Estudio de un Router Software para la implementación de una Pasarela Residencial

Jaime García, Francisco Valera, David Díez, Hugo Gascón, Carmen Guerrero, Arturo Azcorra
Departamento de Ingeniería Telemática. Universidad Carlos III de Madrid
Avda. de la Universidad, 30.
28911 - Leganés (Madrid)
E-mail: jgr, fvalera, ddiez, hgascon, guerrero, azcorra@it.uc3m.es

Abstract *Residential Gateways (RGW) are the last (or first, it depends on the point of view) element in the access network and the interface between Home Network and access provider. In MUSE [1] our work is related to the design of a RGW compliant with QoS requirements and to the development of a RGW prototype. In this paper we propose a new model to design RGW, based on the Click modular router [2], but using user level applications too. This is what we call an Hybrid Model due to the capacity to create new functionalities at different levels: as a normal application or extending Click. This capacity helps the developer to extend the RGW device. A complete set of tests validations are presented to support this innovative idea.*

1. Introducción

El proyecto europeo MUSE [1] tiene como misión la especificación de la arquitectura y el modelo de la futura red de datos de banda ancha para Europa, con la idea de crear un acceso de bajo coste y alta calidad para los usuarios. Aunque las redes de acceso y central son unos puntos importantes de investigación, los dispositivos ubicados en los entornos residenciales son también componentes clave para poder proporcionar buenas calidades de servicio, especialmente los llamados Residential Gateway (RGW) o Pasarelas Residenciales.

La RGW es el primer elemento de la red accesible por el usuario como se muestra en la Fig. 1. Cada dispositivo del hogar estará conectado (de forma directa o inalámbrica) a través del RGW a una red de banda ancha, pero compartida. Por lo tanto, información de tiempo real como pueden ser datos de voz o video pueden estar compartiendo el mismo medio que otras conexiones menos prioritarias como conexiones web. Como se puede ver, algún tipo de *priorización* es necesaria antes de que los paquetes salgan a la red. Además, se necesita conformar el tráfico de los distintos flujos para administrar el ancho de banda disponible.

Además de todas estas funcionalidades, la RGW debe realizar muchas otras tareas por el usuario final: auto configuración, control, administración, configuración de los servicios, etc. Existen algunas funcionalidades que no se describirán en este artículo, como la señalización de la calidad de servicio (QoS). En MUSE se están estudiando diferentes modelos para la señalización y provisión de la QoS extremo a extremo, centrándose en la red de acceso. Una de las alternativas consideradas es IP Multimedia Subsystem (IMS) [3] para la configuración de los servicios. Realmente esta especificación tendrá que ser adaptada al escenario de redes de acceso fijas ya que hoy en día el IMS está sólo especificado para escenarios móviles y de redes WLAN. MUSE trabajará junto al ETSI-TISPAN [4] para realizar esta adaptación, en la que la RGW tiene un papel importante. El prototipo de la RGW en MUSE incorporará esta funcionalidad, no en las primeras etapas del desarrollo, pues el trabajo se tiene que hacer en paralelo con la estandarización del ETSI-TISPAN.

Se ha escogido utilizar el sistema operativo Linux para esta etapa de implementación del prototipo de la RGW que se ejecutará en un dispositivo PC compatible *i386* de reducidas dimensiones [5]. Debido a que la RGW debe manejar paquetes de bajo nivel (del nivel de enlace) y Linux no provee mecanismos para realizar esta manipulación de forma nativa, se ha decidido utilizar el Click modular router [2] para el tratamiento de las tramas de nivel de enlace. Es importante señalar que los resultados que aquí se presentan son extrapolables a otra herramienta software que trabaje al mismo nivel de Click, es decir, cualquier herramienta que capture tramas a nivel de enlace y sea capaz de enviarlas sin modificar al nivel de aplicación. En la sección 2 se hará una introducción a las principales características de Click así como a su elección como potencial herramienta de desarrollo. La sección 3 se dedica a la descripción del modelo propuesto para la implementación de la RGW. La sección 4 describe los escenarios de pruebas planeados y las pruebas ya realizadas. Por último, la sección 5 finaliza con las principales conclusiones extraídas de todas las pruebas realizadas.

[1] MUSE (Multi Service Access Everywhere) es un gran proyecto de I+D en las redes de banda ancha. Dentro del sexto Programa Marco, MUSE contribuye al objetivo estratégico de "Ancho de Banda para Todos" dentro de las IST (Information Society Technologies) y está parcialmente financiado por la Comisión Europea.

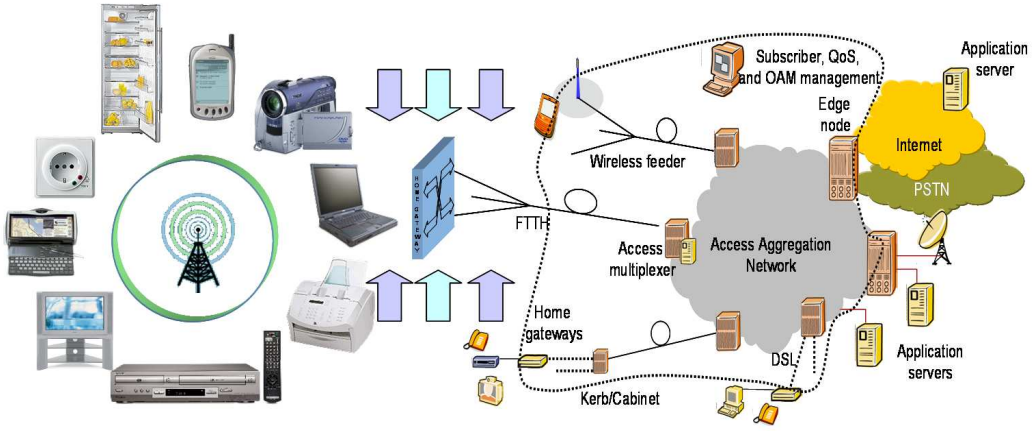

Figura 1: La red de banda ancha en MUSE

2. Plataforma Click

Click [6] es un *software router* modular desarrollado conjuntamente por el grupo LCS's Parallel and Distributed Operating Systems del MIT, Mazu Networks, el ICSI Center for Internet Research y últimamente UCLA. Un kernel de Linux ejecutando Click es capaz de actuar como un router de una forma flexible y configurable. Las tareas de encaminamiento se realizan de una manera extremadamente rápida para un software ejecutándose en un hardware común. En un Pentium III a 700 MHz, un router IP con Click puede manejar hasta 333.000 paquetes de 64 bytes por segundo [2].

Configurar un router con Click se basa en interconectar módulos llamados *elementos* que controlan cada aspecto de la operación del router: comunicación con los dispositivos, modificación de paquetes, colas, políticas de descarte, envío de paquetes, etc. Como la modularidad es la principal ventaja de Click, es posible escribir nuevos elementos en C++ con la funcionalidad requerida. La configuración del router resulta de la unión de elementos en un fichero de configuración usando un lenguaje simple propio de Click.

En Click se puede trabajar a nivel de *aplicación* o usando un *módulo del kernel* de Linux, cambiando ligeramente el funcionamiento del mismo. Se ha escogido la segunda opción para la realización del prototipo, ya que con ella se permite la manipulación de tramas de una manera más rápida, eliminando los retardos introducidos por la pila de protocolos TCP/IP. Actualmente Click sólo funciona en kernel de Linux de la versión 2.2 y 2.4 aunque se espera que pronto sea implementado para las versiones 2.6. Como es un proyecto de software abierto, muchos desarrolladores crean nuevas funcionalidades y varios están trabajando para soportar el kernel 2.6 y añadir nuevas funcionalidades para IPv6.

3. Modelo Híbrido Click / Aplicación

Para la implementación del prototipo de la RGW en el proyecto MUSE se necesita un software capaz de capturar todos los paquetes de la capa de nivel de enlace, modificarlos y volver a enviarlos a la red; enviarlos al nivel de aplicación, etc. Por lo tanto, se decidió utilizar Click (exactamente el módulo kernel de Click). Como hemos visto en el apartado 2, en Click se encuentran implementadas varias funcionalidades, pero en nuestro desarrollo necesitaremos crear nuevas. Existe la posibilidad de crear nuevos elementos de Click que se integren con los que actualmente existen. Otra posibilidad sería capturar tramas a nivel de Click y enviarlas todas al nivel de aplicación para ser procesadas ahí. Realizarlas de una forma u otra presenta los siguientes problemas:

1. Programar nuevos elementos a nivel de kernel es, en ocasiones, considerablemente complejo y más aún cuando se trata de aplicaciones con una gran carga de componentes de servicios de red.

2. La programación de nuevas aplicaciones hardware o software pueden ser necesarias en el futuro y deben crearse independientes de la plataforma cuando sea posible (desarrollándolas en Java, por ejemplo).

Sin embargo es evidente que trabajar a nivel de aplicación presenta además de problemas de eficiencia, problemas técnicos importantes cuando hay que gestionar los niveles más bajos de la torre de protocolos. Para solventar estos problemas, se decidió crear un nuevo *modelo híbrido* donde no se usa un modelo puro de Click o de aplicación, sino que se crean nuevas implementaciones de software en el nivel o capa más conveniente (como un elemento de Click o como una aplicación, según sea mejor). La Fig. 2 muestra el *modelo híbrido* presentando tres bloques principales:

Click es la implementación de Click trabajando a nivel de kernel. Recibirá cada paquete, comprobará el tipo de paquete, y lo enviará directamente por otros elementos de Click o lo enviará hacia el nivel de aplicación según tenga configurado.

El Manager recibirá paquetes *nuevos* del módulo Click y los procesará. Dependiendo de las características del paquete, el Manager puede configurar el módulo Click para que envíe paquetes con las mismas características a otro proceso de nivel de aplicación.

Procesos P1-Pn son las aplicaciones de nivel de aplicación desarrollados para realizar ciertas funciones.

Esta propuesta de modelo debe ser validada primero para comprobar que su rendimiento es aceptable para el funcionamiento de un RGW. Cuando se realizan todas las operaciones al nivel de Click nos ahorramos el tiempo de tránsito de un paquete por la pila TCP/IP del kernel de Linux. En el modelo híbrido propuesto, este tiempo de tránsito debe ser otra vez tenido en cuenta ya que algunos paquetes (paquetes de señalización por ejemplo) se enviarán desde el nivel Click al nivel de aplicación. Debido a esto es imprescindible estudiar el retardo impuesto por enviar una trama desde el nivel de Click al Manager puesto que este retardo limitará y marcará las posibilidades que tiene esta plataforma como base del desarrollo de la pasarela residencial para un entorno de acceso a banda ancha como el planteado en el proyecto MUSE.

4. Validación del Modelo Híbrido

4.1. Retardo de procesamiento de paquetes

Con esta prueba se va a medir el retardo adicional que supone enviar los paquetes desde el nivel

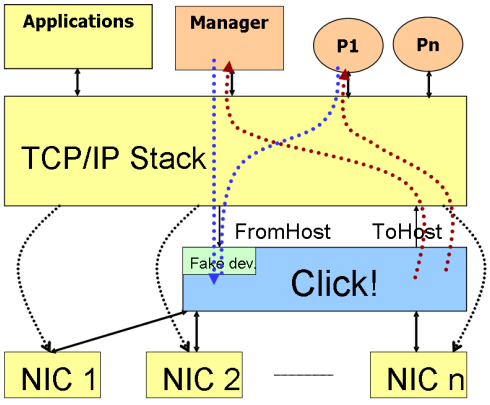

Figura 2: Modelo híbrido

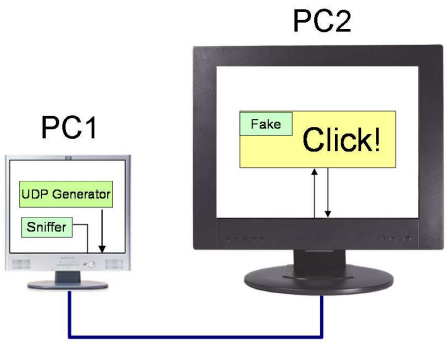

Figura 3: Conexión directa

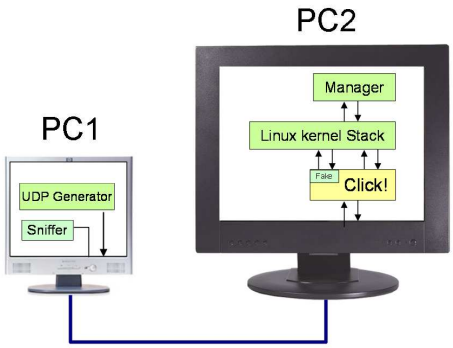

Figura 4: Conexión con Manager

Click hacia la aplicación Manager y de vuelta a Click.

Se crearon dos escenarios para comparar estos valores (Figs. 3 y 4). En ambos escenarios se utilizan dos ordenadores directamente conectados. Un ordenador actúa de generador y el otro ejecuta Click que, en cada escenario, realiza una función diferente. En el primer escenario denominado *conexión directa* el ordenador con Click cambia las cabeceras Ethernet e IP para devolver inmediatamente el paquete al ordenador origen. En el otro escenario (*conexión con Manager*) el nivel Click encapsula la trama entera en un paquete UDP y lo envía hacia la aplicación Manager que lo devuelve inmediatamente al nivel Click que realiza el mismo cambio de cabeceras que en el escenario anterior. En cada escenario se realizaron experimentos con diferentes tamaños de paquetes y se obtuvo la media de dichos experimentos. El resultado final se puede ver en la Tabla 1.

Como resultado de estas pruebas se puede concluir que la diferencia del retardo es independiente de la longitud del paquete y además que esta diferencia está tan sólo entre los 130 y 140μs. Hay que mencionar finalmente que estas pruebas se realizaron en un ordenador *compacto* con las siguientes especificaciones:

- Placa Lex Solution SV860A
- Procesador VIA C3 533MHz
- 512 Mbytes de RAM
- 2 tarjetas de red 10/100 BaseTx

Tamaño de paquete	Conexión directa	Conexión con Manager	Diferencia
100 bytes	$120\mu s$	$250\mu s$	$130\mu s$
540 bytes	$200\mu s$	$330\mu s$	$130\mu s$
1060 bytes	$290\mu s$	$430\mu s$	$140\mu s$
1440 bytes	$365\mu s$	$500\mu s$	$135\mu s$

Tabla 1: Retardo introducido por el modelo híbrido

- 1 tarjeta de red 10/100/1000 BaseTx
- 1 tarjeta de red inalámbrica Atmel 802.11b

Es importante destacar que *ToHost* fue el elemento de Click utilizado para las pruebas con el Manager. El elemento *ToHost* envía un paquete que recoge en su entrada hacia la pila TCP/IP de Linux. *ToHostSniffers* es otro elemento similar a *ToHost* que envía tramas hacia aplicaciones que leen directamente de los interfaces de red como haría la librería *libpcap* [7]. Debido a que en Java no existe un soporte nativo para leer directamente de un interfaz de red, se tuvieron que realizar pruebas con otras APIs que implementan una funcionalidad similar a la de *libpcap* para poder probar el elemento *ToHostSniffers* que disminuiría el tiempo de tránsito de un paquete al no tener que atravesar la pila TCP/IP de Linux. Las dos APIs probadas ([8] y [9]) se denominan *jpcap* y no mejoran los resultados dados por *ToHost*. Los resultados obtenidos se muestran en la Fig. 5. Se han eliminado los resultados de la API [8] ya que son mucho peores que los de [9].

Estos resultados pueden deberse a la forma de implementar las APIs de Java ya que hacen uso de la librería *libpcap* como paso intermedio. Por ello, para las siguientes pruebas sólo se programará el Manager para que trabaje con el elemento *ToHost*.

4.2. Carga soportada por el modelo

Con este escenario se desea probar si el uso de una aplicación en el nivel de aplicación, llamada el Manager, aumenta o no considerablemente el retardo de tránsito (si realmente se reduce el rendimiento utilizando un Manager se podría realizar las mismas funciones en el propio nivel de Click, pero esto reduciría la flexibilidad del desarrollo y complicaría el mismo). Para estas pruebas, se instaló Click en el ordenador *compacto* mencionado en apartados anteriores.

En el escenario de pruebas se conectaron dos ordenadores de diferentes redes IP a través de este ordenador que denominamos RGW. Este escenario se puede ver en la Fig. 6.

En estas pruebas se desea calcular el *throughput* y el *jitter* obtenidos en el equipo *servidor* al enviar paquetes desde el terminal *cliente*. Para comparar prestaciones, se configuró el equipo *RGW* para que realizase funciones de NAT (Network Address Translation). Se implementó el NAT de tres formas distintas para poderlas comparar:

1. Linux con IPTables. Para realizar esta prueba se necesita compilar el núcleo de Linux para incluir soporte para IP Tables. Una vez compilado, tan sólo es necesario activar el servicio de NAT con la instrucción *iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE* También es necesario activar el *ip_forwarding* activando el bit del fichero */proc/sys/net/ipv4/ip_forward*.
2. NAT implementado en Click. En Click existen elementos que permiten realizar un NAT a este nivel.
3. NAT implementado en una aplicación (Manager). En estas pruebas el nivel Click envía todos los paquetes (utilizando el elemento *ToHost* de Click, como se comentó anteriormente) a una aplicación Java que realiza el NAT.

En las pruebas se ha utilizado el programa *iperf* [10] para la generación de tramas y obtención de estadísticas de las mismas. Existe un programa *cliente* que genera tramas a un ordenador donde se esté ejecutando una instancia *servidor* del programa *iperf*. Este programa *servidor* captura tramas y genera estadísticas del *throughput* y del *jitter*. La ventaja del *iperf* es que en el programa cliente podemos especificar la duración de la simulación, el tamaño de paquete y la tasa de generación de tramas.

Se realizaron las pruebas para los escenarios mencionados anteriormente y se utilizaron paquetes de 1470, 850 y 200 bytes para tener tamaños representativos. Los resultados se pueden ver en la Fig. 7. Se han omitido los resultados del escenario con IPTables ya que son idénticos a los del escenario con Click.

Los resultados más importantes que podemos extraer de estas pruebas son los siguientes:

- La máxima tasa de generación de paquetes depende del tamaño de dichos paquetes. En el caso de utilizar paquetes de 1470 bytes, la tasa máxima que puede generar el equipo *cliente* es

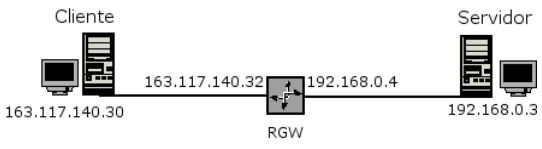

Figura 6: Primer escenario de pruebas

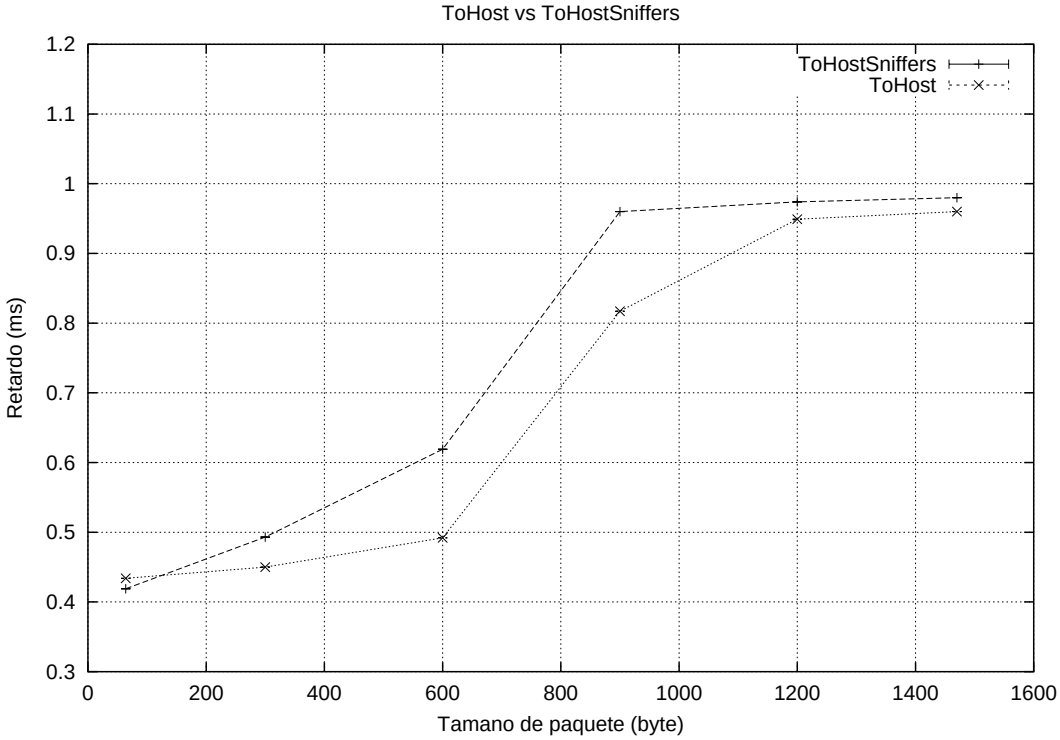

Figura 5: Resultados con la API jpcap

de 95,1 Mbps. Cuando se transmiten paquetes de 850 bytes ese máximo se encuentra en 92,7 Mbps mientras que para paquetes de 200 bytes no se pueden emitir más de 75,1 Mbps.

- Los escenarios de NAT con Click y NAT con IPTables dan resultados prácticamente idénticos.

- El escenario donde se realiza el NAT a nivel de aplicación (Manager) nos muestra resultados parecidos a los anteriores para paquetes mayores de 850 bytes y tasa de generación menor de 40 Mbps. Cuando nos salimos de estos valores el rendimiento ya no es tan bueno.

A la vista de estos resultados, es importante reseñar que estas pruebas están orientadas a comprobar el funcionamiento del modelo híbrido, donde algunas tramas pasarán del nivel Click al de aplicación. Se debe reafirmar el hecho de que los paquetes de datos atravesarán el nivel de Click sin "subir" al de aplicación por lo que sólo tramas de señalización y los primeros paquetes de flujos nuevos tendrán que ser procesados en dicho nivel.

Viendo los resultados mostrados en la Fig. 7 se puede ver que el tamaño de los paquetes de señalización, que han de subir al Manager y ser procesados de manera diferente a los paquetes de datos, influye en el throughput conseguido en el modelo híbrido. Si, como hemos indicado anteriormente, se considera señalización para la configuración de los servicios mediante el protocolo SIP (Session Initiation Protocol) [11] (que es el protocolo de señalización utilizado en IMS), la pregunta inmediata es qué tamaño medio tendrán estos paquetes que deben ser procesados por una aplicación y cuál será su tasa de generación. Considerando, en el caso peor, tamaños medios de mensajes de SIP pequeños del orden de unos *465 bytes* para el caso de mensajes INVITE o *388 bytes* para mensajes OK, nos situamos en estas pruebas siempre con tasas no superiores a 40 Mbps. En situaciones más complejas, donde las cabeceras de los mensajes SIP pueden ser más extensas, se debe de considerar, como se recoge en [12], que los mensajes de SIP pueden estar entre los *838* y *1024 bytes*. Lo que se indica como tal en [11], que estandariza SIP, es que la longitud de sus mensajes no superen el valor de la MTU (Maximum Transmission Unit) si es previamente conocida, o 1200 bytes en su defecto.

De todas formas, una prueba interesante es cambiar el equipo que realiza las funciones del RGW por otro más potente de *sobremesa*, para comprobar el efecto del procesador en estas pruebas. Para ello se utilizó un ordenador Pentium 4 con 2,4 GHz y 512 Mbytes de memoria RAM (igual que el ordenador compacto) en vez del *compacto* que se utilizará como RGW. En la Fig. 8 se pueden observar los resultados obtenidos. En esta gráfica también se han eliminado los resultados del escenario que se utiliza IPTables como NAT.

Cabe destacar la mejora con respecto a las pruebas realizadas con el ordenador *compacto* por lo que se puede concluir que el procesador es un elemento que definitivamente influye en el caso de relegar el procesamiento de las tramas al nivel de aplicación.

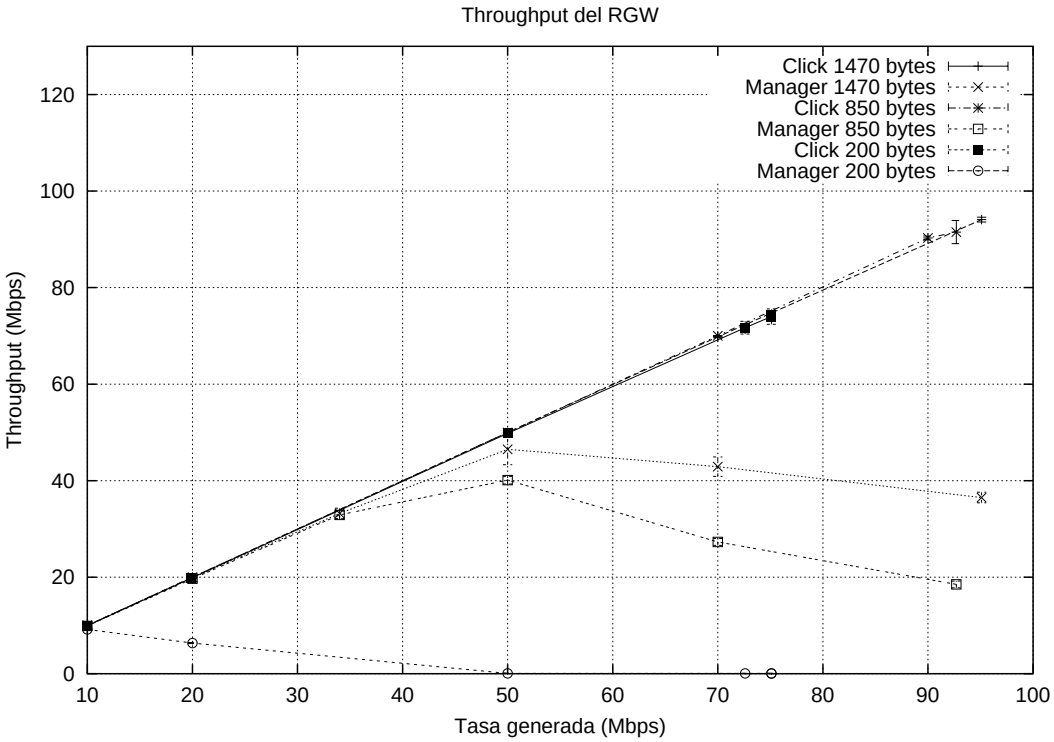

Figura 7: Resultados usando el ordenador compacto.

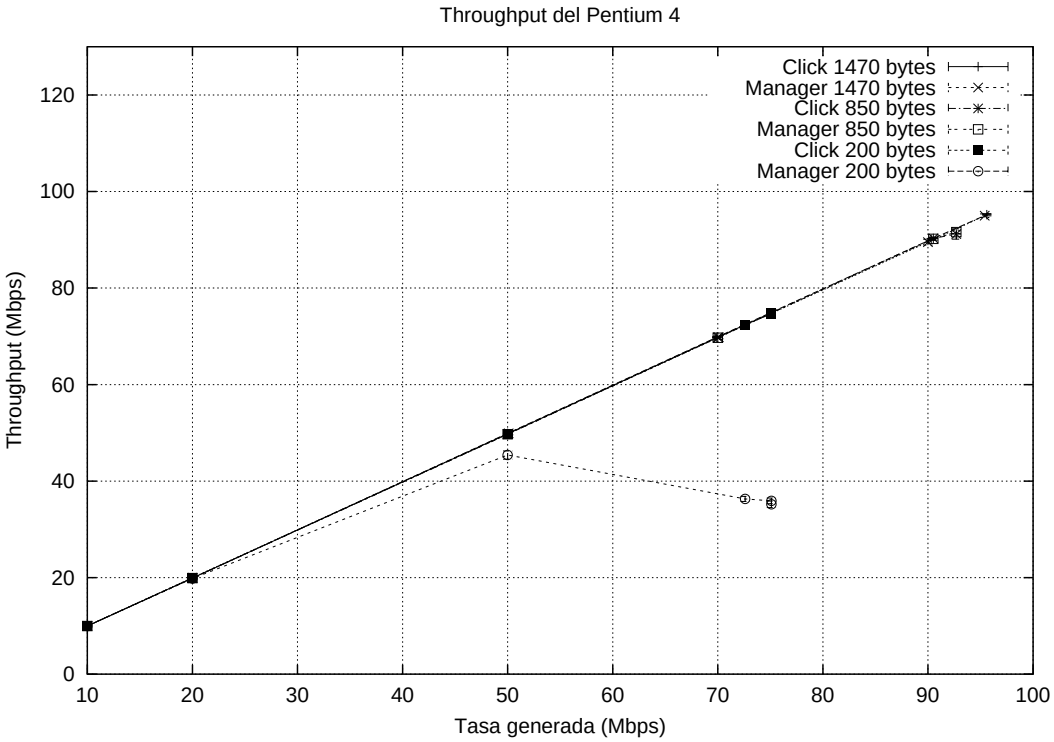

Figura 8: Resultados usando el ordenador de sobremesa.

5. Conclusiones

En el artículo se han presentado las diferentes pruebas realizadas con la plataforma de desarrollo de routers modulares Click con el objetivo de validar su utilización dentro del proyecto MUSE.

Este artículo propone la utilización de un modelo híbrido que se ha probado en diferentes escenarios en donde se han planteado escenarios reales utilizando los equipos hardware que se planean utilizar en el prototipo final.

Entre los diferentes resultados cabe destacar el

obtenido al probar la carga soportada en las diferentes configuraciones en donde queda patente la importancia que tiene un procesador potente en el equipo RGW y el tamaño de los paquetes que deben ser procesados en el nivel de aplicación, es decir, de los paquetes que el nivel de Click enviará hacia el nivel de aplicación para ser tratados por el Manager. En el artículo se ha demostrado como incluso con un ordenador que no tenga demasiados recursos, será posible tratar los mensajes de señalización que habitualmente son mensajes pequeños soportando un caudal más que suficiente. Una vez terminado el proceso de señalización, el tráfico de usuario será tratado ya a nivel de Click y en las diferentes pruebas se ha comprobado que a este nivel no hay perjuicio apreciable de prestaciones.

A la hora de tratar los mensajes al nivel de aplicación, otro resultado importante es el obtenido al comparar los elementos de Click *ToHost* y *ToHostSniffers* (mecanismos habituales para transferir los datos a niveles superiores). Las librerías *jpcap* de Java estudiadas no presentan ninguna ventaja con respecto a los *sockets* normales de Java, por lo que se ha estimado que el mecanismo habitual de envío de estos datos (*toHost*) es suficiente como para tratar estos flujos y además presenta un menor nivel de complejidad. Queda como trabajo futuro la realización de las mismas pruebas, esta vez con un Manager programado en lenguaje C, donde se pueden utilizar las funciones de la librería *libpcap*.

Agradecimientos

Este artículo ha sido financiado parcialmente por la Comisión Europea a través del proyecto MUSE.

Referencias

[1] Multi service access everywhere (muse) european project. [Online]. Available: http://www.ist-muse.org/

[2] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The click modular router.acm transactions on computer systems," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, 2000.

[3] *IP Multimedia Subsystem (IMS)*, 3GPP TS 23.228 V6.6.0, Rev. Stage 2 (Release 6), 2004.

[4] "Release 1: Release definition," TISPAN: Draft ETSI, 2004.

[5] Lex system. [Online]. Available: http://www.lex.com.tw:8080/ home.htm

[6] Página web de click. [Online]. Available: http://www.pdos.lcs.mit.edu/click/

[7] N. R. G. Lawrence Berkeley National Labs. libpcap. [Online]. Available: http://www.tcpdump.org/

[8] Jpcap: Java package for packet capture. [Online]. Available: http://netresearch.ics.uci.edu/kfujii/jpcap/doc/

[9] Jpcap: a network packet capture library. [Online]. Available: http://jpcap.sourceforge.net/

[10] Iperf. [Online]. Available: http://dast.nlanr.net/Projects/Iperf/

[11] J. Rosemberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," IETF RFC 3261, 2002.

[12] H. Schulzrinne and J. Rosemberg, "The session initiation protocol: Internet-centric signalling," *IEEE Communications Magazine*, vol. 38, no. 10, 2000.