RC 21785 (Log 98085) (06/28/2000)
Computer Science/Mathematics

IBM Research Report

On the Effectiveness of DNS-based Server Selection

A. Shaikh, R. Tewari

IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, New York 10598

M. Agrawal

University of Michigan
Dept. of Electrical Engineering and Computer Science
Ann Arbor, MI 48109

Research Division
Almaden · Austin · Beijing · Delhi · Haifa · T.J. Watson · Tokyo · Zurich

This page intentionally left blank.

On the Effectiveness of DNS-based Server Selection

Anees Shaikh   Renu Tewari   Mukesh Agrawal

Network Architecture and Services
IBM T.J. Watson Research Center
Hawthorne, NY 10532
{aashaikh,tewarir}@watson.ibm.com

Department of EECS
University of Michigan
Ann Arbor, MI 48109
quichem@eecs.umich.edu

Abstract

The rapid growth of the Internet in users and content has fueled extensive efforts to improve the user's overall Internet experience. A growing number of providers deliver content from multiple servers or proxies to reduce response time by moving content closer to end users. An increasingly popular mechanism to direct clients to the closest point of service is DNS-based redirection, due to its transparency and generality. This paper studies draws attention to two of the main issues in using DNS: 1) the negative effects of reducing or eliminating the cache lifetimes of DNS information, and 2) the implicit assumption that client nameservers are indicative of actual client location and performance. We quantify the impact of reduced DNS TTL values on web access latency and show that they can increase name resolution latency by two orders of magnitude. Using HTTP and DNS server logs, as well as a large number of dial-up ISP clients, we measure client-nameserver proximity and show that a significant fraction are distant, more than 8 hops apart. Finally, we suggest protocol modifications to improve the accuracy of DNS-based redirection schemes.

1 Introduction

An emerging focus of Internet infrastructure services and products is to improve each user's overall Web experience by reducing the latency and response time in retrieving Web objects. Numerous content distribution services claim improved response time by placing servers closer to clients, at the edges of the network, and transparently directing clients to the "nearest" point of service, where near refers to low round-trip delay, small number of hops, or least loaded server.

An increasingly popular technique for directing clients to the nearest server is to execute the server selection function during the name resolution phase of Web access, using the Domain Name System (DNS). The DNS provides a service whose primary function is to map domain names such as `www.service.com` to the IP address(es) of corresponding machine(s). The transparent nature of name resolution can be exploited to redirect clients to an appropriate server without requiring any modification to client software, server protocols, or Web applications. The appeal of DNS-based server selection lies in both its simplicity – it requires no change to existing protocols, and its generality – it works across any IP-based application regardless of the transport-layer protocol being used. Other approaches such as application-layer redirection (e.g., HTTP redirection), application-specific communication protocols, or routing protocol modifications, are often too complex or too limited in function.

Several commercial content distribution services (e.g., Akamai), currently use modified DNS servers to dynamically redirect clients to the appropriate content server or proxy. When the nameserver receives a name resolution request, it determines the location of the client and returns the address of a nearby server. In addition to these distribution services, several commercial products use DNS-based techniques for wide-area load balancing for distributed Web sites. Examples of such products include Cisco Distributed Director, F5 3/DNS, and Alteon WebOS.

Given the increasing use of DNS for associating clients with the right server, the question of whether DNS is the right location for this function remains unexplored. This paper investigates this question by considering two key issues in DNS-based server selection. First, in order to remain responsive to changing network or server conditions, DNS-based schemes must avoid client-side caching of decisions, which potentially limits the scalability of the DNS. Second, inherent in the DNS-based approach is the assumption that clients and their local nameservers are proximal. When this assumption is violated it leads to poor decisions since server-selection is typically based on the nameserver's identity, not the client's.

The Domain Name System (DNS) is a distributed database of records (e.g., name-to-address mappings) spread across a semi-static hierarchy of servers [1, 2]. The system scales by caching resource records at intermediate name servers. Each resource record has a time-to-live (TTL) value that determines how long it may be cached, with typical TTL values on the order of days [3]. When the DNS is used for server selection it requires that caching of name resolution results be disabled, by setting TTL values to zero (or very small values). Small TTL values allow fine-grained load balancing and rapid response to changes in server or network load, but disabling caching requires that clients contact the authoritative nameserver for every name resolution request, increasing Web access latency. In addition, small TTL values could significantly degrade the scalability of the DNS, since many more requests would have to transmitted in the network, rather than being served from local nameserver caches.

Another, more subtle, issue with DNS-based redirection is that it assumes that the client's local nameserver is representative of the client with respect to location or network performance. If the client and nameserver are distant from each other, the client could be directed to an unsuitable server. It is easy to imagine cases where clients and their nameservers are not co-located, for example in large dial-up or broadband ISPs where widely distributed clients share a nameserver. Moreover, the local nameserver could easily be misconfigured. On the other hand, when a client proxy or firewall doubles as a nameserver, basing redirection decisions on the nameserver location is likely to be quite accurate.

In this paper, we draw attention to these issues and quantify their impact on DNS-based server selection schemes. We use data from ISP proxies, popular Web sites, DNS and Web server logs, and dial-up ISP clients to empirically study the effects of small TTLs and client-nameserver proximity mismatches. Our results show that without careful tuning of TTL values, client latency can increase by up to two orders of magnitude, especially as more embedded objects in Web pages are served from content distribution services. Additionally, many clients and their nameservers are topologically distant from each other. Our experiments show that typical client-nameserver distance is 8 or more hops. Furthermore, we find that latency measurements from server sites to nameservers are poor indicators of the corresponding client latencies.

In the next section we give a brief overview of basic DNS operation. Section 3 discusses and quantifies the effects of using small TTL values on client-perceived Web access latency. Section 4 presents a quantitative analysis of the distance between clients and their local nameservers using DNS and HTTP logs from a commercial web site, as well as a large number of dial-up ISP clients. Section 5 proposes a modification to the DNS protocol to address the problem of identifying clients during name resolution. Section 6 summarizes some representative related work and we conclude the paper in Section 7.

2 DNS: A Brief Overview

At its most basic level, the DNS provides a distributed database of name-to-address mappings spread across a hierarchy of nameservers. The namespace is partitioned into a hierarchy of domains and subdomains with each domain administered independently by an authoritative nameserver. Nameservers store the mapping of names to addresses in resource records, each having an associated TTL field that determines how long the entry can be cached by other nameservers in system. A large TTL value reduces the load on the nameserver but limits how frequently updates to the database propagate through the system. The different types of resource records and additional details about the DNS are described in [1, 4]. The most widely used nameserver implementation in the DNS is the Berkeley Internet Name Domain (BIND) [5].

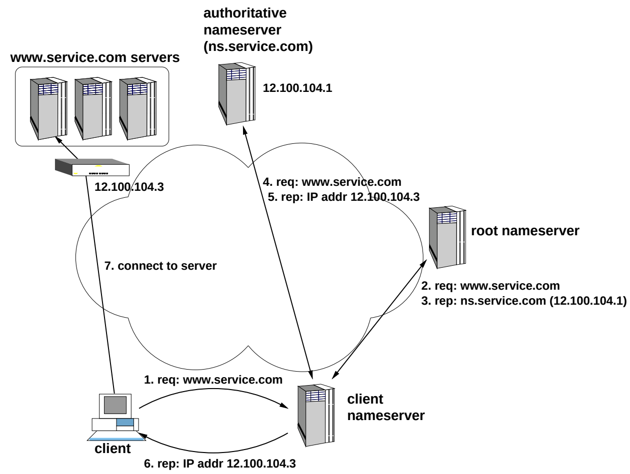


Figure 1: **Basic DNS operation**: This example shows the basic steps required for a client to resolve the address of a service at `www.service.com`.

Nameservers can implement iterative or recursive queries. In an iterative query, the nameserver returns either an answer to the query from its local database (perhaps cached data), or a referral to another nameserver that may be able to answer the query. In handling a recursive query, the nameserver returns a final answer, querying any other nameservers necessary to resolve the name. Most nameservers within the hierarchy are configured to send and accept only iterative queries. Local nameservers that handle queries from clients (i.e., end-hosts), however, typically perform recursive name resolution.

Figure 1 illustrates how a client typically finds the address of a service using DNS. The client application uses a resolver, usually implemented as a set of operating system library routines, to make a recursive query to its local nameserver. The local nameserver may be configured statically (e.g., in a system file), or dynamically using protocols like DHCP or PPP. After making the request, the client waits as the local nameserver iteratively tries to resolve the name (`www.service.com` in this example). The local nameserver first sends an iterative query to the root to resolve the name (steps 1 and 2), but since the sub-domain `service.com` has been delegated, the root server responds with the address of the authoritative nameserver for the sub-domain, i.e., `ns.service.com` (step 3)[1]. The client's nameserver then queries `ns.service.com` and receives the IP address of `www.service.com` (steps 4 and 5). Finally the nameserver returns the address to the client (step 6) and the client is able to connect to the server (step 7).

3 Impact of DNS TTL Values

The scalability of the DNS largely depends on the caching of resource records across intermediate nameservers. The caching is controlled by the TTL value, which in turn depends on the frequency with which administrators expect the data to change. For example, Internet RFC 1912 recommends minimum TTL values around 1–5 days [3]. Earlier documentation had recommended 1 day as the minimum TTL for most servers and around 4 days for top-level domains [6]. These values are now considered too small. Once a domain stabilizes, values on the order of three or more days are recommended. A recent study shows, however, that a majority of nameservers use a default TTL value of 86400 seconds (or 1 day) for their domain [7].

Apart from intermediate nameservers, name resolution results are also cached by Web browsers as a performance optimization. The resolver typically does not return the TTL value with the query result, so browsers use their own policies for caching. For example, the default value used in recent versions of Netscape Communicator is around 15 minutes. Since client-side caching by browsers is often not configurable, we only focus on caching effects at nameservers in this section.

DNS-based server selection radically changes the magnitude of TTL values and, correspondingly, the benefits of caching at local nameservers. To achieve fine-grained load balancing in these schemes, the TTL values returned by authoritative nameservers are typically very small (e.g, 20 sec) or set to zero. These small TTL values affect performance in two ways: (i) they increase cache misses, thereby increasing the number of queries sent to the authoritative nameserver (along with the corresponding network traffic), and (ii) they increase the client latency due to the extra name resolution overhead for each URL access.

One might argue that an increase in request traffic to authoritative DNS servers is not a major concern, given the CPU power of modern servers. Clearly, in the case of Web access, the number of name resolutions is bounded by the number of URL accesses. Processing and servicing HTTP GET requests is likely to incur much higher overhead than handling name resolution requests, which require simple lookups and single-packet responses. However, the increase in network traffic due to additional UDP DNS requests is not insignificant [8].

For client-observed latency, on the other hand, TTL values have a much greater impact. To quantify this effect, we first analyze the overhead of a single name resolution and compare it to the total Web page download latency. Second, we determine the distribution of embedded objects (e.g., images and advertisements) in Web pages across multiple servers by analyzing logs at an ISP proxy site as well as from the top-level pages of the most popular Web sites. Based on this data, we compute the fraction of time the client spends in the name resolution phase for a typical Web page access when the TTL values are small or 0.

[1]Presumably, the client's nameserver caches the address of the `ns.service.com` to avoid repeatedly querying the root servers.

total HTTP requests	34868
unique hostnames	581
unique URLs	7632
duration of trace	6 hrs (10am-1pm, 6pm-9pm)
trace date	February 1999

Table 1: ISP proxy log statistics

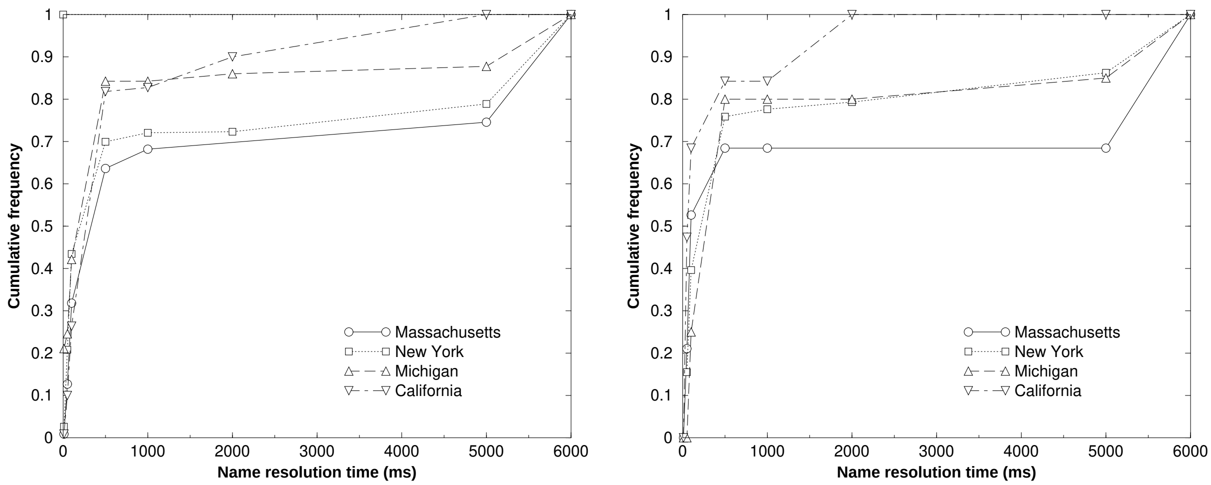
Nameserver cache contents	Median latency
root and `.com` only (case i)	200 ms
domain nameserver (case ii)	60 ms
server address (case iii)	2.3 ms

Table 2: Name resolution latency

3.1 Name Resolution Overheads

To quantify name resolution overhead we analyzed the time spent in the various phases of a typical Web page access. A Web page download consists of the following basic steps: server name resolution, TCP connection establishment, transmission of the HTTP request, reception of the HTTP response, reception of data packets, and TCP connection termination. Using HTTP/1.0 results in repeating the the above steps for each embedded object within a composite page. Note that when the embedded objects are stored on another server (e.g., servers in a content distribution service), having HTTP/1.1 support for persistent TCP connections across multiple HTTP requests does not eliminate the first two steps.

To compute the DNS overheads we compiled a list of server names from the proxy logs at a single POP location of a medium-sized ISP. Table 1 shows the statistics for the fraction of the trace analyzed. We ran a local nameserver (BIND version 8.2.1) at four different locations (Massachusetts, Michigan, California, and New York) and used it to resolve the various server names found in the logs. We measured the name lookup overhead by timing the `gethostbyname()` system call for each server hostname. The measurements were for three levels of caching: (i) the local nameserver cache had neither the server address nor the address of the authoritative nameserver for that sub-domain, (ii) the local nameserver cache had the authoritative nameserver's address, and (iii) the local nameserver cache had the server's address in its cache.



(a) Proxy logs

(b) Popular sites

Figure 2: **Name resolution overheads**: The graph in (a) shows the distribution of name resolution latency for the sites from ISP proxy logs when neither server nor authoritative DNS server addresses are cached locally. In (b) we show the overheads for the most popular sites.

We initially configured the local nameserver to have the addresses of the 13 root DNS servers in its cache. The cache was then primed to contain the addresses of the `.com` domain nameservers. Together, this setup represents case (i) discussed above where the local nameserver had neither the server IP address nor the corresponding authoritative nameserver address in its cache. After each run of the experiment, the local nameserver was restarted to flush the local cache contents. For case (ii), the nameserver cache was primed to contain the address of the authoritative nameserver for each of the domains. Case (iii) measured the time for a cache hit, i.e., when the server address was in the local cache. The median name resolution times for the three levels of caching, measured from the New York site, are shown in Table 2. The results show that caching reduces the median name resolution time by more than two orders of magnitude (from 200 ms to 2.3 ms). With TTL values set to 0 this extra overhead adds to the client-observed latency.

To further validate these results, we obtained a list of popular Web sites compiled by an Internet measurement service [9] and repeated the name lookup experiments. These sites had a combined user population of 76 million. Figures 2(a) and (b) show the distribution of name resolution latency for servers (case i) for both the proxy logs and the popular Web sites, respectively. The times were measured from four different locations on the Internet. The results show that 25% of the name lookups (with no caching) add an overhead of more than 650 ms and more than 3 seconds, for the popular sites and ISP proxy log sites, respectively. It is interesting to observe that nearly 15% of the popular sites required more than 5 seconds to contact the authoritative nameserver and resolve the name. This is likely to be related to the 5-second default request timeout in BIND-based resolvers [2].

3.2 Impact of Embedded Objects

Most Web pages accessed today contain a number of embedded components. These components, including images and advertisements, may be stored at the same Web server or possibly at a different server belonging to a content distribution service. In cases where the embedded objects are not colocated, each page access may result in multiple name resolutions, as the client resolves the address of other servers. In this section we quantify the name resolution overhead per embedded object, beginning with a determination of the distribution of embedded objects per Web page.

The logs we obtained from the ISP proxy (see Table 1) were packet traces collected using the `iptrace`[2] tool available on AIX. The packet traces logged information about the packet contents including IP and TCP headers, HTTP request and response headers and the list of embedded objects within each request (i.e., all < `img src ...` > tags). From these traces we extracted a list of embedded objects within each composite page. To further substantiate the results, and also study more current data, we also analyzed the top-level pages from the popular Web sites, determining the number of embedded objects for each. The distribution of the number of embedded objects in both data sets is shown in Figure 3. The ISP logs show an average of 14 and a median of 5 embedded objects per page. The index pages of the popular sites have much higher values, an average of 35 and a median of 25 objects per page. These results are similar to those observed in [10].

For the index pages of the popular Web sites we determined the download time for each embedded object along with the composite page, and compared it to the name resolution latency. We use a tool called *Page Detailer* [11] that measures the individual components contributing to Web

[2] `iptrace` is similar in function to `tcpdump`.

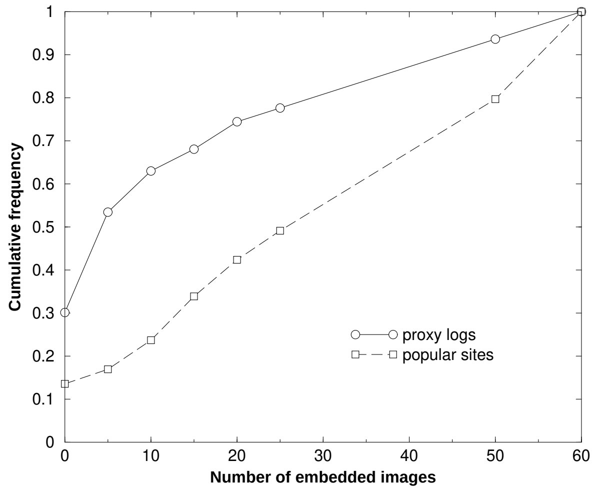


Figure 3: **Embedded object distribution**: This graph shows the distribution of the number of embedded objects per Web page from the ISP proxy sites and the most popular Web sites.

page access latency, including the download time for each embedded object. We primed the local nameserver cache and the browser cache to contain all the server addresses (of the popular sites) such that the measured time consisted only of the page download time and had no name resolution overhead. The average page download times and the object sizes are shown in Table 3.

The results show that if all the embedded components were stored on the same server, such that only one name resolution was required for the entire composite page (e.g., with HTTP/1.1 or when the TTL returned by the nameserver is non-zero), the name resolution overheads are quite small. When neither the nameserver nor the server address is in the cache (case i), however, the overhead grows to around 3% (200 ms for the name resolution vs. about 6 seconds for the entire page download).

The name lookup overhead becomes an order of magnitude higher when each embedded component requires an additional name lookup. This might occur, for example, when objects are served from different servers belonging to a content distribution service. From our experimental results, performing a name lookup for each embedded object adds an overhead of 48% (around 200 ms for the name lookup and around 400 ms for the embedded object download) on a cache miss. The large name resolution overhead suggests several considerations: additional DNS queries should be amortized over large page downloads, and embedded components should be co-located to avoid excessive DNS queries.

The DNS TTL value needs to balance the tradeoff between responsiveness of DNS-based server selection, client-perceived latency, and overall scalability of the system. It is important for site administrators to understand these tradeoffs before selecting small TTL values. The problem of selecting TTLs arises from the basic limitation of having

avg. complete page download time	6.3 sec
avg. total page size	30.9 KB
avg. embedded object size	1.22 KB
avg. embedded object download time	0.415 sec

Table 3: Page download statistics

no mechanism to flush cached name-to-address mappings in client-side nameserver caches. One simple solution is to use larger TTL values to provide only coarse-grained load balancing at the DNS level. Another approach avoids overloading basic DNS functionality, but instead relies on new services or protocols for load-balancing and server selection. For example, Web servers can direct clients to the best proxy or alternate server by creating dynamic HTML pages with embedded links pointing to the best server, or by using HTTP redirection. These approaches are not without drawbacks, however. Dynamic pages with rewritten hyperlinks cannot be cached and HTTP redirection suffers from additional TCP connection establishment latency.

4 Client-Nameserver Proximity

DNS-based server selection schemes typically assume that clients and their primary nameservers share network performance characteristics by virtue of being located close to each other. When handling a name resolution request, the DNS server performing the server selection typically sees only the client nameserver as the originator. It has no way of knowing who the actual client is, or how far the client is from its nameserver. The conventional solution to this problem is simply to assume that the client and nameserver are located nearby each other. In this section we evaluate the validity of this assumption empirically using two approaches, first based on data traces and then on experiments with several ISPs.

Proximity could be measured directly between the client and nameserver, in terms of network hops, intradomain routing metrics, or round-trip time. But for the purposes of DNS-based server selection, the direct client-to-nameserver distance is less relevant. The accuracy of server selection decisions is more directly influenced by whether clients and nameservers appear nearby when observed externally, for example from server sites. Hence, in this section we focus on proximity metrics that are measured from arbitrary sites in the Internet.

Our initial approach is to collect traces of HTTP and DNS requests from a production web site and use them to match clients to their nameservers. We then determine the distance between these clients and their nameservers, as seen from a probe site in the network. In Section 4.4 we use dial-up ISP accounts to conduct experiments to determine client-nameserver proximity as seen from multiple probe sites.

	Clients	Client nameservers
unique IP addresses	$32,919$	3807
common IP addresses	497	
unique AS numbers	886	805
	HTTP requests	DNS requests
no. of requests	$1,455,199$	$288,581$
duration of trace	48 hrs	39.5 hrs
avg. request rate	8.42 req/s	2.03 req/s

Table 4: DNS and HTTP log statistics

4.1 DNS and HTTP Data

We obtained DNS and HTTP server logs from a commercial web site hosted by IBM Global Services. The site is configured with a group of several servers that provide access to a Web-based service. Incoming connections from clients are directed to one of the servers by a load-balancing layer-4 switch which accepts requests on virtual IP address(es). The authoritative DNS server for the subdomain, co-located at the site, handles name resolution requests, and returns answers with a TTL 0. The logs, collected over 2 days, contain DNS requests and the client HTTP requests on the corresponding web servers. The DNS logs contain the IP address of the requesting nameserver, the name being resolved, the IP address returned, and the timestamp. The HTTP logs contain only the client IP address and the timestamp. Table 4 shows some basic statistics about both sets of logs.

We use information in the global Internet Routing Registry (IRR) to determine autonomous system (AS) numbers for each IP address. The IRR is a collection of routing policy databases that includes servers operated by several ISPs along with several other networking organizations [12, 13]. We constructed a local copy of the available IRR databases and used it to lookup AS numbers. ISPs voluntarily publish policy and route information in the IRR, thus its contents are incomplete. In our traces we could not identify the AS numbers for 6% of client IP addresses and 5% of nameserver IP addresses using the routing registry.

4.2 Matching Clients and Nameservers

Before we can characterize client-nameserver proximity we use the logs to match clients with their configured nameservers. We rely primarily on timestamps for the correlation of DNS requests with HTTP requests. Since the authoritative DNS server returns addresses with a zero TTL, we expected each HTTP request to have a corresponding DNS request. Several factors complicated this process, however:

- **clock skew**: The DNS server and HTTP servers run on separate machines which are not synchronized. Moreover, the clock skew of the DNS machine relative to each HTTP server machine may be different.

- **client caching**: Although the DNS server at this site is configured to return answers with a zero TTL, client browsers typically cache the result of name resolutions. Some informal experiments using *Page Detailer* suggest that Netscape Communicator 4.72 on the Microsoft Windows platform, caches name resolutions for approximately 15-20 minutes.So despite the zero TTL, a request in the HTTP server log may not have a corresponding request in the DNS server log.

- **mishandling of TTLs**: Some older BIND nameservers are known to enforce a minimum TTL on received DNS information, even if the TTL is zero [6]. Thus, some HTTP requests may not have corresponding DNS requests even after accounting for client-side caching.

The process of matching clients and their nameservers is subject to inaccuracy (due to the factors above); hence, we develop a multi-step algorithm to remove as much uncertainty as possible.

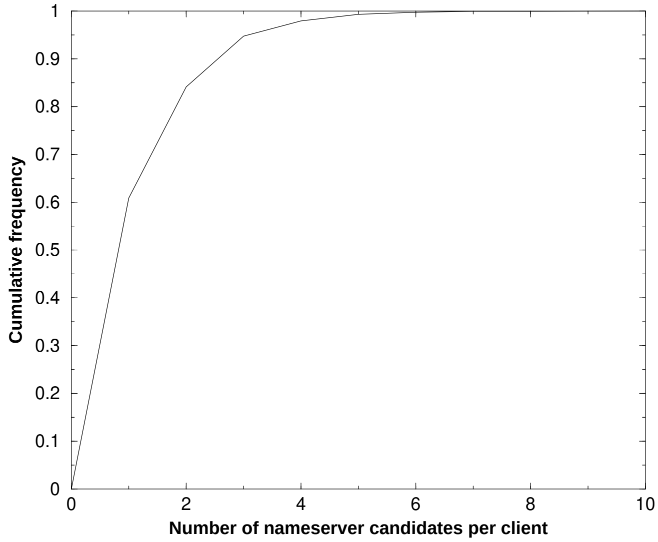
Since we rely on timestamps to perform the matching, we first try to identify the relative clock skew between the DNS server and each of the web server machines using IP addresses that are common to both the DNS and HTTP logs. We assume that these addresses are proxies or firewalls that perform both HTTP and DNS requests on behalf of clients, and consider such cases to be certain matches. Using these certain matches we determine the mean clock skew and use it in the subsequent steps. What we refer to here as clock skew also includes the delay between the name resolution request and corresponding HTTP request.

In the first pass we consider each HTTP request in turn and construct a list of candidate nameservers with a nearby timestamp[3], subject to the skew and the expected browser caching (which we assume is approximately 15 minutes). On subsequent sightings of the same client in the HTTP log, we refine the list of candidate nameservers by intersecting the existing list with the new list generated by the new sighting. At the end of this process we have a variable-length list of candidate nameserver addresses for each client IP address.

Since this first pass did not always narrow the list sufficiently to find a single nameserver for each client, we introduced a second pass that performs a similar process in reverse. We consider each nameserver address sighting in the DNS logs and construct a list of likely clients served by the nameserver, according to the timestamp and the name being queried (again, taking the measured clock skew into account). This results in a second list of candidate clients served by each nameserver. These lists are naturally much longer than the candidate nameserver lists.

Finally we combine the two sets of candidate lists to identify client-nameserver pairs that appear in both lists. Using this process we were able to find candidate lists for 2394 clients (approximately 10% of all clients). Each final candidate list had an average length of 1.6. Figure 4(a) shows the CDF of the length of the nameserver candidate lists. More than 60% of these clients matched to one nameserver, though we were not always able to take advantage of this (as discussed below). Note that these candidate lists are based on matching clients and nameservers using only timestamps.

[3] We use a window of 4-10 seconds.



(a) Candidate list distribution

Matching heuristic	% matches (324 total matches)
AS	31%
domain name	16%
AS and domain name	53%

(b) Matching heuristics

Figure 4: **Candidate lists and matching heuristics**: The graph in (a) shows the cumulative distribution of the length of the nameserver candidate lists. This distribution is taken across each of the 2394 clients for which we were able to find matches. In (b) we tabulate the percentage of client-nameserver pairs that match according to various heuristics.

At this point we must decide how to finally pick one nameserver when a client has more than one possible candidate. We adopt a conservative approach that applies some simple heuristics based on AS number and domain name to decide if a client and nameserver do in fact belong together. Basically, when presented with several candidate nameservers, we pick the nameserver that has either the same AS number or domain name as the client. We further rank the matches so that matching domains are ranked higher than matching AS, since AS grouping is relatively broader. Of course a match on both criteria is ranked highest. When the AS or domain name is unavailable (no IRR entry or no PTR record in the DNS), we assume a mismatch for that criteria. In the case of ties between candidates we use random choice as the final tie-breaker. According to these heuristics, we were able to find 324 clients-nameserver pairs (14% of the clients for which a candidate list was found using timestamps) . The table in Figure 4(b) shows the percentage of these pairs that matched according to the combinations of these heuristics.

When none of the candidate nameservers for a particular client match according to these heuristics, we consider it a mismatch, even if the candidate list consists of only one nameserver. This, in effect, removes most of the cases in which a client may not be using the correct or assigned nameserver. While these cases are of particular interest, we believe from inspection that timestamp-based correlation alone may be inaccurate, thus requiring a conservative approach using additional heuristics.

4.3 Log-Based Proximity Evaluation

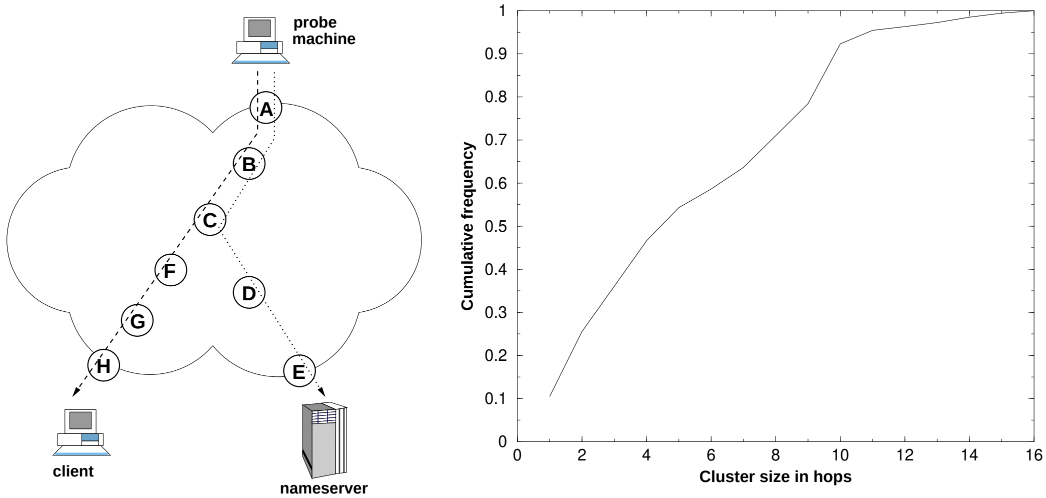
After determining the set of client-nameserver pairs from the DNS and HTTP logs, the next step is to determine the proximity of clients to their nameservers. Some simple metrics of proximity include relatively static parameters such as AS number, domain name, and IP address prefix.

Proximity measure	% matches		
matching AS	31 (3% n/a)		
matching domain	16 (23% n/a)		
matching AS and domain	53 (26% n/a)		
matching IP prefix	1 octet	2 octets	3 octets
	37	19	10

Table 5: Client and nameserver proximity measures

In Table 5 we show the percentage of client-nameserver pairs that are “nearby” according to these metrics. In parentheses are the percentage of pairs for which the corresponding metric could not be determined. Since we use domain names and AS numbers as heuristics to *determine* matching pairs, these metrics are somewhat misleading. For example, Table 5 indicates that 31% of the client-nameserver pairs had the same AS number but this really means that 69% of the pairs had either matching domain names only or matching AS and domain name. We found that only about half of the client-nameserver pairs had matching domain names *and* AS number.

Table 5 also shows the percentage of client-nameserver pairs that share the same prefix in their IP addresses when prefix lengths are assumed to be one, two, or three octets. It should be noted, however, that although nearly 50% of actual Internet address prefixes are 24 bits, there are a large number that are between 16 and 24 bits [14]. Therefore, the numbers in Table 5 may underestimate the actual matches if the real prefix length is not 8, 16, or 24 bits.



(a) Clustering clients and nameservers (b) Distribution of cluster sizes

Figure 5: **Client and nameserver clustering (router hops)**: In (a) we illustrate how clients and nameservers are clustered within x number of hops using path information gathered from the probe site. The graph in (b) shows the distribution of the cluster sizes.

A better metric for determining client-nameserver proximity is network hops which we measure from a probing site in the network. We use the `traceroute` tool to learn the network path from the probing site to the client and nameserver. Then we find the maximum hopcount until a common ancestor appears in the paths to determine the "cluster" size of the client-nameserver pair. This process is illustrated in Figure 5(a). Router C is the first common ancestor on the two paths from the probe site. Since C is 4 hops away from the client and 3 hops away from the nameserver, we say that this pair belongs to a $\max(3, 4) = 4$-hop cluster. If both paths were the same except for the last hop (i.e., client and nameserver both connected to router H), then the client and nameserver belong to a 1-hop cluster.

Figure 5(b) shows the distribution of cluster sizes for the client-nameserver pairs we identified. Notice that only about 15% of the pairs are in 1-hop clusters. The median cluster size is 5 and more than 30% of the pairs are in 8-hop clusters, indicating that a large fraction of clients are topologically distant from their nameservers when measured from an arbitrary point in the network. Furthermore, since the matching process removed misconfigured client-nameserver pairs, the actual number of clients that are topologically distant from their nameservers is likely to be higher.

4.4 ISP Proximity Experiments

To further evaluate client-nameserver proximity, we conducted experiments with ISP clients that connect using dial-up PPP connections. In most cases, dial-up ISPs provide primary and secondary nameserver IP addresses along with the local (dynamic) IP address during the PPP network-layer protocol configuration [15, 16]. This allows us to know with certainty the nameserver addresses for the client, thus overcoming the major challenge of matching clients to their nameservers using only DNS and HTTP request timestamps in logs.

ISP accounts	11
POPs dialed	27–54, avg: 45.8
unique client addresses	498
unique nameserver addresses	54
nameserver addresses per ISP	2–15, avg: 7.4

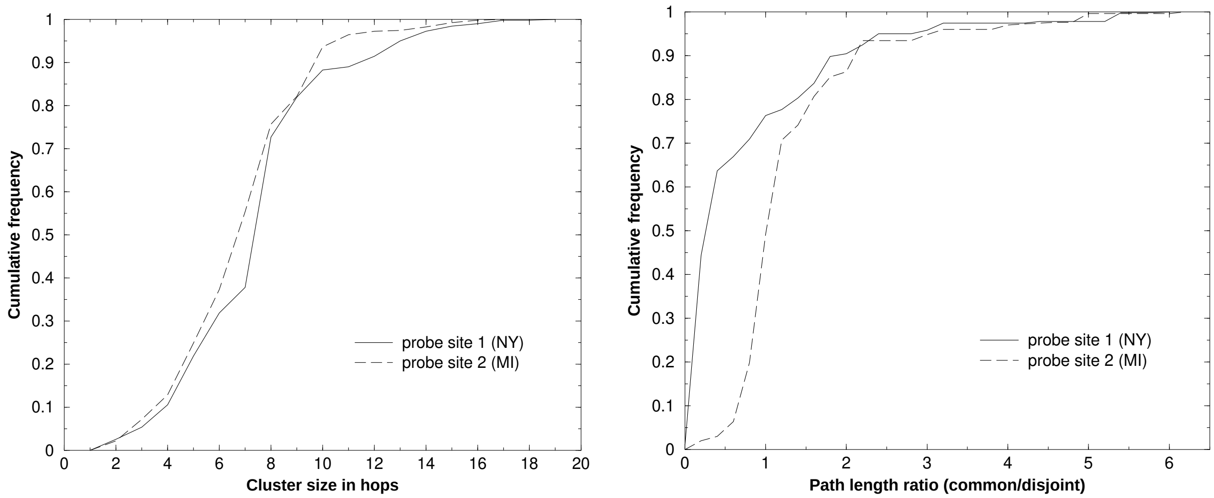
Table 6: ISP address statistics

We obtained dial-up accounts from 9 National retail ISPs [17] and two "free" ISPs. For each ISP, we dialed into approximately 50 POPs across the U.S. Our dataset includes 1090 distinct client-nameserver pairs. Table 6 summarizes the ISP data. Note that we limited our study to those ISPs that use standard link-layer and authentication protocols to simplify the process of automating the experiments.

From two probe points in the Internet (located in New York and Michigan) we collected path and latency measurements to the dial-up client and each of its nameservers using the `traceroute` and `ping` tools. In addition we determined the path and network latency from the client to its nameservers.

4.5 ISP Proximity Evaluation

In our evaluation of ISP client-nameserver proximity we focus on path and latency measurements from the probing points rather than other proximity heuristics such as AS number or domain name. In most cases the AS numbers and domain names of clients and nameservers matched, though some dial-up ISPs employ nameservers from third-party providers. It is interesting to note that some larger network providers that provide DNS services for dial-up ISPs appear to use network-layer anycast for their DNS server ad-



(a) Client-nameservers clusters

(b) Ratio of common to disjoint path length

Figure 6: **ISP client-nameserver paths**: In (a) we show the distribution of cluster sizes as viewed from both probe sites. The graph in (b) shows the distribution of the ratio between the common and disjoint portions of client-nameserver paths.

dresses. We found several cases, for example, where the path to the advertised DNS server address consistently ended at a different address from both probe sites. In the case of one ISP, the nameserver ultimately contacted depended on the POP location, or where the `traceroute` was taken.

We first measured the size of client-nameserver clusters as viewed from the two probing points, using the same technique shown in Figure 5(a). The graph in Figure 6(a) shows similar clustering to the log-based results in Section 4.3. Again, nearly 30% of client-nameserver pairs fall in clusters that are 8 or more hops. The median cluster sizes are larger than in the earlier results, 8 and 7 hops from probe sites 1 (New York) and 2 (Michigan), respectively. The results from both probe sites are generally equivalent, though the clusters are slightly smaller when viewed from probe site 2.

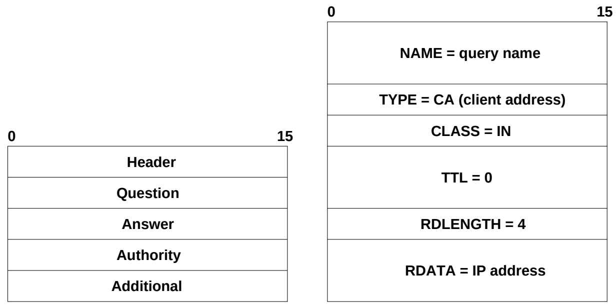
We compared these results with the direct client-nameserver topological distance and found that that the average distance over all pairs was 7.6 hops, with a median of 8. Some clients were as far as 15 hops from their nameservers. The average client-to-nameserver round-trip latency was 234 ms, though this was dominated by the average first-hop latency which was 188 ms. These results show that even when considering direct distances, clients and nameservers are often topologically quite far apart.

Another indicator of how performance from the client and its nameserver may differ is the length of the common versus disjoint portions of the paths. Suppose the path from a Web server to a client and its nameserver is common for many hops, and then diverges near the ends. Then it might be expected that the client and nameserver share similar network performance characteristics to the server, more than if the paths diverged nearer to the server. To measure this, we compute the ratio of the length of the common portion of the paths to the disjoint portion. For example, in Figure 5(a), the common path is A-B-C with length 2 and the (maximum) disjoint portion is C-F-G-H with length 3, resulting in a ratio of $2/3 = 0.66$. A smaller ratio implies that a smaller portion of the paths to the client and nameserver is shared, suggesting that similar network performance to the client and nameserver is less likely.

Figure 6(b) shows the distribution of path length ratios from both probe sites. As expected, the path ratios depend heavily on the probe site location. For probe site 1, around 35% of client-nameserver paths have disjoint paths that are twice as long as the common paths (i.e., ratio 0.5). For probe site 2, however, only 5% of the client-nameserver pairs have a 0.5 ratio and nearly 50% have ratio 1.0. For both probe sites, though, no more than 10% of the client-nameserver paths had a ratio greater than 2.0. Thus, in most of the cases, the disjoint portion of the path is significantly long, relative to the common portion. One interpretation of these results is that the nameserver and client paths are sufficiently divergent, such that similar network performance is unlikely.

We also examined the network latency to clients and nameservers to determine if measurements to nameservers are in general indicative of the relative performance from the corresponding clients. For example, several DNS-based server selection products collect measurements from each server site to the requesting nameserver, and direct the client to the site reporting the smallest round-trip latency. For each client-nameserver pair, we obtain a round-trip latency measurement (using `traceroute`) to the client and nameserver from each of the probe sites[4]. We denote the measured latency from probe site 1 to the client and nameserver as t_c^1 and t_d^1, respectively (similarly for probe site 2). If we suppose that the probe sites represent Web server sites, an interesting question is: does $t_d^1 < t_d^2$ imply that $t_c^1 < t_c^2$? In our experiments, this relationship was violated in 21% of the

[4]The client latency is measured to the last hop router rather than the client itself, to remove the effect of the large delay introduced by the dial-up link



(a) DNS message format (b) CA resource record

Figure 7: **DNS protocol modifications**: The general DNS message format is shown in (a), and (b) shows the proposed CA resource record carried in the additional records section.

cases. We also consider the case when two probe sites look roughly equivalent with respect to nameserver latency, i.e. $|t_d^1 - t_d^2| \leq w$, where w is, say, 10 ms. In this case we wish to determine if the corresponding client latency is also roughly the same, subject to the same value of w. We found that this was true in only about 12% of the cases, suggesting that a random choice among two equivalent-looking server sites, when measurements are relative to the nameserver, may be misguided. In general, the correlation between nameserver latency and actual client latency was quite low. Specifically, we computed the correlation coefficient between $a = t_d^1 - t_d^2$ and $b = t_c^1 - t_c^2$, and found that $\rho = 0.32$. Thus, a and b are positively correlated, but only weakly so.

5 DNS Protocol Modifications

As stated at the outset, DNS-based server selection schemes assume that clients and their primary nameservers are located near each other, such that they would experience similar performance when accessing a server. As shown in Section 4, however, clients and nameservers are often topologically quite distant from each other, casting doubt on the validity of this assumption.

One way to address this problem is to modify the DNS protocol to carry additional information to identify the actual client making the request. In this section we propose a simple scheme that carries the IP address of the client requesting name resolution in the DNS query message. A DNS server performing load balancing or server selection can use the client IP address to decide more accurately which address to return in the answer. This is of course only applicable in the common case where client resolvers make recursive queries to the local nameserver, which then operates iteratively to find the answer.

As shown in Figure 7(a), the standard DNS message format consists of five sections: `header`, `question`, `answer`, `authority`, and `additional` [4]. This scheme could be implemented by modifying the format of the `question` section in DNS messages, but a more backward compatible approach is to define a new DNS resource record with type `CA` (client address) to accompany the query in the `additional` records section of the message. Figure 7(b) illustrates the format of the new resource record. The type field is set to `CA` and the data section of the record simply contains the client IP address. The TTL is zero since the record applies only to the current transaction and should not be cached. Note that this extension can be incrementally deployed, similar to other experimental resource records. Nameservers that do not understand the new type will simply ignore it. This is a slightly unusual use of a new resource record since it pertains to a specific query instead of providing additional information in the database about a host, nameserver, or network.

6 Related Work

There are several areas of research and standardization efforts relating to DNS-based server selection. In this section we summarize some representative work.

The general problem of determining distance between Internet hosts or networks has received a great deal of recent attention. For example, the IDMaps architecture attempts to provide a service in which `traceroute` measurements are distributed over the Internet using IP multicast [18]. Clients of the service use the raw measurements to compute distance estimates. The SONAR service provides an interface between applications and proximity estimation services [19]. SONAR defines a query/response protocol that clients can use to find the distance between a nearby SONAR server and a set of IP addresses. SONAR does not specify a means for estimating proximity, depending instead on services such as IDMaps for distance information.

Related to the issues of how to measure proximity and make information available to clients is the question of which metrics provide the best indication of actual latency when selecting among multiple servers. Recent work has considered metrics such as network hops, AS hops, and RTT, along with various means of collecting them including active probing using `traceroute`, `ping`, or HTTP, and passive participation in BGP peering sessions [20–22].

Several modifications to DNS have been proposed, both to provide additional location information about hosts, and specifically to facilitate server selection. The LOC resource record allows geographic location information to be expressed in the DNS as latitude and longitude [23]. Similarly, the GL resource record encodes location information in terms of hierarchical locator (country code, postal code) and an textual address [24]. NetGeo is a related tool that performs `whois` queries on a variety of databases to return all available geographic information about an IP address, domain name, or AS number [25]. The SRV DNS resource record is a proposed standard which specifies the identity of servers that provide a specific service (e.g., LDAP) using a specified protocol (e.g., TCP), in a specified domain (e.g., `service.com`) [26]. SRV records allow clients to identify the servers providing the specified service but does not give any indication of their actual location beyond what is discernible from the server's name or IP address.

In [27], the authors describe a scheme in which client DNS servers directly query gateway routers for routing metric information about paths to different content servers. Using these metrics the DNS server selects an appropriate server. The Location Data System (LDS) defines an extension to the DNS for resource records that map URLs to lists of servers holding the specified object [28]. LDS also uses the dynamic update facility in DNS to learn about changes in the location of objects from object servers (e.g., web caches).

Finally, some recent work has proposed new mechanisms to reduce client latency related to name resolution. One approach is to pre-resolve server names by considering the user's access history along with the links appearing on the Web page being viewed [29]. Other work proposes proactive cache management in which cached name resolution results are refreshed automatically, without waiting for a query from the client [7]. This work further affirms that DNS caching plays a crucial role in determining client-perceived latency.

7 Conclusion

This paper explored two important issues related to DNS-based server selection. The DNS-based schemes typically disable client-side caching of name resolution results, raising the question of what impact this policy has on client-perceived Web access latency. Our experiments show that without caching, name resolution overhead can grow up to two orders of magnitude. Furthermore, as the number of embedded objects served from multiple sources increases, name lookup overheads can grow nearly 50%. DNS-based server selection also relies on clients and their local name-servers being in close proximity, since redirection is based on the nameserver originating the request rather than the client. Our experiments show that this assumption is often violated, with clients typically 8 or more hops from their nameservers. Also, our ISP experiments showed that latency measurements to local nameservers are generally weak predictors of latency to the actual clients.

A growing number of content and service providers are turning to DNS-based schemes, and associated commercial products, for distributed server selection and load balancing. This study draws attention to the pitfalls associated with this approach. Our results suggest that careful consideration is necessary when choosing DNS TTL values to balance responsiveness against extra client latency. Also, additional mechanisms may be necessary to ensure the accuracy of server selection decisions when client proximity is a deciding factor. In this paper, we propose one such mechanism in the form of a new, simple DNS resource record that identifies the client originating a name resolution request.

Acknowledgment

We are grateful to David Bolthouse and Steven Woodruff of IBM Global Services for their assistance in obtaining the HTTP and DNS data. We also thank Nat Mills for providing us access to the *Page Detailer* tool, and Srinivasan Seshan and Erich Nahum for their suggestions and feedback.

References

[1] P. Mockapetris, "Domain names – concepts and facilities," Internet Request for Comments (RFC 1034), November 1987.

[2] Paul Albitz and Cricket Liu, *DNS and BIND*, O'Reilly and Associates, 1998.

[3] David Barr, "Common DNS operational and configuration errors," Internet Request for Comments (RFC 1912), February 1996.

[4] P. Mockapetris, "Domain names – implementation and specification," Internet Request for Comments (RFC 1035), November 1987.

[5] Internet Software Consortium, "Berkeley internet name domain (BIND)," `http://www.isc.org/products/BIND`, June 2000.

[6] Anant Kumar, Jon Postel, Cliff Neuman, Peter Danzig, and Steve Miller, "Common DNS implementation errors and suggested fixes," Internet Request for Comments (RFC 1536), October 1993.

[7] Edith Cohen and Haim Kaplan, "Proactive caching of dns records: Approaches and algorithms," available from `http://www.research.att.com/~edith`.

[8] Peter B. Danzig, Katia Obraczka, and Anant Kumar, "An analysis of wide-area name server traffic: A study of the Internet Domain Name System," in *Proceedings of ACM SIGCOMM*, Baltimore, MD, August 1992, pp. 281–292.

[9] "Media Metrix top 50," `http://www.mediametrix.com/usa/data/thetop.jsp`, May 2000.

[10] Mikhail Mikhailov and Craig E. Wills, "Embedded objects in web pages," Tech. Rep. WPI-CS-TR-00-05, Worcester Polytechnic Institute, Worcester, MA, March 2000.

[11] IBM Corporation, "Page Detailer," Distributed with IBM WebSphere Studio, `http://www.ibm.com/software/webservers`, June 2000.

[12] "RADB internet routing registry," `http://www.radb.net`.

[13] "Internet routing registry daemon," http://www.irrd.net.

[14] Balachander Krishnamurthy and Jia Wang, "On network-aware clustering of web clients," in *to appear in Proceedings of ACM SIGCOMM*, January 2000, Also available as AT&T Labs–Research Technical Memorandum #000101-01-TM.

[15] Glenn McGregor, "The PPP internet protocol control protocol (IPCP)," Internet Request for Comments (RFC 1332), May 1992.

[16] Steve Cobb, "PPP internet protocol control protocol extensions for name server addresses," Internet Request for Comments (RFC 1877), December 1995.

[17] "Top ISPs, quarterly report," Network World Fusion, April 2000, http://www.nwfusion.com/research/2000/0417isp.html.

[18] Paul Francis, Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel F. Gryniewicz, and Yixin Jin, "An architecture for a global Internet host distance estimation service," in *Proceedings of IEEE INFOCOM*, March 1999.

[19] Keith Moore, "SONAR – a network proximity service," Internet Draft (draft-moore-sonar-03.txt), August 1998.

[20] Patrick McManus, "A passive system for server selection within mirrored resource environments using AS path length heuristics," Tech. Rep., AppliedTheory Corp., March 1999, http://proximate.appliedtheory.com.

[21] Katia Obraczka and Fabio Silva, "Looking at network latency for server proximity," Tech. Rep. USC-CS-99-714, Department of Computer Science, University of Southern California, 1999.

[22] Mehmet Sayal, Yuri Breitbart, Peter Scheuermann, and Radek Vingralek, "Selection algorithms for replicated web servers," in *Proceeding of Workshop on Internet Server Performance (WISP98)*, Madison, WI, June 1998.

[23] Christopher Davis, Paul Vixie, Tim Goodwin, and Ian Dickinson, "A means for expressing location information in the domain name system," Internet Request for Comments (RFC 1876), January 1996.

[24] "Definition of the DNS GL resource record used to encode geographic locations," Internet Draft (draft-costanzo-dns-gl-02.txt), May 2000.

[25] David Moore, Ram Periakaruppan, and Jim Donohoe, "Where in the world is netgeo.caida.org?," in *Proceedings of the Internet Society Conference (INET)*, 2000, http://www.caida.org/Tools/NetGeo/inet2000_paper.

[26] "A DNS RR for specifying the location of services (DNS SRV)," Internet Request for Comments (RFC 2782), February 2000.

[27] Wenting Tang, Fan Du, Matt W. Mutka, Lionel M. Ni, and Abdol-Hossein Esfahanian, "Supporting global replicated services by a routing-metric-aware DNS," in *Proceedings of 2nd International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS 2000)*, Milpitas, CA, June 2000.

[28] Jussi Kangasharju, Keith W. Ross, and James W. Roberts, "Locating copies of objects using the domain name system," in *Proc. 4th Web Caching Workshop*, San Diego, CA, March 1999.

[29] Edith Cohen and Haim Kaplan, "Prefetching the means for document transfer: A new approach for reducing web latency," in *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March 2000.