On the Effectiveness of DNS-based Server Selection

Anees Shaikh Renu Tewari Mukesh Agrawal

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 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 reducing DNS TTL values on web access latency and show that it 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.**

I. 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 perform 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 both in 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, Digital Island), 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 across distributed Web sites. Examples of such products include Cisco Distributed Director, F5 3/DNS, and Nortel/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 largely 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 performance and scalability of the DNS. Second, inherent in the DNS-based approach is the assumption that clients and their local nameservers are proximal. When DNS-based server selection is used to choose a nearby server, the decision is based on the nameserver's identity, not the client's. Thus when clients and nameservers are not proximal, the DNS-based approach may lead to poor decisions.

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 be transmitted in the network, rather than being served from local nameserver caches.

Another, more subtle, issue arises when DNS-based redirection is used to find a server or replica nearby the client. DNS-based redirection 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

A. Shaikh and R. Tewari are with the IBM T.J. Watson Research Center, Hawthorne, NY, USA. E-mail: {aashaikh,tewarir}@watson.ibm.com .

M. Agrawal is with Carnegie Mellon University, Pittsburgh, PA, USA. Email: mukesh@cs.cmu.edu. This work was done while he was visiting the IBM T.J. Watson Research Center from the University of Michigan.

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 two issues and quantify their impact on DNS-based server selection schemes. Our main contributions are:

- quantification of the impact of low DNS TTL values on client-perceived latency;
- extensive measurement and analysis of client and local nameserver proximity.

Using data from an ISP proxy as well as popular Web sites, we empirically study and quantify the impact of low DNS TTL values used in many content distribution networks (CDNs). In particular, we quantify their effect on client-perceived latency when accessing web pages with multiple embedded objects. Previous studies have measured name resolution latency in a general sense, but not in situations where the amount of DNS caching is controlled or where the effects of embedded objects are quantified.

We conduct an extensive study of the proximity of ISP clients to their local nameservers to gauge the potential inaccuracy of using the identity of the client's nameserver to make server selection decisions. To our knowledge there has been no earlier work on analyzing client-nameserver proximity as it relates to server selection.

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 III discusses and quantifies the effects of using small TTL values on client-perceived Web access latency. Section IV 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 V briefly suggests a modification to the DNS protocol to address the problem of identifying clients during name resolution. Section VI summarizes some representative related work and Section VII concludes the paper.

II. 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 the system. A large TTL value reduces the load on the nameserver but limits the frequency of update propagation through the system. The different types of resource records and additional details about the DNS

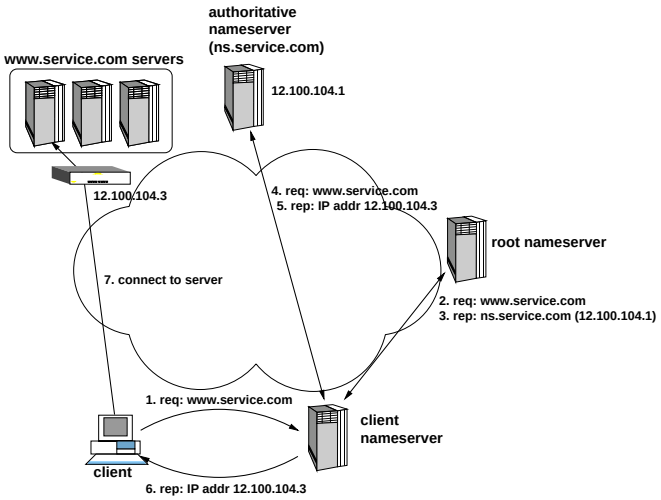


Fig. 1. **Basic DNS operation**: This example shows the basic steps required for a client to resolve the address of a service at `www.service.com`.

are described in [1], [4]. The most widely used nameserver implementation in the DNS is the Berkeley Internet Name Domain (BIND) [5].

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, however, typically accept recursive queries from clients (i.e., end-hosts).

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).

III. Impact of DNS TTL Values

The scalability and performance of the DNS largely depends on the caching of resource records across intermediate name-

[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 server names	581
unique URLs	7632
duration of trace	6 hrs (10am-1pm, 6pm-9pm)
trace date	February 1999

TABLE I
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 II
NAME RESOLUTION LATENCY

servers. 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 library 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 related ways: (i) they increase cache misses, thereby increasing the number of queries that must be handled by 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. Current high-end Web servers are able to process and service several thousands of simultaneous HTTP GET requests [8], each of which is likely to incur much higher overhead than handling name resolution requests, which require simple lookups and single-packet responses. And in the case of Web access, the number of name resolutions is bounded by the number of URL accesses. Thus it is reasonable to expect that modern servers would be able to keep pace with DNS requests. Early studies showed that the increase in network traffic due to additional UDP DNS packets is not insignificant [9], however, and would be even higher as caching in the DNS is reduced.

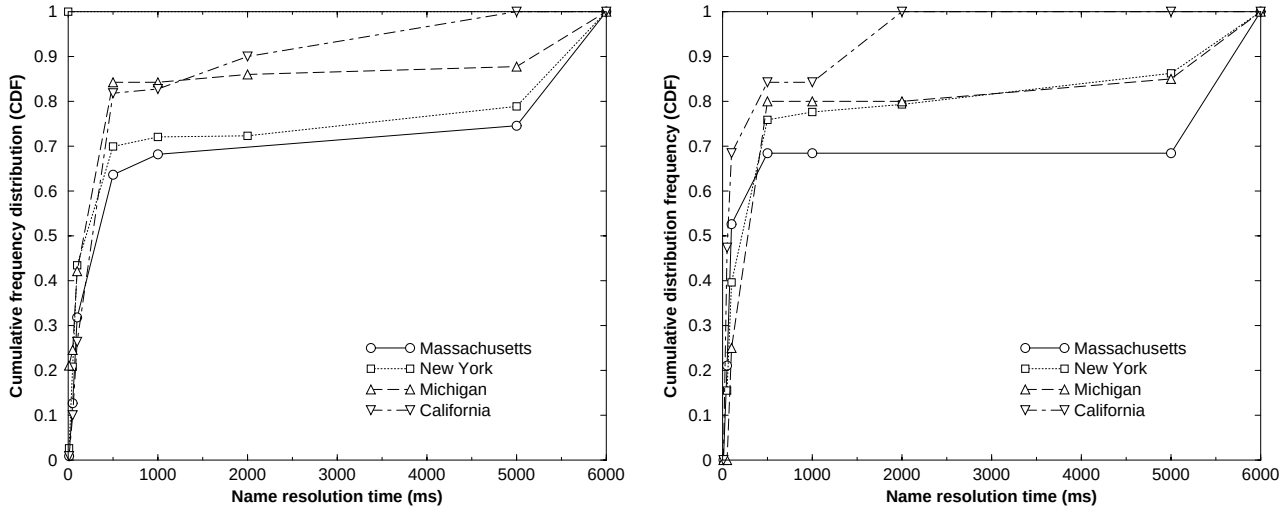
For client-observed latency, on the other hand, TTL values have a much greater impact. To quantify this effect, we first analyzed the overhead of a single name resolution and compared it to the total Web page download latency. Second, we determined the distribution of embedded objects (e.g., images and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

A. Name Resolution Overhead

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 overhead we compiled a list of server names from the proxy logs at a single point-of-presence (POP) location of a medium-sized ISP. Table I 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.

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 II. The results show that caching reduces the



(a) Proxy logs (b) Popular sites

Fig. 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 overhead for the most popular sites.

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 [10] and repeated the name lookup experiments. According to the service, these sites had a combined user population of 76 million clients. 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 New York results show, for example, that 25% of the name lookups (with no caching) add an overhead of more than 3 seconds for the ISP proxy log sites, and more than 650 ms for the popular 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].

B. Impact of Embedded Objects

Most Web pages accessed today contain a number of embedded objects. These objects, 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 co-located, each object access may require an additional name resolution. 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 I) 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. For the popular sites the embedded objects included images, advertisements, and also objects or page fragments that are generated via a URL-designated script. An example of the latter might be a link to an off-site server script which generates weather data for a personalized Web page.

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 [11].

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 used *Page Detailer* to measure the download time for each object. *Page Detailer* is a tool that enumerates the objects contributing to Web page access latency and measures the time to retrieve each embedded object [12] .

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, measured from the New York site, are shown in Table III.

The results show that if all the embedded objects 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 overhead is quite small. When neither the nameserver nor the server address is in the cache (case i), how-

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

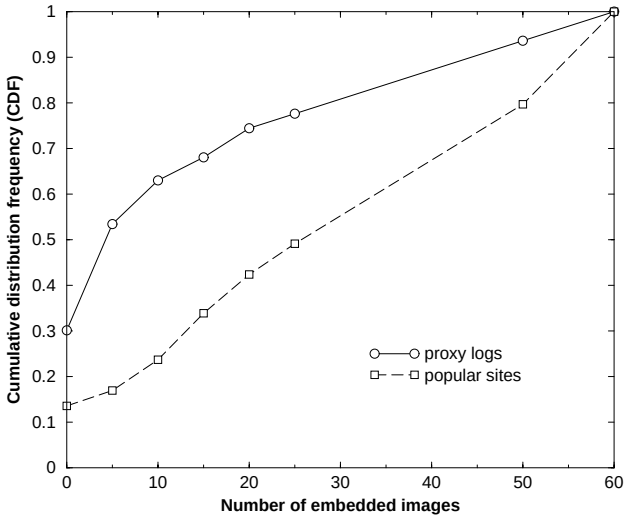


Fig. 3. **Embedded object distribution**: This graph shows the distribution of the number of embedded objects per Web page from the ISP proxy sites and the most popular Web sites.

ever, the overhead grows to about 3% (200 ms for the name resolution versus about 6 seconds for the entire page download).

The name lookup overhead becomes an order of magnitude higher when each embedded object 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 in Table II, performing a name lookup for each embedded object adds an overhead of 48% on a cache miss (200 ms for the name lookup and 415 ms for the embedded object download). The large name resolution overhead suggests several considerations: additional DNS queries should be amortized over large page downloads, and embedded objects should be co-located, as much as possible, to avoid multiple DNS queries.

Our analysis of top-level pages at the popular sites showed that each Web page requires, on average, 3.7 name resolution operations. That is, one for the server name itself, and 2.7 additional name lookups for embedded objects located on other servers. Thus, when accessing a single page, the overhead due to name resolution can be significant, and is likely to increase as more objects are delivered from alternate servers or content distribution services. For example, we observed one case among the popular sites with 12 objects on the top-level page, 8 of which were located on different servers.

We also computed the average number of name resolution operations required per object as the ratio of name lookups per page to the number of embedded objects per page. The ratio came to 0.13, indicating that, on average, about one name lookup is required for every 8 objects downloaded.

The DNS TTL value should 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 no mechanism to flush cached name-to-address mappings in client-side nameserver caches.

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 III
PAGE DOWNLOAD STATISTICS

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.

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

A. 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

	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 IV
DNS AND HTTP LOG STATISTICS

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 of 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 IV shows some basic statistics about both sets of logs.

We used information in the global Internet Routing Registry (IRR) to determine autonomous system (AS) numbers for each IP address [13], [14]. 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.

B. Matching Clients and Nameservers

Before characterizing client-nameserver proximity we used the logs to match clients with their configured nameservers. We relied primarily on timestamps for the correlation of DNS requests with HTTP requests. Since the authoritative DNS server returns addresses with a zero TTL, we would expect 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. 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 developed a multi-step algorithm to remove as much uncertainty as possible. Since we relied on timestamps to perform the matching, we first tried 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 assumed that these addresses are proxies or firewalls that perform both HTTP and DNS requests on behalf of clients, and considered such cases to be certain matches. Using these certain matches we determined the mean clock skew and used it in the subsequent steps. What we refer to here as clock skew also includes the delay between the name resolution request and the corresponding HTTP request.

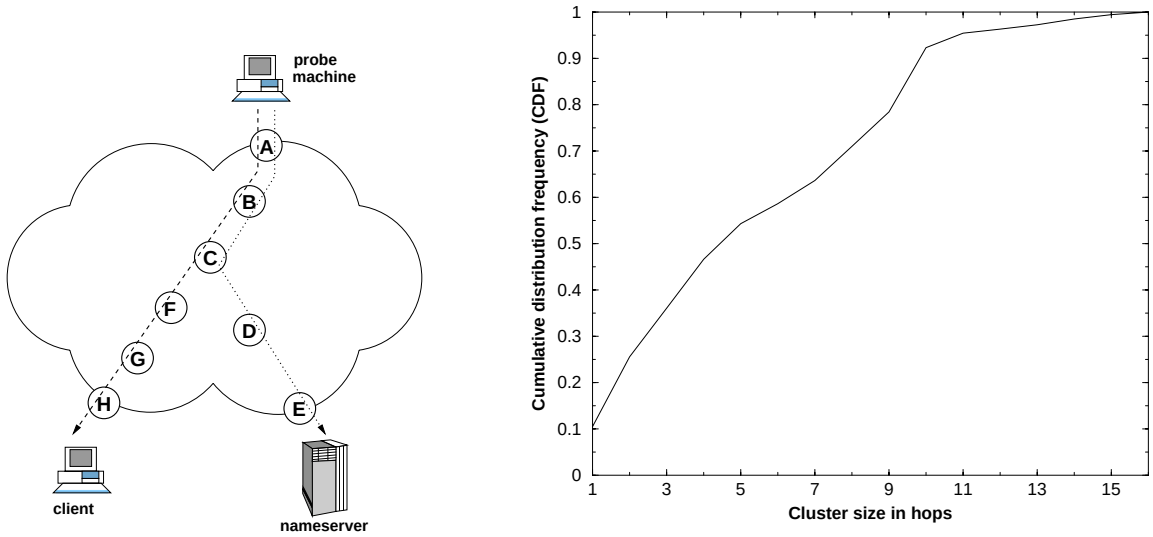
In the first pass we considered each HTTP request in turn and constructed a list of candidate nameservers with a nearby timestamp (with "nearby" defined as within a 4-10 second window), accounting for the skew and the expected browser caching (which we assume is approximately 15 minutes). The second pass performed the same process in reverse, considering each nameserver address sighting in the DNS logs and constructing a list of likely clients served by the nameserver, according to the timestamp and the name being queried. Finally, we combined the two sets of candidate lists to identify client-nameserver pairs that appeared in both lists. Using this process we were able to find candidate lists for 2394 clients (approximately 10% of all clients). More than 60% of these clients matched to one nameserver. Note that these candidate lists are based on matching clients and nameservers using only timestamps.

When a client matched to more than one candidate nameserver, we used 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. Using these conservative heuristics, we were able to find 324 client-nameserver pairs (14% of the clients for which a candidate list was found using timestamps).

C. Proximity Evaluation Using Access Logs

After determining the set of client-nameserver pairs from the DNS and HTTP logs, the next step was 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. Since we used domain names and AS numbers as heuristics to *determine* matching pairs, evaluating these metrics is somewhat misleading. Hence, we did not consider them further. If we assume that IP prefix lengths are one, two, or three octets, we found that 37%, 19%, and 10% of the client-nameserver pairs shared the same prefix. 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 [15]. Therefore, these numbers may underestimate the actual matches if the real prefix length is not 8, 16, or 24 bits.

A better metric for determining client-nameserver proximity is network hops which we measured from a probing site in the network that might represent, for example, a candidate server site. We used the *traceroute* tool to learn the network



(a) Clustering clients and nameservers

(b) Distribution of cluster sizes

Fig. 4. **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.

path from the probing site to the client and nameserver. Then we found the maximum hopcount until a common ancestor appears in the paths to determine the "cluster" size of the client-nameserver pair. The cluster size is thus defined as the maximum disjoint path length. This process is illustrated in Figure 4(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 4(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 the matching process is conservative, since it removed misconfigured client-nameserver pairs. The actual number of clients that are topologically distant from their nameservers is likely to be higher.

D. 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 [16], [17]. 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.

We obtained dial-up accounts from 9 National retail ISPs [18] 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 V 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.

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

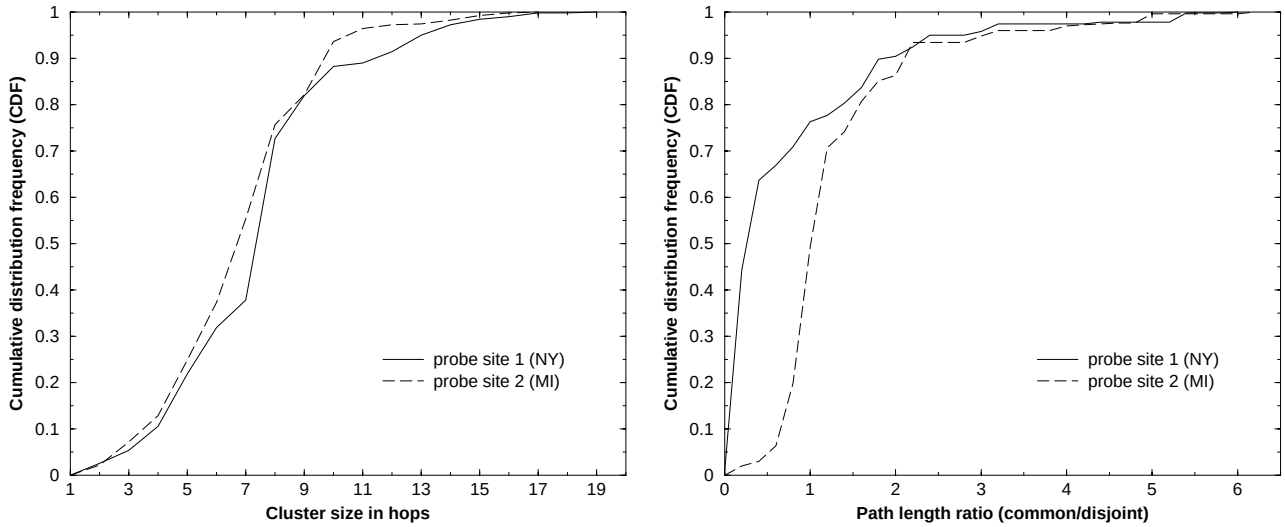
TABLE V
ISP ADDRESS STATISTICS

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.

E. Proximity Evaluation Using ISP Measurements

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 addresses. We found several cases, for example, where the path to the advertised DNS server address ended at a different address when traced from different probe sites.

We first measured the size of client-nameserver clusters as viewed from the two probing points, using the same technique shown in Figure 4(a). The graph in Figure 5(a) shows simi-



(a) Client-nameservers clusters

(b) Ratio of common to disjoint path length

Fig. 5. **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.

lar clustering to the log-based results in Section IV-C. 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.

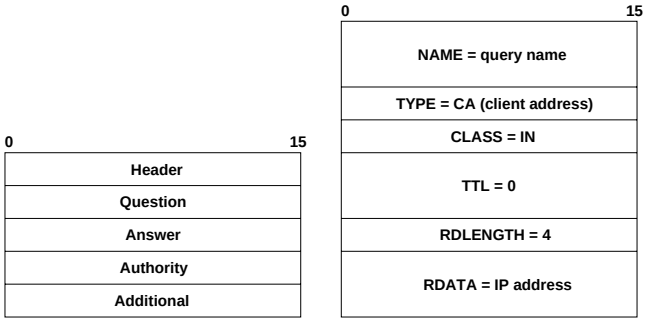
We compared these results with the direct client-nameserver topological distance and found that the average distance over all pairs was 7.6 hops, with a median of 8. Some clients were as far as 15 hops from their nameservers. The average client-to-nameserver round-trip latency was 234 ms, though this 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 4(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 5(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 at least 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 or less, 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[3]. 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 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 ρ between $a = t_d^1 - t_d^2$ and $b = t_c^1 - t_c^2$, and

[3]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

Fig. 6. **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.

found that $\rho = 0.32$. Thus, a and b are positively correlated, but only weakly so.

V. 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 IV, 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 6(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 6(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.

This scheme will also work when clients are behind firewalls or proxies that hide actual client IP addresses. In many cases the firewall or proxy makes the DNS query on behalf of the clients. Thus, the address carried to the DNS load-balancing nameserver is precisely what is needed, since the HTTP requests also originate from the firewall address.

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.

VI. 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 [19]. The SONAR service provides an interface between applications and proximity estimation services [20].

Related to proximity measurement is the question of which metrics provide the best indication of actual latency. Recent work has considered network hops, AS hops, and RTT metrics, along with various means of collecting them, including active probing or passive participation in BGP peering sessions [21], [22], [23].

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 [24]. Similarly, the GL resource record encodes location information in terms of hierarchical locator (country code, postal code) and a textual address [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]. Earlier work suggests using the existing DNS zone transfer mechanism as a way to add flexible load-balancing capability to a nameserver [27].

Finally, some recent work has proposed new mechanisms to reduce client latency related to name resolution using prefetching or proactive cache management techniques [28], [7]. This work further affirms that DNS caching plays a crucial role in determining client-perceived latency.

VII. Conclusion

This paper explored two important issues related to DNS-based server selection. 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 nameservers 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.

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: Addressing a performance bottleneck," in *Proc. of 2001 Symposium on Applications and the Internet (SAINT-2001)*, San Diego, CA, January 2001, `http://www.research.att.com/~edith`.

[8] Standard Performance Evaluation Corporation, "SPECweb 99 benchmark, performace results," `http://www.spec.org/osg/web99`, June 2000.

[9] 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.

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

[11] 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.

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

[13] "RADB Internet routing registry," `http://www.radb.net`.

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

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

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

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

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

[19] 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.

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

[21] 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`.

[22] 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.

[23] 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.

[24] 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.

[25] "Definition of the DNS GL resource record used to encode geographic locations," Internet Draft (draft-costanzo-dns-gl-03.txt), June 2000.

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

[27] Thomas P. Brisco, "DNS support for load balancing," Internet Request for Comments (RFC 1794), April 1995.

[28] 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.