Cross-Domain Cache Cooperation for Small Clients

Amy S. Hughes and Joe Touch
USC / Information Sciences Institute
{ahughes, touch}@isi.edu

Abstract

Caches are used throughout systems to increase performance and reduce load. Networking caches include Web caches and DNS caches; algorithms in these caches, from replacement policy to prefetching, currently rely only on the information within the individual cache. Web caches organize new Web pages based on currently cached Web pages; DNS caches organize new DNS responses based on cached DNS responses. Cross-domain cooperation allows the cached information in Web caches and DNS caches to be shared, and to affect each others' algorithms. This paper presents and explores the feasibility and issues of such cross-domain cache cooperation. Analysis of Web connection times shows that the DNS request occupies up to 1/2 of the total connection overhead. Squid log request streams demonstrate 85-95% reuse of server names. These two results mean that a DNS cache would be highly useful on a user machine. Cooperation between the Web and the DNS cache enables DNS anticipation. For a single client, the additional storage required for anticipated cache entries is a factor of 2-3, but further work must be done to assess the impact of cooperation between the Web cache and the DNS cache.

1: Introduction

Cross-domain cache cooperation is caching that involves more than one domain. Cache cooperation currently occurs in a single domain, i.e., Web caches [1][2][3]. Cache cooperation can be extended to include caches from more than one domain. Because Web requests include implicit DNS requests, there is the possibility of cooperation between a Web cache and a DNS cache to reduce the effect of the DNS request on the per-connection overhead. This document presents the case for maintaining a DNS cache on individual clients and explores cooperation between that cache and the local Web cache.

Timing measurements of Web requests indicate that clients with a high-latency first hop would benefit from a local DNS cache. The DNS request represents a significant portion of the connection overhead on these clients, 1/3 - 1/2 of the total connection overhead. Request streams from Squid[1] logs indicate a high percentage of server name reuse suggesting that a DNS cache would be heavily used, with hit ratios of 85-95%. This hit rate is much higher the demonstrated reuse rate of 30-50% for Web caches[4][5][6]. Many clients already use a Web cache because the most popular browsers, Netscape Navigator[7] and Microsoft's Internet Explorer[8], include one.

A trace-driven simulation of the DNS cache using Squid logs for a single client was created to measure the burden of a DNS request on small clients such as PDAs or similar personal network presence devices [9]. The simulation downloaded HTML documents in the request stream and preloaded the DNS entries corresponding to internal references. The result is that anticipation increases the size of the DNS cache by a factor of 2-3. The improvement to the hit ratio was unclear.

Using Squid logs to simulate the behavior of single-user browsing is imperfect at best. For more accurate results, these simulations should be performed on traces from single user browsing sessions under live network situations where more complete reference information is available. The benefit of caching DNS items was based on a simple hit metric. This metric needs to be refined.

The rest of this document explains the analysis and results in more detail. Section 2 presents an analysis of the connection overhead for a Web request, focusing on the role that the DNS request plays. Section 3 presents information about server name reuse in request streams. Section 4 introduces cooperation between a DNS cache and a Web cache and outlines the opportunity for DNS anticipation. It also presents the results of the initial analyses. Section 5 discusses future work to refine the experiments presented in the previous sections, and Section 6 summarizes the conclusions.

2: DNS Overhead in Web Requests

When a client performs a Web request, it includes an implicit DNS lookup. Depending on the configuration of the network and the location of the DNS server, this implicit request can add significant delay to the connection. In general, if the first hop is a high-latency link and there is no local DNS cache, the DNS transaction will be a significant component of the overhead.

An individual Web transaction begins when the browser resolves the domain name for a new request and ends when the final closing ACK has been received. An individual Web request was divided into components and the durations of each component was tracked

for a set of Web sessions. In these sessions, a Web request consists of five timed components which are illustrated in Figure 1.

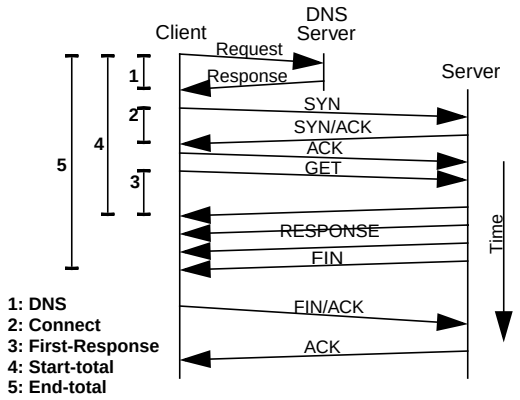

Figure 1: Web connection components

The first component of a Web request is the *DNS* request. This request goes to the client's configured nameserver, which may involve one or more network hops. Once the server name is resolved, the client can open a connection to the Web server. This second component, *Connect*, starts when the client issues the first TCP SYN packet and ends with the arrival of the SYN/ACK. The third component begins with transmission of the GET request, and ends when the client receives the first packet of response data, called *First-Response*. The fourth component, *Start-total*, is the sum of the first 3 components. Start-total begins with the DNS request and ends when the first data packet is received. The last component, *End-total*, starts with the DNS request and ends when the TCP FIN is received, indicating the end of a simple Web transaction.

Timing these five components for 200-300 user requests resulted in the plots in Figures 2-5. In each configuration, the client makes requests from a single Web server. The client does not use persistent HTTP connections and does not maintain a DNS cache. In the first pair of figures, the client's first hop is a low-latency LAN connection. In the second, the client is connected over a high-latency ISDN line. In each set, the first plot represents requests made from a distant server, and the second plot represents requests made from a server on the local LAN.

Figure 2 shows the connection times for a client connected via a LAN. As illustrated by the network diagram below the plot, the remote server is located on the far side of the network cloud. The DNS server is located on the client side of the network cloud. In the plot, the times for the *DNS* component are represented by the line on the far left, labelled by the number 1. Because the DNS server is located closer to the client relative to the Web server, the *DNS* overhead is negligible.

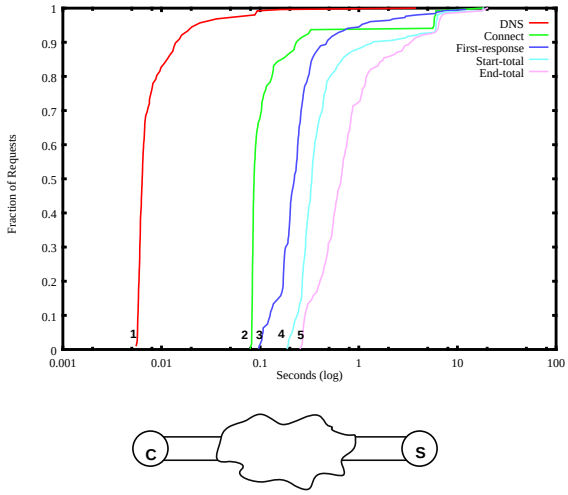

Figure 2: Web connection breakdown for LAN with remote requests and network diagram

In Figure 3, the same client is connected to a Web server on the local LAN. In this case, the DNS server is the same distance from the client as the Web server. The Web server connection, line 2, costs the same as the DNS request, line 1. Again, for a LAN connection, the *DNS* component is a very small part of the connection overhead.

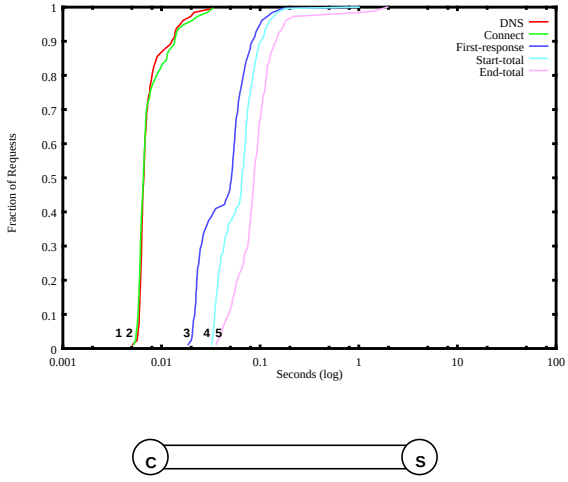

Figure 3: Web connection breakdown for LAN with local requests with network diagram

The impact of the DNS request changes for a client connected to the network with an ISDN line. In this case, the bottleneck is located in the first hop. Figure 4 and Figure 5 show the connection times for this client. The configured DNS server is located on the far side of the first-hop bottleneck, but before the network cloud in Figure 4.

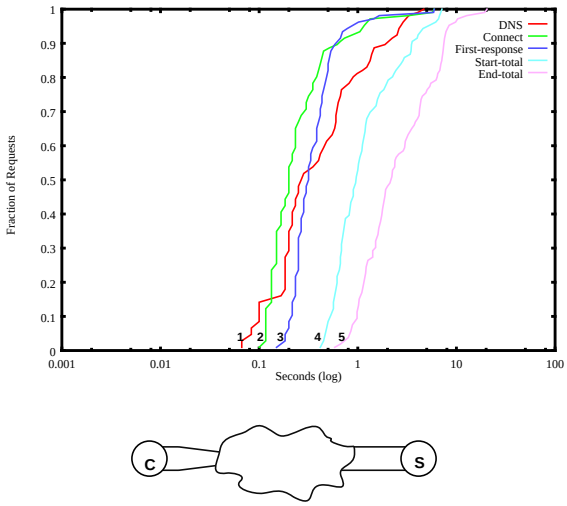

Figure 4: Web connection breakdown for ISDN with remote requests with network diagram

In Figure 4, where the Web server is located on the far side of the network cloud, the DNS request is similar to the initial Web server connection, line 2, and the time needed for the first response, line 3. It accounts for about one-third of the *Start-total*. In most cases, the DNS request takes longer than a tenth of a second, a delay that is noticeable to most users.

The network configuration in Figure 5 eliminates the delay imposed by the remote connection, and shows the case where the DNS server and the Web server are equidistant from the client. In this case, the DNS request is the largest delay in the *Start-total* and takes upwards of one-tenth of a second.

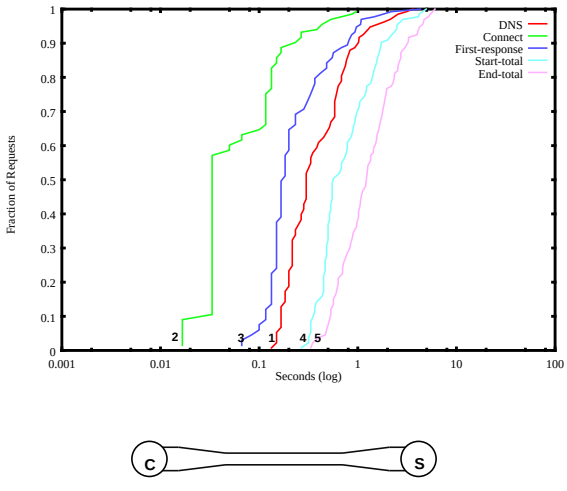

Figure 5: Web connection breakdown for ISDN with local requests with network diagram

In each of the above cases, the client contacts a single Web server. If the client was using persistent HTTP connections, the aggregate connection overhead would be reduced. However, connecting to a single Web server is a simplification of connecting to a variety of Web servers. With multiple Web servers, persistent connections would not be relevant. In either case, the ISDN client would benefit from maintaining a DNS cache. The rest of this paper quantifies the possible benefit and costs of doing so.

3: Server Name Reuse

The analysis in Section 2 indicates that users connected via modems or other high-latency first hops would benefit from a local DNS cache. The next two sections present data from an analysis of Squid logs. The requests from an individual client were extracted from aggregate logs and examined individually. This section measures the potential for reuse in Web request streams. The next section analyzes the impact of cooperation and anticipation.

A simple analysis indicates that caching all DNS entries on a busy client would satisfy 85% or more of all DNS requests. In addition, the overall size of the cache for a 24-hour request cycle is relatively low.

Figure 6 compares the size of the cache to the probability of a hit for one client trace. At 5000 items, the hit rate peaks near 90%. Using an approximate entry size of 250 bytes, 9000 entries translates to a local DNS cache of about 1.25MB. Caching all the entries for the day would increase the cache size to about 2.25MB. Although cache size varies with the activity level of the client, the hit rate below is representative of the logs examined.

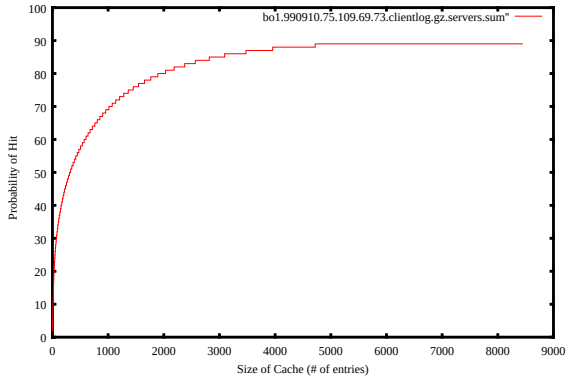

Figure 6: Hit probability vs. Size of Cache

4: Anticipation and DNS-Web Cooperation

Web caches are used to reduce user latency and preserve network resources. However, the reuse rate is often less than 50%[4][5][6]. Web documents contain inherent prefetching hints, in the form of internal references, so prefetching has been explored as a way to fur-

ther reduce user latency. DNS entries and requests do not contain any prefetching hints. Because every Web request triggers a DNS request, there may be benefits possible through cooperation with the Web cache to make DNS prefetch requests.

Because Web documents vary widely, one of the problems with Web prefetching is that it is impossible to predict the size of the documents that will be requested and the amount of time it will take to make the request. In contrast, DNS records are compact, approximately 250 bytes, and the transaction can be handled relatively quickly. To demonstrate the behavior of a DNS cache cooperating with a Web cache, a Squid client log was used as a request stream. Each HTML item requested was also downloaded and its internal reference information harvested, treating new server names as DNS prefetches. The benefit from prefetching each of the internally referenced DNS names can be measured.

4.1: Cache Size

Figure 7 examines the size of the DNS cache over time for a single Squid client. The following plots represent the same trace used in Figure 6. The cache grows as the client makes more requests. The lower line represents simple caching, where entries are added to the cache as they are requested. By the end of the day, the cache has grown to approximately 9000 entries, as shown in Figure 6. The upper line represents anticipation, where all internal references are harvested from HTML documents and cached in addition to the original DNS request. With anticipation, the size of the DNS cache increases by a factor of 2-3. For most clients, this is not prohibitive.

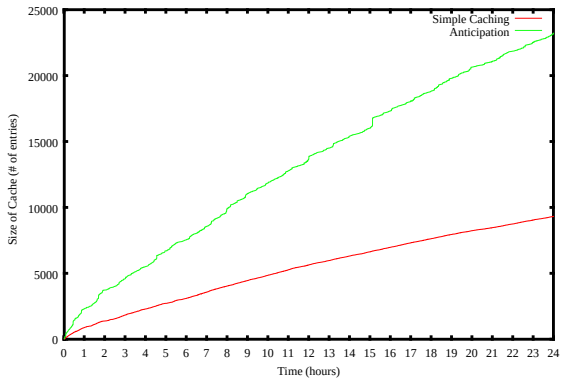

Figure 7: DNS Cache Size over Time

4.2: Hit Benefit

Figure 8 examines the hit rate over time. After the initial cache loading period in the first hour, the hit rate for simple caching approaches the rate expected in Figure 6. The hit rate for anticipation is represented by the upper line. Although anticipation does yield an improvement, the total benefit to individual clients is unclear, due to a range of factors that could not be analyzed in the available Squid traces. These factors will be discussed in the next section.

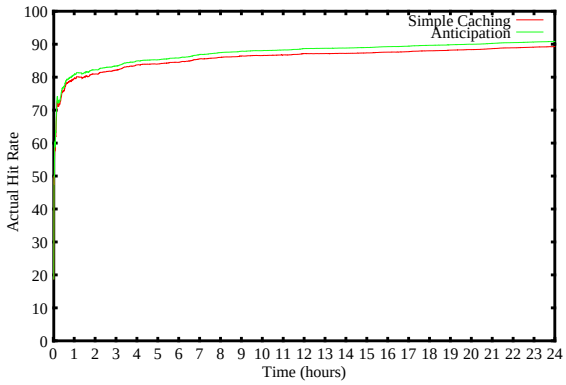

Figure 8: Percentage of Requests Satisfied by Cache Hits

5: Future Work

The Squid traces used for this analysis come from highly cached systems. Individual Squid clients extracted from the daily logs do not reasonably represent individual users. Directly measured user traces are required, because it is likely that Web-DNS cache cooperation is of direct benefit to individual users, and the network latency that they incur. The simulation attempted to retrieve every URL in a given log, but some items were no longer available, or required missing cookies or specific authorization. A better trace would include all communications with the browser to get a complete set of requests and reference URLs.

To properly represent the impact of cooperation, traces of thin clients with limited resources are required. The request rates in the logs examined are far higher than those for an individual user. In addition, browsing patterns change depending on the responsiveness of the network.

As noted before, our simulations do not consider persistent connections. Making multiple requests on the same connection reduces the initial connection overhead, but it does not affect the DNS overhead. A DNS request is still required for every connection made. Figure 9 presents the relationship of connection overhead to connection goodput. The total connection time is the sum of the overhead and the goodput. The overhead is the sum of the DNS request and the connection establishment. As persistent connections reduce the connection overhead, the impact of the DNS connection becomes more pronounced. Thus, reducing the number of DNS lookups through caching becomes more significant.

A = Connection Goodput
B = Connection Establishment
C = DNS Requests
A + B + C = Total Connection Time

$$\frac{C}{A+B+C} < \frac{C}{A+C}$$

Figure 9: Relation of Overhead to Goodput

For a more complete examination, the natures of DNS hits and misses need to be defined. For a Web request, the lookup is binary, the document is in the cache or it is not. For a DNS request, the relationship is less concrete. A DNS cache entry stores a collection of information. For example, consider a request for *www.yahoo.com.* The initial request is a complete miss. After the request, the DNS cache stores the IP address for this site as well as information about the authoritative DNS server for the domain *yahoo.com*. Later requests for *www.yahoo.com* would yield a complete hit. A later request for *maps.yahoo.com* would not find the IP address in the DNS cache, but would find domain server information. This is a partial hit. It is not clear whether the partial hit is as costly as a full miss.

Lastly, this examination of cooperative anticipation ignores the implementation of such a system and the additional computation needed to share information between the caches. We are currently exploring the nature of cross-domain cooperation and it's implementation.

6: Summary

This document proposed and explored cross-domain cooperation between a Web cache and a DNS cache. Several different analyses were presented that suggest the use of a DNS cache on a client machine. More work was done to examine how the DNS cache could cooperate with the local Web cache and the impact of such cooperation was presented.

Section 2 quantified the DNS component of Web requests. Time traces of actual client sessions were broken down into parts. For clients with a high-latency first hop, the DNS request is as time-consuming as the connection to the target Web server, comprising 1/3-1/2 of the initial connection time. Caching the DNS requests on the client would reduce the connection overhead in the case of a DNS hit.

Section 3 examined server name reuse in Web request streams. Analysis of Squid logs showed a high degree of reuse, 85-95%. Coupled with the results in Section 2, this means that a local client DNS cache could significantly reduce the overall Web request overhead.

Section 4 explored anticipation in a cooperative Web/DNS system using Squid logs as request steams and downloading HTML files. By caching the server names found in the internal references, the DNS cache size grows by a factor of 2-3. For a low-traffic client machine, this is an acceptable burden. However, the increase to the overall hit rate is minimal and the total benefit to an individual client cannot adequately be determined with this model.

More work needs to be done to determine whether anticipation is useful for a DNS cache. Most importantly, actual browser traffic needs to be examined that includes persistent connections and all internal references. In addition, the nature of hits and misses in a DNS cache needs to be quantified.

[1] The Squid Internet Object Cache, http://www.nlanr.net/Squid/

[2] The LSAM Project, http://www.isi.edu/lsam

[3] Scott Michel, Khoi Nguyen, Adam Rosenstein, Lixia Zhang, Sally Floyd, and Van Jacobson, "Adaptive web caching: towards a new global caching architecture," Computer Networks and ISDN Systems 30 (1998) 2169-2177

[4] Steven Glassman, "A Caching Relay for the World Wide Web." In Proceedings of the First International Conference on the WWW, 1994

[5] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams and Edward A. Fox, "Caching Proxies: Limitations and Potentials." In Proceedings of the Fourth International Conference on the WWW, Boston, MA, December 1995.

[6] Ramon Caceres, Fred Douglis, Anja Feldmann, Gideon Glass, and Michael Rabinovich, "Web Proxy Caching: The Devil in the Details", in ACM SIGMETRICS Workshop on Internet Server Performance, June 1998.

[7] Netscape Communications Corporation, http://www.netscape.com

[8] Microsoft Corporation, http://www.microsoft.com

[9] Gregory G. Finn and Joe Touch, "The Personal Node," http://www.isi.edu/~touch/pubs/pnp/