Caracterización de los enlaces de Internet utilizando tecnología de Redes Activas

Marifeli Sedano[1], Bernardo Alarcos[1], María Calderón[2] y David Larrabeiti[2]

[1] Área de Ingeniería Telemática.
Dpto. Automática
Escuela Politécnica. Universidad de Alcalá.
Carretera Madrid-Barcelona, Km 33,600 - 28871
Alcalá de Henares (Madrid)
E-mail: {marifeli, bernardo}@aut.alcala.es

[2] Dpto. de Ingeniería Telemática.
Universidad Carlos III.
Avd. de la Universidad, 30 – 28911
Leganés (Madrid)
E-mail: {maria, dlarra}@it.uc3m.es

Abstract. *This paper presents the design, implementation and trials of **a-clink**, which is a hop-by-hop performance estimation tool based on active networks. The paper begins by analyzing different alternatives of hop-by-hop performance estimation tools: pathchar, clink, pchar and nettimer. Based on this analysis, several deficiencies are identified on the different tools. In order to improve the efficiency and accuracy of the estimations, one of the tools is selected, clink, to design an extension based active network technology. This extension, a-clink, has been implemented over the public domain active network platform SARA. The implementation of a-clink has been trialed on a simple active network prototype spanning two universities connected through public Internet, and its results compared with those obtained by the original clink. The paper concludes describing the advantages of the active version of clink over the conventional passive performance estimation tool.*

1 Introducción

La topología densamente interconectada de Internet y la tecnología IP hacen de Internet una red bastante robusta, por lo tanto, es muy poco frecuente que se pierda la conectividad a través de la red. Sin embargo, el dinamismo y la complejidad de la red, tanto en la configuración de las rutas, como en la distribución del tráfico en múltiples enlaces, hace que las prestaciones sean difíciles de predecir y fuertemente variables.

En este entorno comienza a ser imprescindible disponer de herramientas de diagnostico que sean capaces de determinar las características, en cuanto a prestaciones y posibles cuellos de botella, del camino entre un origen y un destino. Dicho análisis ayudará a los administradores y usuarios de la red a detectar donde se encuentran los cuellos de botella que están causando situaciones indeseables, y en general a conocer las prestaciones que se pueden obtener para una determinada comunicación.

Entre las herramientas de diagnostico más utilizadas en la actualidad podemos encontrar el *traceroute*. Dicha herramienta permite averiguar el número y dirección de los nodos intermedios por los que pasarán los paquetes en la comunicación entre un origen y un destino, proporcionando adicionalmente datos de retardo de tránsito desde el origen a cada uno de los nodos intermedios. Dicha herramienta fue diseñada con el objetivo principal de determinar el camino que siguen los paquetes en Internet.

En 1997 Van Jacobson [1] ante los problemas de congestión que ya comenzaban a materializarse en la red, desarrollo una nueva herramienta de diagnostico de redes, el *pathchar*, que permite a un usuario determinar entre otras características el ancho de banda y retardo en cada salto entre un origen y un destino.

Esta herramienta, que ha sido la base de la mayoría que se han planteado posteriormente[2, 3], adolece de una serie de problemas entre los que cabe destacar: relación señal/ruido baja, los errores de las estimaciones se propagan, y sobrecarga de la red. Las posteriores herramientas que se ha propuesto han ido modificando los mecanismos de cálculo de las estimaciones y resolviendo parte de los problemas que presentaba el *pathchar*. Pero hay dos problemas principales que siguen sin ser resueltos como son la propagación de los errores en las estimaciones de un salto a los siguientes y la sobrecarga que se origina en la red.

Por otro lado, la tecnología de redes activas [4, 5, 6] se ha propuesto como evolución de los modelos de red tradicionales. La idea fundamental es añadir programabilidad a las redes. Las redes activas constituyen una arquitectura de red en la que los nodos de la misma pueden realizar procesamiento a medida sobre los paquetes que los atraviesan. Las redes activas proporcionan un cambio en el paradigma de red: de nodos capaces exclusivamente de transportar octetos de forma pasiva, a nodos capaces de procesar los paquetes a cualquier nivel de la pila de protocolos.

Las redes activas introducen el concepto de procesamiento específico de los paquetes en base a código móvil que se ejecuta en los nodos de la red. Esto quiere decir que los nodos de la red no son sistemas de procesamiento especializados en un

* Este trabajo ha sido financiado por la CICYT dentro del proyecto TEL99-0988-C02-02

protocolo de red dado (para el caso de redes multiprotocolo en un limitado número de ellos), como sucede en la actualidad, sino que son plataformas de ejecución genéricas en las que se puede descargar dinámicamente código específico para el procesamiento de los distintos tipos de paquetes que se desee definir.

En este artículo se plantea la aplicación de la tecnología de redes activas al desarrollo de herramientas de diagnóstico de redes, con el objeto de solucionar parte de la problemática actual que presentan este tipo de herramientas. En concreto, se propone una herramienta activa, *a-clink*, que proporcionará una mayor precisión en las estimaciones evitando la propagación de errores, una mayor rapidez en los cálculos y una limitada sobrecarga de la red. Los motivos que justifican la aplicación de la tecnología de redes activas se basan en la conveniencia de disponer en la red de elementos programables que realicen un análisis de las características de la red distribuido y de alcance limitado.

El resto del artículo se ha organizado de la siguiente forma. Primero analizaremos como han ido evolucionando las herramientas de diagnostico de redes y su problemática actual, para a continuación describir la herramienta de diagnostico activa, *a-clink*, que presentamos en este artículo. Mediante dicha herramienta se demostrará como la tecnología de redes activas puede ayudar a mejorar el comportamiento de dichas herramientas. Seguidamente describiremos la implementación que hemos realizado sobre la plataforma de red activa SARA (Simple Active Router-Assistant architecture) [7] y los resultados que hemos obtenido en las pruebas realizadas. Terminaremos con las conclusiones obtenidas y las líneas de futuros trabajos.

2 Evolución de las herramientas de evaluación de prestaciones de red

En este apartado vamos a describir distintas herramientas que se han desarrollado en los últimos años para medir las características de los enlaces de la red, analizando las ventajas e inconvenientes de cada una de ellas. Aunque no pretende ser un listado completo de todas las aplicaciones, si se intenta mostrar una selección de las más significativas. Finalmente seleccionaremos una de ellas como base de nuestra aplicación.

2.1 Traceroute

La primera herramienta a la que haremos referencia es *traceroute* desarrollada por Van Jacobson en 1988. Su funcionamiento se basa en el envío de una secuencia de paquetes IP desde una fuente a un destino incrementando sucesivamente el valor del campo TTL de la cabecera IP. Los routers por los que pasan los paquetes decrementan el valor TTL y si alcanza el valor 0, descartan el paquete y envían al emisor un mensaje de error ICMP. El emisor aprovecha estos mensajes de error para averiguar la dirección IP de los routers que hay en el camino entre el origen y el destino y su localización en número de saltos desde el origen (igual al valor TTL de salida). El emisor también calcula el rtt (round trip time), tiempo desde que sale el paquete IP hasta que llega el mensaje ICMP del router. Esta herramienta únicamente nos permite averiguar de forma fiable el camino seguido desde una fuente hasta llegar a un destino.

2.2 Pathchar

Desde 1991 Van Jacobson trabaja en el desarrollo de herramientas que además de descubrir rutas obtengan información sobre las características de los enlaces que componen el camino entre una fuente y un destino. En 1997 lanza una herramienta denominada *pathchar* [1] que muestra el camino entre una fuente y un destino y parámetros del estado de cada enlace en el camino: rtt y ancho de banda.

Basada en el funcionamiento del *traceroute*, *pathchar* envía una secuencia de paquetes a un destino. Para cada router que se encuentra en el camino (manteniendo un determinado valor de TTL) realiza una serie de pruebas que consisten en enviar una secuencia de paquetes de distintos tamaños. De esta forma mide el rtt total desde el origen a los distintos routers del trayecto en función del tamaño de los paquetes (Fig. 1).

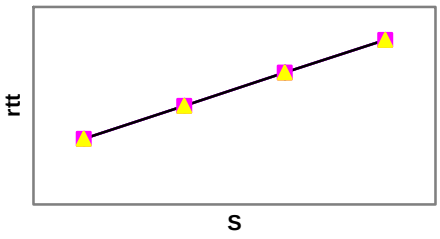

Fig. 1: Valor de rtt en función del tamaño de los paquetes S

Para expresar analíticamente esta curva, que se puede observar que se aproxima a una recta cuando se toman los valores de rtt(S) menores, podemos utilizar el modelo de la Fig. 2 en la que se observa el flujo de los datos en un enlace entre dos routers.

Los retardos que sufre un paquete en este escenario son:

1) Tiempos de propagación en cada uno de los enlaces (lo denominamos latencia).
2) Tiempos de transmisión de los paquetes a la velocidad nominal del enlace.
3) Tiempos de procesamiento de los paquetes en los sistemas finales y en los routers.
4) Tiempo de espera en las colas de los routers y de los sistemas finales.

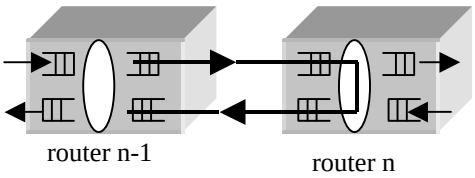

Fig. 2: Modelo del camino recorrido por la información.

Todos estos parámetros se pueden considerar deterministas excepto el tiempo de espera en las colas, que tiene un carácter aleatorio. Para eliminar este tiempo se recurre a un método de repetición. Consiste en repetir las pruebas con cada tamaño de paquete un número suficiente de veces para poder tener una alta probabilidad de que una de las muestras no haya sufrido retardo de colas. Para dibujar la recta se utiliza un filtro que selecciona el valor mínimo de cada batería de repeticiones.

Una simplificación de este modelo en el que no se consideran: los tiempos de procesamiento en los sistemas finales y routers, el tiempo invertido en transmitir el mensaje ICMP de vuelta, y los tiempos de espera en cola, es expresada en la ecuación 1:

$$rtt(s) = \sum_{i=1}^{n} 2lat_i + \frac{S}{B_i}$$

Ecuación 1. Modelo matemático simplificado.

Se representa el rtt desde el origen al router n teniendo en cuenta el retardo introducido en cada uno de los enlaces. Donde S es el tamaño del paquete, B_i es el ancho de banda del enlace i y lat_i representa el retardo de propagación del enlace i. Con esta simplificación lo que obtenemos es la ecuación de una recta de pendiente 1/B. Donde podemos estimar el valor de ancho de banda B.

Para realizar los cálculos de cada enlace se utiliza un método de cálculo regresivo (Fig. 3). Se comienza realizando los cálculos del primer enlace (n = 1), aplicando directamente la ecuación 1 y obteniendo el rtt en función de S para el dicho primer enlace. De la recta obtenida se mide:

1. **Latencia** igual a la mitad del valor de la recta cuando S = 0.
2. **Ancho de banda** igual a la inversa de la pendiente de la recta $1/B_1$.

Para el siguiente enlace realizamos los mismos cálculos pero basándonos en los resultados del enlace anterior. Es decir, la latencia será la diferencia entre la latencia medida (total) y la latencia del enlace anterior.

De la misma forma el ancho de banda del siguiente enlace vendrá dado por la diferencia de pendientes de las curvas actual y la del enlace anterior. De esta forma se van calculando sucesivamente los parámetros para los distintos enlaces hasta llegar al destino.

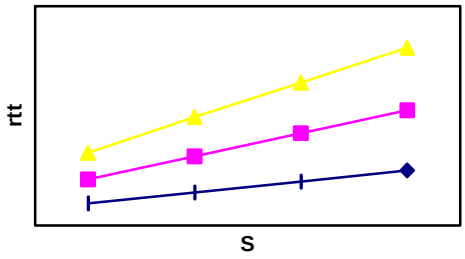

Fig. 3: Curvas de rtt(S) para distintos enlaces.

Los principales problemas de *pathchar* son:

1. Relación señal/ruido baja: en algunos casos el tiempo de transmisión es mucho menor que el de propagación (1500 octetos sobre alta velocidad T3, OC-3 u OC-12). En estos casos las fluctuaciones en las medidas de retardo debidas a tiempos de cola pueden llegar a ser 100 veces menores que el retardo de propagación. Esto impide obtener valores precisos en las medidas.
2. Amplificación de ruido: los errores en las estimaciones se propagan al salto siguiente en el cálculo regresivo.
3. No funciona bien ante enlaces con múltiples canales que reparten el ancho de banda total del enlace en canales paralelos de menor ancho de banda.
4. Limitaciones de mensajes ICMP por temas de seguridad: puesto que la aplicación utiliza mensajes ICMP encontrará problemas en sistemas como Linux y Solaris que limitan la velocidad de envío de estos mensajes. También se pueden encontrar rutas que pasan por routers o cortafuegos que filtran estos mensajes.
5. Sobrecarga en la red: debido a la gran cantidad de pruebas que se deben realizar y a que todos los paquetes se mandan desde la fuente a cada router.

2.3 Clink

En 1999 Allen B. Downey desarrolla una nueva herramienta denominada *clink* [2]. Basada en *pathchar*, utiliza las mismas técnicas de recolección de datos, presentando algunas mejoras en las técnicas de cálculo del ancho de banda con las que consigue estimaciones más precisas.

Al mismo tiempo Downey realiza un estudio para averiguar si es más eficiente aumentar el número de tamaños de paquete (más precisión en la curva) o el número de paquetes por tamaño (más posibilidades de encontrar el mínimo rtt). Concluyendo que es mejor aumentar la variación en el número de tamaños.

En la aplicación que implementa realiza por cada salto 8 pruebas por cada uno de los 93 tamaños distintos de paquete (28 y 1500 octetos con saltos de 16). Esto supone reducir la carga que se introduce en la red a prácticamente la mitad respecto a *pathchar*.

Con el objetivo de limitar el numero de muestras al estrictamente necesario para obtener una determinada precisión en las medidas, Downey propone realizar una recogida de datos adaptativa que consiste en ir aumentando progresivamente el número de muestras por cada tamaño y recalculando el rtt y ancho de banda hasta que se vea que los valores estimados convergen.

2.4 Pchar

También en 1999 Bruce A. Mah desarrolla *pchar* [3] una nueva herramienta basada en *pathchar*. Como principal mejora soporta algoritmos alternativos con el objetivo de tener una mayor precisión en el cálculo del rtt y ancho de banda. En concreto permite elegir entre dos algoritmos: 1) mínimos cuadrados o 2) método basado en tests estadísticos de Kendall.

Con esta herramienta se consiguen resultados similares a los obtenidos con *clink*, con el agravante de que es más agresiva en cuanto a tráfico generado.

2.5 Nettimer

En el año 2000 Kevin Lai y Mary Baker proponen una nueva técnica denominada P*acket Tailgating* y la desarrollan en una herramienta denomina *nettimer*[8]. En esta herramienta se utiliza un nuevo modelo denominado Pair-Packet, que consiste en enviar dos paquetes seguidos, uno muy pequeño y otro muy grande, que permitan medir diferencias de tiempos de llegadas en el receptor. A partir de estos datos se pude estimar el ancho de banda del cuello de botella.

La principal ventaja de este método es que las pruebas son menos agresivas que las propuestas por Jacobson, al tener que enviar una cantidad considerablemente menor de paquetes. Obteniéndose resultados con precisiones similares a las de las aplicaciones anteriores.

Como limitación principal podemos destacar que no obtiene buenos resultados si en el camino existe un enlace muy rápido después de uno lento. En concreto la relación entre el enlace rápido y el lento debe ser superior a 37.5.

2.6 Selección de herramienta base

En este artículo planteamos como la tecnología de redes activas puede ayudar al desarrollo de este tipo de herramientas de evaluación de prestaciones de red. Para lo cual, partiendo de una de estas herramientas desarrollaremos una herramienta activa, que aporta las ventajas que proporciona la tecnología de redes activas. A la hora de decidir la herramienta base de entre las analizadas, nos hemos basado en el estudio comparativo que se realiza en [8]. Dicho estudio se realiza en base a dos aspectos:

1. **Estimación de ancho de banda de los enlaces**: no destaca considerablemente ninguna herramienta con respecto a las demás.
2. **Número de paquetes enviados en las pruebas**: *nettimer* es la menos agresiva seguida por *clink*, que en pruebas referenciadas es entre 2 y 6 veces más agresiva, y finalmente *pathchar* y *pchar* que son el doble de agresivas que *clink*.

Por lo tanto se puede decir que *clink* es más agresiva que *nettimer*, pero dado que *nettimer* tiene la limitación de no obtener buenos resultados sobre enlaces rápidos seguidos de lentos, situación que se da en Internet, y que en nuestra propuesta pretendemos reducir la sobrecarga que introducen estas herramientas en la red, hemos decidido seleccionar *clink.*

3 Aplicación activa a-clink

La aplicación activa *a-clink* se basa en utilizar la tecnología de redes activas para resolver parte de los problemas que presentan las actuales herramientas de evaluación de prestaciones. En concreto, se ha visto que uno de los principales problemas de los que adolecen este tipo de herramientas es la propagación de los errores en las estimaciones en un enlace a los siguientes enlaces. Mediante la aplicación *a-clink* se lanzan varias aplicaciones de estimación de características de los enlaces en paralelo, una en cada router activo que exista en el camino entre el origen y el destino. Cada una de estas aplicaciones realizará los cálculos correspondientes a los enlaces entre dicho router activo y el siguiente router activo o el destino si es el último. De esta forma, el número de enlaces que estimará cada aplicación será mucho menor, limitándose por tanto la propagación de errores a los enlaces del tramo entre routers activos.

Otro de los problemas de dichas herramientas es la sobrecarga que producen en la red, sobre todo en los enlaces más cercanos a la fuente que deben soportar no solo el tráfico que se envía para calcular sus características, sino también el tráfico que se envía para calcular las características del resto de enlaces que se encuentran entre ellos y el destino. En concreto, cada enlace deberá soportar de forma adicional el tráfico necesario para realizar los cálculos de un enlace multiplicado por el número de enlaces que existen hasta el destino. En a-click la sobrecarga se reduce dado que el número de enlaces que hay entre un enlace determinado y el router activo o destino contra el que se lanza el cálculo de estimaciones, se ve reducido considerablemente y por lo tanto también el tráfico que se inyecta.

Al mismo tiempo, dado que se distribuye el proceso de realización de pruebas entre los routers activos que hay en el camino entre la fuente y el destino, se reducirá el tiempo necesario para realizar las pruebas y obtener las estimaciones pertinentes,

frente a las soluciones tradicionales que únicamente lanzan un proceso en la fuente.

3.1 Descripción del escenario de la aplicación

En la aplicación *a-clink* se divide el camino entre los sistemas finales en *segmentos* delimitados por los routers activos. De esta forma cada router activo realizará los cálculos de estimación de los enlaces de su segmento y enviará los resultados al sistema final que actúe como origen. Por lo tanto, en este nuevo escenario podemos distinguir cuatro tipos de sistemas que intervienen en la aplicación:

- **Sistema final origen**: inicia la aplicación y recopila los resultados de los routers activos.
- **Router activo**: realiza cálculos en su segmento devolviendo los resultados al sistema final de origen.
- **Router no activo**: situado dentro de los segmentos no intervienen directamente en el proceso activo, sólo en el proceso tradicional descrito en el funcionamiento de *clink*.
- **Sistema final de destino**: delimita el final de la ruta.

Los routers activos descargarán la aplicación *a-clink* que básicamente tendrá dos funciones principales:

- Descubrir cual es su segmento de ruta.
- Realizar los cálculos de estimación de parámetros de los enlaces en su segmento utilizando la aplicación *clink* tradicional.

Los routers activos mantiene la siguiente información de estado sobre la aplicación *a-clink*:

- **Id. Sesión**: identifica la sesión de forma única.
- **IP Origen**: sistema final origen.
- **IP destino**: sistema final destino.
- **Next**: siguiente router activo en la ruta, contra el que realizan los cálculos de medidas de los enlaces.
- **Saltos**: número de saltos de routers activos desde el origen hasta este router.

3.2 Proceso de segmentación de la ruta

El sistema final origen comienza el proceso enviando un paquete **explorador** dirigido al sistema final destino. El paquete va acumulando la siguiente información:

- **Sentido**: sentido del paquete explorador (origen a destino o destino a origen).
- **Saltos**: número de routers activos en el camino hasta el destino.
- **ACK[n]**: confirmación de cada router activo del mensaje explorador de vuelta. Hay un campo por cada router activo.
- **IP[n]**: dirección de cada router activo en la ruta. Hay un campo por cada router activo.

Cuando el paquete explorador, en dirección origen a destino, pasa por un router activo, éste realiza los siguientes procesos:

- Registra el identificador de sesión.
- En el paquete explorador incrementa el valor del campo s*altos* y le añade su *dirección IP*.
- Actualiza sus variables de estado (*origen*, *destino* y *saltos*).
- Reenvía el paquete explorador hacia el destino.

Cuando el paquete explorador llega al destino, lleva una lista ordenada de las direcciones IP de cada uno de los routers activos en la ruta.

El destino cambia el bit de sentido del paquete explorador y lo reenvía al origen siguiendo el camino inverso. Cada router activo al recibir el paquete explorador de vuelta actualiza su variable de estado *next* con la dirección IP del sistema siguiente a él en el camino de la fuente al destino, este sistema puede ser el siguiente router activo o el propio destino. Pone el bit ACK correspondiente a uno y reenvía el paquete hacia el origen.

Cuando el origen recibe el paquete explorador de vuelta, comprueba que todos los routers activos han visto el mensaje de vuelta (ACK=1), y han registrado por tanto la dirección de su siguiente salto en la ruta, y actualiza su variable de estado *next* apuntando al primer router activo en la ruta.

Cuando termina este proceso de exploración de routers activos en la ruta entre el origen y el destino, el origen conoce exactamente los routers activos que se encuentran en su camino al destino y cada uno de los routers activos conoce quién es su propio destino para el cálculo de estimaciones.

3.3 Cálculos de estimación de parámetros de los enlaces

Cada uno de los routers activos y el origen lanzarán la aplicación de cálculo de estimación de las características de los enlaces cuando reciban el paquete explorador de vuelta, dado que en ese momento ya conocen cuál es el destino de su segmento. Entonces cada uno lanzara la aplicación *clink* contra dicho destino. Terminado dicho proceso cada router activo enviará a la fuente los resultados de los cálculos en su segmento.

El origen, una vez que haya terminado el cálculo de su propio segmento de ruta y recibido los resultados del resto de routers activos, ordenará y mostrará las estimaciones calculadas para cada uno de los enlaces de la ruta completa.

4 Implementación y pruebas realizadas

Para la implementación de la aplicación *a-clink* se han analizado las principales plataformas de redes activas. ANTS [9] desarrollada por el MIT y basada

en una máquina virtual java (JVM) es una de las plataformas más utilizadas debido a su relativa facilidad de utilización y a que fue una las primeras plataformas de red activa desarrolladas. Pero como se pudo comprobar en [10] las prestaciones que proporciona hace que no sea la más adecuada para implementaciones en las que uno de los objetivos principales sea evaluar prestaciones. Otra de las plataformas más conocidas se corresponde con la propuesta desarrollada en el proyecto Switchware por la Universidad de Pensilvania. Dicha plataforma se basa en PLAN[11], un lenguaje funcional de propósito específico basado en OCAML, lo que complica bastante las tareas propias de la implementación. BBN en el proyecto Smart Packets [12] propone una plataforma específica para usarla en tareas de gestión de red. La Universidad Carlos III de Madrid ha desarrollado en el proyecto europeo IST-GCAP[7], una plataforma de redes activas, denominada SARA, caracterizada por estar orientada a proporcionar altas prestaciones y por su facilidad de utilización. Es debido a estas características el que se haya seleccionado dicha plataforma para la implementación de *a-clink*. Esta plataforma utiliza Java para el desarrollo de aplicaciones y se instala sobre el sistema operativo Linux.

La aplicación *a-clink* se descarga bajo demanda desde un servidor de código ante la llegada del primer paquete activo que lleve el identificador de la aplicación. Para poder integrar la aplicación *clink* que esta desarrollada en c, se ha añadido a la aplicación *a-clink* mediante JNI (Java Native Interface).

4.1 Descripción del escenario de pruebas

El escenario real sobre el que se han realizado las pruebas está formado por dos redes, una en la Universidad de Alcalá y otra en la Universidad Carlos III de Madrid, unidas por un túnel. En la Fig. 4 se puede ver el esquema. Las características de los equipos y los enlaces se describen a continuación.

Universidad de Alcalá:

- SF1: K6/266, 98 MB de RAM.
- R1: Smart Switch Router 2000 de Cabletron.
- RA1: Pentium III/600, 64 MB de RAM.
- R2: Pentium II/350 , 64 MB de RAM.

Universidad Carlos III de Madrid:

- RA2 y R3: Pentium III 733MHz, 128 MB de RAM.
- SF2: Pentium III 700MHz, 128 MB de RAM.

Todos los ordenadores tienen instalado el sistema operativo Linux. Los enlaces tienen las siguientes características:

- Los enlaces E1, E2 y E3 son Ethernet a 10Mbps.
- El enlace E4 es un túnel que une las redes de la Universidad de Alcalá y la Universidad Carlos III.
- Los enlaces E5 y E6 son Fast Ethernet a 100Mbps.

En los enlaces E1, E2, E3, E5 y E6 no se ha introducido más tráfico que el generado por la propia aplicación de prueba.

Nos hemos encontrado dos problemas a la hora de implementar este escenario:

1. En la entrada de la red de la Universidad Carlos III se filtra el tráfico ICMP. Esto no permite el funcionamiento de *clink* que se basa en la recogida de paquetes ICMP. Para resolverlo se ha implementado un túnel IP entre R2 y RA2 (Fig. 4).
2. Linux limita la velocidad de envío de paquetes ICMP. Hemos resuelto este problema desprotegiendo dicha limitación en el kernel del sistema operativo para los paquetes ICMP que utiliza la aplicación *clink*.

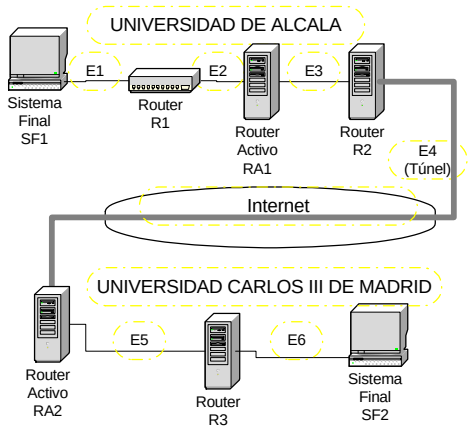

Fig. 4. Escenario de pruebas.

4.2 Pruebas realizadas y análisis de los resultados

Para probar la aplicación activa *a-clink* y comparar sus resultados con los de *clink*, hemos realizado diversas pruebas de ambas aplicaciones en el escenario de la Fig. 4. Las pruebas se han realizado enviando paquetes a cada router de 91 tamaños distintos comprendidos entre 28 y 1468 bytes con saltos de 16. Cada tipo de paquete se ha repetido 8 veces. Por lo tanto, el total de paquetes enviados a cada router del camino es 728 (91*8).

La tabla 1 muestra los resultados de las pruebas para cada una de las aplicaciones. La columna LAT corresponde a la estimación de la latencia y la columna BW a la estimación del ancho de banda. Cada fila muestra los resultados en cada uno de los enlaces del camino en el escenario de pruebas. Este camino, desde el punto de vista de la aplicación activa, se divide en tres segmentos; el primero

formado por E1 y E2, el segundo por E3 y E4, y el tercero por E5 y E6.

De los resultados de estas pruebas podemos deducir que los valores de estimación de latencia no corresponden realmente con el retardo de propagación, según suponía el modelo teórico. En dicho modelo los retardos de procesamiento interno se despreciaban frente al retardo de propagación. Esto puede ser cierto en el entorno de aplicación de Internet, en el que las distancias son mayores con dispositivos de conmutación rápidos. En nuestro escenario las distancias son cortas, abarcando valores que van desde unos metros hasta unas pocas decenas de kilómetros (en el caso del túnel). Esto se traduce en unos tiempos de propagación que pueden estar en un rango aproximado entre 50 ns y 150 µs. Por otra parte los tiempos de procesamiento interno dependen de la CPU del equipo, pudiendo llegar a alcanzar valores de varios milisegundos y por lo tanto nada despreciables. Resumiendo podemos decir que la estimación de latencia (columna LAT de la tabla 1), realmente nos da idea de los tiempos consumidos en procesamiento de paquetes en cada enlace.

También podemos observar que las estimaciones de ancho de banda no se aproximan con suficiente precisión a los valores nominales de los enlaces. La explicación está en la suposición del modelo teórico de que el único tiempo que depende del tamaño de los paquetes es el tiempo de transmisión. En las máquinas hay procesos internos cuyo tiempo de proceso depende del tamaño de los paquetes. Este tiempo se suma al de transmisión, ocasionando desviaciones en la estimación del ancho de banda.

Tabla 1. Resultados de las pruebas.

Enlace	Seg-mento	CLINK LAT (ms)	ACLINK LAT (ms)	CLINK BW (Mbs)	ACLINK BW (Mbs)
E1	Seg. 1	0,277	0,263	6,98	6,994
E2		0,384	0,368	4,131	4,009
E3 (RA)	Seg. 2	0,211	0,238	6,667	6,668
E4		50,683	49,953	0,276	0,302
E5 (RA)	Seg. 3	-8,424	0,051	-0,813	65,852
E6		2,655	0,035	2,269	64,496

Analizando los resultados que se obtienen con *clink* vemos que las estimaciones de los enlaces E5 y E6 no son buenas, llegando incluso a dar valores negativos. Analizando los resultados que se obtienen con *a-clink* vemos que las estimaciones correspondientes a los enlaces E1, E2, E3, E5 y E6 son correctas, con las matizaciones comentadas previamente. El enlace E4 queda fuera de nuestro control con lo cual es difícil determinar la precisión de sus estimaciones. Lo que si queda claro es que *a-clink* aísla los posibles errores de las estimaciones que produce este enlace, al iniciarse de nuevo los cálculos en el segmento 3, evitando como ya hemos comentado la propagación de los errores. Estos posibles errores pueden ser debidos a varios motivos entre los que podemos destacar congestión en los routers y ocultación de nodos.

Otra mejora que se obtienen con la aplicación *a-clink* se refiere a los tiempos invertidos en la ejecución de la aplicación frente a *link*, reduciéndose considerablemente el tiempo necesario para la realización de las pruebas. Esto es debido a que la aplicación *a-clink* se ejecuta en paralelo en cada uno de los tres segmentos del camino. En nuestro caso los tiempos invertidos en la ejecución de *clink* y *a-clink* han sido 11:47 y 6:09 minutos respectivamente.

Por último en la tabla 2 se representa el número de paquetes que pasan por cada uno de los enlaces, para cada una de las aplicaciones. Para calcular el número de paquetes que soporta cada enlace podemos utilizar la expresión $\mathbf{N_i=2n(k-i+1)}$. Donde N_i es el número de paquetes que soporta el enlace i, n es el número de paquetes generados para realizar las pruebas, k es el número de enlaces del camino (*clink*) o del segmento (*a-clink*) e i es el número de orden del enlace en el camino o segmento. En nuestro caso n = 728 paquetes. Remarcar que en estos cálculos se incluyen los mensajes ICMP de respuesta.

Analizando dicha formula se deduce claramente que el número de paquetes que debe soportar un enlace depende directamente del número de enlaces en el camino entre el origen y el destino. Para c*link* ese número se corresponderá con el número total de enlaces entre el origen y el destino. Para *a-clink* dicho número queda reducido en función del número de segmentos que se establezcan, que dependerá del número de routers activos en el camino entre la fuente y el destino.

Tabla 2. Cantidad de paquetes en cada enlace.

Enlace	Segmento	CLINK (paquetes)	ACLINK (paquetes)
E1	Seg. 1	8736	2912
E2		7280	1456
E3 (RA)	Seg. 2	5824	2912
E4		4368	1456
E5 (RA)	Seg. 3	2912	2912
E6		1456	1456

En la tabla 2 vemos como los primeros enlaces con *clink* soportan una carga elevada al acumular los mensajes del resto de los enlaces. Con *a-clink* únicamente soportan los menajes de su segmento.

En base a los resultados obtenidos podemos concluir que la aplicación *a-clink* al dividir el camino en segmentos, reduce el tiempo de cálculo y el número de paquetes soportados por los enlaces, y aísla las perturbaciones introducidas en determinados enlaces, de forma que no afecten a las medidas en enlaces posteriores.

5 Conclusiones y trabajos futuros

En este artículo hemos descrito una nueva herramienta de caracterización del estado de los enlaces basada en la tecnología de redes activas y que utiliza como base una herramienta ya existente. Se han realizado pruebas comparativas de ambas herramientas y hemos observado que nuestra propuesta reduce el número de paquetes necesarios para las estimar el estado de los enlaces, reduce el tiempo de ejecución de la aplicación y se recupera de errores en enlaces en los que las perturbaciones no permiten obtener buenos resultados. Al mismo tiempo se ha visto como las estimaciones obtenidas con la herramienta *clink*, que como se demostró en [8] proporciona estimaciones equivalentes a las obtenidas con otras herramientas de estimación de prestaciones en red, hay que manejarlas con cierta cautela.

Queremos remarcar que dichos resultados se han obtenido en un entorno de pruebas donde el número de enlaces totales es reducido. Si extrapolamos los resultados obtenidos a un entorno más real, Internet, en donde el número de enlaces entre un origen y destino es bastante mayor, podemos concluir que el impacto de las ventajas obtenidas se verá en gran medida amplificado.

En cuanto a la plataforma de red activa SARA, podemos concluir que la familiarización y utilización de dicha plataforma ha sido fácil y rápida. Y que la plataforma se encuentra en un estado de desarrollo lo suficientemente maduro como para poder realizar implementaciones de aplicaciones activas en un entorno real. En relación a la tecnología de redes activas, la herramienta desarrollada es una más de los ejemplos de aplicaciones que aprovechan las ventajas de dicha tecnología.

Para terminar, podemos destacar tres líneas de actuación futuras. 1) Completar el diseño del protocolo de segmentación de la ruta para que soporte pérdida de paquetes de control. 2) Modificaciones del protocolo para que no sea necesario tener un sistema final de destino con procesamiento de routers activos. 3) Simulación de la aplicación para poder comparar eficiencias en escenarios más complejos.

Agradecimientos

Queremos mostrar nuestro agradecimiento a los becarios José Alberto Fernández, Clara Santos y Manuel Urueña y a los proyectandos Andrés Navarro y Graciela Garrido por haber participado en la implementación de la aplicación *a-clink* y en la realización de las pruebas.

Referencias

[1] V. Jacobson. Pathchar–A tool to infer characteristics of Internet paths. Presented at the Mathematical Sciences Research Institute, April 1997.

[2] A. B. Downey. *Using pathchar to Estimate Internet Link Characteristics*. In proceedings of ACM SIGCOMM 1999, pp. 241-250, July 1999.

[3] Bruce A. Mah. Pchar: Child of Pathchar. Presented at the DOE NGI Testbed Workshop, Berkeley, CA, 21 July 1999.

[4] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall and G.J. Minden. A survey of Active Network Research. IEEE Communications Magazine, pp. 80-86, January 1997.

[5] M. Calderón, M. Sedano, S. Eibe García. *Principios y Aplicaciones de las redes Activas*. Proceedings de Jitel´99. pp 311-319. Sep 1999.

[6] Jonathan T. Moore Scott M. Nettles. *Towards Practical Programmable Packets*. Technical Report MS-CIS-00-12 . May 2000.

[7] Servidor www proyecto IST GCAP (Global Communication Architecture and Protocols for new QoS services over IPv6 networks). http://www.laas.fr/GCAP/

[8] K Lai, M Baker. *Measuring Link Bandwidths Using a Deterministic Model of Packet Delay*. In Proceedings of ACM SIGCOMM 2000. August 2000.

[9] D. Wetherall, J. Guttag and D.L. Tennenhouse. *ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols*. *IEEE OPENARCH'98*, San Francisco, CA, April 1998.

[10] M. Calderón, M. Sedano, A. Azcorra and C. Alonso. Active Networks Support for Multicast Applications. IEEE Network Magazine, Special Issue: Active and Programmable Networks, Vol. 12 No. 3 pp 46-52, Mayo/Junio 1998.

[11] S. Alexander, W. A. Arbaugh, M. W. Hicks, P. Kakkar, A. D. Keromytis, J. T. Moore, C. A. Gunter, S. M. Nettles and J. M. Smith. *The SwitchWare Active Network Architecture*. IEEE Network, Special Issue: Active and Programmable Networks, Vol 12(3) pp. 29-36, May/June 1998.

[12] B. Schwartz, A. Jackson, T. Strayer, W. Zhou, R. Rockwell, and C. Partridge. *Smart packets for active networks*. In proceedings of the 1999 IEEE 2nd Conference on Open Architectures and Networks Programming (OPENARCH'99), March 1999.