Piggybacking Related Domain Names to Improve DNS Performance[*]

Hao Shang and Craig E. Wills

Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609
Email: {hao,cew}@cs.wpi.edu

Abstract

In this paper, we present a novel approach to exploit the relationships among domain names to improve the cache hit rate for a local DNS server. Using these relationships, an authoritative DNS server (ADNS) can piggyback resolutions for future queries as part of the response message for an initial query. The approach improves the cache hit rate as well as reducing the total queries and responses. The approach is particularly attractive because it can be implemented with no changes to the existing DNS protocol.

Trace-based simulations show more than 50% of cache misses can be reduced in the best case while straightforward policies, using frequency and relevancy data for an ADNS, reduce cache misses by 25-40% and DNS traffic by 20-35%. These percentages improve if we focus the policies on resource records with smaller authoritative TTLs. We also show improved performance for hybrid approaches that combine the approach with renewal-based approaches.

In conjunction with this work we also did a study on current DNS performance for 20 locations in the United States. The outcome of this study is that the current average DNS latency is generally in the range of 200-300ms, but range from 500ms to multiple seconds if we look at the 95% response time. Approaches, such as what we propose, that reduce the amount of DNS traffic will improve the overall response time for applications.

Keywords: DNS, Network Applications and Services, Caching

[*]This work is partially supported by a WPI Research Development Grant.

1 Introduction

As part of work on studying the correlations among network data flows we find many flows have a temporal relationship with each other [16]. They happen concurrently or sequentially within a short period of time. Many application flows such as streaming, Web or Instant Messaging are preceded by a DNS (Domain Name System) flow. Applications like the Web also generate many concurrent HTTP flows between a client and server. These correlations exist due to inherent behavior of applications, content organization by a provider, or user access patterns.

Our overall work seeks to exploit relationships between flows for improved performance [16]. Some relationships have been exploited to improve application performance. The persistent connection mechanism specified in HTTP/1.1 [6] is motivated by the observation of many short concurrent or sequential connections existing between two end hosts. Motivated by the same problem, a previous study uses an approach that bundles multiple objects in one response [20]. Krishnamurthy et al. proposed a DNS-enabled Web approach that uses DNS messages to piggyback Web content [10], which seeks to exploit the relationship between DNS and HTTP flows. Another study [18] presents the methodology to use DNS queries to infer the relative popularity of any Internet server that can be identified by a name. It exploits the relationship that DNS lookups foreshadow the access of those Internet servers. Unlike these previous works the work presented in this paper is focused on exploiting the relationships that exist within a DNS flow.

As part of our work, we found that a local domain name server (LDNS) frequently sends more than one query to the same authoritative domain name server (ADNS) for different names within a short period of time. Using network flow data obtained from the WPI campus network in early December 2003, we observed that 42% of flows involving the DNS protocol for name server lookups result in multiple packet exchanges between client and server.

As a means to reduce multiple-packet DNS flows between a local DNS server and an authoritative DNS server, we hypothesize that in many cases the authoritative DNS server can predict subsequent requests by a local DNS server based on knowledge of site usage and history of its DNS accesses. For example, the content of Web pages at busy Web sites is often served by multiple servers at the site, each with distinct names. Similarly, streaming or Instant Messaging applications use their own servers and are often combined with access to Web servers.

If an ADNS can piggyback resolutions of those related names in the response to the first query, it will save the LDNS from sending further queries. We call this approach *Piggybacking Related Names* (PRN). It benefits end-users as they experience less latency for DNS lookups. It also benefits ADNSs as they receive fewer DNS requests. Assuming that the piggybacked resolutions do not require additional packets, then the approach reduces the number of packets needing to be routed through the Internet.

In order to better understand the PRN approach, we briefly overview the DNS mechanism. DNS is a distributed database providing mappings between addresses and names [12, 13]. The domain name space is a hierarchical structure, where each node has a label. Associated with each node is a set of resource records (RRs) that comprise the domain database. The database is divided into non-overlapping zones that are distributed among name servers with a group of root name servers at the top of the hierarchy. Below the root servers are generic Top Level Domain (gTLD) servers, which have NS RRs about delegated name servers for zones within a domain such as `.com`. These delegated name servers, called authoritative domain name servers (ADNSs), have complete information for the zone. There are many types of RRs. The most common type at an ADNS is an “A” RR that gives the mapping from a domain name to an IP address. For each RR, there is an associated time-to-live (TTL) parameter that indicates how long the RR can be cached.

The most important function of DNS is to return IP addresses for a given domain name. The process is typically initiated by a local application that calls an underlying resolver routine and passes the name as a parameter. The resolver sends a query to the local domain name server (LDNS), which in turn sends queries to responsible domain name servers if RRs for the name are not cached locally. As needed, the LDNS iteratively communicates with a root server then to a gTLD server then the authoritative domain name server for the name. Once the LDNS obtains the resolution, it returns the result to the caller application.

A domain name lookup is required for any application that identifies servers by names instead of IP addresses. The latency incurred by the lookup procedure can influence the application’s performance as a whole. In a previous study [19] we found that the median and mean lookup time for non-cached domain names is on the order of several hundred milliseconds. About 20% of the lookups took more than one second. Cohen et al. [4] indicates that DNS lookup time exceeds three seconds for over 10% of Web servers. This result is consistent with the measurement conducted by Chandranmenon and Varghese [2]. Jung et al. found that 10-20% of DNS lookups take more than one second based on traces collected in MIT and KAIST [9]. A more recent study shows the average latency for resolving non-cached domain names ranges from 0.95 seconds to 2.31 seconds for a variety of clients [11]. All of these measurements suggest that the DNS lookup time for non-cached domain names can influence the performance of interactive applications.

Caching is effective for reducing user perceived latency and is commonly adopted by DNS implementations. Previous studies [19, 9, 8, 3, 2] have shown cache hit rate varies from 50-90%. However with the increasing number of networked applications, there are many DNS requests generated for a single application and many of the cached copies of DNS mappings have a short time-to-live value in the cache. The result is that many requests for non-cached or stale DNS entries still exist. For example, in one day of WPI network flow data we observed over 40,000 DNS flows per hour. Thus further reduction of cache misses is still necessary for improving application performance.

The PRN mechanism reduces the cache miss rate by ADNSs predicting and piggybacking DNS resolutions. Previous work has also examined approaches to reduce the cache miss rate. We compare our PRN approach with these approaches [3, 8]. In addition, we examine how the PRN approach can be used in combination with these other approaches.

The remainder of the paper begins in Section 2 with a study that seeks to better understand current DNS latency performance. We then move on to a discussion of the PRN approach in Section 3 and how it compares with other works in Section 4. We investigate the potential usefulness of PRN in Section 5. Implementation and policy issues are discussed in Section 6. We evaluate the performance of different PRN policies in Section 7. Section 8 compares performance among PRN, other related approaches, and hybrid ones. We conclude with a summary of our findings and directions for future work in Section 9.

2 DNS Latency

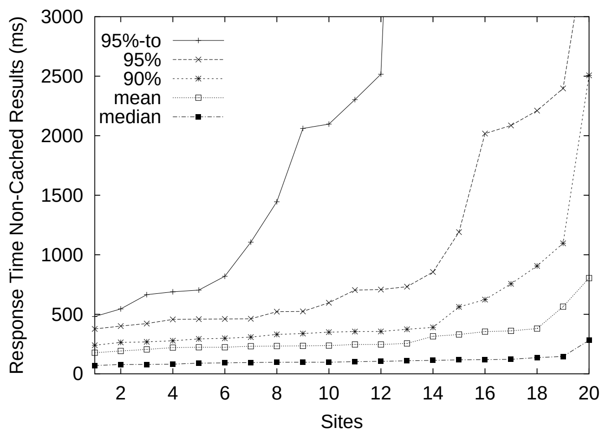
A central question about any proposal to improve DNS performance is to understand to what extent DNS performance is an issue. Recent work on DNS performance found that the average time to resolve non-cached domain names ranged from 0.95 seconds to 2.31 seconds for a variety of clients [11]. Previous studies had also found 10-20% of DNS resolution times greater than one second [19, 4, 2, 9]. These collective results indicate that DNS latency performance is an issue for applications.

In conjunction with our primary work on improving DNS performance, we performed a study in June 2004 and again in February 2005 to better understand current DNS performance for a subset of locations. We used 20 LDNSs that we identified as part of previous work [18]. These servers are all located in the United States and comprise four categories: commercial sites, educational sites, Internet Server Providers (ISPs) serving commercial companies, and ISPs serving home customers. Servers in the first three categories were found by using the `dig` tool to obtain the ADNS for an institution and then using directed DNS "A" record queries to determine if these authoritative name servers also played the role of LDNS for the institution. Servers in the last category were found by using published addresses for DNS servers of ISPs known to serve home customers.

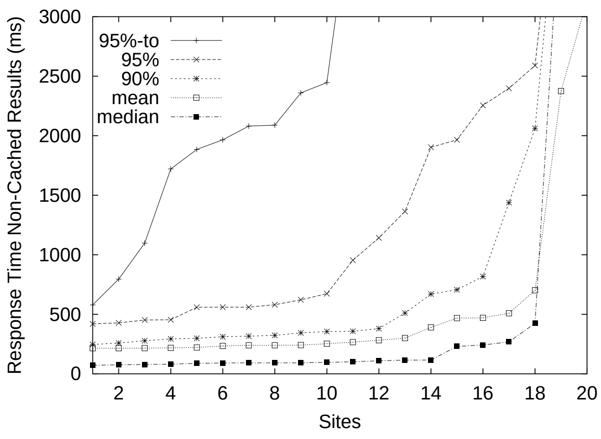
For testing DNS performance, we used a list of just over 5000 unique names randomly drawn from 164,000 domain names from the logs discussed in Section 5. This list initially contained only valid names that were successfully resolved by using a local client, although in the second study about 100 of the names were no longer valid. Each test involved using the `dig` DNS tool to direct a DNS query for each domain name to each of the 20 LDNSs. There are two possible situations. First, a LDNS could have the resolution for a domain name cached, in which case the total time is simply the round-trip time (RTT) between the `dig` client and the LDNS. Second, the LDNS needs to recursively perform a DNS lookup before returning the resolution. As part of the study we repeatedly measured the time for an entry known to be cached at a LDNS for establishing a baseline RTT measure between our `dig` client and each LDNS. We subtracted the mean of these RTT measures from all resolution times to determine the DNS resolution time by the LDNS for a given name. This is a similar approach as used in [7] of using DNS to measure the RTT between arbitrary points in the Internet, but we use valid domain names rather than randomly generated names.

To distinguish between cached and uncached entries during analysis of the results, we first examined the distribution of resolution times from our separate study of entries known to be cached. For most LDNSs, the difference between the minimum and 95% RTT value was within 10ms and we used the 95% RTT value to determine cached/uncached entries. For LDNSs with more variation in RTT values, we used the mean RTT value as the threshold. We confirmed the validity of these time-based thresholds by comparing results against one where we checked for the presence of the authoritative Time-to-Live (ATTL) value in the returned entry to determine non-cached entries. We had independently found the ATTL value for each entry. We were able to make this check for the latter timeframe in our study. We applied these time-based thresholds for cached/uncached entries for both timeframes in the study.

We used results for completed requests only with Mean, Median, 90% and 95% values for the 20 LDNSs in rank order shown in Figure 1. More than 5% of the queries did time out for six of the 20 LDNSs. We determined an effective timeout rate for each LDNS by subtracting the timeout rate to the LDNS itself from the overall timeout rate. The “95%-to” values in Figure 1 show the rank order 95% levels if the effective timeout rates are included for each LDNS.



a. June 2004



b. February 2005

Figure 1: DNS Response Time for Non-Cached Results

An observation about the results is that the average response time is generally between 200 and 300ms for both timeframes, which is consistent with [19], but less than the average times on the order of a second reported in [11]. This difference may occur because the tested LDNSs are all in the relatively well-connected United States as well as the vast majority of the server names, so better average results than [11] are not surprising. The results do show that over half of the tested LDNSs exhibit a 95th percentile response time of over a half-second and 25% of the LDNSs yield a 95% response time over one second. If we include the timed out responses for valid server names, then the majority of the LDNSs have a 95% response time of greater than one second with about half over three seconds.

The outcome of this study is that the current average DNS latency is generally in the range of 200-300ms, but poor DNS performance is still a problem. While we do not know the source of all latency problems, potential causes are packet RTT variance and loss combined with relatively long timeouts typically used by DNS clients. In addition, [15] found that request overload and competition from periodic tasks can cause response problems for DNS servers. Approaches that reduce the amount of DNS traffic will improve the overall response time for applications. A clear direction for future work is to consider extending this methodology to a wider range of LDNSs with more work on the methodology to better understand how to handle the effects of caching and timeouts.

3 The Piggybacking Related Names Approach

As a means to improve DNS performance, the *Piggybacking Related Names* (PRN) approach is motivated by the observation that many applications and related applications generate a sequence of DNS requests for "A" resource records to the same ADNS for domain names within the same DNS zone of authority. The approach exploits the observation that most DNS packets are smaller than the allowed size of the UDP packet in which they are carried and hence there is potential for ADNSs to include "Additional Records" in response to a client's request. In RFC1034, it says the "Additional Records" response field "carries RRs which may be helpful in using the RRs in the other sections." In our work we propose that this field can also be used to contain additional RRs that the ADNS expects the client to subsequently request based upon the current request. As long as including the additional records does not exceed the maximum allowed size of a DNS packet then these additional records are delivered to the client with no additional packets and minimal cost on the packet-switched Internet.

Figure 2 illustrates the approach with the query/response dialogue between a LDNS client and an ADNS for resolution of multiple server names. In this example, names from a1.b.c to a5.b.c belong to zone b.c. The first query (in this example is a1.b.c) triggers a response that includes resolutions for the additional names in the zone. Once it obtains the response, the LDNS caches all included entries. Queries for names a2.b.c, a3.b.c, a2.b.c and a4.b.c will be cache hits. We assume the interval between T6 and T1 is bigger than the authoritative TTL (ATTL) for a2.b.c. So at T6, the entry for a2.b.c is stale and a query for it causes a cache miss. A cache miss causes a new query to be sent to the ADNS by the LDNS.

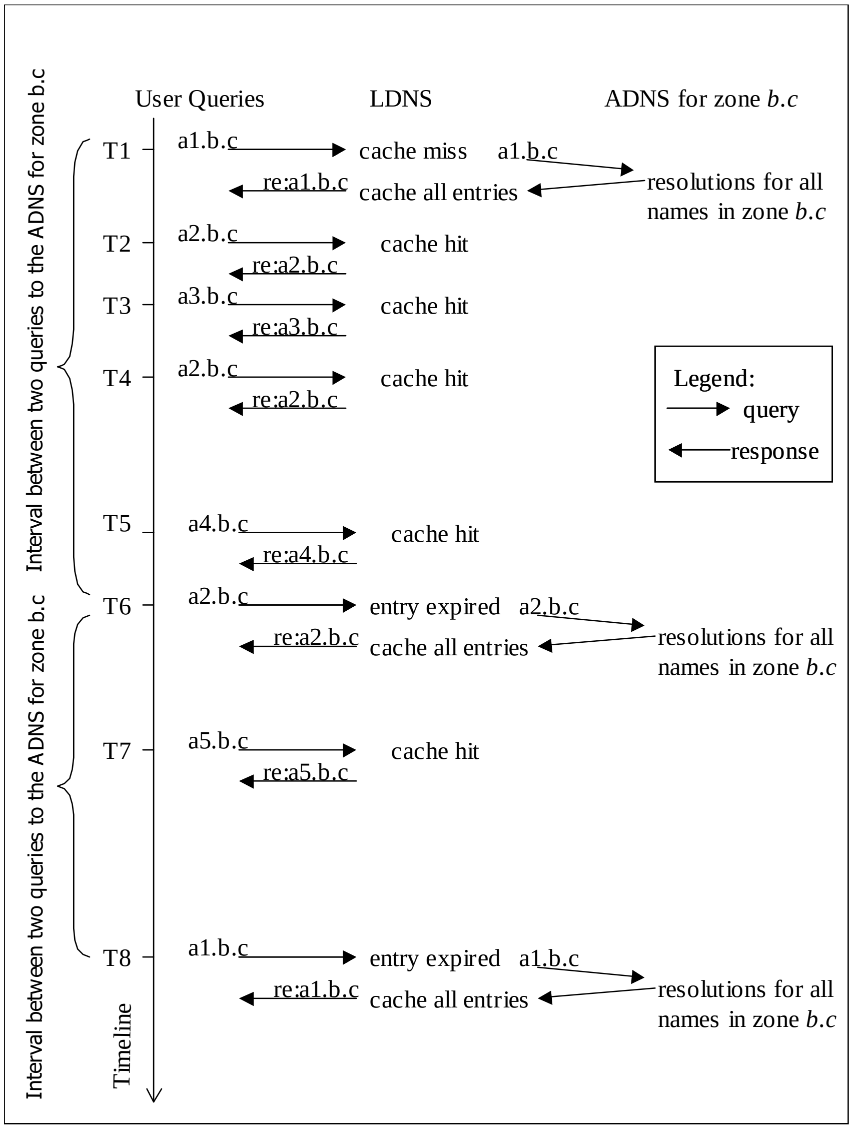


Figure 2: Illustration of the Piggybacking Mechanism

It is obviously not practical for an ADNS to piggyback resolutions for all the names in a zone. In reality, an ADNS only needs to piggyback resolutions it expects to be used in the interval before the next query is needed. In the example both “T1 to T6” and “T6 to T8” are such intervals. In Section 6 we study the expected number of these additional records as well as the amount of available room in a DNS response.

A clear advantage of the *Piggybacking Related Names* (PRN) approach is that it requires no changes to the existing DNS protocol while reducing the amount of DNS traffic for local and authoritative DNS servers. However the approach does require changes to the implementation of LDNSs and ADNSs. An ADNS must determine which names to be piggybacked and add them to the Additional Records section of a response message. This determination can be based on existing DNS queries as well as from knowledge of the site contents. A LDNS must extract the additional records and store them in its cache, which could be a problem in unnecessarily filling up the DNS cache, but in practice DNS cache records are small and cache space is not expected as a limitation. In our experiments, we assume all piggybacked records can be cached and will not be evicted before they expire. In the situation when cache space is limited, those piggybacked records can be tagged and be the first to be replaced if the cache is full.

One potential security issue with including resource records in the additional records field is a DNS-based attack called "cache poisoning" that is caused by allowing non-authoritative RRs to be cached by LDNSs [1]. Our approach does not lead to this problem because a ADNS only piggybacks RRs for which it is the authoritative server.

4 Related Approaches

Previous research has examined other approaches for reducing the cache miss rate at a LDNS. This section discusses three proposed approaches and compares them with the PRN approach.

One approach to improve DNS performance is for clients to pre-resolve server names [4]. This approach requires applications such as Web browsers to predict, based on Web content, which DNS lookups will be required and to issue those lookups before the content is retrieved. While this type of predictive policy is similar to the server-side predictions of our PRN approach it requires changes in applications and allows predictions to be made based only on client-available information.

A second approach is to use separate DNS queries to renew stale DNS cache entries [3]. This approach has the advantage that these queries are done outside of the critical path of an application and will improve the performance of an application. The problem with this approach is that it can generate many DNS queries for which the result is never used.

The third approach is to piggyback requests for stale entries onto a needed request to an ADNS [8]. This “renewal using piggybacking” (RUP) approach causes no additional DNS packets to be generated, but requires each LDNS to organize all resource records according to their zone. As in previous methods it also causes resolutions of names that may not be used again. This approach also requires that the DNS protocol support more than one request in a message.

To compare these approaches we examine the types of cache misses that they avoid. Using the terminology of [3], cache misses can be divided into two types: “first-seen” (FS) misses, indicating the first lookup of a DNS name; and “previously-seen” (PS) misses, indicating entries that have been previously seen, but expired. The two renewal approaches only reduce PS misses because they can only renew entries that have been seen before. The pre-resolving approach of [4] can reduce both FS and PS misses, but will only do so based on the immediate needs of the application. The PRN approach not only reduces FS misses based on server knowledge, but if those entries are already cached, it can be used to restore these entries to their full TTL duration, thus reducing PS misses.

5 Potential Impact

Before looking at the details of implementing the PRN approach, an important question is to examine its potential impact in terms of miss rates. It is known that LDNS caches satisfy over 50% of DNS requests received from local applications. With these hit rates, an argument can be made that DNS performance is not a problem. However, a number of factors justify the need to further reduce the number of non-cached lookups. First, despite the high hit rates, a substantial number of DNS queries must still be satisfied by contacting the appropriate ADNS and as previously mentioned 40% of the DNS traffic in WPI flow data indicate multiple DNS requests. Second, our own study in Section 2 along with recent studies have shown that latency is an issue for a portion of requests. Third, in the presence of a dropped packet the delay is much larger as LDNSs use a three or five second timeout. Fourth, more applications leads to more domain names at a site that must be looked up and many of these names carry shorter ATTLs to allow flexible load balancing.

We used three logs summarized in Table 1 to study the performance of DNS and examine the potential impact of improvements. The first log is of data from WPI’s primary DNS server, which serves as both a LDNS and ADNS for the campus. For our purposes the log was filtered to only consider queries from WPI clients that are handled by the server in its role as a LDNS. We augmented these data by fetching the ATTL and the ADNS(s) for each unique name in the log.

Table 1: Summary of Trace Logs Used

Name	Queries	Date	Dur.	From
WPI	1169569	Apr ’03	28 hrs	WPI DNS
RTP	1041275	Oct ’03	7 days	NLANR
SJ	457070	Oct ’03	7 days	NLANR

The other two logs are generated from two NLANR Web traces [14] as done in [3]. Each entry of the Web trace is a request to an object identified by a uniform resource locator (URL). We extract the host name part from a URL as a query in a DNS trace. Because many browsers themselves cache name-to-IP address mappings for a short time, we make the same assumption as in [3] that there is no DNS lookup incurred if the same name is requested again within a 60-second window.

We used these three logs along with the augmented data to determine the miss rate performance of DNS using a trace-driven simulation assuming the cache is empty at the beginning of the simulation. The simulation mimics the regular behavior of a DNS cache as well as an ideal behavior where whenever an ADNS receives a query, it returns resolutions for all the names in its zone. Subsequent queries that belong to this zone are satisfied locally as long as these entries are still fresh.

Results in Figure 3 show that 26% of requests in the WPI DNS log result in misses and this percentage can be potentially reduced to 10% for a relative improvement of over 60%. Similar results are shown in Figure 4 for the RTP log where the percentage of total misses is over 45% with a potential reduction to under 25% for a relative improvement of about 50%. Similar results were obtained for the SJ log and are not shown.

The collective results show significant reductions are possible in reducing both first-seen and previously-seen misses. Prediction of first-seen requests are not possible in renewal-based approaches while prediction of previously-seen requests extend the lifetime of the corresponding cached entries.

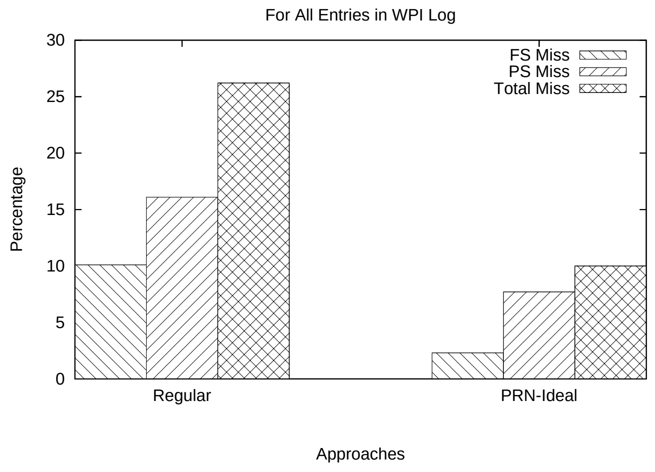


Figure 3: Potential Performance Improvement for Ideal PRN Policy with WPI Log

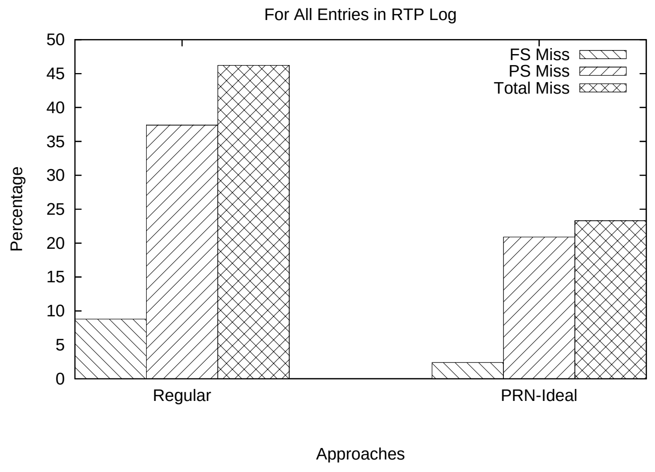


Figure 4: Potential Performance Improvement for Ideal PRN Policy with RTP Log

6 Implementation and Policy Issues

Having established the potential usefulness of the PRN approach, in this section we discuss specific implementation issues regarding the number of resource records that need to and can be piggybacked on a DNS response. We also describe specific policies for an ADNS to make decisions on what records to piggyback and what information the ADNS must maintain for these policies.

6.1 Piggybacked Responses

We used the data from the WPI DNS log to determine the number of responses that would ideally be piggybacked on a response. We used the request intervals in Figure 2 to define DNS "bundles" for a zone. A DNS bundle includes all unique server names for the zone that occur in a request interval. The size of this bundle determines the number of DNS responses that would be useful for the ADNS of the zone to return.

Using this definition, we found about 20,000 DNS bundles in the DNS log. Figure 5 shows the cumulative distribution function (CDF) for the number of names inside each bundle. As shown, about half of bundles have only one name—the response itself—while the other half have two or more names. The results show that only 5% of bundles have more than 15 entries and only 15% of the bundles have more than 5 entries. These results are encouraging for the PRN approach as they indicate it is useful for half of the bundles and the number of names that need to be piggybacked is not large.

6.2 DNS Response Message Capacity

DNS messages are limited to 512 bytes in size when sent over UDP [13], however DNS extension mechanisms [17] extend the limit to 1280 bytes. These mechanisms are supported in the latest 9.0 version of the widely-used BIND software [5]. We checked the sizes of DNS response packets for the 164K unique domain names collected from the three logs in Table 1. The CDF for the response message size for the unique names as well as for message sizes based on access patterns are shown in Figure 6. With respect to the trace-based statistic, most responses are 100-300 bytes, which affords 200-400 remaining bytes if we use the traditional 512B limit and many more bytes if we use the limit for extended DNS.

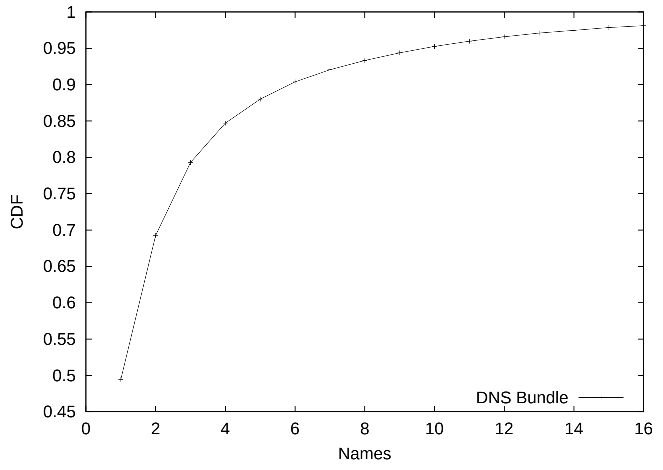


Figure 5: CDF of Size of DNS Bundles

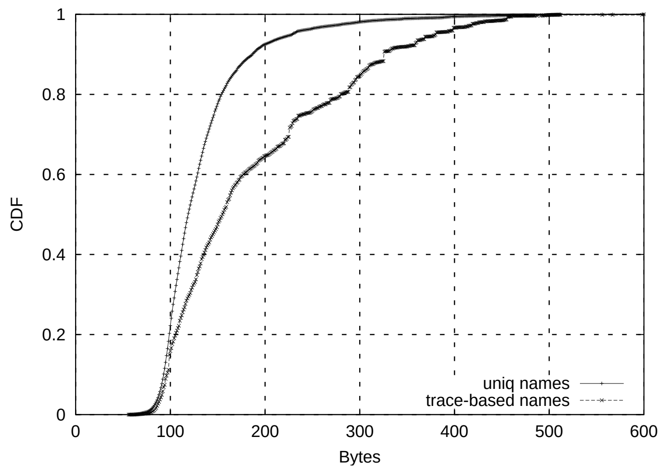


Figure 6: CDF of Sizes for DNS Response Packets on a Unique Name and a Trace-Based Set

Given the available room, we examined the number of additional type “A” records that can be piggybacked on a response. If we consider type “A” RRs in IPv4, the size of all its fields are fixed except the name. While a domain name can be long, it is not necessary to put the full name in that field. DNS provides a mechanism that enables domain names to share their common suffix. Using the same trace-based statistics, we observe that over 90% of names have a first distinguishing label (excluding “www”) less than 10 characters while the median and average are between 4 and 5. Putting those statistics together, the length for a piggybacked record is likely between 18 and 27 bytes (14 bytes for all fields with fixed length, 2 bytes for the pointer to the common suffix, 1 byte for the length count for the first label, and 1-10 bytes for the first label itself). With available space of 200-400 bytes, the total RRs that can be piggybacked are between 7 (200/27) and 22 (400/18). For extended DNS, the range is between 36 and 65.

We considered the situation when one domain name maps to multiple IP addresses, which requires multiple RRs for one name. We find that over 90% of domain names have less than five associated IP addresses while 72% have only one or two. Taking this factor into account, the total names that can be piggybacked are 1-22 for the traditional DNS length and 7-65 for the extended DNS length.

6.3 Piggyback Policies

The previous two sets of results indicate that a sizeable percentage of the records that could be piggybacked will fit in the additional space of a DNS response. In cases where there are more potential names than can be piggybacked, an ADNS needs to have a policy to decide which names to include. In addition to the *ideal* policy, which we described in Section 5, we define two practical policies: *Most Frequently Queried (MFQ) First* and *Most Related Query (MRQ) First*. The former policy gives preference to piggybacking names that are popular in the zone independent of the current request, while the latter policy gives preference to piggybacking names that are most related to the current query. These policies are described in more detail as follows.

MFQ(n): The ADNS selects up to n names in the order of their requested frequencies. For this policy, the ADNS needs to track query frequencies for each name in its zone and maintain them in a Frequency Ordered List (FOL).

MRQ(n, r): The ADNS selects names in the order of their relevancy to the current query. A Relevancy Ordered List (ROL) is maintained for each name. ROL(a) denotes the relevancy list for domain name "a". The MRQ policy chooses names from the ROL list with a relevancy greater than r for the current query up to the bound of n. If there is still remaining space then names from the FOL are added.

Figure 7: FOL and ROLs for zone "cnn.com."

Figure 7 shows an example of a FOL and ROLs for the zone "cnn.com" based on queries from the WPI DNS log. The first line contains the name of first ADNS (in sorted order) for the zone "cnn.com." The second line is the FOL for the zone and has all names in the order of their query frequency. All subsequent lines are the ROLs for each name. The first element on each line is the name and the remaining elements are its related names in the order of their relevancy values.

6.4 Maintenance of Information

Each ADNS must maintain data structures as shown in Figure 7 to support piggybacking of related names. In the combination of the logs in Table 1, we observed the maximal number of names in a zone is 1650 and the maximal number of ROLs is 627.

The FOL and ROLs can either be set up manually by administrators who know the internal connections among names, or by tracking query patterns. The example shown in Figure 7 is generated by analyzing the query patterns for the zone "cnn.com." The FOL is created by counting queries to each name. For generating ROLs, we group queries from the same client to the same ADNS that occur within a short period of time (5 minutes in our experiment). Whenever a name happens to be the first query in a group, the counter for its corresponding ROL is increased by 1. For all other names (after removing duplicates) in the group, each is counted once in the ROL for the first query. The relevancy value from name "a" to name "b" is calculated by dividing the counter of "b" in ROL("a") by the counter of ROL("a"). For instance, in Figure 7, line 3 is the ROL for query "www.cnn.com". The following number "723" is the count for the ROL and indicates there are 723 times "www.cnn.com" is the first query in a group. The number "0.78" following “i.cnn.net” is the relevancy value from “www.cnn.com” to “i.cnn.net”, which indicates out of 723, 78% of times “i.cnn.net” follows “www.cnn.com”. The relationship table is created based only on the WPI DNS trace. In general, it is expected that a server could create more accurate lists based on a larger number of client users. The internal relationships and access patterns between these domain names are not expected to change in the time scale of hours so these relevancy tables can be computed offline or when the ADNS server is not busy.

7 Evaluation

7.1 Methodology

We evaluate the PRN approach by trace-drive simulations of the ideal policy as described in Section 5 as well as the MRQ and MFQ policies described in Section 6.3. We use the relative decrease in the cache miss percentage as the metric to evaluate each policy. The regular policy is used as the baseline to compare effects of other policies with all results shown as the relative decrease in misses compared to the total number of first-seen (FS) and previously-seen (PS) misses for the regular DNS policy. Results for the ideal policy from Section 5 are shown for reference.

We studied the MFQ and MRQ policies with fixed upper bounds. We choose 5 and 15 as two upper bounds based on results from Section 6. In addition to these bounds, the MRQ policy is tested with relevancy values of 0.5 and 0 for a total of four MRQ policy combinations. Note that relevancy bound equal to 0 means the relevancy value should be bigger than 0, hence a qualified name must have some relevancy, even if weak.

7.2 Results

We used the first half of a log to generate relevancy tables and conducted the simulation on the second half of the log beginning with an empty DNS cache. The results are shown in Figure 8 and Figure 9 for the WPI and RTP logs respectively. Simulation on the SJ log produced similar results as the RTP log and they are not shown here. In the figures, each set of bars corresponds to a policy. The first bar in a set shows the relative (to the total misses for the regular DNS approach) decrease of first-seen misses, the second bar indicates the relative decrease of previously-seen misses, and the third bar is the relative decrease of the total misses.

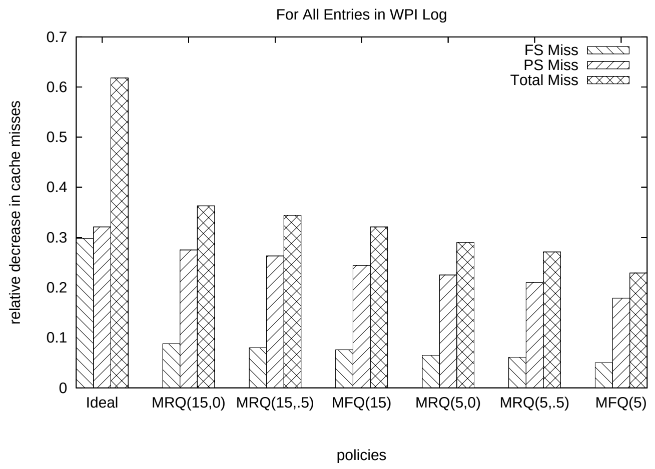


Figure 8: Relative Decrease in Cache Misses Over Different Policies on WPI Log

The results show that both the FS misses and PS misses are significantly reduced when any piggyback policy is in use. The MRQ and MFQ policies reduce the total misses in the range from 25% to close to 40%. The results are significant because they are obtained by also reducing the number of queries by the same amount.

In terms of the policies, MRQ policies consistently outperform MFQ policies when they have same bound constraints. Among MRQ policies, those having a smaller relevancy bound perform better. As MFQ(n) is similar to MRQ($n, 1$), we can summarize the performance relationship among the policies as $MFQ(n) < MRQ(n, .5) < MRQ(n, 0)$. These results indicate names with relevancy, even weak, should be given higher preference than simply piggybacking popular names. Increasing the bound helps reduce cache misses, but even the smaller bound results in a 25% reduction in cache misses.

The same methodology is used for the RTP log with the results shown in Figure 9. The relative decrease in cache misses for this log varies between 25% and 35% with similar variation between the policies as we found with the WPI log.

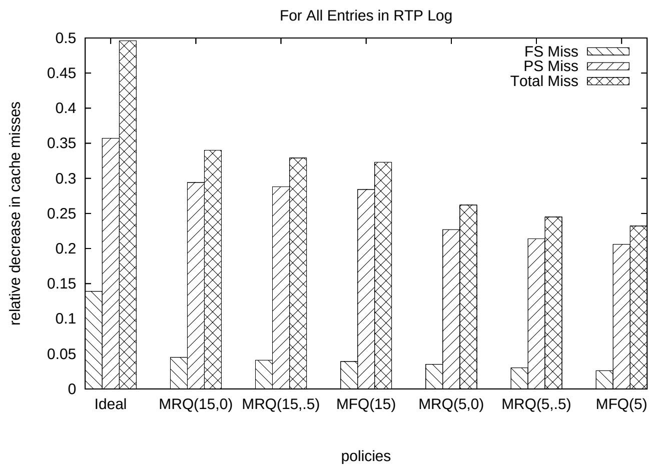


Figure 9: Relative Decrease in Cache Misses over Different Policies on RTP Log

7.3 Results for Short ATTLs

As a means to test the PRN approach for resource records with relatively short ATTLs, we filtered the log for queries to servers whose resolution have ATTLs of 30 minutes or less. This filter removed roughly half of the original DNS requests. These records must be requested more frequently by a LDNS and we hypothesize that the PRN approach would be relatively more effective at reducing the number of cache misses. Results for this analysis for the WPI log are shown in Figure 10 where the total miss rate for regular DNS is 32% as compared to 26% in Figure 3. The results in Figure 10 show relative decreases in cache misses from nearly 30% to over 40%.

We pushed this analysis further and filtered the log to include only entries with an ATTL of 5 minutes or less. This filter removed roughly 80% of the log entries with 46% of requests for these entries resulting in a cache miss. As shown in Figure 11, the PRN policies reduce the cache miss rate by over 40%. We found a similar tone of results when we did the same analysis for the RTP log. The results indicate this approach is more useful as the ATTLs grow shorter in duration.

7.4 Results for Total DNS Queries

Another direction we explored was the total number of DNS queries reduced by our approach. This avenue of exploration is relevant because while the PRN approach reduces DNS query traffic to ADNSs, it has little effect on traffic to root and gTLD servers. The previous results treat all

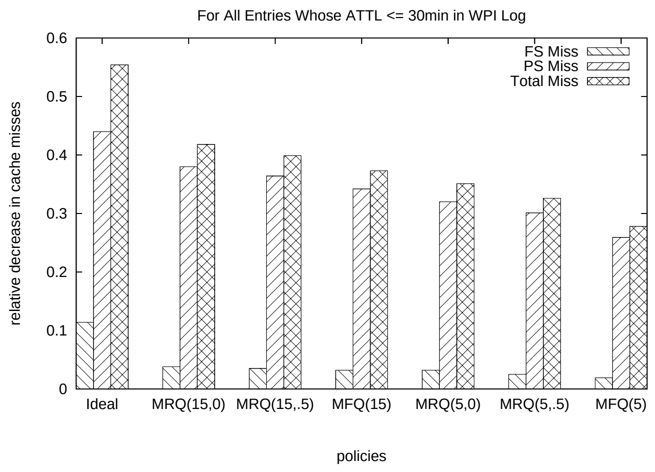


Figure 10: Relative Decrease in Cache Misses over Different Policies on WPI Log Entries with ATTL $\leq$ 30min.

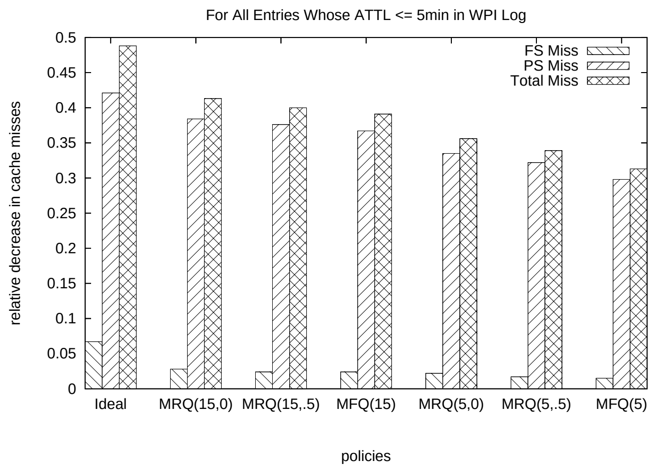


Figure 11: Relative Decrease in Cache Misses over Different Policies on WPI Log Entries with ATTL $\leq$ 5min.

cache misses as incurring the same cost when in fact the first time a LDNS encounters a domain name such as *x.foo.com* it must first find the ADNS for *foo.com*, which may involve contacting a root server as well as a *.com* gTLD server before the query is sent to the ADNS for *foo.com*. A subsequent access to *y.foo.com* would only require a query be sent to the ADNS for *foo.com* assuming the information about the ADNS is still fresh.

To model the situation where multiple DNS queries may be needed to resolve a domain name we obtained the ATTL for all ADNSs in the WPI log. We then reran our simulation on the WPI log to determine the relative decrease of not only requests to the ADNS, but the decrease for all DNS requests, which include those to obtain the authority for a zone. We did ignore queries to root name servers, which are relatively small in number. The results are shown in Figure 12.

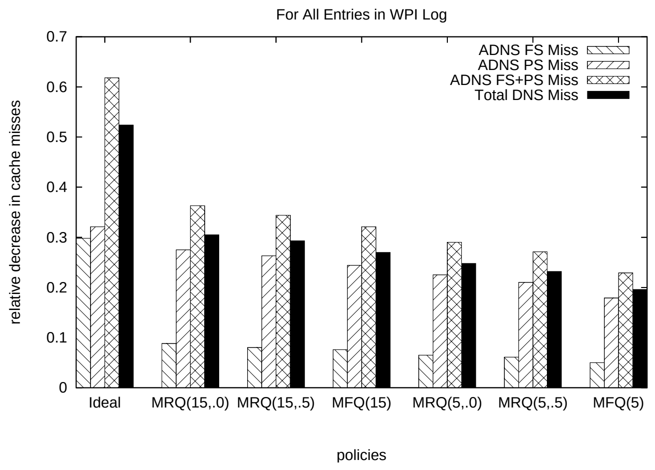


Figure 12: Relative Decrease in All DNS Cache Misses Over Different Policies on WPI Log

The results show that the additional DNS queries generated to obtain the authority for a zone lower the relative decrease in cache misses by less than 5%. This small reduction indicates that the number of queries generated to gTLD servers is much less than the number of queries sent directly to the ADNSs (about 20%) because the ATTLs for "A" records are generally smaller than those for records of ADNSs. Figure 12 shows that the PRN approach reduces the total number of DNS requests by 20-35%.

To better understand the costs of query to a gTLD server versus an ADNS we used the `dig` DNS client from WPI to measure the respective times. Using the names from the WPI log we found a mean response time of 47ms for queries to gTLD servers and a mean of 145ms for queries to the ADNSs. For queries from a home DSL client we found queries to gTLD servers take 63ms on average versus an average of 142ms for queries to the ADNSs. These results, along with the simulation results, indicate that the requests to ADNSs are the dominant DNS costs so that an approach such as PRN does yield significant cost savings.

8 Comparison and Combination with Other Approaches

Our final analysis was to compare the performance between our approach and others proposed to reduce cache misses. Because our approach is compatible with the others, a combination with these approaches is possible. We evaluate these hybrid approaches on all the three logs with results for the WPI log shown.

8.1 Performance Comparison Among Approaches

The proactive caching approach proposed in [3] has several policies. Among them, R-LFU is one of the better and more straightforward policies. We implemented R-LFU(r) for comparison purposes. The renewal using piggyback (RUP) approach proposed in [8] also has several policies. We implemented RUP-MFU, which performs best among all practical approaches. Among our PRN policies, MRQ performs better than MFQ. We choose MRQ(15,0) for the comparisons between approaches. We refer it as PRN-MRQ(15,0).

Figure 13 shows the relative decrease in cache miss percentages for the three approaches relative to normal DNS. The reduction in total cache misses is close for all the three approaches. When considering FS misses and PS misses separately, RUP-MFU and R-LFU policies behave almost the same, where FS misses are untouched and the reduction rates for PS misses are close. While PRN-MRQ does not reduce PS misses as much as the other two, its reduction on FS misses compensates for the difference. Despite the fact that the performance gains among the three approaches are similar, their costs are different. For the variation of the R-LFU policy we studied, it introduces 56% more queries and responses than normal DNS. The PRN-MRQ and RUP-MFU policies do not produce additional queries. Instead, by reducing the total misses, the total queries and responses are reduced as well.

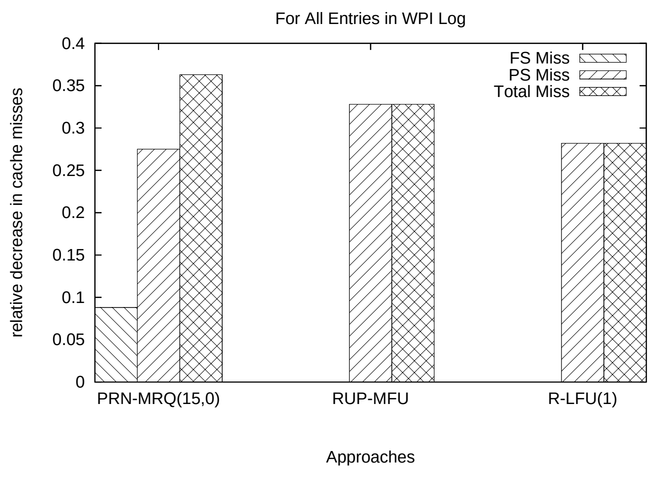


Figure 13: Performance Comparison among Approaches on WPI Log

8.2 Combination of PRN and RUP

In Figure 13 we observe that PRN-MRQ reduces more FS misses while RUP-MFU reduces more PS misses. This result encourages us to consider the possibility of combining the two approaches. Both approaches use piggybacking, but one makes the decision on the server side while the other does on the LDNS client side.

To combine these policies we define a new policy called "piggyback related names with client hint first" (PRN-CHF), where the client DNS server piggybacks its stale names in the query message and the ADNS uses these hints as well as its own relevancy table. The policy is described as:

> PRN-CHF(n, r): The total number of names that can be piggybacked is bounded by n, but instead of first looking at the corresponding ROL, the ADNS gives priority to the names piggybacked in the query message. If there is still extra space left, the ROL and FOL are checked in turn.

We show the performance of PRN-CHF and its two component approaches in Figure 14. As we expected, PRN-CHF has the same FS miss rate as PRN-MRQ and the same PS miss rate as RUP-MFU, so it performs the best among the three.

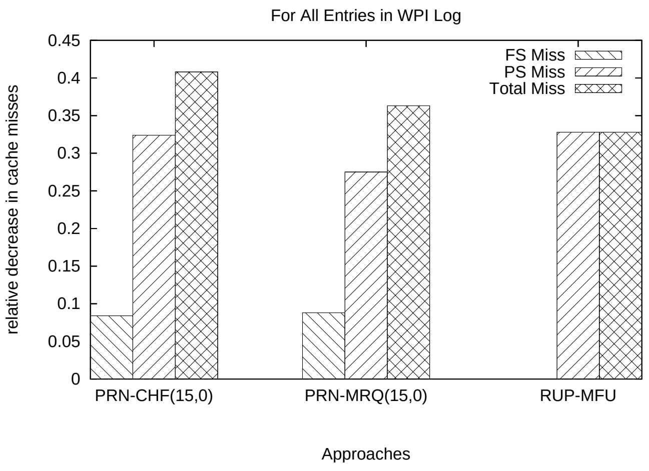


Figure 14: Performance Comparison among PRN-CHF and its Two Component Approaches on WPI log

8.3 Combination of PRN and R-LFU

We also studied the combination of these policies with active renewal. As the R-LFU approach is initiated by the LDNS cache and the PRN approach is initiated by an ADNS, the two approaches can complement each other.

We show performance of the various hybrid approaches along with R-LFU in Figure 15. In order to distinguish our original PRN approaches from their hybrid versions with R-LFU, we refer to those three hybrid approaches with a prefix "R-" to their original names. As with the R-LFU approach, each approach is tested with different aggressiveness in prefetching. As aggressiveness increases, the cache misses are further reduced, but more queries are generated.

The hybrid approaches show significant performance gains in Figure 15 compared with either R-LFU or their original PRN approaches. With about the same number of queries, R-PRN-MRQ performs much better than R-LFU. For instance, having 1.5 times queries as the regular approach, R-PRN-MRQ reduces 54% of total misses while R-LFU reduces 28%. R-PRN-CHF performs slightly better than R-PRN-MRQ as it is the combination of the PRN, RUP and R-LFU approaches. Renewal also benefits the original PRN approach at the expense of more queries.

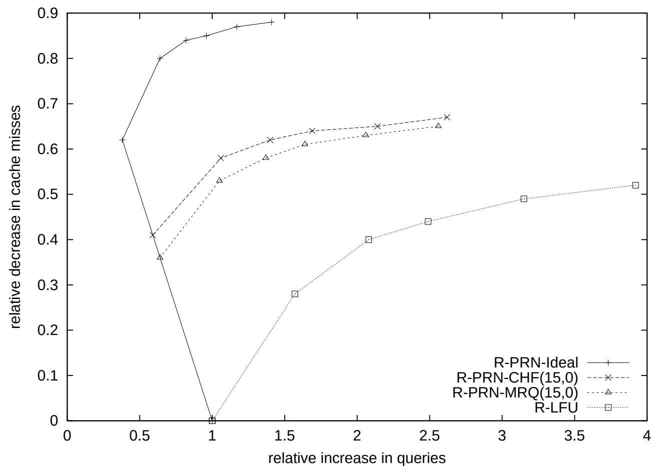


Figure 15: Performance for Hybrid Approaches on WPI Log

9 Summary and Future Work

This work is motivated by research on studying the relationships among network data flows. We found many cases where a local DNS server sends multiple DNS queries to the same authoritative DNS server within a short period of time. If the ADNS can predict these near-future queries once it receives the first one then it can send answers for all of them with the first response. We call this the piggybacking related names (PRN) approach. It helps reduce local cache misses and therefore reduces user-perceived DNS lookup latency. By piggybacking multiple answers in one response packet, the total queries and responses are also reduced, which alleviates the workload on both LDNSs and ADNSs.

Compared with other approaches that also address improving local cache hit rate, our approach is novel. We explictly use the relationships among queries and allow an ADNS to push resolutions for predicted names to the LDNS. The PRN approach reduces both first-seen misses as well as previously-seen misses while other approaches reduce just the latter. The cost of PRN is also low as it reduces the number of query and response packets while requiring no changes to the DNS protocol.

Trace-base simulations show more than 50% of cache misses can be reduced if prediction is perfect and response packet space is plentiful. Realistic policies, using frequency and relevancy data for an ADNS, reduce cache misses by 25-40% and all DNS traffic by 20-35%. These percentages improve if we focus the policies on resource records with smaller ATTLs. We also show improved performance by combining the PRN approach with renewal-based approaches to create hybrid approaches that perform significantly better than their component approaches.

In conjunction with this work we also did a study on current DNS performance for 20 locations in the United States. The outcome of this study is that the current average DNS latency is generally in the range of 200-300ms, but range from 500ms to multiple seconds if we look at the 95% response time. The reduced cache misses for the PRN approach will be reflected in improved response latency and timeout performance.

An obvious direction for future work on the PRN approach is to examine alternate policies such as ones to consider the ATTL for an entry. Policies should also be tested with additional logs. Another direction of future work is to deploy the PRN approach at an ADNS. We expect it should perform better than our simulation because an ADNS has more complete knowledge of its site contents and it can also aggregate reference patterns from a greater number of clients. An ADNS will not know if predicted names are actually used, but it can detect and modify its piggybacked list as it learns new access patterns that could have been predicted. A final direction to explore with the approach is the different types of sites and contents for which it is most useful. Sites with few servers and long authoritative TTLs likely do not need improvement in DNS performance while we expect more dynamic sites would be the first to benefit from this approach. Another clear direction for future work is to extend the DNS latency performance methodology to a wider range of LDNSs with more work on the methodology to better understand how to handle the effects of caching and timeouts.

10 Acknowledgments

We thank the National Laboratory for Applied Network Research for making proxy logs available. Data collection at the NLANR is supported by National Science Foundation grants NCR-9616602 and NCR-9521745.

References

[1] CERT/CC. Vulnerability Note VU109475.
`http://www.kb.cert.org/vuls/id/109475`.

[2] Girish P. Chandranmenon and George Varghese. Reducing web latency using reference point caching. In *Proceedings of IEEE Infocom 2001*, pages 1607–1616, 2001.

[3] Edith Cohen and Haim Kaplan. Proactive caching of DNS records: Addressing a performance bottleneck. In *Proceedings of the Symposium on Applications and the Internet*, pages 85–94, San Diego-Mission Valley, CA, USA, January 2001. IEEE-TCI.

[4] Edith Cohen and Haim Kaplan. Prefetching the means for document transfer: A new approach for reducing web latency. *Computer Networks*, 39(4):437–455, July 2002.

[5] Internet Software Consortium. BIND DNS Server.
`http://www.isc.org/products/BIND/`.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, June 1999. RFC 2616.

[7] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of the Second ACM SIGCOMM Internet Measurment Workshop*, Marseille, France, 2002.

[8] Baekcheol Jang and Kilnam Chon. DNS resolution with renewal using piggyback. In *Proceedings of the Twelfth International World Wide Web Conference (Poster)*, Budapest, Hungary, May 2003.

[9] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. Dns performance and the effectiveness of caching. *IEEE/ACM Transactions on Networking*, 10(5):589–603, October 2002.

[10] Balachander Krishnamurthy, Richard Liston, and Michael Rabinovich. DEW: DNS-enhanced web for faster content delivery. In *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungary, May 2003.

[11] Richard Liston, Sridhar Srinivasan, and Ellen Zegura. Diversity in DNS performance measures. In *Proceedings of the Second ACM SIGCOMM Internet Measurment Workshop*, pages 19–31, Marseille, France, 2002.

[12] P. Mockapetris. Domain Names - Concepts and Facilities, November 1987. RFC 1034.

[13] P. Mockapetris. Domain Names - Implementation and Specification, November 1987. RFC 1035.

[14] NLANR. network traffic packet header traces.
`http://pma.nlanr.net/Traces/`.

[15] KyoungSoo Park, Vivek S. Pai, Larry Peterson, and Zhe Wang. CoDNS: Improving DNS performance and reliability via cooperative lookups. In *Symposium on Operating Systems Design and Implementation*, San Francisco, CA, December 2004.

[16] Hao Shang and Craig E. Wills. Exploiting flow relationships to improve performance of networked applications. Technical Report WPI-CS-TR-04-13, Computer Science Department, Worcester Polytechnic Institute, May 2004.
`http://www.cs.wpi.edu/˜hao/tech-rep/use_relation.pdf`.

[17] P. Vixie. Extension Mechanisms for DNS (EDNS0), August 1999. RFC 2671.

[18] Craig E. Wills, Mikhail Mikhailov, and Hao Shang. Inferring relative popularity of Internet applications by actively querying DNS caches. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, Miami, Florida, November 2003.

[19] Craig E. Wills and Hao Shang. The contribution of DNS lookup costs to web object retrieval. Technical Report WPI-CS-TR-00-12, Worcester Polytechnic Institute, July 2000.
`http://www.cs.wpi.edu/˜cew/papers/tr00-12.ps.gz`.

[20] Craig E. Wills, Gregory Trott, and Mikhail Mikhailov. Using bundles for web content delivery. *Computer Networks*, 42(6):797–817, August 2003.