EMULACIÓN DE ESCENARIOS DE RED MEDIANTE UN TESTBED

José Mª Saldaña, Jenifer Murillo, Julián Fernández-Navajas, José Ruiz-Mas, Eduardo A. Viruete, José Ignacio Aznar
E-mail {jsaldana, jenifer.murillo, navajas, jruiz, eviruete, jiaznar}@unizar.es
Grupo de Tecnologías de las Comunicaciones (GTC) – Instituto de Investigación en Ingeniería de Aragón (I3A)
Universidad de Zaragoza
Ed. Ada Byron, C/ María de Luna nº 1, 50018 Zaragoza (España)
Tlf: (+34) 976 761 963 – Fax: (+34) 976 762 111

***Abstract*- This work presents a virtualization based testbed, which is able to emulate both wired and wireless networks. Although big research infrastructures have been developed around the globe, small testbeds can also be useful, especially in the first stages of the development of new systems and protocols. The presented system is hybrid, as it includes simulation and emulation tools. It uses a previous simulation stage, that has to be carried out offline in order to avoid computational load. In this article two uses of the testbed are also presented: in the first one, some MIPv4 handovers are measured. In the second one, a Call Admission Control for an IP Telephony system is tested and measured. These two examples show that the measurements obtained with the testbed are very similar to the ones published by other research groups using real hardware.**

I. INTRODUCCIÓN

Aunque en sus comienzos Internet se entendía como una infraestructura de investigación, hoy en día se ha convertido en un entorno operativo y es más difícil de usar como plataforma de pruebas. Muchas universidades y departamentos de I+D que trabajan en redes y protocolos necesitan herramientas y entornos para verificar su funcionamiento en condiciones reales. En ocasiones, las pruebas son muy difíciles de realizar debido al gran número de máquinas y entornos que se requieren.

Hay varias soluciones posibles para este problema. Una pasa por el uso de herramientas de simulación, como Opnet o ns-2, que permiten realizar medidas controladas y repetibles a bajo coste. Tienen dos inconvenientes: el primero es la carga computacional, especialmente cuando el escenario de red incluye un gran número de máquinas; el segundo se debe al hecho de que los simuladores simplifican el sistema a medir, alejándolo de la realidad, usando implementaciones específicas de los protocolos, y no permitiendo el uso de cualquier aplicación o sistema operativo (SO).

Otra opción es utilizar *hardware* real, que incluye la pila de protocolos completa del SO. Esta puede ser una buena solución en muchos casos, pero también se han desarrollado muchos *testbed* y emuladores [1] para facilitar las pruebas y reducir el coste. Algunos son híbridos, combinando las ventajas de la simulación, la emulación y las pruebas con equipos reales.

En este trabajo se explica el desarrollo de un *testbed* que emula una red mediante el uso de virtualización, y permite así implementar un conjunto de máquinas virtuales dentro de una sola máquina física. Los nodos que participan en la comunicación son máquinas virtuales adecuadamente conectadas.

También presentamos dos usos diferentes que hemos dado al *testbed*, tanto para redes cableadas como inalámbricas. Servirán para validarlo, al comparar nuestros resultados con los obtenidos por otros grupos que han usado *hardware* real. Estas comparaciones ilustran la utilidad del *testbed*, especialmente en los primeros pasos del desarrollo de nuevos sistemas y aplicaciones distribuidas.

En la siguiente sección se presentan los trabajos relacionados. La arquitectura del sistema se resume en la sección III. La sección IV presenta dos ejemplos de uso del *testbed*. El trabajo termina con las conclusiones.

II. TRABAJOS RELACIONADOS

En los últimos años se han desplegado grandes infraestructuras donde probar protocolos y aplicaciones [2], [3]. Estos entornos suelen ser compartidos por grupos de investigación de diferentes países. Algunos integran emuladores para imitar el comportamiento de diferentes elementos del sistema, como los movimientos de los nodos, los cambios en el canal radio, etc. Una de estas grandes plataformas de pruebas es PlanetLab [4], que permite desarrollar y evaluar nuevos protocolos y servicios. Cada nodo contiene un monitor de máquinas virtuales que aísla entre sí los servicios y aplicaciones.

Cuando se utiliza emulación, el sistema a medir se representa con algunas de sus partes modificadas y otras tratadas exactamente igual que en el caso real [1]. Algunas partes de la prueba tienen un mayor nivel de abstracción que otras; algunas son simuladas y otras reales.

Como la emulación combina elementos reales con otros modificados o simplificados, se debe ejecutar en tiempo real. Esto hace que las pruebas no sean totalmente repetibles, ya que puede haber pequeñas diferencias entre unas y otras. Por eso es importante que el *testbed* proporcione un aislamiento suficiente entre las máquinas como para garantizar un nivel aceptable de repetibilidad.

Como plataforma híbrida podemos destacar en primer lugar EMWin [5], que emula la capa MAC, mientras que la pila de protocolos y las aplicaciones son reales. NCTU [6] emula nodos que generan tráfico, que luego es capturado por el simulador para introducir el comportamiento de un

escenario inalámbrico. Netbed/ Emulab [7] y W-NINE [8] usan conformadores de tráfico después de una etapa de simulación. También hay emuladores que usan virtualización, como vBET [9].

La ventaja principal de la virtualización es que permite usar las implementaciones reales de los SO y aplicaciones, consiguiendo un tráfico igual al del caso real. La desventaja es que el comportamiento de la red debe emularse, puesto que los *bridge* virtuales que conectan las máquinas no tienen limitaciones de ancho de banda. Otro inconveniente es que se debe monitorizar la CPU para evitar que los retardos sean causados por su carga de trabajo y no por la red.

Xen es una solución de paravirtualización, que requiere que el SO esté específicamente compilado para correr en la máquina virtual; pero a cambio proporciona un rendimiento casi igual al de un sistema no virtualizado.

III. Arquitectura del Testbed

Cada nodo es emulado con una máquina virtual. Para emular el comportamiento de los enlaces se usa la herramienta *Traffic Control* (*tc*), que permite introducir anchos de banda en los enlaces entre nodos. También se ha usado *Mackill* para emular el comportamiento de redes inalámbricas. Esta herramienta forma parte del *testbed* APE [10], y consiste en un módulo del *kernel* de Linux que añade un filtro MAC en la pila de protocolos, de forma que se pueden descartar paquetes que vengan de ciertas direcciones. *Mackill* lee la información de visibilidad de un fichero generado previamente, y descarta los paquetes que proceden de direcciones que en ese momento no están visibles, emulando así el comportamiento de una red inalámbrica (Fig. 1).

Para emular el comportamiento a nivel de red se ha usado *Netem*, que permite introducir retardos controlados en los paquetes con distintas distribuciones estadísticas. Se han creado dos redes: una de control y otra para de pruebas, para evitar que un tráfico influya en el otro.

El sistema se usa en tres fases (Fig. 2). En primer lugar hay una etapa de generación del escenario. Se pueden usar diferentes herramientas de simulación para obtener los momentos en que ocurrirán los diferentes eventos. Se obtiene así un conjunto de ficheros de escenario que contienen los diferentes parámetros de la emulación: posiciones de los nodos en cada momento, instantes de las llamadas, etc. Una vez generados los ficheros, se puede ejecutar varias veces la misma realización. También se pueden repetir las mismas situaciones, pero variando la configuración del escenario: tráfico de fondo, atenuación, etc.

Posteriormente tiene lugar la fase de emulación del escenario, en la que se utilizan diferentes herramientas para generar tráfico. El objetivo de esta etapa es lograr un conjunto de ficheros *log*, que serán analizados después.

Dado que en la segunda fase se produce el envío del tráfico, es importante que el procesador no se ocupe de ningún otro trabajo en ese momento. Por eso, en la primera fase se prepara *offline* el escenario, y en la tercera se analizan, también *offline*, los resultados.

A partir de los ficheros de salida obtenidos podemos calcular algunos parámetros de QoS como el retardo en un sentido (*One Way Delay*, OWD), la tasa de pérdidas, el *jitter* o el MOS (*Mean Opinion Score*).

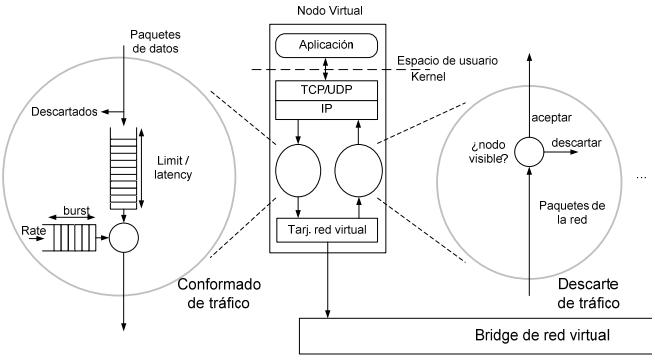


Fig. 1. Modelo de red con el conformador de paquetes y *Mackill*

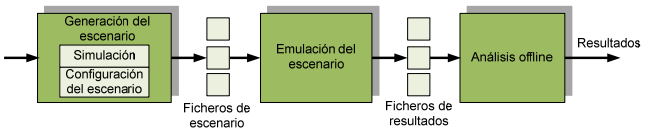


Fig. 2. Etapas de uso del *testbed*

La máquina empleada usa el SO CentOS 5. La versión del núcleo de Linux es la 2.6.18-8.1.15. Dispone de un procesador Core 2 Duo a 2.40 Ghz, 2MB de Cache nivel 2, y 4GB de RAM. Las máquinas virtuales tienen instalado también el SO CentOS 5. La versión de Xen instalada es la 3.03-25.0.4. Se han usado las herramientas *top* y *mpstat* para monitorizar el uso del procesador durante las pruebas, asegurando que la utilización no exceda el 10%.

IV. Ejemplos de uso del Testbed

A. Medidas de traspasos usando Movilidad IP

En este primer ejemplo se muestra el uso del *testbed* para medir tiempos de traspaso usando el protocolo MIPv4. Para la fase de generación del escenario se usa una modificación de *AnSim* [11], una aplicación que permite al usuario generar los movimientos aleatorios de los nodos. Se pueden modificar algunos parámetros, como el nivel de atenuación, el modelo de movilidad, el número de nodos, etc.

Se han añadido a *AnSim* algunas funcionalidades, como nodos fijos, y la posibilidad de especificar qué nodos actúan como *router* para otros. También se ha mejorado el modelo de pérdidas, para incluir no sólo pérdidas en el espacio libre sino también las causadas por la onda reflejada. Podemos también definir la velocidad y características de los enlaces que unen los nodos. A partir de la información del escenario, *Mackill* filtra a nivel MAC los paquetes que proceden de nodos no visibles.

Hemos construido un escenario (Fig. 3) que emula el paso de una red a otra durante una transmisión, para comparar diferentes situaciones. MIPv4 introduce agentes de movilidad para dar servicio a los nodos que no se encuentran en su red original. Existe un Agente Local (*Home Agent*, HA) que se ocupa del tráfico de los nodos que están fuera y una serie de Agentes Externos (*Foreign Agent*, FA) que se encargan del tráfico de los nodos que están visitando su red. El Nodo Correspondiente (*Correspondent Node*, CN) es el que se está comunicando con el Nodo Móvil (*Mobile Node*, MN).

Se ha usado la implementación Dynamics-Hut de MIPv4, que puede trabajar en modo MN y FA *decapsulation*, y

puede hacer *tunneling* reverso o triangular. En la Fig. 4 se puede ver la gráfica obtenida durante un traspaso con un tiempo entre paquetes de 12 ms. El ancho de banda de información es de 28 kbps. Vemos que se han perdido 7 paquetes. Por eso podemos estimar el tiempo de traspaso en 84 ms. Los paquetes posteriores al traspaso llegan a la vez, porque se almacenan en el FA.

Hemos medido algunos traspasos, basándonos en las pérdidas de paquetes, para comparar los dos modos de funcionamiento: con *FA decapsulation* se obtenía una media de 9,2 paquetes perdidos, y con *MN decapsulation* era de 9,4. Se enviaban paquetes de 42 bytes cada 10 ms. Este tiempo de traspaso concuerda con [12] en un caso similar, ya que el tiempo de traspaso medido estaba en torno a los 100 ms.

Se ha observado una diferencia en el retardo de los traspasos *directos* (en los que el *MN* va de su red a otra) y los *inversos*, en los que vuelve a su red. Enviando el mismo tráfico que en el caso anterior, se perdían en media 1.9 paquetes. Por tanto, los traspasos *inversos* resultan más rápidos.

B. Pruebas de un sistema de telefonía IP

Los servicios de telefonía IP representan una solución interesante para las empresas porque no sólo aportan ahorro, sino también disponibilidad y seguridad. VoIP es un servicio en tiempo real en el que el retardo y las pérdidas influyen directamente en la calidad de las llamadas, y los usuarios requieren una QoS similar a la que proporciona la Red Telefónica Conmutada (RTC). Actualmente nuestro grupo está trabajando en un control de admisión de llamadas (*Call Admission Control*, CAC) para un sistema de telefonía IP basado en el protocolo SIP, y las primeras pruebas se están realizando en el *testbed* presentado. En este caso el *testbed* emulará redes cableadas, por lo que no serán necesarias herramientas para simular movilidad y visibilidad.

El sistema CAC está pensado para una situación en la que los usuarios no tienen control sobre Internet ni sobre los parámetros y la infraestructura de la red. Una nueva llamada se aceptará sólo en el caso de que ni ella ni el resto de llamadas en curso vean disminuida su calidad.

El sistema de telefonía se corresponde con la situación de una empresa con sucursales en diferentes países (Fig. 5). La PBX se encuentra en el centro de datos. Se usa Internet para el envío de tráfico, prescindiendo de enlaces dedicados. En cada sucursal se ha introducido un agente local, que incluye un *proxy* SIP encargado de las decisiones de admisión. Las llamadas entre los terminales pueden ser rechazadas debido a la falta de QoS o por la no disponibilidad del terminal destino. Las llamadas son rechazadas mediante un mensaje SIP *480 Temporarily Unavailable* a la PBX.

Cuando un agente local recibe un *INVITE* de la PBX enviado a un terminal, lo acepta o rechaza dependiendo del número de llamadas establecidas en ese momento. Hay un número máximo de llamadas simultáneas permitido por el sistema, que es el parámetro principal del CAC. Para las llamadas destinadas a RTC se puede usar redirección si el *gateway* local tiene todas sus líneas ocupadas. Aunque los mensajes SIP pasan por la PBX, el tráfico RTP se envía directamente entre los teléfonos.

Se ha utilizado *Matlab* para generar los instantes y las duraciones de las llamadas durante la hora cargada, con diferentes distribuciones estadísticas para emular el tráfico telefónico entre las sucursales.

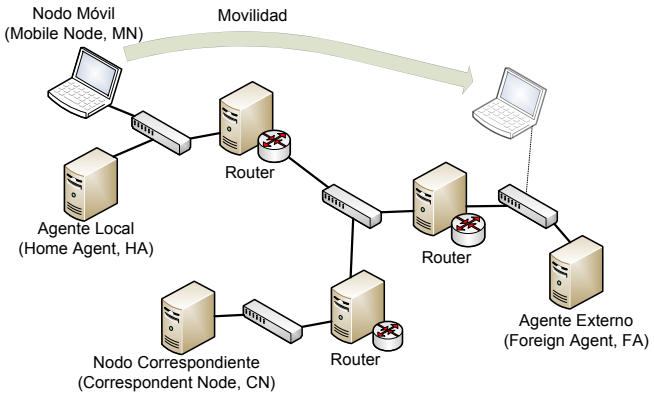


Fig. 3. Escenario de Movilidad IP

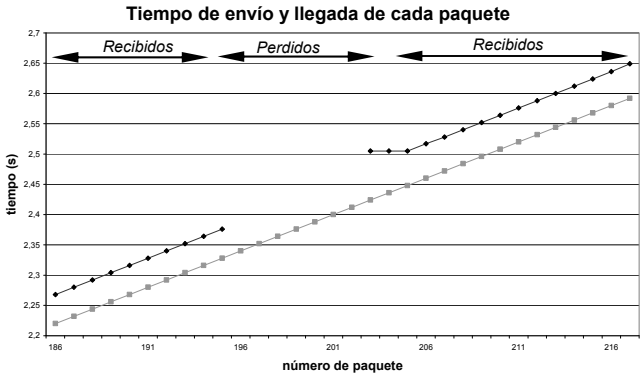


Fig. 4. Tiempo de transmisión y recepción durante un traspaso

Las herramientas *software* elegidas deberán tener poca carga computacional, por correr en un entorno virtualizado, compartiendo los recursos con otras máquinas. Para la centralita hemos elegido *Asterisk 1.6.* También se ha usado *OpenSIPS 1.4*, un *proxy* SIP que se ha adaptado para tomar las decisiones de CAC según la información de una base de datos externa. Para emular los teléfonos IP y los *gateway* hemos usado el *softphone* PJSUA 1.0.

Los accesos de cada sucursal tienen un ancho de banda de subida de 1 Mbps. El tamaño del *buffer* se ha calculado para que el tiempo máximo de encolado sea de 50 ms. Con *Netem* se ha añadido un retardo medio de 40 ms en cada sucursal, y 20 ms en el centro de datos. Cada flujo RTP tiene un ancho de banda de 24 kbps a nivel IP.

Se ha usado la herramienta *expect* para automatizar el manejo de los *softphone*, de forma que realicen las llamadas en el momento adecuado, según los ficheros obtenidos inicialmente con *Matlab*. Estos ficheros pueden ejecutarse en diferentes situaciones, para poder realizar comparativas entre usar o no los agentes locales, diferentes distribuciones de tráfico, número de usuarios, etc.

El tráfico de fondo tiene la siguiente distribución de tamaños: el 50% de los paquetes son de 40 bytes a nivel IP, el 10% de 576 bytes, y el 40% de 1500 bytes [13]. Este tráfico se ha utilizado para saturar el acceso de cada sucursal. Hemos usado UDP para evitar el efecto del control de flujo que realiza TCP. Así conseguimos que el sistema trabaje siempre en el peor caso, pues el tráfico de fondo es siempre el mismo porque no se adapta al ancho de banda disponible.

Los ficheros *log* obtenidos se analizan después de las ejecuciones para obtener gráficas de los parámetros de QoS. Las Fig. 6 (a) y (b) muestran el comportamiento del sistema en términos de OWD y pérdidas. Estas curvas suponen la

cota superior para el caso en que el número máximo de llamadas se establezca en cada valor. Están representadas en función del tráfico total.

La existencia de una cola que descarta paquetes lleva a un límite para el OWD. La Fig. 6 (c) muestra el MOS [14]. El *jitter* del sistema está por debajo de 12 ms.

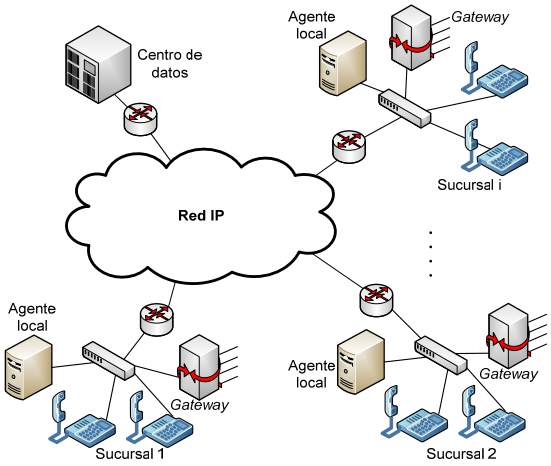
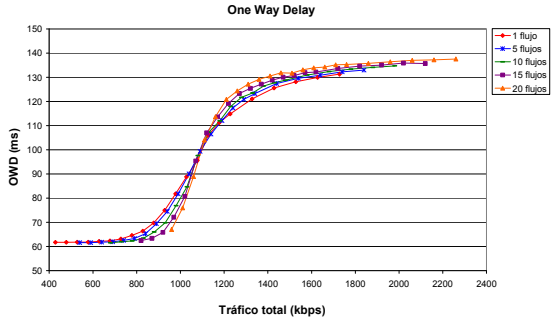
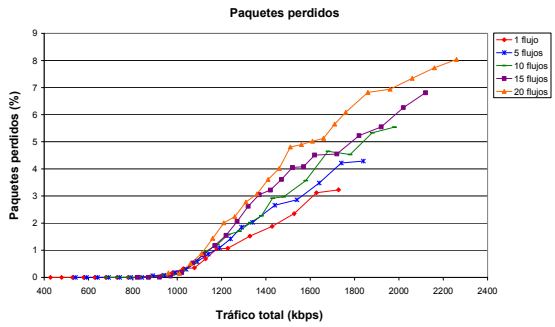


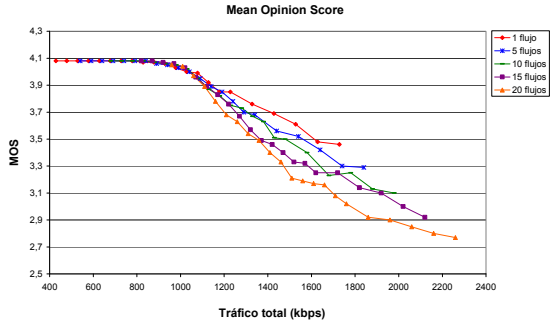
Fig. 5. Escenario de telefonía IP



(a)



(b)



(c)

Fig. 6. OWD, pérdidas y MOS del sistema

V. Conclusiones

Se ha presentado un *testbed* híbrido que utiliza la virtualización Xen emular una red dentro de una sola máquina física. Se usa en tres etapas. En primer lugar se genera el escenario y se simulan los movimientos, llamadas y demás eventos. Posteriormente, los nodos simulados se trasladan a máquinas virtuales, y se envía tráfico, que se captura para su análisis en una tercera fase, en la que se obtienen *offline* diferentes parámetros de QoS.

Uno de los objetivos era validar si el *testbed* obtenía resultados comparables a los que se obtienen con *hardware* real. Se han presentado dos usos del *testbed*: el primero simula un escenario de movilidad y se han obtenido tiempos de traspaso usando MIPv4, que son similares a los que se obtienen con máquinas reales. El segundo uso muestra la utilidad del *testbed* para probar un sistema distribuido, como un CAC para telefonía IP. Se ha mostrado que el *testbed* tiene flexibilidad para llevar a cabo diferentes pruebas, tanto de redes cableadas como inalámbricas.

Agradecimientos

Este trabajo ha sido financiado parcialmente por el proyecto Cheque Tecnológico 2009/2010, de la Agencia Aragón I+D, del Gobierno de Aragón, y el proyecto Cátedra Telefónica, de la Universidad de Zaragoza.

Referencias

[1] E. Göktürk, “A stance on emulation and testbeds”, in Proceedings of the 21st European Conf. on Modelling and Simulation ECMS 2007.

[2] J. S. Turner, “A proposed architecture for the GENI backbone platform”, In Proc. Arch. for Network and Comm. Systems, 2006.

[3] A. Gavras, A. Karila, S. Fdida, M. May, M. Potts, “Future internet research and experimentation: the FIRE initiative”, ACM SIGCOMM Computer Communication Review, v.37 n.3, 2007.

[4] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, M. Wawrzoniak, “Operating System Support for Planetary-Scale Network Services”, in Proc. of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04), San Francisco, CA, 2004.

[5] P. Zheng, L. M. Ni, “EMWin: emulating a mobile wireless network using a wired network”, In Proceedings of the 5th ACM international Workshop on Wireless Mobile Multimedia, Atlanta, 2002.

[6] S. Y. Wang, “Using the innovative NCTUns 3.0 network simulator and emulator to facilitate network researches”, Testbeds and Research Infrastructures for the Development of Networks and Communities, TRIDENTCOM, pp.4-184, Barcelona, 2006.

[7] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, Ch. Barb, A. Joglekar, “An integrated experimental environment for distributed systems and networks”, in Proc. 5th symposium on Operating systems design and implem., Boston, 2002.

[8] E. Conchon, J. Garcia, T. Pérennou, M. Diaz, “Improved IP-level Emulation for Mobile and Wireless Systems”, in Proc. IEEE Wireless Communications & Networking Conference, Hong Kong, 2007.

[9] X. Jiang, D. Xu, “vbet: a vm-based emulation testbed”, in Proc. ACM SIGCOMM workshop Models, methods and tools for reproducible network res. MoMeTools’03, ACM Press, pp. 95–104, New York 2003.

[10] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordströ, C. Tschudin, “A Large-scale Testbed for Reproducible Ad hoc Protocol Evaluations”, Proc. IEEE Wireless Comm. and Networking Conf. WCNC'02, 2002.

[11] H. Hellbrück, S. Fischer, “Towards analysis and simulation of ad-hoc networks”, in ICWN02: Proceedings of the International Conference on Wireless Networks, pages 69–75, Las Vegas, 2002.

[12] A. Cabellos-Aparicio, H. Julian-Bertomeu, J. Núñez-Martínez, L. Jakab, R. Serral-Gracià, J. Domingo-Pascual, “Measurement-Based Comparison of IPv4/IPv6 Mobility Protocols on a WLAN Scenario”, in Proceedings of Networks UK HET-NET Ilkley, UK, 2005.

[13] Cooperative Association for Internet Data Analysis “NASA Ames Internet Exchange Packet Length Distributions”.