# Metodología para realizar pruebas de carga a un servidor Web sobre un sistema empotrado. Un caso de estudio

***Jorge Arturo Hernández Perales***

Universidad Tecnológica de la Mixteca, Carretera a Acatlima km 2.5 Huajuapan de León, Oax., 953 5320399 Ext.200

*jahdezp@mixteco.utm.mx*

***María Esperanza Pérez Cordoba Sánchez***

Universidad Tecnológica de la Mixteca, Carretera a Acatlima km 2.5 Huajuapan de León, Oax., 953 5320399 Ext.200

*mapercor@mixteco.utm.mx*

***Mónica Edith García García***

Universidad Tecnológica de la Mixteca, Carretera a Acatlima km 2.5 Huajuapan de León, Oax., 953 5320399 Ext.200

*mgarcia@mixteco.utm.mx*

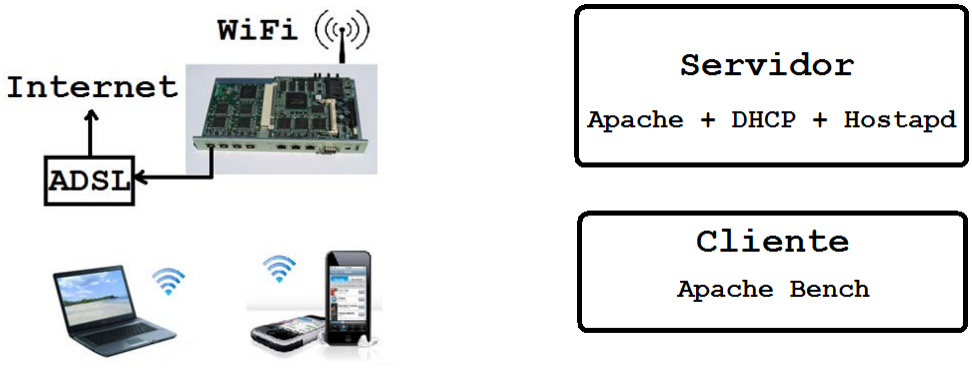
## Resumen

En el presente trabajo se propone una metodología para hacer pruebas de carga a un sistema empotrado que se usará para brindar el servicio de conexión inalámbrica a Internet desde un portal cautivo. Las pruebas de carga y en general aquellas pruebas que se denominan como “Benchmark” son muy útiles porque nos ayudan a tomar decisiones a la hora de desarrollar sistemas, configurarlos o para elegir la solución más óptima dado un conjunto de parámetros. Se muestra la aplicación de la metodología y el diseño de un conjunto de pruebas básicas. El objetivo de las pruebas consiste en encontrar la mejor configuración de hardware y software para el sistema, para poder hacer las comparativas fue necesario controlar algunas variables como el tráfico en la red.

**Palabras Claves:** Metodologías, pruebas de carga, servidores web, sistemas empotrados.

## 1. Introducción

Este trabajo presenta la propuesta de una metodología para realizar pruebas de carga sobre un sistema empotrado que alberga un servidor Apache, mismo que será usado para crear un portal cautivo por medio del cual se brindará el servicio de acceso a Internet de manera inalámbrica. Un sistema empotrado o embebido es un sistema limitado en software y/o hardware en comparación con una computadora personal, que tiene una funcionalidad particular, y físicamente la mayoría de los componentes se encuentran incluidos en una placa base [1]. Actualmente un sistema empotrado tiene gran importancia por su aplicación en cualquier ámbito: comunicaciones móviles, tráfico y transporte (aéreo y terrestre), medicina, hogar, automotriz, entretenimiento [2, 3]. Este tipo de sistemas son construidos para su configuración como un componente de red que sea altamente flexible y de bajo costo. A diferencia de otras metodologías como las que se presentan en [4, 5, 6], nuestra propuesta es diseñada y aplicada a un sistema con características muy particulares y no consideradas en estudios anteriores [7, 8, 9].




(a) (b)

**Fig. 1. Esquema general del sistema y diagrama de módulos de software.**

En la Fig. 1(a) se muestra el esquema general del sistema, donde el elemento central es una tarjeta Soekris que puede ser configurada como servidor web, brindando además los servicios de punto de acceso inalámbrico con direcciones dinámicas y por medio de una aplicación a la medida servirá de ruteador/proxy entre los dispositivos inalámbricos y la salida a Internet. En la Fig. 1(b) se detallan los módulos de software principales del sistema. En el presente trabajo se detalla una metodología para probar la capacidad de carga que el servidor web es capaz de soportar para una determinada configuración de software y hardware, donde las opciones son cuatro diferentes tarjetas (dos Soekris y dos Alix) y como sistema operativo Debian y OpenBSD. Las pruebas se restringieron a comparar una de las tarjetas Soekris y una Alix usando el sistema operativo OpenBSD, porque en esta etapa del trabajo fue necesario desarrollar una metodología y una estrategia, que permitieran arrojar resultados confiables para hacer las comparativas. Es importante mencionar que para poder hacer las comparativas entre las diferentes pruebas es necesario aislar la mayor cantidad de variables porque de lo contrario el tiempo medido para dos pruebas idénticas arroja dos valores diferentes, convirtiéndose en una fuente de error.

Un elemento importante de las pruebas, como las que se proponen en el presente trabajo, es el “benchmarking”, que se define como el proceso de comparar dos o más sistemas mediante la obtención de medidas [7]. Estas pruebas proporcionan diversos resultados que dan apoyo a la toma de decisiones para elegir entre dos o más sistemas, herramientas, o para determinar cuál de ellas funciona mejor, tal como lo hacen las pruebas de estrés (stress testing), carga (load testing) y de rendimiento (performance testing) [10]. Para escoger o diseñar un buen conjunto de pruebas “benchmark” deben de seguirse algunos pasos como los descritos en [11, 12, 13]. Para automatizar el proceso de las pruebas se requiere del uso de una herramienta especializada para servidores web, como: ApacheBench (ab), Webserver Stress Tool, Visual Studio Ultimate, Jetstress, entre otros.

Dentro del apartado de desarrollo se describe la metodología propuesta que incluye objetivo, la selección de la herramienta, entorno y diseño de las pruebas base. En la sección de

resultados se presentarán las gráficas obtenidas con las pruebas diseñadas al aplicar la metodología propuesta. En el apartado de discusión se hace un análisis de aquellos parámetros que fueron considerados y su impacto en los resultados. En las conclusiones se resumen las principales aportaciones de la metodología y se sugiere como trabajo futuro, el diseño de algunas pruebas con parámetros adicionales y la inclusión de otras herramientas de medición que permitan robustecer el criterio de selección final.

## 2. Desarrollo

La metodología que se propone en el presente trabajo se basa en aquellas que se usan para hacer pruebas de carga sobre servidores web. En este caso nos interesamos en medir el rendimiento del servidor web sometido a diferentes tipos de carga y hacer una comparativa de su comportamiento. La metodología consiste en definir un objetivo, seleccionar una herramienta que nos ayude a cumplir dicho objetivo, definir el escenario considerando las variables básicas que pudieran afectar las pruebas y por último diseñar las pruebas mínimas que se harán al servidor y que pudieran servir para hacer una comparativa inicial y establecer los parámetros base requeridos. A continuación se listan los elementos que forman parte de la metodología:

### a) Objetivo

Diseñar un conjunto de pruebas de carga a un servidor web montado sobre un sistema empotrado, para obtener el mejor rendimiento basado en la capacidad de carga y menor tiempo de respuesta promedio por petición.

### b) Selección de la herramienta

Probar el desempeño de los servidores web es una labor compleja por la cantidad de variables a considerar tal como tiempo y recursos. Por tal motivo organismos [14], empresas e instituciones se encuentran en constante desarrollo de metodologías y herramientas de monitoreo. Como ejemplo en [15] se describen algunas de estas para el servidor Exchange 2003, mientras que en [16,17] se exponen las que utilizan la plataforma Studio de Microsoft©, o bien como las que se describen en [18]. Todas estas herramientas ponen a prueba diferentes elementos dentro de los servidores, por lo que es necesario evaluar si dichas herramientas tienen soporte para realizar pruebas de rendimiento (performance testing), de carga (load testing) o de estrés (stress testing), descritas en [19], según sea el caso.

En nuestro caso realizamos pruebas de carga con diferentes entornos, por lo que fue necesario realizar una comparativa entre distintas herramientas que ayudarán a cumplir el objetivo de las pruebas. Las características discriminatorias que se utilizaron para la selección son: tipo de licencia, el sistema operativo que soporta la herramienta para su instalación, el número máximo de conexiones soportadas promedio y el tipo de reportes que arroja la herramienta (pudiendo ser en formato gráfico, personalizado o bien que sea compatible con el manejado por hojas de cálculo). En la tabla 1 se resumen los resultados obtenidos en dichas características.

| **Herramienta** | **Licencia** | **Sistema operativo** | **Conexiones soportadas** | **Formato de reportes[1]** |
|---|---|---|---|---|
| Webserver Stress Tool[20] | Con costo | Windows XP o posterior | 10.000 | Texto y gráfico. |
| JMeter [21] | FreewareOpensource | Cualquier SO con Java | --- | Texto y gráfico(limitada) |
| OpenSTA | FreewareOpensource | Windows XP o posterior | --- | Grafico |
| ApacheBench (ab) [22] | FreewareOpensource | Cualquiera | 100,000 | Texto y gráfico (limitada). |

[1] La mayoría incluye ancho de banda, número de peticiones, tasa de error, tiempo de conexión y petición, entre otros

**Tabla 1. Comparativa de herramientas para realizar pruebas de carga en servidores web.**

De acuerdo con los resultados de la tabla y al objetivo planteado, se decidió usar la herramienta ApacheBench (ab) porque los resultados que arroja corresponden con el tipo de variables que se desean medir (tiempo de respuesta promedio por petición) y porque permite simular la concurrencia.

### c) Entorno de prueba

Las pruebas de carga, permiten evaluar el comportamiento de una aplicación de software bajo la influencia transaccional de determinado número de usuarios. En la industria, predecir cómo se comportará una aplicación con niveles de concurrencia específicos es de suma importancia.

Dentro del presente trabajo las pruebas se realizaron sobre un sistema empotrado que actúa como un punto de acceso inalámbrico que brinda el servicio DHCP y que tiene montado un servidor web Apache. La descripción del entorno de las pruebas se presenta a continuación.

**Configuración del sistema**

- Tarjetas Soekris 5501 y Alix: las características de las tarjetas pueden encontrarse en [23, 24].
- Tarjetas Compact Flash como almacenamiento primario.
- Tarjeta de red inalámbrica CM9-GP configurada como punto de acceso inalámbrico.
- Servidor HTTP: Apache 1.26

- **Sistema operativo:** OpenBSD.
- **Tipo de archivo de prueba**
  - Página de inicio del servidor Apache 2.2KB

- **Software para realizar las pruebas:** ApacheBench, Versión 2.3
- **Consideraciones especiales para hacer las pruebas:** La estrategia para hacer las pruebas consistió en tratar de aislar algunos parámetros como el tráfico de la red, las variaciones que resultan de la carga a la que se somete la máquina que se usa como cliente y las que naturalmente resultan al hacer pruebas con diferentes parámetros que nos permitan darnos una idea de cómo se comporta el servidor. La configuración de la red se realizó de la siguiente forma: buscando aislar el tráfico se decidió crear una subred sin salida a Internet con la subred 10.10.10.x, y se configuró el servidor DHCP para servir direcciones en el rango 10.10.10.1 a 10.10.10.254, la dirección del servidor (sistema empotrado) se estableció de manera fija en 10.10.10.1. Esto fue necesario para reproducir el ambiente de producción requerido. Se aisló de Internet porque de ese modo pudimos medir el rendimiento del servidor a su nivel máximo de carga buscando minimizar el efecto del tráfico en la red. Con esto, se obtuvo el menor tiempo de respuesta considerando diferentes niveles de concurrencia.

**d) Diseño de las pruebas**

Para esta etapa, se diseñaron 13 pruebas que miden el número máximo de peticiones y el número máximo de conexiones simultáneas que soporta el servidor web. Dentro de los parámetros que se probaron están la ejecución del comando **ab** con una sola máquina (cliente).

Las pruebas se diseñaron usando un documento HTML (página de bienvenida de Apache) como objetivo de las peticiones. Dado que las pruebas nos ayudarán a seleccionar los parámetros base, no se probó con diferentes tipos de archivos y/o páginas dinámicas. Las 12 primeras pruebas se hicieron para la tarjeta Alix2d3, mientras que la prueba 13 se realizó en ambas tarjetas (Alix y Soekris). Estas pruebas sirvieron para probar el comportamiento del

servidor con distintos niveles de concurrencia y de peticiones, con lo que se probó su respuesta a distintos niveles de carga. La tabla 2 resume las principales características de las pruebas realizadas.

| Núm. de Prueba | Núm. de Peticiones | Concurrencia |
|---|---|---|
| **1** | 500 | 10 |
| **2** | 500 | 50 |
| **3** | 500 | 100 |
| **4** | 1000 | 10 |
| **5** | 1000 | 50 |
| **6** | 1000 | 100 |
| **7** | 10000 | 10 |
| **8** | 10000 | 50 |
| **9** | 10000 | 100 |
| **10** | 10000 | 150 |
| **11** | 50000 | 50 |
| **12** | 50000 | 100 |
| **13** | 50000 | 150 |

**Tabla 2. Parámetros de concurrencia y número de peticiones para cada prueba realizada.**

La primera columna especifica el número de prueba que sirve para identificar las gráficas que se muestran en el apartado de resultados. Las dos últimas columnas indican los parámetros introducidos en la herramienta.

## 3. Resultados

Una vez finalizadas las pruebas se realizó un análisis para evaluar que parámetros son los más significativos considerando lo tiempos de: promedio de conexión, de procesamiento, de espera y total de petición.

De este análisis se concluye que las pruebas que permiten realizar las comparativas son:

a) Las primeras 12 pruebas se hicieron con la página de bienvenida de Apache (2.2KB), solo para probar de manera incremental el número de peticiones y el valor máximo del nivel de concurrencia que el servidor es capaz de soportar. Como resultado de esta prueba se comprobó que el nivel de concurrencia máximo es de 150 peticiones simultáneas que corresponden al valor de maxclients configurado por default en Apache. También fue importante medir el valor del número total de peticiones antes de que la herramienta ab marcara algún error como conexión restablecida por el servidor o tiempo de espera agotado.

   Pudimos darnos cuenta que el tiempo entre conexiones, procesamiento y espera, se reparte de forma equitativa sin importar número de usuarios, o nivel de concurrencia, tal como puede observarse en la Fig. 2.

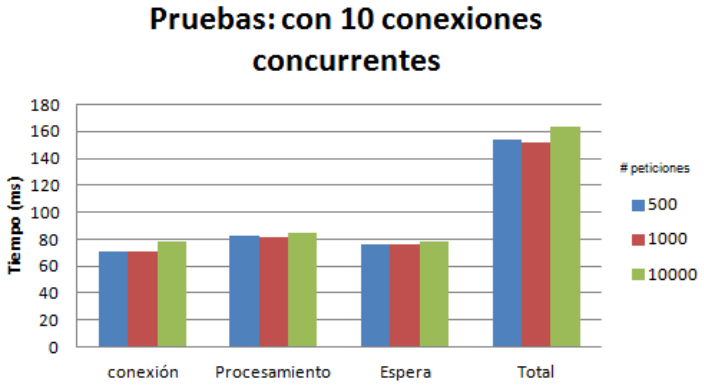



**Fig. 2. Tabla comparativa de las Prueba 1, 4 y 7.**

Lo anterior es importante debido a que aunque se esperaría que a mayor concurrencia para igual número de peticiones se debiera reducir el tiempo de atención, esto no se cumple exceptuando para las pruebas 12 y 13, tal como lo muestra en la Fig. 3(d), donde se ve una reducción de tiempo cuando tenemos 50000 peticiones y la concurrencia aumenta de 100 a 150.

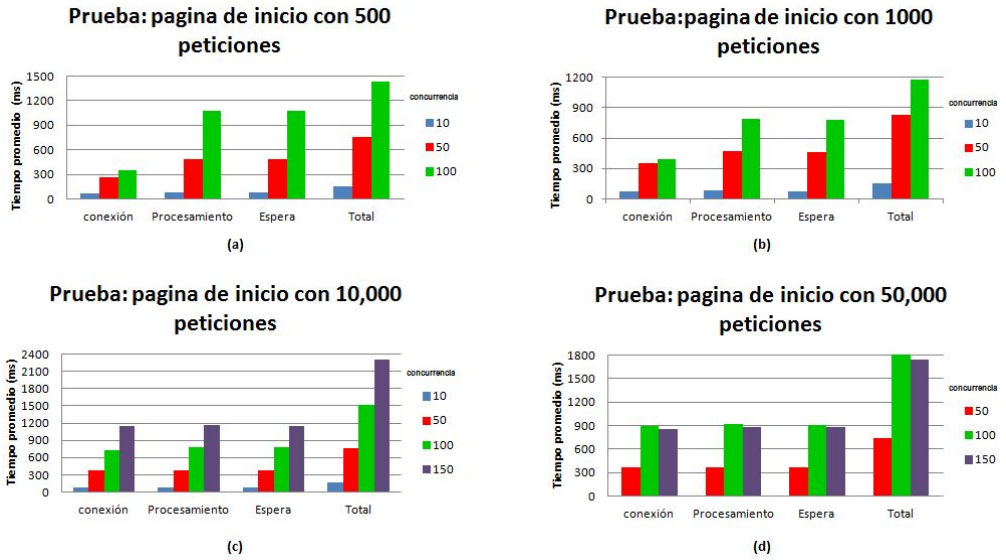



**Fig. 3. Graficas de resultados de las 13 pruebas. (a) 500, (b) 1000, (c) 10000 y (d) 50000 peticiones.**

b) La prueba 13 se usó como base para establecer el comportamiento del servidor respecto al tiempo requerido para completar un determinado porcentaje del número de peticiones, con 2 diferentes tarjetas. La siguiente tabla resume los valores obtenidos para las dos tarjetas para la prueba 13. Los valores en la tabla son el tiempo en milisegundos que se necesita para atender un cierto porcentaje de peticiones.

| % de peticiones atendidas | Alix 2d3 | Soekris 5501 256 RAM |
|---|---|---|
| **50%** | 2114 | 2448 |
| **66%** | 2262 | 2640 |
| **75%** | 2355 | 2755 |
| **80%** | 2428 | 2851 |
| **90%** | 2620 | 3158 |
| **95%** | 2838 | 3685 |
| **98%** | 3261 | 4919 |
| **99%** | 3567 | 5686 |
| **100%** | 65849 | 65487 |

**Tabla 3. Prueba 13 Página de bienvenida de Apache sobre el sistema operativo OpenBSD.**

Estos resultados muestran en la siguiente Fig.4 que las características de las tarjetas son importantes para dar respuesta a las peticiones enviadas. Tal como se ve, en esta prueba se probaron 2 tarjetas diferentes: Alix2d3 y Soekris 5501, donde el comportamiento al menos para esta prueba fue mejor para la Alix2d3, el cuál va mejorando en relación al número de peticiones que son atendidas.

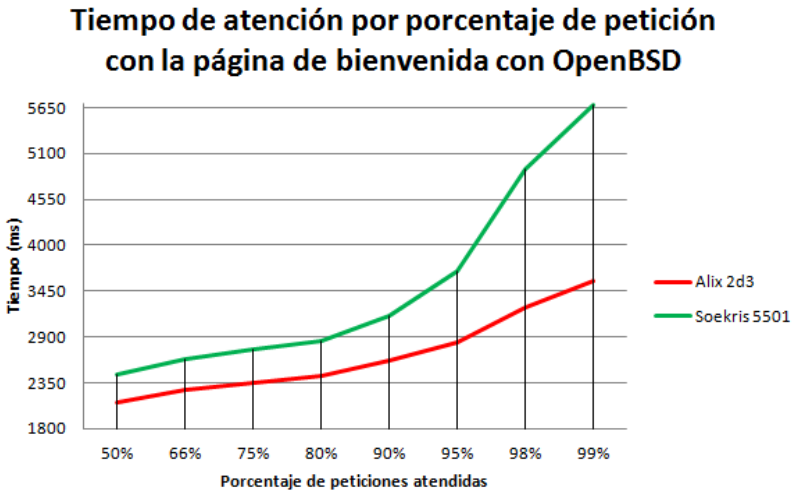



**Fig. 4. Prueba 13 Página de bienvenida de Apache con Alix2d3 y Soekris 5501.**

De las gráficas anteriores podemos concluir que el mejor rendimiento en tiempo por petición se obtiene con la tarjeta Alix 2d3, lo cual nos afirma que el hardware utilizado en las pruebas es un factor primordial cuando se usan sistemas empotrados.

## 4. Discusión

De lo anterior podemos concluir dado que el número de peticiones totales y el número de peticiones concurrentes no son suficientes para determinar la mejor configuración del sistema empotrado. Para tener un diagnóstico más preciso sobre el rendimiento de cada instalación en particular y el uso que se hace de los recursos (procesador y memoria), sería necesario correr algunas otras pruebas, e incluso en otras tarjetas y sistemas operativos, que escapan del objetivo del presente estudio. Esto permitiría hacer un ajuste más preciso de la configuración del software que saque el mayor provecho a los recursos de hardware.

## 5. Conclusiones

De acuerdo con los resultados obtenidos al aplicar la metodología propuesta podemos concluir que se tuvo éxito al conseguir el objetivo de diseñar las pruebas de carga y obtener parámetros para decidir cuál es la configuración más óptima para el hardware.

A partir de los resultados obtenidos pudimos darnos cuenta que es necesario probar con diferentes configuraciones de hardware y software para poder decidir cuál de ellas ofrece el rendimiento más óptimo, tal como se plantea en el objetivo de las pruebas. En este conjunto de pruebas no se observa un comportamiento distinto del servidor a distintos niveles de carga, por lo que será necesario probar con distintos tipos de archivo para evaluar la capacidad de carga del procesador y de la memoria, incluso sería aconsejable medir el uso de dicho recursos mediante las pruebas. Otro factor importante es el uso de la red, para ello es necesario controlar variables como el tráfico y en dado caso medir el mismo.

Estas mejoras se proponen como trabajo futuro y tienen la intención de arrojar resultados más precisos sin cambiar la aplicación de la metodología puesto que sólo se estaría cambiando el diseño de las pruebas. Es necesario hacer notar que las mejoras propuestas requieren del uso de herramientas adicionales y se tendría que evaluar cuáles son las más apropiadas.

## 5. Agradecimientos

Agradecemos al Consejo Nacional de Ciencia y Tecnología (Conacyt) por el apoyo brindado para la realización del presente trabajo, el cual fue financiado por medio del programa PROINNOVA con el número de proyecto 197758 de la convocatoria 2013. Así mismo agradecemos a la empresa LOGICALBRICKS quienes nos tuvieron la confianza para la realización del estudio citado, usando como base uno de sus desarrollos.

## 6. Referencias

[1] T. Noergaard, Embedded Systems Architecture, A Comprehensive Guide for Engineers and Programmers. Ed. Elsevier Inc., ISBN-13: 978-0750677929 e ISBN-10: 0750677929, 2005, páginas 5 y 6.

[2] Reporte de estudio de oportunidades para el sector sistemas embebidos del programa innovación orientada, presentado por la Fundación México-Estados Unidos para la Ciencia (FUMEC), http://fumec.org.mx/v6/htdocs/embebidos.pdf, Consultada por última vez en Agosto de 2014.

[3] Poster informativo "MOOSE Software Engineering MethOdOlogieS for Embedded Systems" presentado por Information Technology for European Advancement (ITEA), http://virtual.vtt.fi/virtual/proj1/projects/moose/docs/moose_2002_poster.pdf, Consultada por última vez en Agosto de 2014.

[4] D.Frost. “Pruebas de carga: Una perspectiva nueva cargando”. Linux Magazine, No. 84, Agosto de 2012, España; páginas. 10-12. Publicada por Linux New Media Spain S.L.; ISSN edición impresa 1576-4079, e ISSN edición online 1699-2237 http://www.linux-magazine.es/issue/84/010-012_PruebasdeCarga_LM84.pdf, Consultada por última vez en Agosto de 2014.

[5] “Performance Testing Guidance for Web Applications”. http://msdn.microsoft.com/en-us/library/bb924375.aspx. Consultada por última vez en Agosto de 2014.

[6] Documento técnico de Oracle: "Identificación rápida de cuellos de botella: Una mejor manera de realizar pruebas de carga", Junio de 2009 http://www.oracle.com/technetwork/es/oem/grid-control/documentation/identificar-cuello-bottella-rbi-427325-esa.pdf. Consultada por última vez en Agosto de 2014.

[7] M.A., Esbrí Palomares, J.V. Higón Valero, "Pruebas benchmark de soluciones cliente/servidor en software libre", Jornadas Técnicas de la IDE de España (JIDEE 05); Madrid, 2005; páginas 228-237 www.geotec.uji.es/pubs/ProceedingsJIDEE05.pdf, Consultada por última vez en Agosto de 2014.

[8] C. de J. Cardona Velásquez, "Propuesta metodológica para la realización de pruebas de software en un ambientes productivos"; Tesis digital; Medellín, Colombia, 04 de Junio de 2009. www.bdigital.unal.edu.co/930/1/8357252_2009.pdf, Consultada por última vez en Agosto de 2014.

[9] B. Afessa; M. T. Keegan, R.D. Hubmayr, J.M. Naessens, O. Gajic, K. H. Long, S.G. Peters, "Evaluating the performance of an institution using an intensive care unit benchmark", Mayo Clinic Proceedings, Elsevier Inc, vol. 80, Issue 2, Febrero 2005, páginas 174–180. DOI: 10.4065/80.2.174.

[10] P. J. Deitel, H. M. Deitel; "Ajax, Rich Internet Applications, and Web Development for programmers", Pearson Education. Inc; 2008. United States, Indiana. página 1025. ISBN-13: 978-0131587380 e ISBN-10: 0131587382

[11] K. Gilly, C. Quesada-Granja, S. Alcaraz, C. Juiz. R Puigjaner; "A Statistically Customisable Web Benchmarking Tool", Electronic Notes in Theorical Computer Science, Proceedings of the Third International Workshop on the Practical Application of Stochastic Modelling , 2009, Elsevier Inc, Vol. 232, páginas 89-99, ISSN 1571-0661.DOI:10.1016/j.entcs.2009.02.052.

[12] "Bench Web Services Performance Benchmark Study" por DOCULABS, 2003, http://people.cis.ksu.edu/~hankley/d764/j2ee/Pets06/DoculabsWebServiceScalability.pdf Consultada por última vez en Junio de 2014.

[13] I. Molyneaux, “The Art of Application Perfomance Testing”, Ed. O’Reilly Media, 2009, páginas 11-76. ISBN 978-0-596-52066-3.

[14] M.C. Ballou, “Driving Business Optimization with End-to-End Performance Testing”, http://h20195.www2.hp.com/V2/GetPDF.aspx%2F4AA4-8541ENW.pdf. Consultada por última vez en Agosto de 2014.

[15] N.Bilic, “Herramientas para poner a prueba el esfuerzo de rendimiento de los servidores de Exchange 2003”, Artículo técnico de Exchange Server del año 2006, http://technet.microsoft.com/es-es/library/aa996207(v=exchg.65).aspx, Consultada por última vez en Agosto de 2014.

[16] “Ejecutar pruebas de rendimiento en una aplicación antes de crear una versión de la misma”, artículo técnico de Microsoft Developer Network, http://msdn.microsoft.com/es-es/library/dn250793.aspx, Consultada por última vez en Agosto de 2014.

[17] “Probar el rendimiento y el esfuerzo mediante pruebas de carga y de rendimiento web de Visual Studio”, artículo técnico de la biblioteca de Visual Studio, http://msdn.microsoft.com/es-es/library/vstudio/dd293540(v=vs.110).aspx, Consultada por última vez en Agosto de 2014.

[18] Performance testing tools, http://www.opensourcetesting.org/performance.php, Consultada por última vez en Agosto de 2014.

[19] C.M. Zapata, C. de J. Cardona Velásquez, “Comparación de las características de algunas herramientas de software para pruebas de carga”, Julio de 2011, Revista Avances en Sistemas e Informática, Vol.8, No.2, ISSN 1657-7663 http://www.bdigital.unal.edu.co/28847/1/26734-93661-1-PB.pdf, Consultada por última vez en Agosto de 2014.

[20] Performance, Load, and Stress-Test for Web Servers, artículo de la compañía Paessler, http://www.paessler.com/webstress, Consultada por última vez en Agosto de 2014.

[21] Página oficial de la herramienta JMeter, http://jmeter.apache.org/, Consultada por última vez en Agosto de 2014.

[22] Página oficial de Apache Benchmark, http://httpd.apache.org/docs/2.2/programs/ab.html, Consultada por última vez en Agosto de 2014.

[23] Descripción de la tarjeta Soekris 5501, fabricante Soekris Engineering, Inc. http://soekris.com/products/net5501.html, Consultada por última vez en Agosto de 2014.

[24] Descripción de la tarjeta Alix2d3, fabricante PCEngines, http://pcengines.ch/alix2d3.htm, Consultada por última vez en Agosto de 2014.

## 7. Autores

M.R.C. Mónica Edith García García. Maestra en Redes de Computadoras. Es profesora de la Universidad Tecnológica de la Mixteca. Trabaja como responsable y miembro del cuerpo académico de Redes y Sistemas Distribuidos de la misma universidad. Su área de interés es Seguridad Computacional y Educación a Distancia.

Ing. Jorge Arturo Hernández Perales es Ingeniero en Electrónica y Comunicaciones por el Instituto Tecnológico de Estudios Superiores de Monterrey Campus Estado de México. Profesor de la Universidad Tecnológica de la Mixteca. Miembro del cuerpo académico de Redes y Sistemas Distribuidos. Sus áreas de interés incluyen las redes de computadoras, sistemas distribuidos y colaborativos, análisis y diseño de protocolos de comunicación.

Lic. María Esperanza Pérez Cordoba Sánchez es Licenciada en Informática por el Instituto Tecnológico de Puebla. Tiene la certificación en PSP por el Software Engineering Institute. Profesora de la Universidad Tecnológica de la Mixteca, miembro del cuerpo académico de Redes y Sistemas Distribuidos.