Prefetching DNS Lookup for Efficient Wireless WWW Browsing

Michael Nidd, Thomas Kunz, and Ertugrul Arik
Department of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1

Abstract

In our opinion, a significant barrier to the popularisation of wireless computing is the frustration that results from the slow connection speed. In the case of browsing the world wide web, the delay between requesting a document, and having it start to appear on your screen can be frustrating. A significant part of this delay, sometimes more than ten seconds, results from attempts to convert URLs from domain name form to IP addresses. Therefore, in an attempt to reduce this part of the delay, we have experimented with an HTML proxy that resides near the wireless basestation, identifies all hotlinks being sent to the mobile, and prefetches DNS resolutions for them. This single modification to the implementation of a standard architecture can often reduce the time between a request being made, and the first pieces of the document being received by five seconds or more.

This paper explains the design of the proxy that implements this strategy, the experiments used to evaluate its effectiveness, and the results of those experiments.

1 Overview

Time spent web browsing is split between reading documents and fetching them. Since most browsers display information as it is retrieved, the phase that most contributes to user-frustration is waiting for the first bits of a document to arrive. A critical issue in designing distributed applications for use over slow network links, which are what wireless provides, is keeping this frustration level low. This paper focuses on a technique for doing that by reducing the time between a document request, and the appearance of the first part of that document on the screen.

In our experience, most URLs identify Internet hosts in domain name form, Before a connection request can be issued, this name must be converted to an IP address, e.g. converting "big.uwaterloo.ca" to 129.97.40.17. This conversion is done through a DNS lookup that, if the address is not cached at the local name server, can often take several seconds. Initiating these lookups before the corresponding documents are requested does not significantly

increase network traffic, and increases perceived performance by reducing the time from a request being issued to the first data bits being received by the client.

A standard design of Web browsers for wireless environments splits the browser functionality into two parts: a user interface, executing at the mobile, and a proxy, executing on a machine that is connected to the wired network and that is close to the base station with which the mobile communicates. Our research is based on this architecture, and enhances the proxy's functionality with prefetching capabilities to reduce the apparent latency of a browsing session. For purposes of this experiment, our proxy server scans all HTML documents for URL links (identified by the HTML "<A...>" tag). For every tag identified, a DNS resolution request is sent to the local name server, and the reply is cached by the proxy. Should the client request something from one of these prefetched addresses, the DNS lookup can hope either to be instant (if the reply has already been received and cached,) or to be sped up (if the request is still pending at the local name server). DNS requests are small, and are causing extra traffic only on the wired part of the network, so we do not consider this antisocial behaviour.

The following sections describe the design of the "prefetching proxy" tool that we used to implement this technique, our experiment to evaluate its effectiveness, and the results of that experiment.

2 The Prefetching Proxy

The proxy is designed to reside on the wired part of the Internet, logically close to (ideally, one hop from) the wireless client application. The proxy establishes HTTP links on behalf of the client, and channels the retrieved documents from the Internet. In our application, the proxy scans these documents while relaying them to the client, and extracts all hyperlink addresses as they go by.

Each address found in the document is passed to a second thread that packages it into a DNS lookup request to the local name server. The replies to these requests are received by a third thread, and inserted into a hash table.

When a client requests an HTTP connection, the proxy checks its hashed list of addresses before doing a standard DNS lookup. If the address is already in the hash table, then the lookup is not necessary.

3 Experimental Design

Method

The first stage of the experiment was to gather lists of valid URLs from typical web-browsing sessions. This was done by using Netscape's ability to connect to a proxy server, redirecting all requests through our data-gathering program. This program recorded the following information for each request issued:

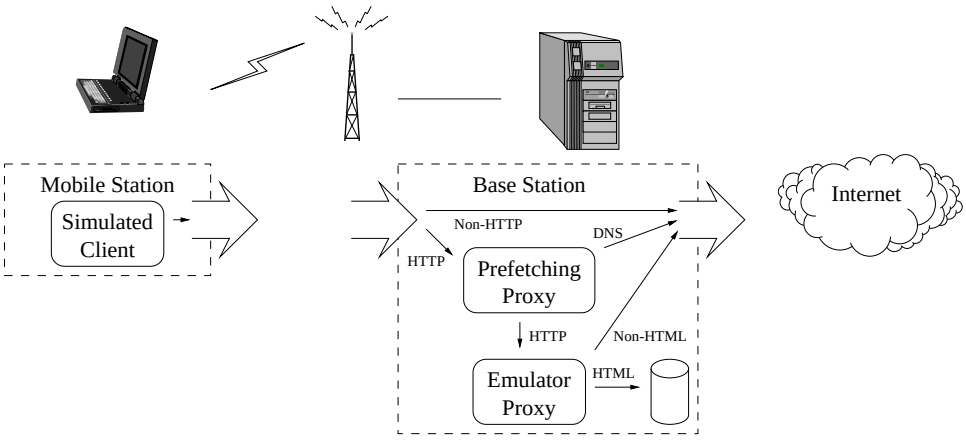

Figure 1: The experiment setup

(URL)
: The Uniform Resource Locator of the document requested.

(RQT) Request Time
: The time that the subject's browser first requested this URL.

(NRT) Name Resolution Time
: The time that the IP address that corresponds to the given host name was returned.

(CET) Connection Establishment Time
: The time that the TCP connection was completely established.

(FBT) First Bit Time
: The time that the first bit of the data was received from the remote server.

(LBT) Last Bit Time
: The time that the last bit of data was received from the remote host. (In the case of an aborted connection, LBT measures the time that the closure was noticed by the proxy, and an indication that the connection was prematurely terminated by the client is recorded.)

The second stage of the experiment was to use this data to examine the network response time with and without DNS prefetching. The experiment is set up as in Figure 1.

The "Simulated Client" drives the data gathering, scrolling through each of the recorded sample sessions. It maintains a constant "think-time" between documents, such that the time from the arrival of the first (and hence, the last, owing to the presence of the emulator proxy) bit until the next request is sent will be the same as in the initial data-gathering session for all trials.

The data file used by the simulated client consists of the following fields:

- URL_i: URL associated with line i
- $HTML_i$: true iff URL_i is an HTML document
- ST_i: think time to leave before requesting line i

Note that lines are in the order in which the URLs were originally requested by the user, and that $ST_i = RQT_i - FBT_j$ where j is the largest value less than i, such that $HTML_j$ is true.

The "Emulator Proxy" attempts to reduce the effects of network variability by caching all of the HTML files in the sample sessions. To determine typical download times, each document in the set was retrieved seven times, and its average retrieval time was stored with its cache entry. All prefetching strategies that we might be interested in parse the content of HTML documents, so controlling the retrieval time of these documents is crucial. Otherwise, the timing of prefetching decisions would be non-repeatable from experiment to experiment, resulting, potentially, in widely differing perceived latencies. While inline images might potentially contain useful information upon which to base prefetching decisions as well (for example, in case the inline image represents a map), online analysis of such images is complex at least and ignored in our experiments. Therefore, controlling the retrieval of these images is of less importance. We opted instead to actually request these images from the Internet, which has two advantages. First, it reduces the disk storage requirements, allowing us to run our experiments on much longer traces. Second, and even more important, downloading inline images creates realistic network traffic that will interact with the ongoing DNS retrievals, similar to the actual deployment of a prefetching proxy.

When a client requests a file, the emulator proxy checks for the presence of that file in its cache (which is the case iff the file is an HTML document). In any case, the host name is resolved, and a connection established. After the connection is established, if the file is in the cache, it is fed back at a uniform rate, such that the first and last bit-times (w.r.t. the time that the name resolution completed) are the same as the established, typical values.

The "Prefetching Proxy" is the central part of our experimental setup. Upon receipt of an HTML document, it scans for links, and asynchronously issues name resolution requests for each hostname that is found, as explained in Section 2. The tests were run both with and without this proxy in place.

The wireless connection used a WaveLAN basestation to communicate with a stationary laptop via a good radio connection. There were occasionally other users on the wireless link, but the amount of external traffic was usually small. The wired network was a normally loaded 10base2 ethernet, with some forty users. Measurements were done at various times of day through both the week and the weekend.

Results

In the results for our test of 865 documents, the average DNS lookup time (NRT) improved from 0.30 seconds, without the proxy, to 0.12 seconds with it (60% closer to ideal). More significantly, the number of delays greater than a second dropped from 102 to 39 (see Figure 2). In this data-set, when multiple consecutive retrieval requests were made for exactly the same URL, only the first time was used. Occasionally, a connection attempt would timeout, and be retried by the client, causing slight differences in the actual list

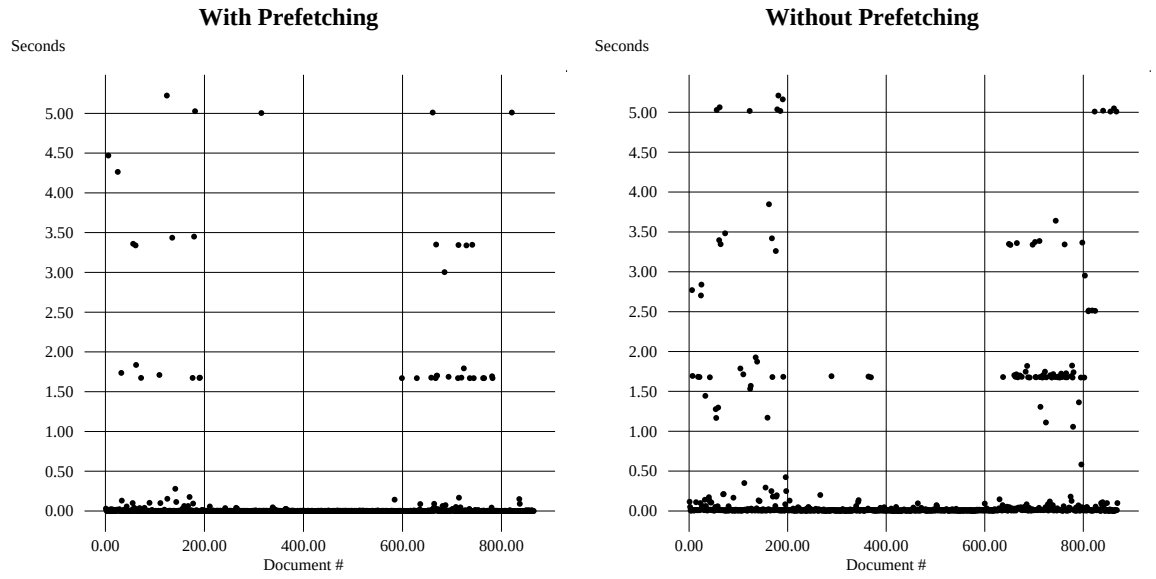

Figure 2: DNS lookup times with and without the prefetching proxy

of URLs retrieved by the proxy (lines would be duplicated). This simplification removed these duplications, making it easier to merge the timing files into a single average time for each URL.

We expected that considering only the retrieval times of those documents that were hotlinks from the previous page retrieved would give even better results. Our tests, however, showed them to be similar to those for the total set of documents. This is probably due to a weakness in the hashing function, since hotlinks should have been prefetched, reducing the delay in their DNS resolution.

Catledge and Pitkow [CP95], based on a test-set of over 43,000 events, found that 52% of all URLs retrieved were selected from hotlinks on the page being displayed. Comparing this to the data used in this experiment, in which only 18% were from hotlinks,[1] suggests that a larger data-set, based on more users, combined with better hashing, would show an even greater improvement. That is to say that this technique can deliver better than half as many noticeable delays, resolving the addresses of half of the documents requested during an average web-browsing session. In our opinion, this suggests that DNS prefetching is a valuable addition to the design of basestation web proxies.

4 Future Work

Having created the utilities and methodology for evaluating prefetching techniques, we plan to use similar experiments to evaluate the value of more conventional strategies[Wac96]. These include different ways of chosing documents and inline images to prefetch. Also of

[1] This difference in behaviour patterns is probably due to atypical behaviour patterns in the small number of people used to generate the data sets that were actually used.

some interest is identifying forwarding markers at the proxy, and loading the redirection site without bothering to send the original page to the client.

Throughout this experimentation, the wireless link was very stable. It gave real throughput of about 1.5Mbps, and a low error rate. Further experiments with lower throughput, and higher error rates might reveal interesting results. The effects of handoff, and techniques for transfer of caches between base stations, or forwarding from one proxy to another will also be investigated.

References

[CP95] Lara D. Catledge and James E. Pitkow. Characterizing browsing strategies in the world-wide web. In *The Third International World-Wide Web Conference*, Darmstadt, Germany, April 1995.

[Wac96] Stuart Wachsberg. Efficient information access for wireless computers. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 1996.

Author Biographies

Michael Nidd, BMath, MMath, is a research associate with the Shoshin distributed computing group, at the University of Waterloo. His current research interests include methods for adapting application behaviour to dynamically adjust to changing network conditions, and architectures for informing applications of changes in network quality.

Thomas Kunz is an Assistant Professor in the Department of Computer Science, University of Waterloo. He received the Dr. Ing. degree from the Technical University of Darmstadt, Federal Republic of Germany, in May 1994. He has been involved in a collaborative research project on "Client-server computing in wireless networks" since joining the Department in July 1994 and published extensively in the area of mobile and distributed computing.

Ertugrul Arik is a Master student in Computer Engineering at Université de Technologie de Compiègne, France. His research interests include client-server computing, computer networks, and telecommunication.