DNS Performance and the Effectiveness of Caching

Jaeyeon Jung, Emil Sit, Hari Balakrishnan, *Member, IEEE*, and Robert Morris

Abstract—**This paper presents a detailed analysis of traces of DNS and associated TCP traffic collected on the Internet links of the MIT Laboratory for Computer Science and the Korea Advanced Institute of Science and Technology (KAIST). The first part of the analysis details how clients at these institutions interact with the wide-area domain name system, focusing on client-perceived performance and the prevalence of failures and errors. The second part evaluates the effectiveness of DNS caching.**

In the most recent MIT trace, 23% of lookups receive no answer; these lookups account for more than half of all traced DNS packets since query packets are retransmitted overly persistently. About 13% of all lookups result in an answer that indicates an error condition. Many of these errors appear to be caused by missing inverse (IP-to-name) mappings or `NS` records that point to non-existent or inappropriate hosts. 27% of the queries sent to the root name servers result in such errors.

The paper also presents the results of trace-driven simulations that explore the effect of varying TTLs and varying degrees of cache sharing on DNS cache hit rates. Due to the heavy-tailed nature of name accesses, reducing the TTLs of address (`A`) records to as low as a few hundred seconds has little adverse effect on hit rates, and little benefit is obtained from sharing a forwarding DNS cache among more than 10 or 20 clients. These results suggest that client latency is not as dependent on aggressive caching as is commonly believed, and that the widespread use of dynamic, low-TTL `A`-record bindings should not greatly increase DNS related wide-area network traffic.

Keywords—**DNS, Measurement, Performance, Caching, Internet**

I. INTRODUCTION

The Domain Name System (DNS) is a globally distributed database that maps names to network locations, thus providing information critical to the operation of most Internet applications and services. As a global service, DNS must be highly scalable and offer good performance under high load. In particular, the system must operate efficiently to provide low latency responses to users while minimizing the use of wide-area network resources.

It is widely believed that two factors contribute to the scalability of DNS: hierarchical design around administratively delegated name spaces, and the aggressive use of caching. Both factors seek to reduce the load on the root servers at the top of the name space hierarchy, while successful caching hopes to limit client-perceived delays and wide-area network bandwidth usage. How effective are these factors? In this paper, we carefully analyze three network traces to study this question.

Prior to the year 2000, the only large-scale published study of DNS performance was by Danzig *et al.* in 1992 [1]. Danzig's study found that a large number of implementation errors caused DNS to consume about twenty times more wide-area network bandwidth than necessary. However, since then, DNS implementations and DNS usage pattern have changeds. For example, the World Wide Web now causes the bulk of traffic. Content distribution networks (CDNs) and popular Web sites now use DNS as a level of indirection to balance load across servers, provide fault tolerance, or route client requests to servers topologically close to the clients. Because cached DNS records limit the efficacy of such techniques, many of these multiple-server systems use TTL values as small as a few seconds or minutes. Another example is in mobile networking, where dynamic DNS together with low-TTL bindings can provide the basis for host mobility support in the Internet [2]. These uses of DNS all conflict with caching.

One concrete way to estimate the effectiveness of DNS caching is to observe the amount of DNS traffic in the wide-area Internet. Danzig *et al.* report that 14% of all wide-area packets were DNS packets in 1990, compared to 8% in 1992. In 1995, the corresponding number from a study of the NSFNET by Frazer was 5% [3]; a 1997 study of the MCI backbone by Thompson *et al.* reported that 3% of wide-area packets were DNS related [4]. This downward trend might suggest that DNS caching is working well.

However, these results should be put in perspective by considering them relative to network traffic as a whole. Thompson's study also showed that DNS accounts for 18% of all flows (where a flow is defined as a *uni-directional* traffic stream with unique source and destination IP addresses, port numbers and IP protocol fields). If one assumes that applications typically precede each TCP connection with a call to the local DNS resolver library, this suggests a DNS cache miss rate of a little less than 25%. However, by 1997, most TCP traffic consisted of Web traffic, which tends to produce groups of about four connections to the same server [5]; if one assumes one DNS lookup for every four TCP connections, the "session-level" DNS cache miss rate appears to be closer to 100%. While an accurate evaluation requires more precise consideration of the number of TCP connections per session and the number of DNS packets per lookup, this quick calculation suggests that DNS caching is not very effective at suppressing wide-area traffic.

These considerations make a thorough analysis of the effectiveness of DNS caching is especially important. Thus, this paper has two goals. First, it seeks to understand the performance and behavior of DNS from the point of view of clients and, second, it evaluates the effectiveness of caching.

A. *Summary of Results*

In exploring DNS performance and scalability, we focus on the following questions:

1. What performance, in terms of latency and failures, do DNS clients perceive?
2. How does varying the TTL and degree of cache sharing impact caching effectiveness?

These questions are answered using a novel method of analyzing traces of TCP traffic along with the related DNS traffic. To facilitate this, we captured all DNS packets and TCP `SYN`, `FIN`, and `RST` packets at two different locations on the Inter-

This research was sponsored by Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare Systems Center San Diego under contract N66001-00-1-8933.

The authors are with the MIT Laboratory for Computer Science, 200 Technology Square, Cambridge, MA 02139 USA (e-mail:jyjung@lcs.mit.edu, sit@lcs.mit.edu, hari@lcs.mit.edu, rtm@lcs.mit.edu).

net. The first is at the link that connects MIT's Laboratory for Computer Science (LCS) and Artificial Intelligence Laboratory (AI) to the rest of the Internet. The second is at a link that connects the Korea Advanced Institute of Science and Technology (KAIST) to the rest of the Internet. We analyze two different MIT data sets, collected in January and December 2000, and one KAIST data set collected in May 2001.

One surprising result is that over a third of all lookups are not successfully answered. 23% of all client lookups in the most recent MIT trace fail to elicit any answer. In the same trace, 13% of lookups result in an answer that indicates an error. Most of these errors indicate that the desired name does not exist. While no single cause seems to predominate, inverse lookups (translating IP addresses to names) often cause errors, as do `NS` records that point to non-existent servers.

DNS servers also appear to retransmit overly aggressively. The query packets for these unanswered lookups, including retransmissions, account for more than half of all DNS query packets in the trace. Loops in name server resolution are particularly bad, causing an average of 10 query packets sent to the wide area for each (unanswered) lookup. In contrast, the average answered lookup sends about 1.3 query packets. Loops account for 3% of all unanswered lookups.

We have also been able to observe changes in DNS usage patterns and performance. For example, the percentage of TCP connections made to names with low TTL values increased from 12% to 25% between January and December 2000, probably due to the increased deployment of DNS-based server selection for popular sites. Also, while median name resolution latency was less than 100 ms, the latency of the worst 10% grew substantially between January and December 2000.

The other portion of our study concerns caching effectiveness. The relationship between numbers of TCP connections and numbers of DNS lookups in the MIT traces suggests that the hit rate of DNS caches inside MIT is between 80% and 86%. Since this estimate includes the effects of web browsers opening multiple TCP connections to the same server, DNS A-record caching does not seem particularly effective; the observed cache hit rate could easily decrease should fewer parallel TCP connections be used, for example. Moreover, we find that the distribution of names is Zipf-like, which immediately limits even the theoretical effectiveness of caching.

The captured TCP traffic helps us perform trace-driven simulations to investigate two important factors that affect caching effectiveness: (i) the TTL values on name bindings, and (ii) the degree of aggregation due to shared client caching. Our simulations show that `A` records with 10-minute TTLs yield almost the same hit rates as substantially longer TTLs. Furthermore, we find that a cache shared by as few as ten clients has essentially the same hit rate as a cache shared by the full traced population of over 1000 clients. This is consistent with the Zipf-like distribution of names.

These results suggest that DNS works as well as it does despite ineffective `A`-record caching, and that the current trend towards more dynamic use of DNS (and lower TTLs) is not likely to be harmful. On the other hand, we find that `NS`-record caching is critical to DNS scalability by reducing load on the root and gTLD servers.

The rest of this paper presents our findings and substantiates these conclusions. Section II presents an overview of DNS and surveys previous work in analyzing its performance. Section III describes our traffic collection methodology and some salient features of our data. Section IV analyzes the client-perceived performance of DNS, while Section V analyzes the effectiveness of caching using trace-driven simulation. We conclude with a discussion of our findings in Section VI.

II. Background

In this section, we present an overview of DNS and survey related work.

A. DNS Overview

The design of the Internet DNS is specified in [6], [7], [8]. We summarize the important terminology and basic concepts here.

The basic function of DNS is to provide a distributed database that maps between human-readable host names (such as `chive.lcs.mit.edu`) and IP addresses (such as `18.31.0.35`). It also provides other important information about the domain or host, including reverse maps from IP addresses to host names and mail-routing information. Clients (or *resolvers*) routinely query name servers for values in the database.

The DNS name space is hierarchically organized so that subdomains can be locally administered. The root of the hierarchy is centrally administered and served from a collection of thirteen (in mid-2001) *root servers*. Sub-domains are *delegated* to other servers that are *authoritative* for their portion of the name space. This process may be repeated recursively.

At the beginning of our study, most of the root servers also served the top-level domains, such as `.com`. At the end, the top-level domains were largely served by a separate set of about a dozen dedicated "generic top-level domain" (gTLD) servers.

Mappings in the DNS name space are called *resource records*. Two common types of resource records are address records (`A` records) and name server records (`NS` records). An `A` record specifies a name's IP address; an `NS` record specifies the name of a DNS server that is authoritative for a name. Thus, `NS` records are used to handle delegation paths.

Since achieving good performance is an important goal of DNS, it makes extensive use of caching to reduce server load and client latency. It is believed that caches work well because DNS data changes slowly and a small amount of staleness is tolerable. On this premise, many servers are not authoritative for most data they serve, but merely cache responses and serve as local proxies for resolvers. Such proxy servers may conduct further queries on behalf of a resolver to complete a query *recursively*. Clients that make recursive queries are known as *stub resolvers* in the DNS specification. On the other hand, a query that requests only what the server knows authoritatively or out of cache is called an *iterative* query.

Figure 1 illustrates these two resolution mechanisms. The client application uses a stub resolver and queries a local nearby server for a name (say `www.mit.edu`). If this server knows absolutely nothing else, it will follow the steps in the figure to arrive at the addresses for `www.mit.edu`. Requests will begin at a well-known root of the DNS hierarchy. If the queried server

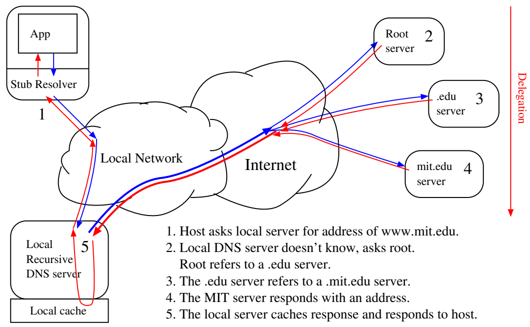


Fig. 1. Example of a DNS lookup sequence.

has delegated responsibility for a particular name, it returns a *referral* response, which is composed of name server records. The records are the set of servers that have been delegated responsibility for the name in question. The local server will choose one of these servers and repeat its question. This process typically proceeds until a server returns an answer.

Caches in DNS are typically not size-limited since the objects being cached are small, consisting usually of no more than a hundred bytes per entry. Each resource record is expired according to the time set by the originator of the name. These expiration times are called *Time To Live (TTL)* values. Expired records must be fetched afresh from the authoritative origin server on query. The administrator of a domain can control how long the domain's records are cached, and thus how long changes will be delayed, by adjusting TTLs. Rapidly changing data will have a short TTL, trading off latency and server load for fresh data.

To avoid confusion, the remainder of this paper uses the terms "lookup," "query," "response," and "answer" in specific ways. A *lookup* refers to the entire process of translating a domain name for a client application. A *query* refers to a DNS request packet sent to a DNS server. A *response* refers to a packet sent by a DNS server in reply to a query packet. An *answer* is a response from a DNS server that terminates the lookup, by returning either the requested name-to-record mapping or an error indication. Valid responses that are not answers must be referrals.

This means, for example, that a lookup may involve multiple query and response packets. The queries of a lookup typically ask for the same data, but from different DNS servers; all responses but the last one (the answer) are typically referrals. This distinction can be seen in Figure 1; the packets in steps 1–4 are all part of the same lookup (driven by the request from the application); however, each step represents a separate query and response.

B. *Related Work*

In 1992, Danzig *et al.* presented measurements of DNS traffic at a root name server [1]. Their main conclusion was that the majority of DNS traffic is caused by bugs and misconfiguration. They considered the effectiveness of DNS name caching and retransmission timeout calculation, and showed how algorithms to increase resilience led to disastrous behavior when servers failed or when certain implementation faults were triggered. Implementation issues were subsequently documented by Kumar *et al.*, who note that many of these problems have been fixed in more recent DNS servers [9]. Danzig *et al.* also found that one third of wide-area DNS traffic that traversed the NSFnet was destined to one of the (at the time) seven root name servers.

In contrast to Danzig *et al.*'s work, our work focuses on analyzing client-side performance characteristics. In the process, we calculate the fraction of lookups that caused wide-area DNS packets to be sent, and the fraction that caused a root or gTLD server to be contacted.

In studies of wide-area traffic in general, DNS is often included in the traffic breakdown [3], [4]. As noted in Section I, the high ratio of DNS to TCP flows in these studies motivated our investigation of DNS performance.

It is likely that DNS behavior is closely linked to Web traffic patterns, since most wide-area traffic is Web-related and Web connections are usually preceded by DNS lookups. One result of Web traffic studies is that the popularity distribution of Web pages is heavy-tailed [10], [11], [12]. In particular, Breslau *et al.* conclude that the Zipf-like distribution of Web requests causes low Web cache hit rates [10]. We find that the popularity distribution of DNS names is also heavy-tailed, probably as a result of the same underlying user behavior. It is not immediately clear that DNS caches should suffer in the same way that Web caches do. For example, DNS caches do not typically incur cache misses because they run out of capacity. DNS cache misses are instead driven by the relationship between TTLs selected by the origin and the interarrival time between requests for each name at the cache. DNS cache entries are also more likely to be reused because each component of a hierarchical name is cached separately and also because many Web documents are present under a single DNS name. Despite these differences, we find that DNS caches are similar to Web caches in their overall effectiveness.

A recent study by Shaikh *et al.* shows the impact of DNS-based server selection on DNS [13]. This study finds that extremely small TTL values (on the order of seconds) are detrimental to latency, and that clients are often not close in the network topology to the name servers they use, potentially leading to sub-optimal server selection. In contrast, we believe that the number of referrals for a lookup is a more important determiner for latency.

Wills and Shang studied NLANR proxy logs and found that DNS lookup time contributed more than one second to approximately 20% of retrievals for the Web objects on the home page of larger servers. They also found that 20% of DNS requests are not cached locally [14]; this correlates nicely with estimates given in Section I and corroborates our belief that DNS caching is not very effective at suppressing wide-area traffic. Cohen and Kaplan propose proactive caching schemes to alleviate the latency overheads by synchronously requesting expired DNS records [15]; their analysis is also derived from NLANR proxy log workload. Unfortunately, proxy logs do not capture the actual DNS traffic; thus any analysis must rely on on measurements taken after the data is collected. This will not accurately reflect the network conditions at the time of the request, and the DNS records collected may also be newer. Our data allows us to directly measure the progress of the DNS lookup as it occurred. Additionally, our data captures all DNS lookups and their related TCP connections, not just those associated with

HTTP requests.

Huitema and Weerahandi measured DNS latency through the `gethostbyname()` interface over a period of 15 months, starting from April 1999. For their study, 29% of DNS lookups took longer than 2 seconds to get an answer [16]. In comparision, our study shows that between 10% and 24% lookups give this much latency. These numbers differ because our latency numbers do not include latency experienced between the client application and the local name server which *is* included in their data. Naturally, DNS latency is also affected by the connectivity and the performance of the network at the point of measurement.

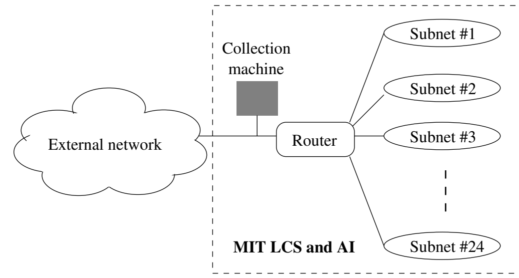
Brownlee *et al.* collected DNS traffic at `F.root-servers.net`, and showed that over 14% of the observed query load was due to bogus queries; their paper provides an analysis of the load and a taxonomy of the causes of errors [17]. Errors include repeated queries generated from same source host, queries for non-existent top level domain names, and malformed A queries. Some of these are observed in our traces and are listed in section IV-C. In particular, we found that between 15% and 27% of the lookups sent to root name servers resulted in negative responses.

Another study by the same authors show passive measurements of the performance of root and gTLD servers as seen from their campus network using NeTraMet meters [18]. The paper presents response time, request rate and request loss rate of the root and gTLD servers seen at the traced network. Their response times indicate the latency between a single query and response, as compared to our latencies which cover the entire lookup process. Their methodology is also targetted to measure overall performance. In contrast, our analyses consider the entire DNS packet header and payload to reveal a more comprehensive view of DNS behavior.

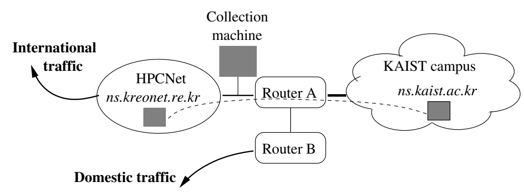
III. The Data

Our study is based on three separate traces. The first two were collected in January and December 2000 respectively, at the link that connects the MIT LCS and AI labs to the rest of the Internet. At the time of the study, there were 24 internal subnetworks sharing the router, used by over 500 users and over 1200 hosts. The third trace was collected in May 2001 at one of two links connecting KAIST to the rest of the Internet. At the time of data collection there were over 1000 users and 5000 hosts connected to the KAIST campus network. The trace includes only international TCP traffic; KAIST sends domestic traffic on a path that was not traced. However, the trace does include all external DNS traffic, domestic and international: the primary name server of the campus, `ns.kaist.ac.kr`, was configured to forward all DNS queries to `ns.kreonet.re.kr` along a route that allowed them to be traced. Figure 2 shows the configurations of the two networks.

The first trace, `mit-jan00`, was collected from 2:00 A.M. on 3 January 2000 to 2:00 A.M. on 10 January 2000; the second, `mit-dec00`, was collected from 6:00 P.M. on 4 December to 6:00 P.M. on 11 December 2000. The third set, `kaist-may01`, was collected at KAIST from 5:00 A.M. on 18 May to 5:00 A.M. on 24 May 2001. All times are EST.



(a) MIT LCS: There are 24 internal subnetworks sharing the border router.



(b) KAIST: The collection machine is located at a point that captures all DNS traffic, but only international traffic of other types. This because the primary KAIST name server, `ns.kaist.ac.kr`, forwards all DNS queries through the traced link to `ns.kreonet.re.kr`.

Fig. 2. Schematic topology of the traced networks.

A. Collection Methodology

We filtered the traffic observed at the collection points to produce a data set useful for our purposes. As many previous studies have shown, TCP traffic (and in particular, HTTP traffic) comprises the bulk of wide-area traffic [4]. TCP applications usually depend on the DNS to provide the rendezvous mechanism between the client and the server. Thus, TCP flows can be viewed as the major driving workload for DNS lookups.

In our study, we collected both the DNS traffic and its driving workload. Specifically, we collected:

1. Outgoing DNS queries and incoming responses, and
2. Outgoing TCP connection start (`SYN`) and end (`FIN` and `RST`) packets for connections *originating* inside the traced networks.

In the `mit-jan00` trace, only the first 128 bytes of each packet were collected because we were unsure of the space requirements. However, because we found that some DNS responses were longer than this, we captured entire Ethernet packets in the other traces.

The trace collection points do not capture *all* client DNS activity. Queries answered by caches inside the traced networks do not appear in the traces. Thus many of our DNS measurements reflect only those lookups that required wide-area queries to be sent. Since we correlate these with the driving workload of TCP connections, we can still draw conclusions about overall performance and caching effectiveness.

In addition to filtering for useful information, we also eliminated information to preserve user (client) privacy. In the MIT traces, any user who wished to be excluded from the collection process was allowed to do so, based on an IP address they provided; only three hosts opted out, and were excluded from all our traces. We also did not capture packets corresponding to reverse DNS lookups (PTR queries) for a small number of names within MIT, once again to preserve privacy. In addition, all packets were rewritten to anonymize the source IP addresses of hosts inside the traced network. This was done in a pseudo-random fashion—each source IP address was mapped using a keyed MD5 cryptographic hash function [19] to an essentially unique, anonymized one.

Our collection software was derived from Minshall's `tcpdpriv` utility [20]. `tcpdpriv` anonymizes `libpcap`-format traces (generated by `tcpdump`'s packet capture library). It can collect traces directly or post-process them after collection using a tool such as `tcpdump` [21]. We extended `tcpdpriv` to support the anonymization scheme described above for DNS traffic.

B. Analysis Methodology

We analyzed the traces to extract various statistics about lookups including the number of referrals involved in a typical lookup and the distribution of lookup latency. To calculate the latency in resolving a lookup, we maintain a sliding window of the lookups seen in the last sixty seconds; an entry is added for each query packet from an internal host with a DNS query ID different from any lookup in the window. When an incoming response packet is seen, the corresponding lookup is found in the window. If the response packet is an answer (as defined in Section II-A), the time difference between the original query packet and the response is the lookup latency. The actual end-user DNS request latency, however, is slightly longer than this, since we see packets in mid-flight. If the response is not an answer, we increment the number of referrals of the corresponding lookup by one, and wait until the final answer arrives. To keep track of the name servers contacted during a lookup, we maintain a list of all the IP addresses of name servers involved in the resolution of the lookup.

This method correctly captures the list of servers contacted for iterative lookups, but not for recursive lookups. Most lookups in the MIT traces are iterative; we eliminated the small number of hosts which sent recursive lookups to name servers outside the traced network. The KAIST traces contain mostly recursive lookups sent to a forwarding server just outside the trace point; hence, while we can estimate lower bounds on name resolution latency, we cannot derive statistics on the number of referrals or the fraction of accesses to a top-level server.

C. Data Summary

Table I summarizes the basic characteristics of our data sets. We categorize lookups based on the DNS code in the response they elicit, as shown in rows 3-6 of Table I. A lookup that gets a response with non-zero response code is classified as a *negative answer*, as defined in the DNS specification [7], [22]. A *zero answer* is authoritative and indicates no error, but has no ANSWER, AUTHORITY, or ADDITIONAL records [9]. A zero answer can arise, for example, when an MX lookup is done for a name that has no MX record, but does have other records. A lookup is *answered with success* if it terminates with a response that has a NOERROR code and one or more ANSWER records. All other lookups are are considered *unanswered*.

TABLE II

PERCENTAGE OF DNS LOOKUPS FOR THE POPULAR QUERY TYPES.

	mit-jan00	mit-dec00	kaist
A	60.4%	61.5%	61.0%
PTR	24.6%	27.2%	31.0%
MX	6.8%	5.2%	2.7%
ANY	6.4%	4.6%	4.1%

DNS queries can be made for a number of reasons: to resolve host names to IP addresses, to find reverse-mappings between IP addresses and host names, to find the hosts that handle mail for a domain, and more. There are twenty query types defined in the DNS specification [7]. Table II lists the four most frequently requested query types in each of our traces. About 60% of all lookups were for A records binding host names to addresses and between 24% and 31% were for the reverse PTR bindings from addresses to host names.

Although most answered A lookups are followed by a TCP connection to the host IP address specified in the returned resource record, there are some notable exceptions. These fall into two main categories: first, there are DNS A lookups which are not driven by TCP connections and second, there are TCP connections which are not preceded by DNS lookups. We excluded both of these classes of queries from our analysis and simulations.

Roughly 50% of the DNS lookups at MIT are not associated with any TCP connection. Approximately, 15% of these A lookups are for name servers, suggesting perhaps that there is a disparity between the TTL values people use for A records as opposed to NS records. Also, roughly 10% of all lookups are the result of an *incoming* TCP connection: some systems will do a PTR lookup for an IP address and then verify that the name is correct by doing an A lookup for the result of the PTR. Finally, a small percentage of these lookups are related to reverse black-lists such as `rbl.maps.vix.com`. This is a service designed to allow mail servers to refuse mail from known spammers. We suspect that the remaining 20% of these correspond to UDP flows but unfortunately, our data does not include any record of UDP flows.

Approximately 20% of TCP connections fall into the second class. Here, the dominant cause is the establishment of `ftp-data` connections: the LCS network hosts several popular FTP servers which results in a fairly large number of outgoing data connections. Other causes include hard-coded and hand-entered addresses from automated services run within LCS such as `validator.w3.org`. Finally, mail servers typically lookup MX instead of A records.

One of the major motivations for our work was the ratio of DNS lookups to TCP connections in the wide-area Internet, as described in Section I. The data in Table I (rows 9 and 15) allow us to estimate this ratio for the traced traffic, as the ratio of the number of TCP connections to the number of successfully answered lookups for A records associated with TCP connec-

TABLE I

BASIC TRACE STATISTICS. THE PERCENTAGES ARE WITH RESPECT TO TOTAL NUMBER OF LOOKUPS IN EACH TRACE.

		mit-jan00	mit-dec00	kaist-may01
1	Date and place	00/01/03-10, MIT	00/12/04-11, MIT	01/05/18-24, KAIST
2	Total lookups	2,530,430	4,160,954	4,339,473
3	Unanswered	595,290 (23.5%)	946,308 (22.7%)	873,514 (20.1%)
4	Answered with success	1,627,772 (64.3%)	2,648,025 (63.6%)	1,579,852 (36.4%)
5	Negative answer	281,855 (11.1%)	545,887 (13.1%)	1,834,942 (42.2%)
6	Zero answer	25,513 (1.0%)	20,734 (0.5%)	51,165 (1.2%)
7	Total iterative lookups	2,486,104	4,107,439	396,038
8	Answered	1,893,882	3,166,353	239,874
9	A lookups with follow-up TCP connections	496,802	941,081	817,937
10	Total query packets	6,039,582	10,617,796	5,326,527
11	Distinct second level domains	58,638	84,490	78,322
12	Distinct fully-qualified names	263,984	302,032	219,144
13	Distinct internal query sources	221	265	405
14	Distinct external name servers	48,537	61,776	8,114
15	TCP connections	3,619,173	4,623,761	6,337,269
16	#TCP : #valid A answers	7.28	4.91	7.75
17	Distinct TCP clients	978	1,216	8,605
18	Distinct TCP destinations	47,427	140,293	32,716

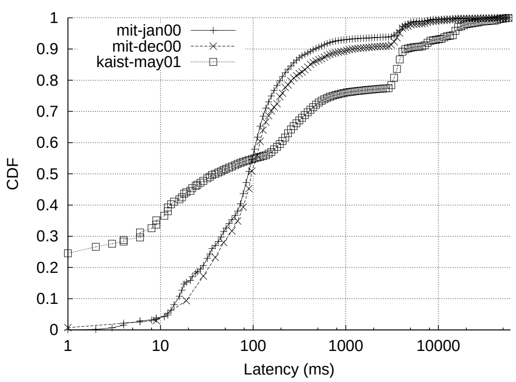


Fig. 3. Cumulative distribution of DNS lookup latency.

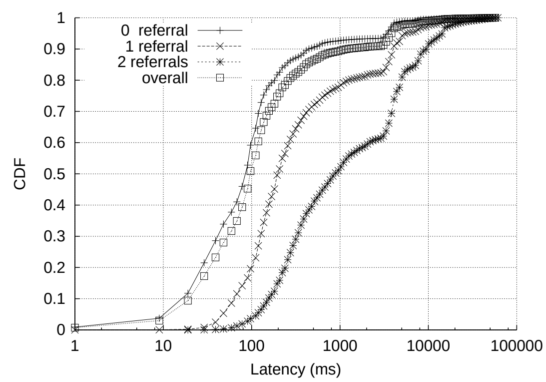


Fig. 4. Latency distribution vs. number of referrals for the mit-dec00 trace.

tions. These numbers are shown for each trace in row 16, suggesting a DNS cache hit ratio (for A-records) between 80% and 87% for all three traces. As explained in Section I, this hit rate is not particularly high, since it includes the caching done by Web browsers when they open multiple connections to the same server.

IV. CLIENT-PERCEIVED PERFORMANCE

This section analyzes several aspects of client-perceived DNS performance. We start by discussing the distribution of time it took clients to obtain answers. We then discuss the behavior of the DNS retransmission protocol and the situations in which client lookups receive no answer. We also study the frequency and causes of answers that are error indications and the prevalence of negative caching. Finally, we look at interactions between clients and root/gTLD servers.

A. Latency

Figure 3 shows the cumulative DNS lookup latency distribution for our data sets. The median is 85 ms for mit-jan00 and 97 ms for mit-dec00. Overall, long latency requests became more common—the latency of the 90th-percentile increased from about 447 ms in mit-jan00 to about 1176 ms in mit-dec00. In the kaist-may01 data, about 35% of lookups receive responses in less than 10 ms and the median is 42 ms. The KAIST trace has more low-latency lookups than the MIT traces because the requested resource record is sometimes cached at ns.kreonet.re.kr, which is close to the primary name server for the campus (see Figure 2(b)). However, the worst 50% of the KAIST distribution is significantly worse than that of MIT. Many of these data points correspond to lookups of names outside Korea, corresponding naturally to increased flight times for packets.

However, latency is also likely to be adversely affected by the number of referrals. Recall that a referral occurs when a server does not know the answer to a query, but does know (*i.e.,* thinks it knows) where the answer can be found. In that case, it sends a response containing one or more NS records, and the agent performing the lookup must send a query to one of the indicated servers. Table III shows the distribution of referrals per lookup. About 80% of lookups are resolved without any referral, which means they get an answer directly from the server first contacted, while only a tiny fraction (0.03%–0.04% for MIT) of lookups involve four or more referrals.

Figure 4 shows the latency distribution for different numbers of referrals for the mit-dec00 data set. For lookups with one

TABLE III

PERCENTAGE OF LOOKUPS INVOLVING VARIOUS NUMBERS OF REFERRALS. THE NUMBER OF LOOKUPS USED IN THIS ANALYSIS FOR EACH TRACE IS SHOWN IN ROW 8 OF TABLE I. THE AVERAGE NUMBER OF QUERIES TO OBTAIN AN ANSWER, NOT COUNTING RETRANSMISSIONS, WAS 1.27, 1.2, AND 1.2, RESPECTIVELY.

	mit-jan00	mit-dec00	kaist-may01
0	74.62%	81.17%	86.09%
1	24.07%	17.86%	10.43%
2	1.16%	0.87%	2.10%
3	0.11%	0.07%	0.38%
≥ 4	0.04%	0.03%	1.00%

referral, 60% of lookups are resolved in less than 100 ms and only 7.3% of lookups take more than 1 second. However, for lookups involving two or more referrals, more than 95% take more than 100 ms, and 50% take more than 1 second.

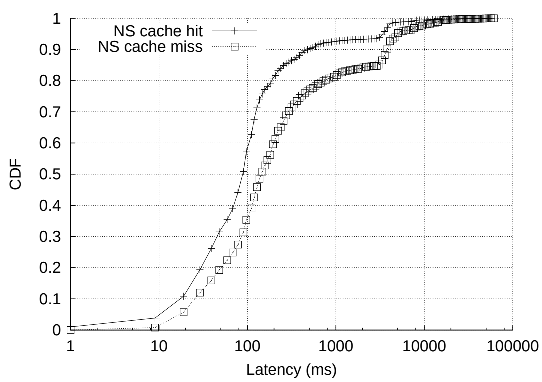


Fig. 5. Distribution of latencies for lookups that do and do not involve querying root servers.

To illustrate the latency benefits of cached NS records, we classify each traced lookup as either a *hit* or a *miss* based on the first server contacted. We assume a *miss* if the first query packet is sent to one of the root or gTLD servers and elicits a referral. Otherwise, we assume that there is a *hit* for an NS record in a local DNS cache. About 70% of lookups in the MIT traces are hits in this sense. Figure 5 shows the latency distribution for each case. It shows that caching NS records substantially reduces the DNS lookup latency even though it may involve some referrals to complete the lookup. Cached NS records are especially beneficial because they greatly reduce the load on the root servers.

B. Retransmissions

This section considers lookups that result in no answer, and lookups that require retransmissions in order to elicit an answer. This is interesting because the total number of query packets is much larger than the total number of lookups; the previous section (and Table III) show that the average number of query packets for a successfully answered query is 1.27 (mit-jan00), 1.2 (mit-jan00), and 1.2 (kaist-may01). However, the average number of DNS query packets in the wide-area per DNS lookup is substantially larger than this.

TABLE IV

UNANSWERED LOOKUPS CLASSIFIED BY TYPE.

	mit-jan00	mit-dec00
Zero referrals	139,405 (5.5%)	388,276 (9.3%)
Non-zero referrals	332,609 (13.1%)	429,345 (10.3%)
Loops	123,276 (4.9%)	128,687 (3.1%)

We can calculate this ratio, r, as follows. Let the total number of lookups in a trace be L, of which I are iteratively performed. This distinction is useful because our traces will not show retransmissions going out to the wide area for some of the $L - I$ recursive lookups. Let the number of query packets corresponding to retransmissions of recursive lookups be X. Let Q be the total number of query packets seen in the trace. Then, $(L - I) + rI = Q - X$ or $r = 1 + (Q - L - X)/I$. The values of L, I, and Q for the traces are shown in rows 2, 7, and 10 of Table I.

The value of r is relatively invariant across our traces: 2.40 for mit-jan00 ($X = 34,728$), 2.57 for mit-dec00 ($X = 18,478$), and 2.36 for kaist-may01 ($X = 448,396$). Notice that in each case r is substantially larger than the average number of query packets for a successfully answered lookup. This is because retransmissions account for a significant fraction of all DNS packets seen in the wide-area Internet.

A querying name server retransmits a query if it does not get a response from the destination name server within a timeout period. This mechanism provides some robustness to UDP packet loss or server failures. Furthermore, each retransmission is often targeted at a different name server, e.g., a secondary for the domain. Despite retransmissions and server redundancy, about 24% of lookups in the MIT traces and 20% of in the KAIST traces received neither a successful answer nor an error indication as shown in the third row of Table I.

We break the unanswered lookups into three categories, as shown in Table IV. Lookups that elicited *zero referrals* correspond to those that did not receive even one referral in response. Lookups that elicited one or more referrals but did not lead to an eventual answer are classified as *non-zero referrals*. Finally, lookups that led to loops between name servers where the querier is referred to a set of two or more name servers forming a querying loop because of misconfigured information are classified as *loops*. We distinguish the *zero referrals* and *non-zero referrals* categories because the former allows us to isolate and understand the performance of the DNS retransmission mechanism. We do not report this data for kaist-may01 since most lookups in that trace were recursively resolved by a forwarder outside the trace point.

The packet load caused by unanswered queries is substantial for two reasons: first, the rather persistent retransmission strategy adopted by many querying name servers, and second, referral loops between name servers.

On average, each lookup that elicited zero referrals generated about *five times* (in the mit-dec00 trace) as many wide-area query packets before the querying name server gave up, as shown in Figure 6. This figure also shows the number of

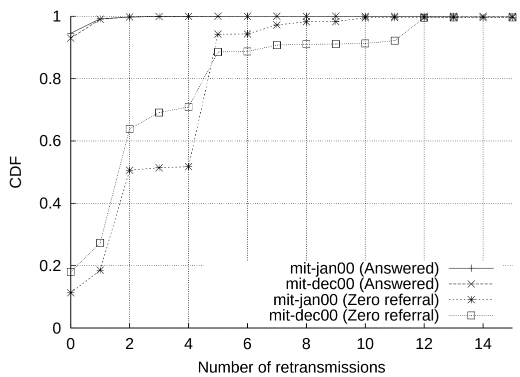


Fig. 6. Cumulative distribution of number of retransmissions for answered (topmost curves) and unanswered lookups.

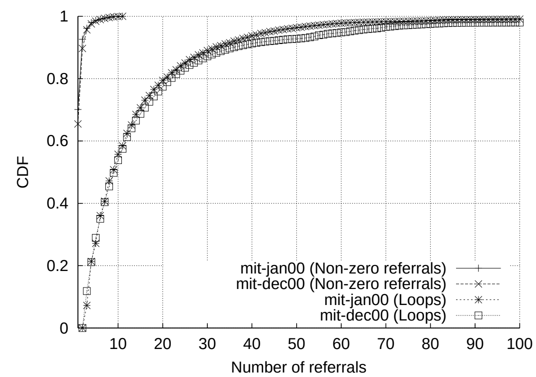


Fig. 7. *Loops* result in a large number of referrals, and hence packets: this graph compares cumulative distributions of number of referrals for the lookups that cause *loops* and that cause *non-zero referrals*.

retransmissions for queries that were eventually answered (the curves at the top of the graph)—over 99.9% of the answered lookups incurred at most two retransmissions, and over 90% involved no retransmissions. What is especially disturbing is that the fraction of such wasted query packets increased substantially between January and December 2000; the percentage of zero referral lookups increased from 5.5% to 9.3% and the percentage of these causing more than 5 retransmissions increased from 10% to 13%.

Given that the queries corresponding to these lookups do not elicit a response, and that most queries that do get a response get one within a small number (two or three) of retransmissions, we conclude that many DNS name servers are too persistent in their retry strategies. Our results show that it is better for them to give up sooner, after two or three retransmissions, and rely on client software to decide what to do. Interestingly, between 12% (`mit-jan00`) and 19% (`mit-dec00`) of unanswered lookups did not see any retransmissions. This suggests either that the resolver was not set to retransmit or was configured with a timeout longer than the 60 second window we used in our analysis.

Figure 7 shows the CDFs of the number of query packets generated for the *non-zero referrals* and *loops* categories of unanswered lookups. As expected, the non-zero referrals (which do not have loops) did not generate as many packets as the loops, which generated on average about ten query packets. Although unanswered lookups caused by loops correspond to only about 4.9% and 3.1% of all lookups, they cause a large number of query packets to be generated.

This analysis shows that a large fraction of the traced DNS packets are caused by lookups that end up receiving no answer. For example, `mit-dec00` included 3,214,646 lookups that received an answer; the previous section showed that the average such lookup sends 1.2 query packets. This accounts for 3,857,575 query packets. However, Table I shows that the trace contains 10,617,796 query packets. This means that over 63% of the traced DNS query packets were generated by lookups that obtained no answer! The corresponding number for `mit-jan00` is 59%. Obviously, some of these were required to overcome packet losses on Internet paths. Typical average loss rates are between 5% and 10% [5], [23]; the number of redundant DNS query packets observed in our traces is substantially higher than this.

C. Negative Responses

As shown in Table I, between 10% and 42% of lookups result in a negative answer. Most of these errors are either `NXDOMAIN` or `SERVFAIL`. `NXDOMAIN` signifies that the requested name does not exist. `SERVFAIL` usually indicates that a server is supposed to be authoritative for a domain, but does not have a valid copy of the database for the domain; it may also indicate that the server has run out of memory.

The largest cause of these error responses are inverse (`in-addr.arpa`) lookups for IP addresses with no inverse mappings. For `mit-jan00`, `in-addr.arpa` accounted for 33,272 out of 47,205 distinct invalid names, and 79,734 of the 194,963 total `NXDOMAIN` responses. Similarly, for `mit-dec00`, `in-addr.arpa` accounted for 67,568 out of 85,353 distinct invalid names, and 250,867 of the 464,761 total `NXDOMAIN` responses. Other significant causes for `NXDOMAIN` responses include particular invalid names such as `loopback`, and `NS` and `MX` records that point to names that do not exist. However, no single name or even type of name seems to dominate these `NXDOMAIN` lookups.

`SERVFAIL`s accounted for 84,906 of the answers in `mit-jan00` (out of 4,762 distinct names) and 61,498 of the answers in `mit-dec00` (out of 5,102 distinct names). 3,282 of the names and 24,170 of the lookups were inverse lookups in `mit-jan00`, while 3,651 of the names and 41,465 of the lookups were inverse lookups in `mit-dec00`. Most of the lookups were accounted for by a relatively small number of names, each looked up a large number of times; presumably the `NS` records for these names were misconfigured.

D. Negative Caching

In this section, we touch on the issue of negative caching, which was formalized in 1998 [22]. The large number of `NXDOMAIN` duplicate responses suggests that negative caching may not be working as well as it could be. In order to study this phenomenon better, we analyzed the error responses to understand their causes. A summary of this analysis is shown in Table V.

We do not know the actual cause for most of the negative responses. Many appear to be typos of correct names. The most clear cause of a `NXDOMAIN` response is a reverse lookup for an address that does not have a reverse mapping. There are several

Cause	mit-jan00	mit-dec00
Non-existent name	82,459 (42%)	150,066 (32%)
No reverse map for PTR	79,725 (41%)	249,236 (54%)
No RBL (or similar) entry	11,552 (6%)	36,955 (7%)
Loopback	7,368 (4%)	11,310 (2%)
Other one-word names	4,785 (3%)	9,718 (2%)
Invalid characters in query	1,549 (1%)	5,590 (1%)

TABLE V

BREAKDOWN OF NEGATIVE RESPONSES BY CAUSE AS PERCENTAGE OF ALL NEGATIVE RESPONSES.

TABLE VI

THE TOTAL NUMBER OF LOOKUPS THAT CONTACTED ROOT AND gTLD SERVERS, AND THE TOTAL NUMBER OF NEGATIVE ANSWERS RECEIVED. THE PERCENTAGES ARE OF THE TOTAL NUMBER OF LOOKUPS IN THE TRACE.

	mit-jan00	mit-dec00
Root Lookups	406,321 (16%)	270,413 (6.4%)
Root Negative Answers	59,862 (2.3%)	73,697 (1.7%)
gTLD Lookups	41,854 (1.6%)	353,295 (8.4%)
gTLD Negative Answers	2,676 (0.1%)	16,341 (0.3%)

other small but noticeable causes of negative answers. Reverse black-list lookups, described in Section III, make up the next largest class of queries causing negative responses. A number of misconfigured servers forward queries for the name `loopback`, instead of handling it locally. It might be a reasonable heuristic for servers never to forward this and other queries for names with a single component (i.e. unqualified names) when resolving queries for the Internet class.

However, we found that the distribution of names causing a negative response follows a heavy tailed distribution as well. Thus, the hit rate of negative caching is also limited.

E. Interactions with Root Servers

Table VI shows the percentage of lookups forwarded to root and gTLD servers and the percentage of lookups that resulted in a negative answer. We observe that 15% to 18% of lookups contacted root or gTLD servers and the percentage slightly decreased between January and December 2000. This was probably caused by an increase in the popularity of popular names (see Section V and Figure 9), which decreased DNS cache miss rates. The table also shows that load on root servers has been shifted to gTLD servers over time. By the end of 2000, the gTLD servers were serving more than half of all top-level domain queries.

Between 15% and 27% of the lookups sent to root name servers resulted in negative responses. Most of these appear to be mis-typed names (e.g. `prodiy.net`), bare host names (e.g. `loopback` or `loghost`), or other mistakes (e.g. `index.htm`). It is likely that many of these are automatically generated by incorrectly implemented or configured resolvers; for example, the most common error `loopback` is unlikely to be entered by a user. As suggested above, name servers could reduce root load by refusing to forward queries for unqualified host names. Note that the number of these lookups resulting in negative answers remains roughly the same during 2000, but because of the shift to gTLDs, the relative percentage of these lookups has increased.

V. EFFECTIVENESS OF CACHING

The previous sections analyzed the collected traces to characterize the actual client-perceived performance of DNS. This section explores DNS performance under a range of controlled conditions, using trace-driven simulations. The simulations focus on the following questions in the context of `A`-records:

1. How useful is it to share DNS caches among many client machines? The answer to this question depends on the extent to which different clients look up the same names.
2. What is the likely impact of choice of TTL on caching effectiveness? The answer to this question depends on locality of references in time.

We start by analyzing our traces to quantify two important statistics: (i) the distribution of name popularity, and (ii) the distribution of TTL values in the trace data. These determine observed cache hit rates.

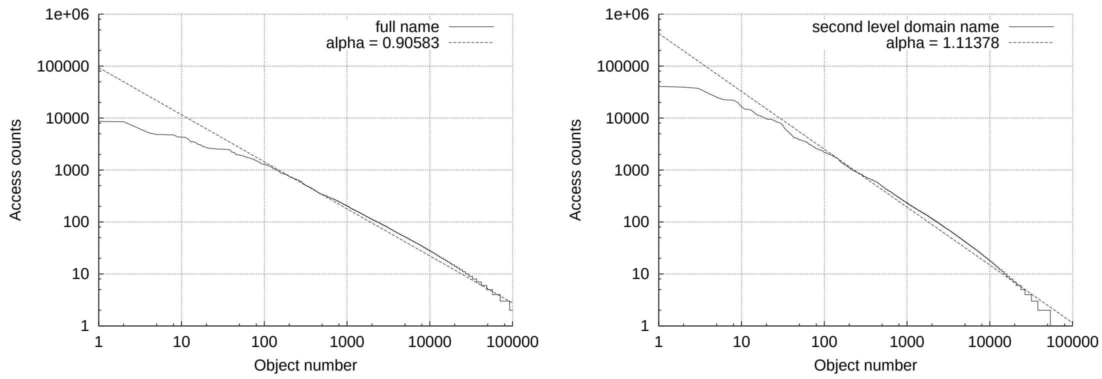
A. Name Popularity and TTL Distribution

To deduce how the popularity of names varies in our client traces, we plot access counts as a function of the popularity rank of a name, first considering only "fully qualified domain names." This graph, on a log-log scale, is shown in Figure 8(a). To understand the behavior of the tail of this distribution, and motivated by previous studies that showed that Web object popularity follows a Zipf-like distribution [10], we represent the access count as a function a/x^{α}, where α is termed the *popularity index*. If this is a valid form of the tail, then a straight line fit would closely fit the tail, with the negative of its slope giving us α. This straight line is also shown in the figure, with $\alpha \approx 0.91$.

We also consider whether this tail behavior changes when names are aggregated according to their domains. Figure 8(b) shows the corresponding graph for second-level domain names, obtained by taking up to two labels separated by dots of the name; for example, `foo.bar.mydomain.com` and `foo2.bar2.mydomain.com` would both be aggregated together into the second-level domain `mydomain.com`. The α for this is greater than one, indicating that the tail falls off a little faster than in the fully qualified case, although it is still a power-law. The slopes calculated using a least-square fit for each trace are shown in Table VII.[1]

Figure 9 illustrates the extent to which lookups are accounted for by popular names. The X-axis indicates a fraction of the most popular distinct names; the Y-axis indicates the cumulative fraction of answered lookups accounted for by the corresponding X most popular names. For example, the most popular 10% of names account for more than 68% of total answers for each of the three traces. However, it also has a long tail, and a large proportion of names that are accessed precisely once. For

[1] We calculated the least-square straight line fit for all points ignoring the first hundred most popular names to more accurately see the tail behavior.



(a) Full name

(b) Second level domain name

Fig. 8. Domain name popularity in the `mit-dec00` trace.

TABLE VII

POPULARITY INDEX α FOR THE TAIL OF THE DOMAIN NAME DISTRIBUTION.

	`mit-jan00`	`mit-dec00`	`kaist-may01`
Fully-qualified	0.88	0.91	0.94
Second-level	1.09	1.11	1.18

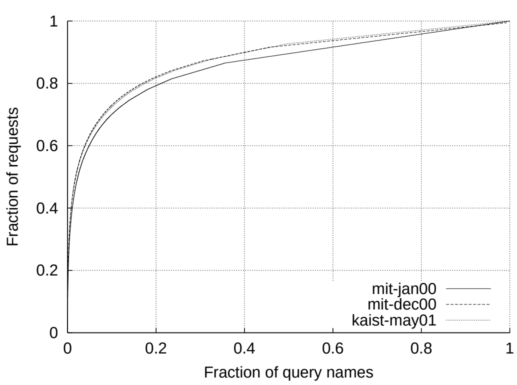


Fig. 9. Cumulative fraction of requests accounted for by DNS name, most popular first. The popular names appear to have become even more popular in December 2000 compared to January 2000, although they are not necessarily the same names.

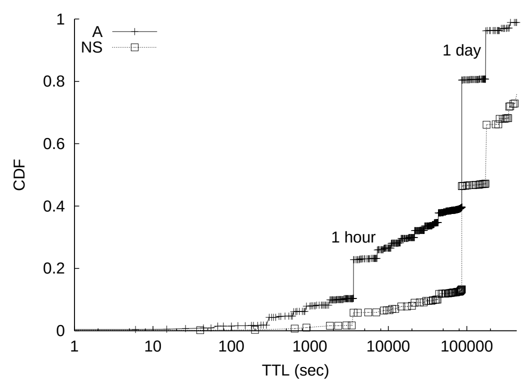


Fig. 10. TTL distribution in the `mit-dec00` trace.

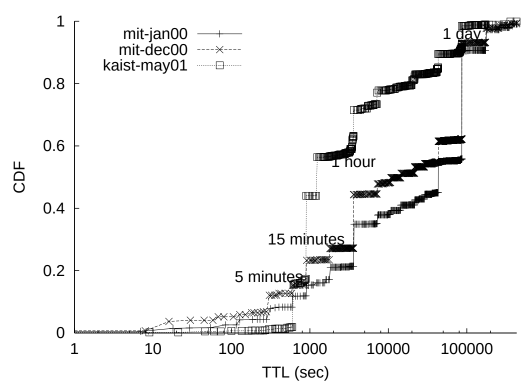


Fig. 11. TTL distribution weighted by access counts for `mit-jan00`, `mit-dec00` and `kaist-may01` traces.

instance, out of 302,032 distinct names involved in successful `A` lookups in `mit-dec00`, there were 138,405 unique names accessed only once, which suggests that a significant number of top-level domain queries will occur *regardless* of the caching scheme.

Figure 10 shows the cumulative distribution of TTL values for `A` and `NS` records. `NS` records tend to have much longer TTL values than `A` records. This helps explain why only about 20% of DNS responses (including both referrals and answers in Table I) come from a root or gTLD server. If NS records had lower TTL values, essentially all of the DNS lookup traffic observed in our trace would have gone to a root or gTLD server, which would have increased the load on them by a factor of about five. Good `NS`-record caching is therefore critical to DNS scalability.

Figure 10 shows how TTL values are distributed, but does not consider how frequently each name is accessed. If it turns out (as is plausible) that the more popular names have shorter TTL values, then the corresponding effect on caching would be even more pronounced. Figure 11 shows the TTL distribution of names, weighted by the fraction of TCP connections that were made to each name. From this, we draw two key conclusions. First, it is indeed the case that shorter-TTL names are more frequently accessed, which is consistent with the observation that DNS-based load-balancing (the typical reason for low TTL values) makes sense only for popular sites. Second, the fraction of accesses to relatively short (sub-15 minute) TTL values has doubled (from 12% to 25%) in 2000 from our site, probably because of the increased deployment of DNS-based server selection and content distribution techniques during 2000.

B. Trace-driven Simulation Algorithm

To determine the relative benefits of per-client and shared DNS caching of `A`-records, we conducted a trace driven simulation of cache behavior under different aggregation conditions. First, we pre-processed the DNS answers in the trace to form two databases. The "name database" maps every IP address appearing in an `A` answer to the domain name in the corresponding lookup. The "TTL database" maps each domain name to the highest TTL appearing in an `A` record for that name. (This should select authoritative responses, avoiding lower TTLs from cached responses.) After building these databases, the following steps were used for each simulation run.

1. Randomly divide the TCP clients appearing in the trace into groups of size s. Give each group its own simulated shared DNS cache, as if the group shared a single forwarding DNS server. The simulated cache is indexed by domain name, and contains the (remaining) TTL for that cached name.
2. For each new TCP connection in the trace, determine which client is involved by looking at the "inside" IP address; let that client's group be g. Use the outside IP address to index into the name database to find the domain name n that the client would have looked up before making the TCP connection.
3. If n exists in g's cache, and the cached TTL has not expired, record a "hit." Otherwise, record a "miss."
4. On a miss, make an entry in g's cache for n, and copy the TTL from the TTL database into the n's cache entry.

At the end of each simulation run, the hit rate is the number of hits divided by the total number of queries.

This simulation algorithm is driven by the IP addresses observed in the traced TCP connections, rather than domain names, because DNS queries that hit in local caches do not appear in the traces. This approach suffers from the weakness that multiple domain names may map to the same IP address, as sometimes occurs at Web hosting sites. This may cause the simulations to overestimate the DNS hit rate. The simulation also assumes that each client belongs to a single caching group, which may not be true if a client uses multiple local forwarding DNS servers. However, because DNS clients typically query servers in a strictly sequential order, this may be a reasonable assumption to make.

C. Effect of Sharing on Hit Rate

Figure 12 shows the hit rates obtained from the simulation, for a range of different caching group sizes. Each data point is the average of four independent simulation runs. With a group size of 1 client (no sharing), the average per-connection cache hit rate is 71% for `mit-dec00`. At the opposite extreme, if all 1,216 traced clients share a single cache, the average hit rate is

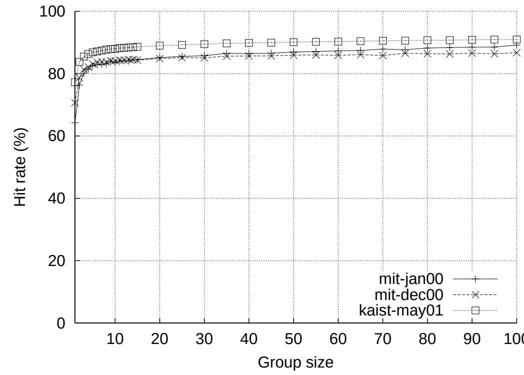


Fig. 12. Effect of the number of clients sharing a cache on cache hit rate.

89% for `mit-jan00`. However, most of the benefits of sharing are obtained with as few as 10 or 20 clients per cache.

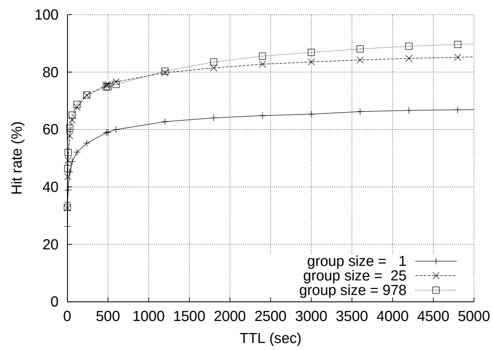
The fact that domain name popularity has a Zipf-like distribution explains these results. A small number of names are very popular, and even small degrees of cache sharing can take advantage of this. However, the remaining names are large in number but are each of interest to only a tiny fraction of clients. Thus very large numbers of clients are required before it is likely that two of them would wish to look up the same unpopular name within its TTL interval. Most cacheable references to these names are likely to be sequential references from the same client, which are easily captured with per-client caches or even the per-application caches often found in Web browsers.

D. Impact of TTL on Hit Rate

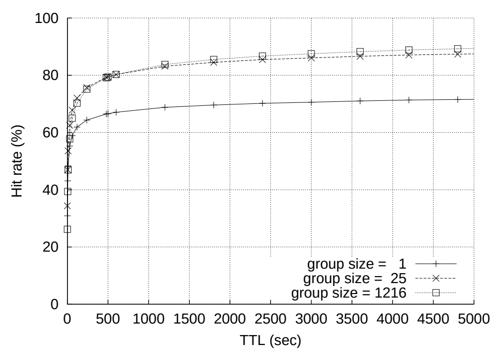
The TTL values in DNS records affect cache rates by limiting the opportunities for reusing cache entries. If a name's TTL is shorter than the typical inter-reference interval for that name, caching will not work for that name. Once a name's TTL is significantly longer than the inter-reference interval, multiple references are likely to hit in the cache before the TTL expires. The relevant interval depends on the name's popularity: popular names will likely be cached effectively even with short TTL values, while unpopular names may not benefit from caching even with very long TTL values. In turn, a name's popularity among a group of clients that share a cache is to some extent determined by the number of clients in the group.

To gauge the effect of TTL on DNS cache hit rate, we perform simulations using a small modification to the algorithm described in Section V-B. Instead of using TTL values taken from the actual DNS responses in the traces, these simulations set all TTL values to specific values. Figure 13 shows the results, with one graph for each of the three traces. Each graph shows the hit rate as a function of TTL. Since the results depend on the number of clients that share a cache, each graph includes separate curves for per-client caches, groups of 25 clients per cache, and one cache shared among all clients. We use a group of size 25 because Section V-C showed that for the actual TTL distribution observed in our traces, a group size of 25 achieves essentially the same hit-rate as the entire client population aggregated together.

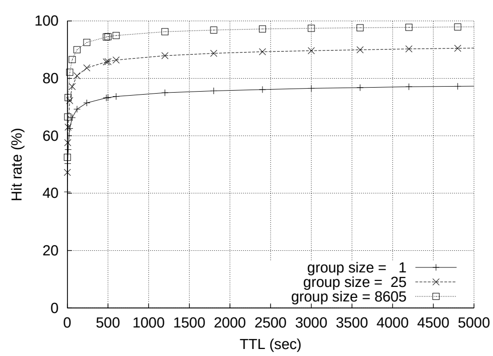
As expected, increasing TTL values yield increasing hit rates. However, the effect on the hit rate is noticeable only for TTL val-



(a) `mit-jan00`



(b) `mit-dec00`



(c) `kaist-may01`

Fig. 13. Impact of TTL on hit rate.

ues less than about 1000 seconds. Most of the benefit of caching is achieved with TTL values of only a small number of minutes. This is because most cache hits are produced by single clients looking up the same server multiple times in quick succession, a pattern probably produced by Web browsers loading multiple objects from the same page or users viewing multiple pages from the same Web site.

Figure 14 may help understand this better. It shows a time sequence of connection (DNS lookup) arrivals to a given destination, where X_i is the inter-arrival time between arrival i and $i-1$. Let $N(t)$ denote the number of queries for the given destination in the interval $(0, t]$, excluding the event at time 0. Define

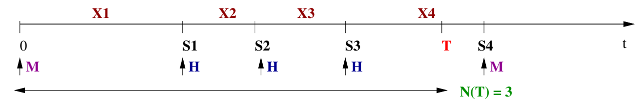


Fig. 14. Time line diagram of DNS queries to a given destination. It also depicts a partial realization of $N(T)$, X_i, and S_i

$S_n = X_1 + X_2 + \ldots + X_{N(t)}$, with $S_0 = 0$. The TTL is T.

At time $t = 0$, there is a DNS cache miss. Subsequently, three DNS lookups occur, at times S_1, S_2, S_3, before the TTL expires at time $t = T$. These three queries are cache hits. The subsequent, fourth query at time S_4 occurs after $t = T$ and is a cache miss. Thus in Figure 14, $N(T) = 3$ and the number of DNS cache hits per miss is 3. In general, $N(T)$ models the number of DNS cache hits per cache miss.

This example also suggests that the DNS hit rate for a given destination is a function of the inter-arrival statistics to that destination. We are currently developing an analytic model based on renewal processes that can predict hit rates for various arrival processes [24]. For the traces studied in this paper, we considered a few different analytic inter-arrival distributions and found that a Pareto with point mass at $t = 0$ was a good fit in all cases. The general form of this fit is $F(x) = w + (1-w)(1 - \frac{k}{(x+k)}^{\alpha})$, where x is the inter-arrival time and w is the weight of the point mass. Figure 15 shows the fit; in all cases $\alpha < 1$, indicating a heavy-tailed distribution with infinite mean.

The infinite mean result suggests that increasing the TTL of a DNS record would have limited additional benefit; essentially all hits in the cache in most cases arrive in close succession, probably typically generated when browsers initiate multiple connections to an address [24].

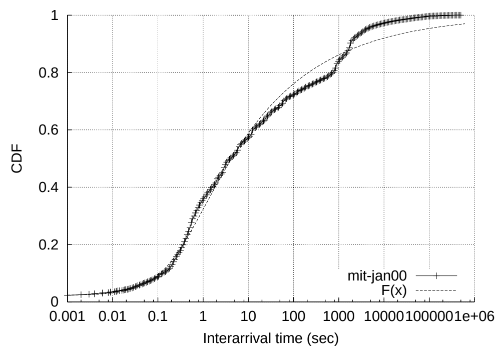
These results suggest that giving low TTL values to `A` records will not significantly harm hit rates. Thus, for example, the increasingly common practice of using low TTL values in DNS-based server selection probably does not affect hit rates much.

Thus caching of `A` records appears to have limited impact on reducing load on remote servers. In terms of overall system scalability, eliminating *all* `A`-record caching would increase wide-area DNS traffic by at most a factor of four, but almost none of that would involve a root or gTLD server. Eliminating all but per-client caching would little more than double DNS traffic. While neither of these cases would be desirable, this evidence favors recent shifts towards more dynamic (and less cacheable) uses of DNS, such as mobile host location tracking and sophisticated DNS-based server selection. The function of caching to reduce client-latency can almost be handled entirely by per-client or per-application caching.

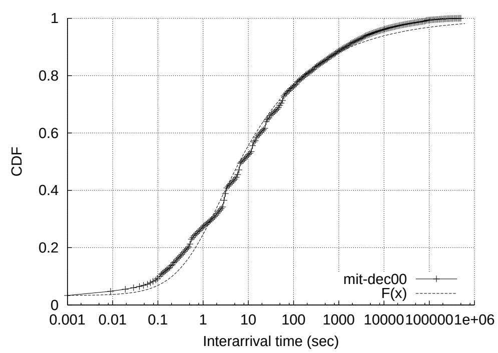
The discussion above applies to `A`-records, specifically for `A`-records for names that do not correspond to name servers for domains. It is *not* a good idea to make the TTL values low on `NS`-records, or for `A`-records for name servers. On the contrary, doing so would increase the load on the root and gTLD servers by about a factor of five and significantly harm DNS scalability.

VI. Conclusion

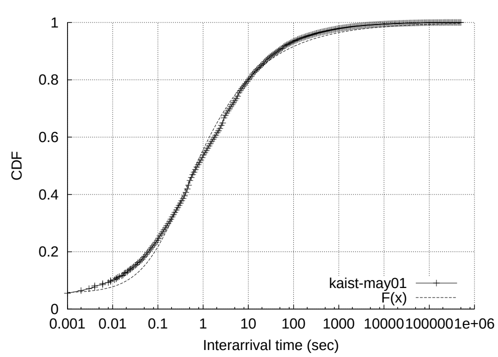
This paper has presented a detailed analysis of traces of DNS and associated TCP traffic collected on the Internet links of the



(a) `mit-jan00` ($\alpha = 0.24, k = 274, w = 0.023$)



(b) `mit-dec00` ($\alpha = 0.29, k = 735, w = 0.033$)



(c) `kaist-may01` ($\alpha = 0.37, k = 153, w = 0.056$)

Fig. 15. Cumulative distributions for TCP connection inter-arrivals.

MIT Laboratory for Computer Science and the Korea Advanced Institute of Science and Technology. We analyzed the client-perceived performance of DNS, including the latency to receive answers, the performance of the DNS protocol, the prevalence of failures and errors, and the interactions with root/gTLD servers. We conducted trace-driven simulations to study the effectiveness of DNS caching as a function of TTL and degree of cache sharing.

A significant fraction of lookups never receive an answer. Further, DNS server implementations continue to be overly persistent in the face of failures. While most successful answers are received in at most 2–3 retransmissions, failures today cause a much larger number of retransmissions and thus packets that traverse the wide-area. For instance, in the most recent MIT trace, 23% of lookups receive no answer; these lookups account for more than half of all traced DNS packets in the wide-area since they are retransmitted quite persistently. In addition, about 13% of all lookups result in a negative response. Many of these responses appear to be caused by missing inverse (IP-to-name) mappings or `NS` records that point to non-existent or inappropriate hosts. We also found that over a quarter of the queries sent to the root name servers result in such failures.

Our trace-driven simulations yield two findings. First, reducing the TTLs of address (`A`) records to as low as a few hundred seconds has little adverse effect on hit rates. Second, little benefit is obtained from sharing a forwarding DNS cache among more than 10 or 20 clients. This is consistent with the heavy-tailed nature of access to names. This suggests that the performance of DNS is not as dependent on aggressive caching as is commonly believed, and that the widespread use of dynamic, low-TTL `A`-record bindings should not degrade DNS performance. The reasons for the scalability of DNS are due less to the hierarchical design of its name space or good `A`-record caching (as seems to be widely believed); rather, the cacheability of `NS`-records efficiently partition the name space and avoid overloading any single name server in the Internet.

Acknowledgments

We are grateful to Garrett Wollman and Mary Ann Ladd, who run the network infrastructure at MIT LCS, for facilitating this collection process and for their prompt help at various times. We thank Dorothy Curtis for help with data collection. This study would not have been possible without the cooperation of the LCS and AI Lab communities at MIT. We are grateful to Heonkyu Park and Taejin Yoon who helped collect traces at KAIST. We thank David Andersen for useful discussions, Nick Feamster for several comments on drafts of this paper, Jennifer Rexford for useful suggestions, and Arthur Berger for his insights in explaining the relationship between cache hit rates and TTLs.

References

[1] P. Danzig, K. Obraczka, and A. Kumar, "An analysis of wide-area name server traffic: A study of the Internet domain name system," in *Proceedings of the ACM SIGCOMM*, Baltimore, MD, Aug. 1992, pp. 281–292.

[2] A. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *Proceedings of the 6th ACM MOBICOM*, Boston, MA, Aug. 2000, pp. 155–166.

[3] K. Frazer, "NSFNET: A partnership for high-speed networking," `http://www.merit.edu/merit/archive/nsfnet/final.report/`, 1995.

[4] K. Thompson, G. Miller, and R. Wilder, "Wide-area traffic pattterns and characteristics," *IEEE Network*, vol. 11, no. 6, pp. 10–23, November/December 1997.

[5] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. Katz, "TCP behavior of a busy Web server: Analysis and improvements," in *Proceedings of the IEEE INFOCOM*, San Francisco, CA, Mar. 1998, vol. 1, pp. 252–262.

[6] P. Mockapetris, *Domain Names—Concepts and Facilities*, Nov. 1987, RFC 1034.

[7] P. Mockapetris, *Domain names—Implementation and Specification*, Nov. 1987, RFC 1035.

[8] P. Mockapetris and K. Dunlap, "Development of the Domain Name System," in *Proceedings of the ACM SIGCOMM*, Stanford, CA, 1988, pp. 123–133.

[9] A. Kumar, J. Postel, C. Neuman, P. Danzig, and S. Miller, *Common DNS Implementation Errors and Suggested Fixes*, Oct. 1993, RFC 1536.

[10] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "On the implications of Zipf's law for web caching," Technical Report CS-TR-1998-1371, University of Wisconsin, Madison, Apr. 1998.

[11] M. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, Dec. 1997.

[12] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, "On the scale and performance of cooperative web proxy caching," in *Proceedings of the 17th ACM SOSP*, Kiawah Island, SC, Dec. 1999, pp. 16–31.

[13] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," in *Proceedings of the IEEE INFOCOM*, Anchorage, AK, April 2001.

[14] C. Wills and H. Shang, "The contribution of DNS lookup costs to web object retrieval," Tech. Rep. TR-00-12, Worcester Polytechnic Institute, July 2000, `http://www.cs.wpi.edu/~cew/papers/tr00-12.ps.gz`.

[15] E. Cohen and H. Kaplan, "Proactive caching of DNS records: Addressing a performance bottleneck," in *Proceedings of the Symp. on Applications and the Internet (SAINT)*, San Diego, CA, Jan. 2001.

[16] C. Huitema and S. Weerahandi, "Internet measurements: the rising tide and the DNS snag," in *Proceedings of the 13th ITC Specialist Seminar on Internet Traffic Measurement and Modelling*, Monterey, CA, Sept. 2000.

[17] N. Brownlee, kc claffy, and E. Nemeth, "DNS measurements at a root server," in *Proceedings of the IEEE GlobeCom*, San Antonio, TX, Nov. 2001.

[18] N. Brownlee, kc claffy, and E. Nemeth, "DNS Root/gTLD performance measurements," in *Proceedings of the 15th USENIX Systems Administration Conference*, Dec. 2001.

[19] R. Rivest, *The MD5 Message-Digest Algorithm*, April 1992, RFC 1321.

[20] G. Minshall, "Tcpdpriv," `http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html`, Aug. 1997.

[21] V. Jacobson, C. Leres, and S. McCanne, "tcpdump," `http://www.tcpdump.org/`.

[22] M. Andrews, *Negative Caching of DNS Queries (DNS NCACHE)*, Mar. 1998, RFC 2308.

[23] V. Paxson, "End-to-end Internet packet dynamics," in *Proceedings of the ACM SIGCOMM*, Cannes, France, Sept. 1997, pp. 139–152.

[24] J. Jung, A. W. Berger, and H. Balakrishnan, "Modeling TTL-based Internet caches," `http://nms.lcs.mit.edu/dns/`, July 2002.

Jaeyeon Jung received her BS and MS from the Korea Advanced Institute of Science and Technology. Since 2000, she has been a graduate student at the Laboratory of Computer Science. She has done extensive work in performance measurement for web caches at CAIDA and while working with the Nationwide Caching Project in Korea. Her e-mail address is: `jyjung@lcs.mit.edu`.

Emil Sit is a third year Ph.D. candidate in the Parallel and Distributed Operating Systems Group at the MIT Laboratory of Computer Science. He received his BS and M.Eng. in Computer Science at MIT, as well as a BS in Mathematics. His research interests include security and its applications to networks and networked applications. His e-mail address is: `sit@lcs.mit.edu`.

Hari Balakrishnan received a B.Tech. from the Indian Institute of Technology (Madras) in 1993 and a Ph.D. in Computer Science from the University of California at Berkeley in 1998. He is an Associate Professor in the EECS Department at MIT, where he holds the KDD Career Development Chair. He leads the Networks and Mobile Systems group at the Lab for Computer Science (http://nms.lcs.mit.edu/), exploring research issues in network protocols and architecture, mobile computing systems, and pervasive computing. Balakrishnan is the recipient of a Sloan Foundation Fellowship (2002), a National Science Foundation CAREER Award (2000), the ACM doctoral dissertation award (1998), and award papers at the ACM MOBICOM (1995 and 2000), IEEE HotOS (2001), and USENIX (1995) technical conferences. He was awarded the MIT School of Engineering Junior Bose Award for Excellence in Teaching in 2002. He is a member of ACM.

Robert Morris is an assistant professor in MIT's EECS department and a member of the Laboratory for Computer Science. He received a PhD from Harvard University for work on modeling and controlling networks with large numbers of competing connections. As a graduate student he helped design and build an ARPA-funded ATM switch with per-circuit hop-by-hop flow control. He led a mobile communication project which won a best student paper award from USENIX. He co-founded Viaweb, an e-commerce hosting service. His current interests include modular software-based routers, analysis of the aggregation behavior of Internet traffic, and scalable ad-hoc routing. His e-mail address is: `rtm@lcs.mit.edu`.