Claves para un correcto modelado y evaluación de arquitecturas web con prebúsqueda[1]

José A. Gil, Johann M. Marquez, Ana Pont

Instituto de Automática e Informática Industrial
Universidad Politécnica de Valencia, España
jagil@disca.upv.es, jomarba@doctor.upv.es, apont@disca.upv.es

Abstract Las técnicas de prebúsqueda para arquitecturas web están demostrando ser soluciones de gran interés para reducir el tiempo de descarga de las páginas web y conseguir de este modo navegaciones más placenteras. Sin embargo, los investigadores rara vez recogen en sus estudios los detalles de implementación y uso que hacen fiables y generalizables los resultados obtenidos. En este trabajo se presenta con detalle el funcionamiento de la técnica de prebúsqueda para la web así como aquellos aspectos más relevantes que se han de tener en cuenta a la hora de su correcto modelado y evaluación de prestaciones. Todo ello se ilustra además con varios casos de estudio que presentan el potencial de esta técnica para mejorar las prestaciones de la web.

Keywords: Evaluación de prestaciones, modelado, web, técnicas de prebúsqueda.

1. Introducción

En los últimos años venimos asistiendo a una evolución de la Web que va, desde un mero repositorio de páginas estáticas y bajo contenido gráfico, a una Web basada en aplicaciones, con páginas dinámicas de alta complejidad gráfica, una gran riqueza de media y a la que recientemente se le añaden funcionalidades basadas en Ajax. Estos cambios han provocado que el número de usuarios conectados crezca de manera vertiginosa, debido principalmente a la agregación de funcionalidades a estas aplicaciones, que son apreciadas como útiles por los usuarios finales. Asimismo, el ancho de banda que disponen los usuarios en la "última milla" se ha incrementado notablemente debido a la posibilidad de acceso mediante DSL desde los hogares y lugares de trabajo, lo que también ha contribuido a su mayor penetración social.

Sin embargo, la latencia de acceso a la Web, definida como el tiempo que transcurre desde que un usuario solicita una página hasta que ésta es representada por su cliente web o navegador, no ha mejorado sustancialmente en los últimos años [1]. La mayor mejora de latencia es la que proviene de las mejoras tecnológicas realizadas

[1]Este trabajo ha sido finanaciado parcialmente por:

- Ministerio Español de Educación y Ciencia y el European Investment Fund for Regional Development (FEDER) ayuda TSI 2005-07876-C03-01.
- Programa Alβan, the European Union Programme of High Level Scholarships for Latin America, scholarship No.E04D031142BO.

en la "última milla" (del orden de unos 110-150 ms de tiempo de retardo de un MODEM frente a los 5-8 ms de un router DSL).

Pese a estas mejoras, los usuarios deben seguir soportando latencias de páginas que oscilan entre 20 y 50 segundos. Estas altas latencias se deben a varios factores, uno de ellos estriba en que, pese a que es posible disponer de un ancho de banda mucho mayor en la última milla, el estándar HTTP, a fin de evitar la sobrecarga en el servidor, limita el número de conexiones TCP simultáneas con el mismo servidor, lo que puede impedir utilizar todo el ancho de banda disponible por el usuario. Otro factor es intrínseco al cambio de la filosofía de páginas estáticas a aplicaciones lo que añade un tiempo de procesamiento de página que en multitud de casos contiene varios accesos a bases de datos. También hay que considerar dentro de estos factores las nuevas aplicaciones basadas en TCP/IP (tales como las aplicaciones P2P de compartir archivos) que consumen una gran parte del ancho de banda disponible en Internet.

Existen varias técnicas que pueden ser utilizadas a fin de reducir la latencia observada por los usuarios, todas ellas explotan alguna de las tres vertientes de la propiedad de localidad que exhiben los accesos a los objetos web, esto es, localidad geográfica, localidad temporal, y localidad espacial [1], [2].

La localidad geográfica consiste en que determinados objetos web son demandados con mayor frecuencia por clientes que operan geográficamente próximos entre sí. Esta propiedad es aprovechada mediante las Redes de Distribución de Contenido (CDN), que replican contenidos y redirigen a los usuarios a los servidores geográficamente más cercanos.

La localidad temporal radica en que, una vez accedido un objeto web, existe una probabilidad bastante alta de que en un futuro cercano se vuelva a acceder al mismo objeto. Las técnicas de Caching intentan explotar esta propiedad almacenando temporalmente los objetos accedidos recientemente para no tener que volver a demandarlos al servidor original.

La localidad espacial se fundamenta en la existencia de patrones o secuencias de acceso a objetos web que se repiten con una alta frecuencia. La localidad espacial se aprovecha mediante técnicas de prebúsqueda ya que esta propiedad permite predecir futuros accesos basándose en accesos anteriores.

Las técnicas de prebúsqueda [2], [3] se basan en obtener los objetos web antes de que estos sean requeridos por el usuario, reduciendo la latencia observada por éste. Para ello existen dos componentes que cooperan entre sí: el motor de predicción y el motor de prebúsqueda. El motor de predicción es el que, mediante la observación de patrones de acceso, elabora las predicciones de accesos futuros, pasándoselas como consejos al motor de prebúsqueda. Este último es el que decide cuándo y cuáles de esos consejos se van a prebuscar. Estos motores se pueden ubicar en cualquiera de los elementos de la arquitectura de la web (Cliente, Proxy, Surrogate o Servidor), pudiendo encontrarse ambos en el mismo elemento o en elementos distintos. La eficacia del sistema de prebúsqueda depende en gran medida de la precisión alcanzada por el algoritmo de predicción, sin embargo, al ser una técnica esencialmente especulativa conlleva unos costes relativos a las prebúsquedas fallidas que se traducen principalmente en incremento del tráfico y sobrecarga del servidor.

En esta ponencia nos centraremos en los sistemas de prebúsqueda que es la línea que presenta en la actualidad una mayor actividad en cuanto a esfuerzo científico en temas de arquitectura de la Web.

2. Particularidades de la Prebúsqueda Web

Como se ha indicado anteriormente, los sistemas de prebúsqueda web se basan en la existencia de un motor de predicción y un motor de prebúsqueda, la configuración más habitual es ubicar el motor de predicción en el servidor original y el motor de prebúsqueda en el cliente. La Fig. 1 muestra el funcionamiento de este esquema tradicional en el que el usuario solicita el objeto A, cuando la petición llega al servidor éste aconseja la prebúsqueda de B, mientras el usuario visualiza A, su cliente prebusca el objeto B, si posteriormente el usuario requiere B, éste ya se encuentra en la cache del navegador, por lo que la latencia asociada a este objeto es prácticamente nula.

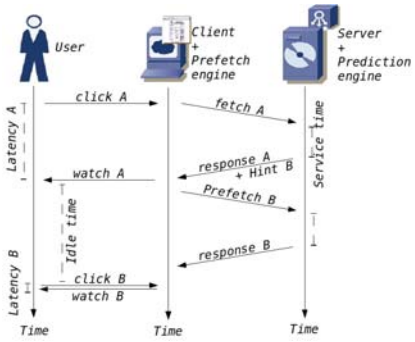


Fig. 1 Esquema tradicional de prebúsqueda

Nótese que no todas las predicciones son prebuscadas, ya que la decisión de prebuscar o no se toma en el motor de prebúsqueda atendiendo a las distintas políticas que se pudieran implementar. La Fig. 2 muestra la relación existente entre las peticiones realizadas por el usuario, las predicciones y las prebúsquedas.

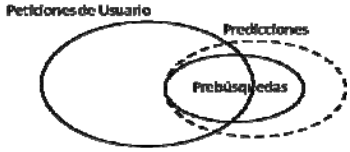


Fig. 2 Relación entre predicciones y prebúsquedas

$$Latency\_Ratio = \frac{Latency\_prefetching}{Latency\_not\_prefetching} \quad \textbf{(1)}$$

Existen dos índices básicos a la hora de evaluar las prestaciones de los sistemas de prebúsqueda web, la tasa de ahorro de latencia y la tasa de incremento de tráfico.

La tasa de latencia (ecuación (1)) mide los beneficios obtenidos por los usuarios cuando se utiliza la prebúsqueda frente a un sistema que carezca de esta técnica. Los valores, comúnmente inferiores a la unidad, son mejores cuanto menores sean. Puede ser calculada por objeto o por página, siendo preferible esta última, ya que supone una mejor aproximación a las exigencias de los usuarios.

La tasa de tráfico evalúa los costes que conlleva la aplicación de la prebúsqueda respecto de un sistema sin prebúsqueda (ecuación (2)). Cuando se calcula en base al número de peticiones da una idea del incremento de carga que sufre un servidor al

realizar prebúsqueda, mientras que si se hace en base a los bytes transferidos, se referiría al incremento de la carga en la red. Los valores son mejores cuanto menores sean en ambos casos.

$$Traffic\_Ratio = \frac{Client\_requests\_prefetching}{Client\_requests\_not\_prefetching} = \\ = \frac{User\_requests + Prefetchs\_not\_used}{User\_requests} \quad \textbf{(2)}$$

El corazón de los sistemas de prebúsqueda es el algoritmo de predicción, existiendo dos aproximaciones a la hora de abordar su diseño, predecir en base a los patrones de acceso observados previamente o predecir basándose en el contenido de las páginas accedidas más recientemente. En la primera categoría se encuentran algoritmos que utilizan la inferencia estadística, ya sea mediante modelos de Markov, tales como el PPM (Prediction by Partial Maching) [3], DG (Dependency Graph) [4] o DDG (Double Dependency Graph) [5], mediante técnicas de Web Mining o mediante otras técnicas estadísticas. Mientras que en la segunda se encontrarían tanto aquellos que consideran los enlaces contenidos en el código HTML [6]como los que se basan en palabras clave u otro tipo de etiquetado.

Existen algunas implementaciones comerciales con prebúsqueda, principalmente de la mano del proyecto Mozilla y de Google. Agentes de usuario basados en Mozilla tales como Firefox disponen incorporado un motor de prebúsqueda que es capaz de realizar prebúsquedas si se le suministran los consejos en la forma adecuada. Google Web Accelerator es una aplicación basada en un motor de prebúsqueda que permite que cualquier navegador se pueda beneficiar de las ventajas de esta técnica. Existe más software comercial que implementa técnicas de prebúsqueda, tales como Robtex, Viking Server, Mentat SkyX Accelerator, AllegroSurf, etc. pero a conocimiento de los autores existen pocos detalles de cómo realizan la prebúsqueda y en general esta se limita hoy por hoy a prebuscar los hipervínculos.

3. Consideraciones para el modelado y evaluación

Cuando se modelan y evalúan los sistemas de prebúsqueda web hay que tomar en consideración varias cuestiones que, asumidas en la fase previa del modelado del sistema o en la realización de los experimentos, pueden afectar a los resultados obtenidos posteriormente.

3.1. El modelo del sistema

Para evaluar cualquier sistema, una de las decisiones más importantes a considerar es a cerca de la técnica a utilizar: modelo analítico, simulación o medición.

La toma de medidas sobre el sistema real tiene como principales inconvenientes que el sistema tiene que existir previamente, que la carga a la que se someta deba ser lo suficientemente representativa, que el tiempo que se tarda en obtener los resultados es considerablemente elevado así como su coste económico y que los experimentos son difícilmente reproducibles.

El modelado analítico permite, mediante la utilización de fórmulas y ecuaciones, hallar a partir de los valores conocidos o estimados de ciertos parámetros los valores que nos interesa evaluar de la forma más rápida. Sin embargo, requiere realizar multitud de supuestos y simplificaciones que pueden restar validez a los resultados.

La técnica más utilizada para el modelado de sistemas informáticos es la simulación, pues permite la reproducción fiel de los aspectos más relevantes del sistema obteniendo resultados de una forma rápida y razonablemente fiable.

En nuestros trabajos, para el modelado de arquitecturas web con prebúsqueda se han utilizado varias de estas técnicas, desde la medición en un sistema real [7], hasta un modelado híbrido con parte real y parte simulada [8].

3.2. La carga de trabajo

La carga con que se alimenta el sistema a evaluar es un factor primordial al realizar los experimentos, ya que en ella viene recogida una serie de características intrínsecas tanto del contenido del sitio como del comportamiento de los usuarios. Existen dos aproximaciones a la hora de generar carga: cargas sintéticas o cargas reales.

Las cargas sintéticas se generan a partir de patrones conocidos de cargas reales. Los generadores de carga sintética son relativamente sencillos de implementar si se conocen previamente las características estadísticas de las cargas reales y capaces de generar el volumen necesario de carga, sin embargo, en la evaluación de arquitecturas web es muy complejo reproducir las características estadísticas que reflejen la localidad geográfica, espacial y temporal de las referencias.

Las cargas reales son las más utilizadas en la evaluación de sistemas web. En la mayoría de casos se basan en trazas previamente recogidas en los elementos de la arquitectura y que se utilizan para reproducir experimentos, aunque en la mayoría de los estudios que se encuentran en la bibliografía no son suficientemente representativas por el número de accesos que incluyen.

Cuando se trata de evaluar sistemas de prebúsqueda web mediante reproducción de trazas hay que poner un especial cuidado en las características de las mismas. Por ejemplo, las trazas de acceso a la web de un periódico, al renovar sus contenidos a diario, tendrán una alta variabilidad en los contenidos accedidos, poniendo a prueba el aprendizaje del motor de predicción.

Puesto que comúnmente las trazas contienen los accesos a objetos web sencillos, si se desea estudiar la latencia por página, habrá que realizar un preprocesado previo de la traza para organizar los accesos por clientes/sesiones/páginas, a fin de poder extraer posteriormente estadísticas adecuadas.

3.3. Cuándo y cómo se realiza la prebúsqueda

Escoger el momento en el que la lista de consejos que da el motor de predicción comenzará a prebuscarse también es una cuestión principal, ya que, comenzar inmediatamente desde que estos se reciben puede interferir con la carga de los objetos de la página actual. Lo ideal es esperar a prebuscar hasta que la página actual se haya cargado en su totalidad, para que la prebúsqueda no interfiera con la navegación del usuario, ya que de otra manera podríamos encontrarnos con que la latencia por página aumenta en lugar de disminuir. Esta técnica es la que se emplea en el navegador Mozilla Firefox[9].

Asimismo, hay que definir cuándo se debe borrar la lista de consejos, en Mozilla Firefox se realiza cada vez que debido a una acción del usuario haya que buscar un nuevo documento, dado que ya no son relevantes, y que se obtendrán nuevos consejos junto con las respuestas a la petición actual.

También hay que tener presente qué consejos serán tenidos en cuenta en la prebúsqueda. Por ejemplo, en Mozilla Firefox los consejos suministrados en los objetos embebidos no se añaden a la lista de prebúsqueda y por lo tanto no son prebuscados, solamente se prebuscan aquellos consejos que se suministran junto al objeto contenedor (HTML). Y otra cuestión distinta es qué ocurre con los consejos obtenidos en peticiones de prebúsqueda, se pueden añadir a la lista de prebúsqueda en el instante en que se reciben o en un instante posterior (cuando se produzca un acierto sobre el objeto donde han sido suministrados)

Obsérvese que, debido a las restricciones mencionadas anteriormente, no todas las predicciones se convierten en prebúsquedas.

3.4. Cómo se modela el tiempo de pensar.

El tiempo de pensar es el tiempo que el usuario necesita para procesar la información contenida en una página web. Es el tiempo que transcurre entre la representación de la página actual y la solicitud de una nueva página. Es como si el tiempo del usuario estuviera ranurado en tiempos de pensar y tiempos de latencia, alternativamente. En estos tiempos de pensar es en los que comúnmente realizan las prebúsquedas los navegadores.

Cuando se utilizan trazas reales para la evaluación, en las mismas generalmente figura un campo en el que se indica el instante de tiempo en el que se realizó cada petición junto con el tiempo de servicio de la misma. Se pueden utilizar estos tiempos para determinar el instante de tiempo en que cada usuario solicita una nueva página. Sin embargo, conducir una simulación en un entorno de prebúsqueda, respetando únicamente estos tiempos es erróneo. Hay que respetar lo que realmente constituye el tiempo de pensar, ya que los éxitos de las prebúsquedas reducen la duración de las ranuras de latencia. Si se preservan los tiempos originales de solicitud de página, los éxitos de las prebúsquedas estarían incrementando erróneamente el tiempo de pensar.

Respecto al tiempo de pensar existe un valor que no puede ser extraído de las trazas y que debe ser estimado. Es el valor del último tiempo de pensar de una sesión, el que transcurre desde que el usuario comienza a visualizar la última página hasta que abandona el sitio, ya por que cierra el navegador, ya por que pasa a navegar por otro sitio. Obsérvese que este tiempo juega un papel muy importante en la evaluación de los sistemas de prebúsqueda, ya que los objetos prebuscados en él no van a ser útiles, incrementando el tráfico sin contribuir al decremento de la latencia y esto ocurre una vez por sesión.

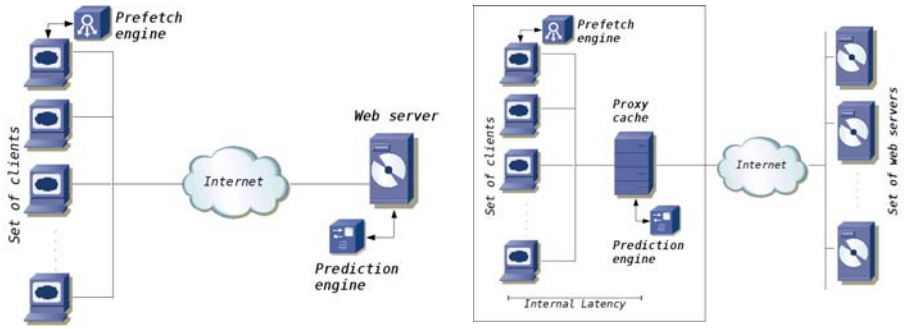
3.5. Tipo de conexión HTTP

El estándar HTTP 1.1 admite tres formas de gestionar las conexiones: conexiones no-persistentes, conexiones persistentes y pipelining. Cuando se utilizan conexiones no-persistentes la conexión TCP utilizada se abre y cierra para la transferencia de cada objeto web, pudiendo existir hasta 4 conexiones simultáneas con el mismo servidor. Las conexiones persistentes permiten la reutilización de las conexiones TCP para

transferir varios objetos, estableciéndose que cada petición debe esperar a que se haya completado la respuesta a la anterior, en este caso solo se permiten un máximo de 2 conexiones simultáneas con el mismo servidor. Finalmente, cuando se utiliza pipelining se descarta el requerimiento de que cada petición deba esperar a la respuesta de la anterior, debiendo el servidor enviar las respuestas en el mismo orden en el que llegaron las peticiones; solamente se permite una conexión TCP con el servidor cuando se realiza pipelining. Las latencias de página obtenidas mediante la utilización de estos tres tipos de conexiones son distintas [10].

4. Casos de estudio

En el presente trabajo, con el objetivo de poder analizar y comparar de forma objetiva los distintos sistemas de prebúsqueda y algoritmos de predicción se ha utilizado un entorno de simulación flexible, alimentado por trazas, que admite la simulación de los distintos elementos de la arquitectura web, así como las distintas configuraciones posibles a la hora de ubicar los motores de predicción y prebúsqueda en ellos. En la Fig. 3 se muestran dos ejemplos de arquitectura, una tradicional (Fig. 3 (a)) con el motor de predicción en el Cliente y el de predicción en el Servidor, y una arquitectura alternativa (Fig. 3 (b)) en la que el motor de predicción se ha ubicado en el Proxy.



(a) (b)

Fig. 3 Ejemplos de arquitecturas de prebúsqueda web

Como carga de trabajo, se han escogido dos trazas; ambas provienen de los accesos de los usuarios del Proxy de nuestra universidad. La primera de ellas contempla la totalidad de los accesos, mientras que en la segunda se han filtrado los accesos al servidor del diario Marca accesible por la Web. Las características de ambas trazas se muestran en la Tabla 1. Las trazas han sido adaptadas para alimentar la simulación identificando usuarios, sesiones de navegación, agrupando los objetos por páginas y calculando el tiempo de pensar entre páginas.

Para el tiempo de pensar de la última página de una sesión se ha considerado el caso peor, esto es, que el cliente permanece activo el tiempo suficiente para que todos los consejos suministrados puedan ser prebuscados ya que ello ocasiona tráfico pero no redunda en ningún beneficio de reducción de la latencia. En los resultados se ha considerado la utilización de conexiones persistentes sin pipelining como compromiso entre las prestaciones de las distintas posibilidades. Los motores de predicción suministran consejos en todas las peticiones que no sean de prebúsqueda,

independientemente de que estas sean de objetos embebidos o no, y los motores de prebúsqueda realizan la prebúsqueda durante el tiempo de pensar sin distinguir si los consejos se obtienen de objetos embebidos o de objetos contenedores.

Tabla 1. Características de las trazas utilizadas

Traza	Proxy	Marca
Fechas	9-12/Marzo/2007	9-12/Marzo/2007
Nº. Accesos	7.324.698	423.559
Nº Páginas	1.326.033	29.942
Media de obj. por pág.	4,52	14
Nº de Usuarios	7.987	1180
Bytes transferidos (GB)	107,3	2,06
Tamaño medio obj. (KB)	15,87	5,10
Tamaño medio pag. (KB)	71,73	75,93
Tamaño medio HTML (KB)	8,82	14,52
Tamaño medio imag. (KB)	17,12	4,38

Para obtener los índices relativos de tasa de ahorro de latencia y tasa de incremento de tráfico se ha realizado una primera simulación de un sistema sin prebúsqueda respecto al cual se relativizan el resto de valores.

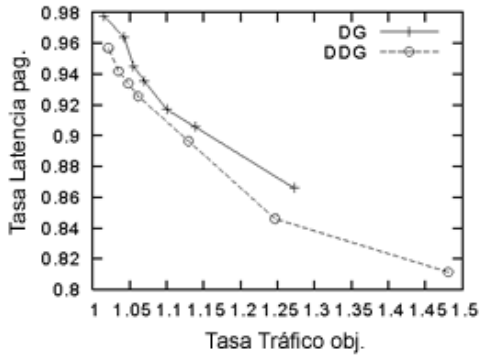


Fig. 4 Comparación de DG frente a DDG para la traza de Marca

4.1. Arquitectura tradicional

El primer caso de estudio es el de la arquitectura tradicional (Fig. 3 (a)) con el motor de predicción en el Cliente y el de predicción en el Servidor. Se ha realizado un estudio comparativo de dos algoritmos de predicción DG (Dependency Graph) [4] y DDG (Double Dependency Graph) [5]. Para obtener los distintos puntos de los resultados se ha variado la agresividad de los algoritmos modificando el parámetro relativo al umbral de confianza de las predicciones. Las referencias mencionadas proporcionan un mayor detalle del funcionamiento de los algoritmos.

La Fig. 4 muestra en términos de coste-beneficio los resultados de las simulaciones, observándose claramente que el DDG es preferible en este entorno.

4.2. Predicción en el Proxy

En el segundo caso de estudio se presenta una arquitectura que incluye un servidor intermedio o Proxy (Fig. 3 (b)) en la que el motor de predicción se ha ubicado en este elemento. En este caso se intenta actuar sobre la latencia de la "última milla". Para ello se estudian varias tecnologías, conexión vía Ethernet 100 Mbps, router Wi-Fi

sobre una conexión ADSL, tecnologías de telefonía celular UMTS y GPRS y conexión mediante módem de 56Kbps, para cada uno de estos métodos de conexión se realiza una primera simulación sin el Proxy a fin de relativizar las medidas.

El motor de predicción solamente suministra consejos de aquellos objetos que se encuentran en la cache, ejecutando el algoritmo DDG, como umbrales de confianza se utilizó 0,3 para los nodos primarios y 0,2 para los secundarios. En cuanto al tamaño de la cache se ha supuesto infinito (111 GB según la traza estudiada).

La Tabla 2 muestra las tasas de latencias obtenidas cuando se considera únicamente el caching y considerando el caching con prebúsqueda. Se puede constatar que ambos son beneficiosos en cualquier caso, siendo más beneficiosos cuanto mayor es la latencia de la "última milla". Cuando se realiza prebúsqueda además de caching, la tasa de latencia mejora respecto de sólo utilizar caching, si bien ésta no es mucho mejor que la del caching, ya que se está reduciendo únicamente la latencia de la "última milla".

Tabla 2. Tasas de Latencia arquitectura con Proxy

	Conexión	Caching	Caching + Prebúsqueda
Tasa Latencia pag.	Ethernet 100Mbps	0.747	0.734
	Wi-Fi	0.734	0.721
	UMTS	0.673	0.657
	GPRS	0.681	0.665
	Modem 56Kbps	0.646	0.594
Tasa Latencia obj.	Ethernet 100Mbps	0.717	0.701
	Wi-Fi	0.690	0.673
	UMTS	0.619	0.597
	GPRS	0.628	0.607
	Modem 56Kbps	0.615	0.561

Tabla 3. Efectos sobre el tráfico en bytes y objetos

		Ethernet 100Mbps	Wi-Fi	UMTS	GPRS	Modem 56Kbps
Reducción tráfico (Proxy-Servidores)	Bytes	28.06%	28.04%	28.06%	28.04%	28.01%
	Obj.	40.04%	40.03%	40.04%	40.03%	40.02%
Incremento tráfico ("última milla")	Bytes	12.33%	11.21%	10.72%	10.66%	8.71%
	Obj.	23.60%	20.56%	18.01%	17.84%	15.33%

En la Tabla 3 se muestran los efectos sobre el tráfico tanto en bytes como en objetos transferidos Respecto del tráfico, por tratarse de un proxy cache, la carga de los servidores finales se ve reducida en un 40% debido al caching, se habilite o no la prebúsqueda, mientras que el tráfico en bytes desde/hacia Internet se ve reducido en un 28%. Cuando se contempla la prebúsqueda, la carga adicional de peticiones que debe soportar el proxy va desde un 23,6% en el caso de conexiones Ethernet hasta un 15,33% en el de las conexiones por módem. El hecho de que el incremento de tráfico de la "última milla" debido a la prebúsqueda sea mayor cuanto más ancho de banda se dispone se debe a que, al suponer inalterados los tiempos de pensar, es posible realizar más prebúsquedas cuanto mayor sea el ancho de banda y menor la latencia.

5. Conclusiones

En este trabajo se ha presentado la técnica de prebúsqueda para la web como una alternativa para reducir la latencia final percibida por los usuarios y mejorar por tanto

la calidad de las navegaciones. Se han realizado una serie de consideraciones que deben ser tenidas en cuenta a la hora de la evaluación de los sistemas de prebúsqueda mediante técnicas de simulación, ya que los supuestos realizados, la carga de trabajo utilizada, etc. pueden afectar a la fiabilidad y generalización de los resultados obtenidos posteriormente.

Asimismo se han presentado dos casos de estudio. El primero evalúa en una arquitectura de prebúsqueda tradicional el comportamiento de dos algoritmos de predicción encontrándose que a igualdad de coste el algoritmo DDG presenta una mayor reducción de la latencia.

Cuando se consideran arquitecturas de prebúsqueda con el motor de predicción en el proxy cache, la prebúsqueda unida al caching mejora las tasas de latencia observadas por los usuarios, esta mejora es tanto más acusada cuanto mayor es la latencia de la "última milla". Si bien la prebúsqueda supone una carga adicional de tráfico, esta solo deberá ser soportada por el proxy.

Bibliografía:

[1] Cheshire, S.: Latency and the Quest for Interactivity. White paper for the Synchronous Person-to-Person Interactive Computing Environments Meeting, San Francisco, November (1996).

[2] Rabinovich, M. ,Spastscheck, O.: Web Caching and Replication. Addison-Wesley, Reading, MA. (2002)

[3] Fan, L., Cao, P., Jacobson, Q.: Web prefetching between low-bandwidth clients and proxies: potential and performance. Proc. of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Atlanta, USA, (1999).

[4] Padmanabhan, V.N., Mogul, J.C.: Using Predictive Prefetching to Improve World Wide Web Latency. Proc. of the ACM SIGCOMM '96 Conference, Palo Alto, USA, (1996)

[5] Domènech J., Gil J.A, Sahuquillo J., Pont A.: DDG: An Efficient Prefetching Algorithm for Current Web Generation. Proceedings of the 1st IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb). (2006)

[6] Davison, B. D.: Predicting web actions from HTML content. Proceedings of the 13th ACM Conference on Hypertext and Hypermedia. College Park, USA (2002)

[7] de la Ossa, B., Gil, J.A.,. Sahuquillo, J. , Pont, A.: Delfos: the Oracle to Predict Next Web User's Accesses. IEEE 21st International Conference on Advanced Information Networking and Applications (AINA'07), 2007

[8] Domènech J., Pont A., Sahuquillo J., Gil J.A.: An experimental framework for testing web prefetching techniques. 30th EUROMICRO Conference, 2004

[9] Fisher, D., Saksena, G.: Link prefetching in mozilla: A server driven approach. Proc. of the 8th International Workshop on Web Content Caching and Distribution (WCW 2003), (2003)

[10] Padmanabhan, V.N., Mogul, J.C.: Improving HTTP Latency. Computer Networks and ISDN Systems, v. 28, pp. 25-35, Dec. (1995).