableada a 100 Mbps). En la figura, el tamaño de la ventana (W) se representa en bits. C_2 es la velocidad de la red inalámbrica.

3 Resultados

En esta sección evaluamos el protocolo SOMA en un escenario real formado una red cableada *fast-Ethernet* a 100 Mbps unida mediante un punto de acceso a otra red inalámbrica a 2 Mbps. Esta acusada diferencia de capacidades (50:1) favorece la aparición de congestión. Para capturar el tráfico hemos utilizado la herramienta *tcpdump* [4] convenientemente modificada en una máquina con Linux con el kernel pacheado con *BSD packet filter* [5].

Fig. 2 muestra la evolución del tamaño de la ventana con el tiempo para valores de *default_*T_{out} de 80, 90, 100 110 y 120 ms. Tal y como se explicó en la Fig. 1, un valor de *default_*T_{out} menor fuerza a l algoritmo de control de flujo a un tamaño de W menor. Las fluctuaciones en torno al valor estacionario son debidas a pérdidas esporádicas en la red inalámbrica (los clientes generan paquetes NACK con lo que el servidor reducirá la ventana). Otro factor que contribuye a estas fluctuaciones es la presencia de paquetes de aplicaciones de control como *spanning-tree* que comparten la capacidad del punto de acceso.

Fig. 3 representa el *throughput* instantáneo del mismo experimento. En todos los casos, independientemente del valor de *default_*T_{out}, el servidor alcanza un valor ligeramente oscilante alrededor de 1,6 Mbps (1,6 Mbps es la capacidad efectiva de la red de 2Mbps). Se puede deducir que el algoritmo de control propuesto es capaz de adaptar la tasa de transmisión del servidor a la capacidad de la red más lenta manteniendo el sincronismo entre todos los clientes.

4 Conclusiones

SOMA es una aplicación multicast para la replica de archivos en cualquier tipo de red intra-campus.

Uno de los aspectos más destacable es la definición de su propio protocolo de transporte, y más concretamente sus algoritmos de control de flujo y de errores. La relación señal ruido de un entorno inalámbrico afecta drásticamente a la capacidad de la red y a la tasa de error. Además, muchos de los puntos de acceso comerciales tienen una limitada capacidad de buffer. El algoritmo de control de flujo propuesto es capaz de reaccionar rápidamente ante estas circunstancias, adecuando el tamaño de la ventana y el tiempo entre bloques de datos para maximizar el *trhoughput*.

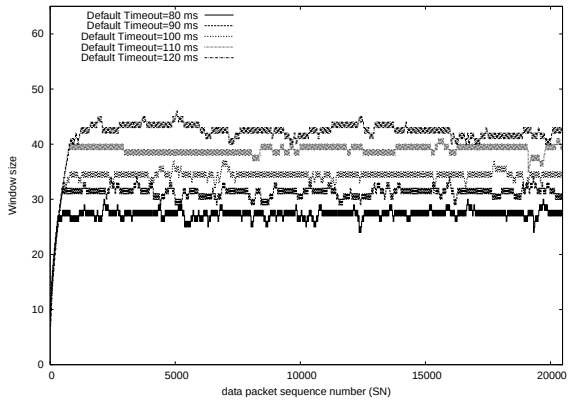

Fig. 2. Evolución del tamaño de la ventana para valores de *default_*T_{out}: 80 ms, 90 ms, 100 ms, 110 ms y 120 ms.

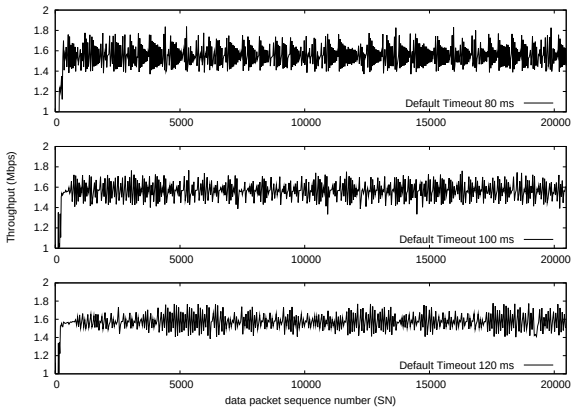

Fig. 3. Evolución del *throughput* instantáneo para valores de *default_*T_{out}: 80 ms, 100 ms y 120 ms.

Agradecimientos

Este trabajo ha sido subvencionado por el proyecto nacional ARPaq (TEC2004-05622-C04-02/TCM).

Referencias

[1] Ivan Stojmenovic, Thomas Kunz, “Handbook of wireless networks and mobile computing”, John Wiley & Soons, Chapter 23, pp 495-507, ISBN 0-471-41902-8, 2002.

[2] Tet Suya Oh-ishi, Koji Sakai, Kazuhiro Kikuma, “Study of the relationship between Peer-to-Peer systems and IP multicast”, IEEE Communications magazine, Vol. 41, No 1, pp. 80-84, January 2003.

[3] P. Manzanares-Lopez, J. C. Sanchez-Aarnoutse, J. Malgosa-Sanahuja, J-Garcia-Haro, “Empirical and Analytical Study of a Multicast Synchronous Transport Protocol for Intra-Campus Replications Services”, International Conference on Communications (ICC-04), ISBN 0-7803-8533-0 (cdrom), Paris, June 2004.

[4] http://www.tcpdump.org

[5] Steven McCanne, Van Jacobson, “The BSD Packet Filter: a New Architecture for User-level Packet Capture”, in Proceedings of the conference on Application, Technologies, Architectures and Protocols for Computer Communications, pp. 162-173, ISBN 0-146-4833, Cambridge (Massachusetts), USA, August 1998.