DNScup: Strong Cache Consistency Protocol for DNS

Xin Chen[1] Haining Wang[2] Shansi Ren[3] Xiaodong Zhang [3]
[1]Ask.com
Piscataway, NJ 08854, USA
xchen@ask.com
[2]College of William and Mary
Williamsburg, VA 23187, USA
hnw@cs.wm.edu
[3]Ohio State University
Columbus, OH 43210, USA
{sren,zhang}@cse.ohio-state.edu

Abstract

Effective caching in Domain Name System (DNS) is critical to its performance and scalability. Existing DNS only supports weak cache consistency by using the Time-To-Live (TTL) mechanism, which functions reasonably well in normal situations. However, maintaining strong cache consistency in DNS as an indispensable exceptional handling mechanism has become more and more demanding for three important objectives: (1) to quickly respond and handle exceptional incidents, such as sudden and dramatic Internet failures caused by natural and human disasters, (2) to adapt increasingly frequent changes of IP addresses due to the introduction of dynamic DNS techniques for various stationed and mobile devices on the Internet, and (3) to provide fine-grain controls for content delivery services to timely balance server load distributions. With agile adaptation to various exceptional Internet dynamics, strong DNS cache consistency improves the availability and reliability of Internet services. In this paper, we propose a proactive DNS cache update protocol, called DNScup, running as middleware in DNS nameservers, to provide strong cache consistency for DNS. The core of DNScup is a dynamic lease technique to keep track of the local DNS nameservers, whose clients need cache coherence to avoid losing service availability. Based on the DNS Dynamic Update protocol, we have built a DNScup prototype with minor modifications to the current DNS implementation. Our trace-driven simulation and system prototype demonstrate the effectiveness of DNScup and its easy and incremental deployment on the Internet.

Keywords — Domain Name System, Cache Consistency, Service Availability, Middleware, Lease.

1 Introduction

The Domain Name System (DNS) is a distributed database that provides a directory service to translate domain names to IP addresses [10], [11]. DNS consists of a hierarchy of nameservers, with thirteen root nameservers at the top. For such a hierarchical system, caching is critical to its performance and scalability, which significantly reduces the workloads of root and TLD nameservers, lookup latencies, and DNS traffic over the Internet. With the deployment of caches, cache consistency has become a serious concern. Strong cache consistency is defined as an insurance in which no stale copy of a modified original will be returned to clients, while under weak cache consistency, a stale copy may be returned to clients.

Existing DNS only supports weak cache consistency by using the Time-To-Live (TTL) mechanism. The majority of TTLs of DNS resource records range from one hour to one day [8]. While most of the domain-name-to-IP-address (DN2IP) mappings are infrequently changed, the TTL approach to coping with an expected mapping change is still cumbersome. Among numerous DNS related Requests For Comments (RFCs), only RFC 1034 [10] briefly describes how to handle an expected mapping change: "if a change can be anticipated, the TTL can be reduced prior to the change to minimize inconsistency during the change, and then increased back to its former value following the change"; but the RFC does not specify in what magnitude the TTL value should be reduced. The propagation of the mapping change may take much longer than expected. This pathology is induced by some local DNS nameservers that do not follow the TTL expiration rule and violate it by a large amount of time [12].

The inefficient and pathological DNS cache update due to weak consistency quite often leads to service disruption. More importantly, three recently-emerged reasons in practice cast a serious doubt on the efficacy of weak DNS cache consistency provided by the TTL mechanism.

1. There are many unpredictable mapping changes due to an emergency, such as terror attacks or natural disasters, in which the loss or failure of network resources (servers, links and routers) is inevitable [7] and we must immediately re-direct the affected Internet services to alternative or backup sites. Maintaining DNS cache consistency is critical under such an

exceptional circumstance, since people highly demand service availability at the crucial moment.

2. The dynamic DNS technique, which provides prompt IP mapping for a server at home or a mobile host using a temporary IP assigned by Dynamic Host Configuration Protocol (DHCP), makes the association between a domain name and its corresponding IP address much less stable.

3. The TTL-based DNS redirection service provided by Content Delivery Networks (CDNs) only supports a coarse-grained load-balance, and is unable to support quick reaction to network failures or flash crowds without sacrificing the scalability and performance of DNS [12].

Thus, *cache inconsistency in DNS poses a potential threat to the availability of Internet services*. Although the existing TTL-based solution can handle DNS dynamics in cases under normal conditions, it lacks an effective mechanism to fulfill the DNS strong cache consistency requirements in some exceptional yet critical cases. In this paper, we propose a proactive DNS cache update protocol, called *DNScup*, working as middleware to maintain strong cache consistency among DNS nameservers and improve the responsiveness of DNS-based service redirection. The core of DNScup is a dynamic lease technique to keep track of the local DNS nameservers whose clients are tightly coupled with an Internet server[1]. Upon a DN2IP mapping change of the corresponding Internet server, its authoritative DNS nameserver proactively notifies these local DNS nameservers still holding valid leases. While the notification messages are carried by UDP, the dynamic lease also minimizes storage usage and communication overhead, making DNScup a lightweight and scalable solution. In addition to maintaining cache coherence among DNS nameservers, DNScup can also be used to improve the responsiveness of DNS-based network controls as suggested in [12]. Also, we can apply the functionality of DNScup to maintain state consistency between a DNS nameserver of a parent zone[2] and the DNS nameservers of its child zones, preventing the lame delegation problem [14].

Based on the DNS dynamic update protocol [17], we have built a DNScup prototype with minor modifications to current DNS implementation [6, 11]. Our trace-driven simulation and a system prototype demonstrate that DNScup achieves strong cache consistency of DNS and significantly improves its performance and scalability. Note that DNScup is backward compatible with the TTL mechanism, and can be incrementally deployed over the Internet. Those local DNS nameservers without valid leases still rely on the TTL mechanism to maintain weak cache inconsistency.

The remainder of the paper is organized as follows. Section 2 surveys related work. Section 3 presents our DNS dynamics measurements. Section 4 gives a detailed description of the proposed DNScup mechanism. Section 5 evaluates the DNScup based on the trace-driven simulations and presents the prototype implementation of DNScup. Finally, we conclude the paper in Section 6.

2 Related Work

DNS performance at either root nameservers [2, 5] or local DNS nameservers and their caching effectiveness [8, 9, 13, 21] have been studied in the past decade. Danzig *et al.* [5] measured the DNS performance at one root nameserver and three domain nameservers. They identified a number of bugs in DNS implementation, and these bugs and misconfigurations produced the majority of DNS traffic. Brownlee *et al.* [2] gathered and analyzed DNS traffic at the F root nameserver. They found that several bugs identified by Danzig *et al.* still existed in their measurements, and the wide deployment of negative caching would reduce the impact caused by bugs and configuration errors. Pang *et al.* [13] characterized the load distribution, availability, and deployment patterns in local and authoritative DNS nameservers.

Jung *et al.* [8] measured the DNS performance at local DNS nameservers (MIT and KAIST) and evaluated the effectiveness of DNS caching. Based on trace-driven simulations, they found that lowering the TTLs of type `A` record to a few hundred seconds has little adverse effect on cache hit rates; and caching of `NS` records and protecting a single nameserver from overload are crucial to the scalability of DNS. Instead of collecting data at a few client locations, Liston *et al.* [9] compared the DNS measurements at many different sites, and investigated the degree to which they vary from site to site. Shaikh *et al.* [18] demonstrated that aggressively small TTLs (on the order of seconds) are detrimental to DNS performance, resulting in the increases of name resolution latency (by two magnitudes), nameserver workload and DNS traffic. Their work further confirmed that DNS caching plays an important role in determining client-perceived latency.

Based on both laboratory tests and live measurements, Wessels *et al.* [21] found that existing DNS cache implementations employ different approaches in query load balancing at the upper levels. They suggested longer TTLs for popular sites to reduce global DNS query load. In order to improve the lookup latency, Wills and Shang [22] explored the technique of actively querying DNS caches to infer the relative popularity of Internet applications. Using graphs, Cranor *et al.* [4] identified local and authoritative

[1] Either the clients frequently visit the Internet server or the services provided by the Internet server is critical to the clients.

[2] Zone is a delegated authority unit that is a manageable domain name space.

DNS nameservers from large DNS traces, which is useful for locating the related DNS caches.

However, none of the previous work focuses on DNS cache consistency. DNS cache inconsistency may induce a loss of service availability, which is much more serious than performance degradation. More recently, P2P-based DNS infrastructures have been proposed to replace the conventional DNS hierarchy for better fault-tolerance and load-balance [3, 16, 19]. Beehive[16] provides O(1) lookup latency by actively pushing replication to DNS caches, which may avoid cache inconsistency problem. Note that these proposed schemes are heavily dependent on the wide deployment of Distributed-Hash-Tables, and the proposed revolutionary changes to the Internet directory service will take a large amount of time and effort to become a reality. In contrast, DNScup is an enhancement to the current DNS implementations, which can fix the problem of cache inconsistency in a timely and cost-effective way.

While DNS caching does not support strong consistency, the DNS Dynamic Update mechanism [17, 20] maintains a strong consistency between the primary master DNS nameserver of a zone and its slave DNS nameservers within the same zone. In terms of DNS semantics, our proposed DNS cache update mechanism can be viewed as an external extension to the DNS Dynamic Update protocol, which makes the implementation and deployment of DNScup much easier.

3 Measurements of DNS Dynamics

The purpose of our DNS dynamics measurements is to answer the question of how often a DN2IP mapping changes. In general, a mapping change may cause two different effects. If the original DN2IP mapping is one-to-one, then the change may lead to the loss of Internet services. We classify this kind of changes as physical changes. However, if the original DN2IP mapping is one to many, the changes may be anticipated to balance the workload of a web site as CDN does. We classify these changes as logical changes.

The various mappings in the DNS name space are called resource records. Any type of resource records may change for different reasons. Among DNS resource records, the type `A` record is the most popular record being queried, accounting for about 60% DNS lookups on the Internet [8]. Moreover, the inconsistency of `A` records may directly lead to service unavailability. Therefore, in the rest of the paper, `A` record is the major target of our study.

3.1 Domain Name Collections

Since Web service is one of the most popular Internet services, our measurements are focused on the dynamics of the mappings between Web domain names and their corresponding IP addresses (`A` records). We collected the Web domain names from the recent IRCache [1] proxy traces. All Web domain names are classified into three categories: domains using CDN techniques, domains using dynamic DNS techniques, and the rest of collected domains. We refer them as CDN domains, Dyn domains, and regular domains. We distinguish CDN and Dyn domains from the rest based on the specific text strings from their providers (e.g., `Akamai` for CDN domains, `DynDns.com` for Dyn domains). The regular domain name distribution with respect to the number of requests in Top-Level Domains (TLDs) is plotted in Figure 1. Most regular domain names fall into the five major groups: `.com`, `.net`, `.org`, `.edu`, and country domains. We select 3,000 domain names from each of the five major groups.

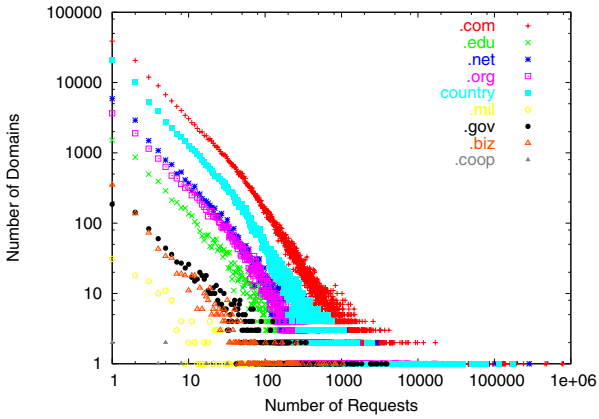


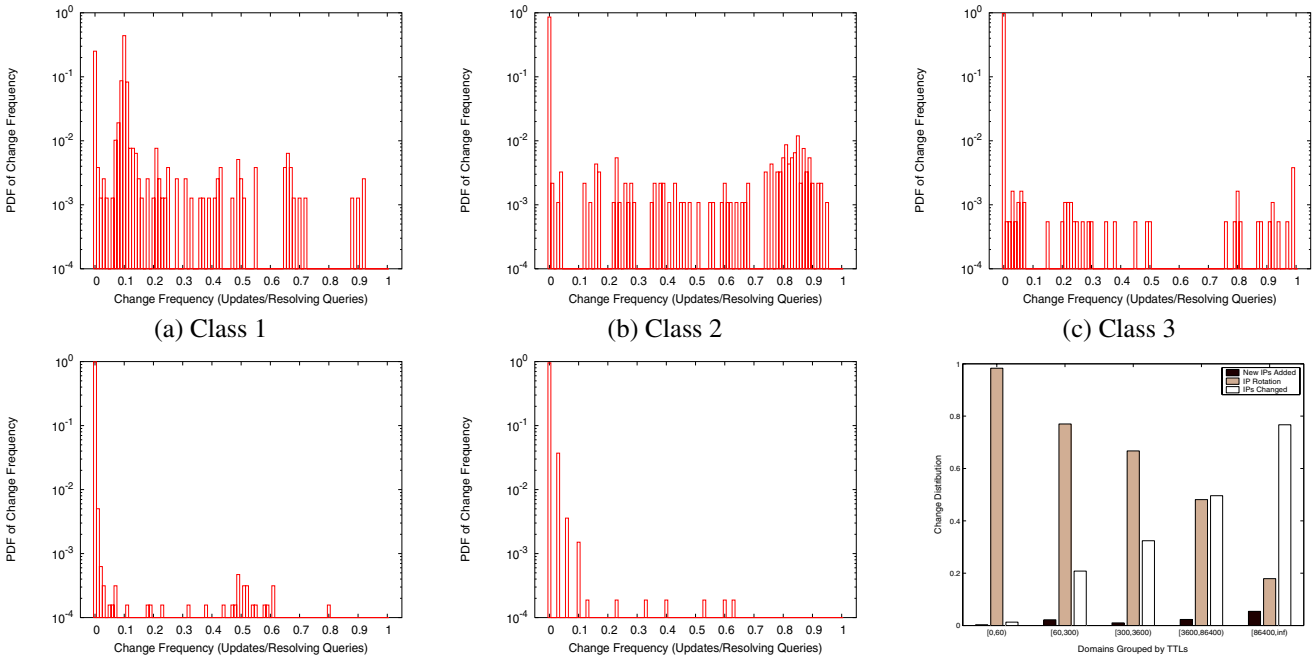
Figure 1. The regular domain name distribution with the number of requests in each groups.

Table 1. Measurement Parameters

Class	TTL (s)	Resolution (s)	Duration
1	[0,60)	20	1 day
2	[60,300)	60	3 days
3	[300,3600)	300	7 days
4	[3600,86400)	3600	7 days
5	[86400,∞)	86400	1 month

3.2 Measurements of Mapping Changes

Each domain name in our collection is periodically resolved to check if the mapping has been changed. The sampling resolution of detecting a DN2IP mapping change is highly dependent upon the values of TTLs. According to the sampling resolution, the Web domain names being



(a) Class 1 (b) Class 2 (c) Class 3

(d) Class 4 (e) Class 5 (f) Change Classifications

Figure 2. The DN2IP mapping change for each class with different TTLs.

probed in our measurements are divided into five classes as shown in Table 1. Since all CDN and Dyn domains' TTL values are bounded by 300 seconds, they belong to either classes 1 or 2. The regular domains of each TLD may fall in all five possible classes, because of the wide spectrum of their TTLs. The duration of a measurement experiment of different class varies from 1 day to 1 month

A DN2IP mapping change is detected when the responses of two consecutive DNS probes for the same domain name are different from each other. We define the relative change frequency of a domain name as the ratio between the number of mapping changes we detected and the total number of DNS probes we sent for that domain name. For ease of presentation, we employ relative change frequency as the metric to study the dynamics of DN2IP mapping changes, and simply call it change frequency in the rest of this paper.

The change frequencies for five different classes are shown in Figures 2 (a), (b), (c), (d) and (e), respectively. Based on the DNS probing results, we identify three causes that lead to the DN2IP mapping changes: (1) a domain name is relocated to a different IP address; (2) the available IP addresses for a domain name increase; and (3) the IP address of a domain name rotates around a set of IP addresses. The first cause results in physical changes, while the second and third causes result in logical changes. The distributions of the changes due to different causes are shown in Figure 2 (f) for all five classes.

Physical Changes: As shown in Figures 2 (c), (d) and (e), the domains in classes 3, 4, and 5 rarely change their DN2IP mappings, with about 95% domains in these classes remaining intact. Moreover, those domains that have changed their DN2IP mappings have very low change frequencies. For instance, in class 5, almost all changed domains have their change frequencies below 10%, which means a change happens every 10 days. On average, the change frequencies are about 3%, 0.1%, and 0.2% for the domains in classes 3, 4, and 5, respectively. This implies that the average life times of DN2IP mappings are 2.5 hours, 42 days, and 500 days, respectively. However, as shown in Figure 2 (f), nearly 40% mapping changes in class 3 and the majorities of mapping changes in classes 4 and 5 are physical changes. Any physical change could cause a cache inconsistency, leading to a loss of service availability. Considering the large number of domain names in classes 3, 4, and 5, the probability of a physical change happening per minute is close to one. Therefore, maintaining strong cache consistency is essential to avoid loss of connection.

Logical Changes: The DN2IP mappings in classes 1 and 2 are frequently changed. In class 1, more than 70% domains change their IP addresses during a one-day measurement. Most changed domains have their change frequencies around 10%. In class 2, only about 20% domains change their IP addresses during a three-day measurement, but most changed domains have relatively high frequencies (e.g., 80%). On average, the change frequencies of classes 1

and 2 are about 10% and 8%, much higher than the previous classes. The average life times of DN2IP mappings are 200 seconds and 750 seconds in classes 1 and 2, respectively. As shown in Figure 2 (f), such frequent changes are mainly due to IP address rotation (e.g., CDN's load balancing over multiple hosts), and most of the DN2IP mapping changes are logical ones.

We observe that CDN domains have very high change frequencies: 10% with TTLs between 0 and 60 seconds; and close to 70% with TTLs between 60 and 300 seconds. Two major CDN providers dominate the domains of the two ranges: Akamai with TTL 20 seconds; and Speedera with TTL 120 seconds. The domain names served by Akamai have change frequencies around 10%, while those served by Speedera have change frequencies close to 100%. In contrast to CDN domains, the Dyn domains have a low mapping change frequencies: 0.4% with TTL larger than or equal to 300 seconds; and close to zero with TTL less than 300 seconds. Compared with the actual change frequencies of CDN and Dyn domains, the corresponding TTL values are aggressively small, resulting in up to 10 and 25 times more DNS traffic than necessary. This redundant DNS traffic would be significantly reduced if server-initiated notification service were used.

4 DNS Cache Update Protocol (DNScup)

To maintain strong cache consistency, DNScup requires the authoritative DNS nameserver to keep track of the recent visitors (i.e., local DNS nameservers) that access and cache a DNS resource record. The *recent* in this context implies that the cached records should have not expired yet in these local DNS nameservers' caches. To make the presentation easier to understand, we refer to these local DNS nameservers, i.e., recent visitors, as DNS caches in the rest of the paper. We design a dynamic lease scheme to balance DNS nameserver storage requirements and DNS traffic between an authoritative DNS nameserver and its DNS caches.

Before detailing the design of dynamic lease, we sketch the cache update process as follows. Once the authoritative DNS nameserver has updated a DNS resource record either manually or via an internal dynamic update message, it retrieves the track file and gets all local DNS nameservers that have queried this record whose leases have not expired yet (i.e., DNS caches). The authoritative DNS nameserver then sends cache update messages to these DNS caches through UDP. The notified DNS caches will update their cached DNS resource records and acknowledge the authoritative DNS nameserver. The cache update process is shown as steps 3 and 4 in Figure 3, in which steps 1 and 2 are the process of granting a lease to a DNS cache.

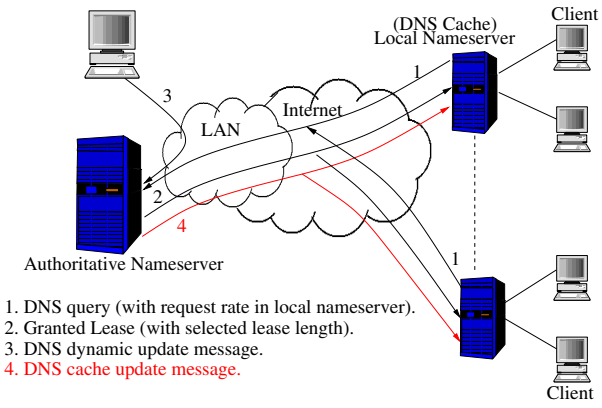


Figure 3. DNScup update process.

4.1 Lease Length Effectiveness

A critical question in applying a lease mechanism is how to choose the appropriate length of a lease to balance the storage usage and communication overhead. Lease storage usage on the authoritative DNS nameserver is represented by the probability of the nameserver holding a lease for each DNS cache. Its upper bound is 1, indicating that the nameserver always keeps a lease for a DNS cache. The communication overhead is represented by the query rate between the nameserver and its DNS caches. Since our practical algorithms always set the maximal lease length much smaller than the lifetime of a resource record, we only consider the lease renewal requests and ignore the update messages.

We assume that the query arrival rate from DNS caches for a DNS resource record follows a Poisson distribution with average arrival rate of λ. The rationale behind this assumption is two-fold: (1) a DNS resolution precedes the beginning of a session communication; and (2) Floyd and Paxson [15] have shown that the session-level (like FTP and Telnet) arrival rate still follows a Poisson distribution, although the packet arrival rate is non-Poisson. The time interval between two contiguous leases is equal to the average interval of two contiguous queries, $\frac{1}{\lambda}$. Suppose that the authoritative DNS nameserver grants a fixed-length lease, t, at the arrival of a query. The expected probability for the nameserver to maintain the lease, P, is thus

$$P = t/(t + \frac{1}{\lambda}). \tag{4.1}$$

The lease renewal message rate is defined as lease renewal frequency. Since a lease is renewed at the interval of $t + \frac{1}{\lambda}$, the lease renewal message rate M is

$$M = \frac{1}{t + \frac{1}{\lambda}}. \tag{4.2}$$

Suppose the lease length is increased from t_1 to t_2. Given the query rate λ, the increase of lease probability on

the nameserver is:

$$\triangle P = t_2/(t_2 + \frac{1}{\lambda}) - t_1/(t_1 + \frac{1}{\lambda}) = \frac{\lambda t_2 - \lambda t_1}{(\lambda t_1 + 1)(\lambda t_2 + 1)}$$

The reduction of message rate is:

$$\triangle M = 1/(t_1 + \frac{1}{\lambda}) - 1/(t_2 + \frac{1}{\lambda}) = \lambda * \frac{\lambda t_2 - \lambda t_1}{(\lambda t_1 + 1)(\lambda t_2 + 1)}$$

Thus, for a given resource record with a query rate λ, the ratio between the reduction of message rate and the increase of lease probability is a constant, which is equal to λ.

4.2 Dynamic Lease Algorithms

Assuming the overhead allowance (storage or communication) is pre-defined, we propose two dynamic lease algorithms: one minimizes the communication overhead given a constraint on storage budget; and the other minimizes the storage overhead, given a constraint on communication traffic.

4.2.1 Storage-constrained Dynamic Lease

We define the storage overhead allowance as the maximal number of leases P_{max} that a nameserver can manage. The storage-constrained dynamic lease minimizes the message exchanges for signing and keeping the leases.

Suppose that a total of n DNS resource records $R_i (i = 1, ..., n)$, are maintained on the authoritative DNS nameserver, each with maximal lease length $L_i (i = 1, ..., n)$. Each record R_i is queried by m DNS caches $C_j (j = 1, ..., m)$, with the query rate λ_{ij}. We define M_{ij} and $P_{i,j}$ as the query rate and lease probability of record R_i by cache C_j. Our objective is to determine the appropriate lease length of every resource record for each DNS cache l_{ij}, in order to minimize the overall communication overhead M_{all}, the sum of M_{ij}. The decision should be made under the following constraints:

- for the record R_i and DNS cache C_j, the lease length l_{ij} should be within the range of 0 and L_i.
- the total storage P_{all} should be less than the storage overhead allowance P_{max}, the sum of P_{ij}.

Thus, this problem can be defined as below:

$$\text{minimize } M_{all} = \sum_{i=1}^{n} \sum_{j=1}^{m} M_{ij}$$

$$\text{subject to for any } R_i \text{ and } C_{ij}, 0 \leq l_{ij} \leq L_i$$

$$P_{all} = \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} \leq P_{max}$$

We refer this kind of optimization as the storage-based lease problem (SLP). Since SLP is equivalent to a Knapsack problem, it is NP-complete, but its approximation solution can be found by utilizing the greedy algorithm.

If we have multiple records with different maximal lease lengths, we need to sort the $\frac{\triangle M_{ij}}{\triangle P_{ij}}$, each of which is equal to λ_{ij}, and then we grant the lease to the DNS cache with the highest query rate. It is clear that, in order to reduce communication overhead, we should grant the lease to the DNS cache with the highest query rate.

If the nameserver always grants leases with their maximal lengths to the DNS caches selected as above until the storage constraint reached, we can guarantee that the total query rate covered by leases is maximal.

4.2.2 Communication-constrained Dynamic Lease

Similarly, given the communication overhead allowance, we can design an algorithm that minimizes the storage overhead. It is also a *NP-complete* problem, and we employ the greedy algorithm to find the optimal solution. Different from the storage-constrained dynamic lease, at the beginning of the algorithm, all DNS caches related to each resource record are granted with the maximum-length leases. After that, we select the DNS cache with the smallest query rate and deprive its lease. This selection and deprivation continue until the communication allowance is satisfied. In this way, we can guarantee that the number of leases maintained by the nameserver under the communication constraint is minimal.

5 System Evaluation

We use trace-driven simulations to evaluate the effectiveness of dynamic lease algorithm, and build a prototype of DNScup on top of Bind 9.2.3 to demonstrate its easy and incremental deployment on the Internet.

5.1 Trace-driven Simulation

Our DNS traces were collected in an academic environment, where three local DNS nameservers provide DNS services for about two thousand client machines. The one-week trace collection is from July 2, 2003 to July 9, 2003. Based on the DNS traces, we simulate a scenario in which a number of clients are using three local DNS nameservers. The local DNS nameservers decide whether or not granting a lease for one cached resource record based on its query rate.

Considering the client caching effect on query intervals, we assume that clients cache each resource record for 15 minutes, which is the default setting in Mozilla. The query rate for each domain name is computed by analyzing the

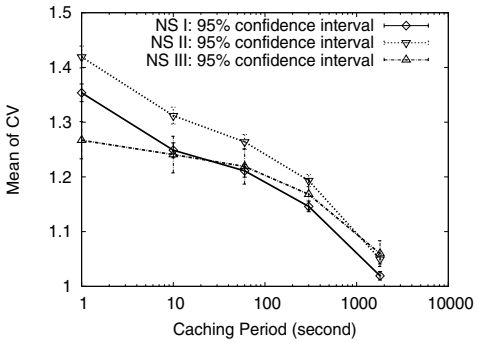


Figure 4. The mean of CV of query interval in DNS traces.

first-day traces. For three categories of domain names (regular, CDN, and Dyn domains), we set different maximal lease length based on their DN2IP mapping change rates. The maximal length for a regular domain is set to six days, while those for DNS and Dyn domains are set to 200 and 6,000 seconds, respectively.

5.1.1 Poisson Distribution Validation

The DNS query behavior is related to the Web request access pattern. As most Web browsers cache DNS responses, the time interval between two continuous queries for one domain name likely follows Poisson distribution. We use the mean of Coefficient of Variation (CV) to study the query interval distribution in our DNS traces. Figure 4 shows the dynamics of the mean of CV with respect to the cache duration at the client side. With the increase of the client cache duration, as the mean of CV is closer to 1, the time intervals are more likely to follow a Poisson distribution. It is also noticeable that the 95% confidence interval of the mean is very small in all cases.

5.1.2 Simulation Results

We introduce two relative system metrics to evaluate the lease algorithms: storage percentage and query rate percentage. The storage percentage is defined as the ratio between the number of leases granted to querying DNS caches and the maximal number of leases that an authoritative DNS nameserver could grant. There are two extreme cases: (1) if the authoritative DNS nameserver grants a lease to each query and all its resource records have valid leases all the time, the storage percentage is 100%; and (2) if no lease is granted to any query, the storage percentage is zero. The query rate percentage is defined as the ratio between the query rate issued from a DNS cache and the maximal query rate that the DNS cache could generate. If no lease is granted, the lease algorithm degrades to the polling scheme and generates the maximal query rate. Thus, the query rate percentage becomes 100% under this extreme scenario.

We compare the simple fixed-length lease scheme, which grants the same length lease to every incoming query, with the proposed dynamic lease. Our simulation results clearly show that the performance of dynamic lease is superior to that of the fixed-length lease. Figure 5 illustrates the simulation results of regular domains based on the traces at the first DNS nameserver. Note that the X-axis in Figure 5 (b) is in logarithmic scale. For CDN and Dyn domains and the cases at the other two DNS nameservers, we have similar results. Due to space limit, we do not present them here. In our trace-driven experiments, the storage percentage is bounded at 60%, since in pratice only a portion of resource records have valid leases at a time.

Dynamic lease is effective to reduce storage overhead. As shown in Figure 5 (a), under the query rate percentage of 20%, the storage percentage of dynamic lease is 19% while that of fixed-length lease is 47%. Thus, dynamic lease reduces storage overhead by 60%. At the same time, dynamic lease is also effective in reducing communication overhead. As shown in Figure 5 (b), under the storage percentage of 1%, the query rate percentage of dynamic lease is 56% while that of fixed-length lease is 88%. The reduction of communication overhead is about 36%.

In our experiments, due to the limitation of the trace length (seven days), the maximal length for regular domains is relatively small. Since regular domains seldom change their DN2IP mappings, we may use a much higher lease length to gain a better performance. Note that the lease selection in our experiment is done off-line based on the trace analyses, and the lease length remains constant. In reality, a DNS cache may monitor the rates of cached records in the incoming queries. When it detects a significant change in query rates, the DNS cache will notify the authoritative DNS nameserver to re-negotiate the current leases.

5.2 Prototype Implementation

We have modified the prompt notification of the zone mechanism in the BIND 9.2.3 implementation. We define a new type of message called `CACHE-UPDATE` with a new opcode 6 in the query/response headers for lease negotiation. Each DNS query includes the query rate originated from the local clients, which is expressed in a new 16-bit field called `RRC` (recent reference counter) at the question section. If a lease is granted, its duration is specified in a new 16-bit field called `LLT` (lease length time) at the answer section of the response from authoritative DNS nameserver.

According to our design, three core components of DNScup have been added to BIND 9.2.3, including the

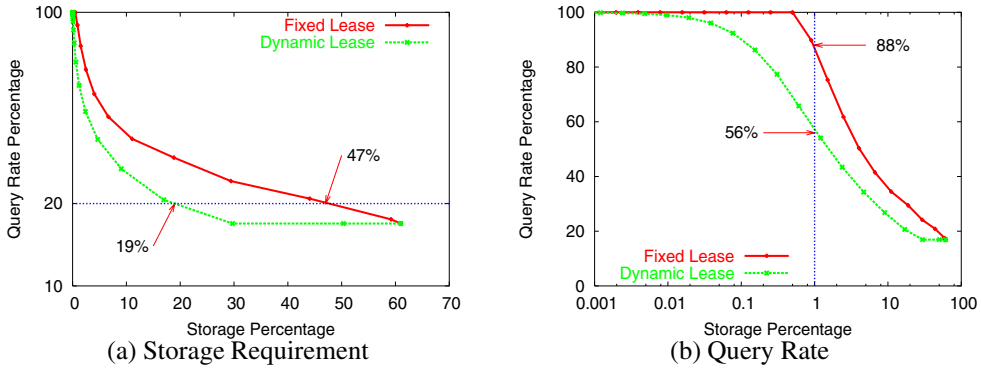


Figure 5. Performance comparison between fixed and dynamic lease

detection module, the listening module, and the notification module. The detection module detects a DNS record change; the listening module monitors incoming DNS queries and updates the track file when necessary; and the notification module propagates DNS `CACHE-UPDATE` messages. The normal DNS operations remain intact. The interactions among all components are illustrated in Figure 6. For DNS resource records of the authoritative DNS nameserver, the `named` daemon creates a database file to keep track of the incoming DNS queries. Each tuple in this file consists of five fields, including the source IP address, queried zone name, query type, query time, and lease length. When a DNS query comes in, the `named` first decides if a lease should be granted based on the query rate carried with the query. If yes, a new tuple is added to the track file, and the corresponding response is sent back.

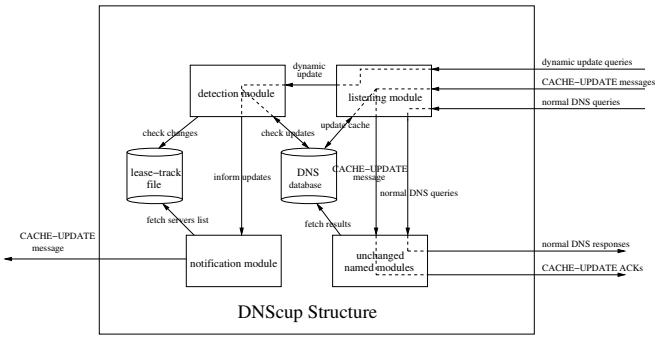


Figure 6. Structure of DNScup Prototype

We validate our implementation in a testbed environment—a hierarchy of DNS nameservers in a LAN. The testbed is shown in Figure 7. By utilizing multiple virtual IP addresses, we run a master authoritative DNS nameserver and its two slaves on a machine. The root nameserver and two DNS caches are mimicked at three different machines, respectively. The machines used in our experiments are 1GHz Pentium IIIs with 128MB RAM running RedHat Linux 9.1, connected by a 100 Mbps Ethernet. We construct 40 zones from the 50 most popular domain names in IRCache proxy traces. The simulated DNS system accepts all kinds of existing messages as well as DNScup messages. Our preliminary experimental results indicate that all message sizes are far below the limitation of 512 bytes, set by RFC 1035 [11]; and the difference in computation overhead between TTL and DNScup is hardly noticeable.

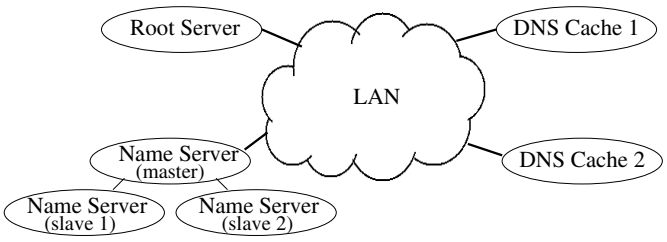


Figure 7. DNScup Implementation Testbed

5.3 Secure DNScup

DNScup may raise some concerns on the DNS security. In our current implementation, we transmit DNS messages in plain text for simplicity and efficiency. However, to protect DNS caches against poisoned CACHE-UPDATE messages originated from a compromised DNS nameserver, we need a secure communication channel for cache update. Fortunately, DNSSEC [6] and the secure DNS Dynamic Update protocols [23] have been proposed. Coupled with the proposed secure DNS mechanisms, DNScup can achieve a secure cache update without much difficulty.

6 Conclusion

In this paper, we have proposed a DNS cache update protocol, called *DNScup*, working as middleware to main-

tain strong consistency in DNS caches. To investigate the dynamics of DN2IP mapping changes, we have conducted a wide range of DNS related measurements, with the following major observations. (1) While the physical mapping changes per Web domain name rarely happen, the probability of a physical change per minute within a class is close to one. (2) Compared with the frequencies of logical mapping changes, the values of the corresponding TTLs are much smaller, resulting in a large amount of redundant DNS traffic.

Based on our measurements, we conclude that maintaining strong cache consistency is essential to prevent potential losses of service availability, particularly under the conditions of sudden and unexpected changes on the Internet. Furthermore, with strong cache consistency support, CDNs and other mechanisms can provide fine-grained load-balance, quick responsiveness to network failure or flash crowd, and end-to-end mobility, without degrading the scalability and performance of DNS.

To keep track of the local DNS nameservers whose clients need strong cache consistency for always-on Internet services, DNScup uses dynamic lease to reduce the storage overhead and communication overhead. Based on the DNS Dynamic Update protocol, we build a DNScup prototype with minor modifications to the current DNS implementation. The components of DNScup implementation include the detection module, the listening module, the notification module, and the lease-track file. Our trace-driven simulation and prototype implementation demonstrate that DNScup achieves the strong cache consistency in DNS and significantly improves its availability, performance and scalability.

Acknowledgment: We thank anonymous reviewers for their comments and suggestions. We appreciate William L. Bynum for reading the paper and his comments. This work is partially supported by the National Science Foundation under grants CNS-0098055, CNS-0405909, and CNS-0509054/0509061.

References

[1] Ircache home. http://www.ircache.net/.

[2] N. Brownlee, K. Claffy, and E. Nemeth. DNS Root/gTLD performance measurements. In *Proceedings of USENIX LISA'2001*, San Antonio, TX, December 2001.

[3] R. Cox, A. Muthitacharoen, and R. Morris. Serving DNS using a peer-to-peer lookup service. In *Proceedings of IPTPS'2002*, Cambridge, MA, March 2002.

[4] C. Cranor, E. Gansner, B. Krishnamurthy, and O. Spatscheck. Characterizing large DNS traces using graphs. In *Proceedings of ACM IMW'2001*, San Francisco, CA, November 2001.

[5] 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 ACM SIGCOMM'92*, Baltimore, MD, August 1992.

[6] D. Eastlake. Domain name system security extensions. In *RFC 2535*, March 1999.

[7] J. Eisenberg and C. Partridge. The internet under crisis conditions: Learning from september 11. *ACM Computer Communication Review*, 33(2), April 2003.

[8] E. S. J. Jung, H. Balakrishnan, and R. Morris. DNS performance and the effectiveness of caching. In *Proceedings of ACM IMW'2002*, San Francisco, CA, October 2002.

[9] R. Liston, S. Srinivasan, and E. Zegura. Diversity in DNS performance measures. In *Proceedings ACM IMW'2002*, Marseille, France, November 2002.

[10] P. Mockapetris. Domain names-concepts and facilities. In *RFC1034*, November 1987.

[11] P. Mockapetris. Domain names-implementation and specification. In *RFC 1035*, November 1987.

[12] J. Pang, A. Akella, A. Shaikh, B. Krishnamurthy, and S. Seshan. On the responsiveness of DNS-based network control. In *Proceedings of ACM IMC'2004*, Taormina, Sicily, Italy, October 2004.

[13] J. Pang, J. Hendricks, A. Akella, R. D. Prisco, B. Maggs, and S. Seshan. Availability, usage and deployment characterisitics of the domain name system. In *Proceedings of ACM IMC'2004*, Taormina, Sicily, Italy, October 2004.

[14] V. Pappas, Z. Xu, S. Lu, A. Terzes, D. Massey, and L. Zhang. Impact of configuration errors on DNS robustness. In *Proceedings of ACM SIGCOMM'2004*, Portland, OR, August 2004.

[15] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[16] V. Ramasubramanian and E. Sirer. The design and implementation of a next generation name service for the internet. In *Proceedings of ACM SIGCOMM'2004*, Portland, Oregon, USA, August 2004.

[17] Y. Rekhter, S. Thomson, J. Bound, and P. Vixie. Dynamic updates in the domain name system. In *RFC2136*, April 1997.

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

[19] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the web from dns. In *Proceedings of USENIX NSDI'2004*, San Francisco, CA, USA, March 2004.

[20] B. Wellington. Secure domain name system dynamic update. In *RFC3007*, November 2000.

[21] D. Wessels, M. Fomenkov, N. Brownlee, and K. Claffy. Measurement and laboratory simulations of the upper DNS hierarchy. In *Proceedings of PAM'2004*, Antibes Juan-les-Pins, France, April 2004.

[22] C. Wills, M. Mikhailov, and H. Shang. Inferring relative popularity of internet applications by actively querying DNS caches. In *Proceedings of ACM IMC'03*, Miami, FL, October 2003.

[23] C. Wills and H. Shang. The contribution of DNS lookup costs to web object retrieval. In *Technical Report TR-00-12*, Worcester Polytechnic Institute, July 2002.