This paper was presented as part of the 14th IEEE Global Internet Symposium (GI) 2011 at IEEE INFOCOM 2011

Reducing DNS Caching

Saleem N. Bhatti
University of St Andrews, UK
saleem@cs.st-andrews.ac.uk

Randall Atkinson
Cheltenham Research, VA, USA

***Abstract*—Motivated by our ongoing work exploring an alternative Internet architecture, we wish to make use of naming services in order to support functionality such as: host and network mobility; application and/or virtual machine migration; and various forms of traffic control (e.g. multi-homing). Currently, the Domain Name System (DNS) is used to resolve names to DNS records, with relatively large time-to-live (TTL) values (several thousands of seconds) for caching the results. To support new agile services and systems, cached results may need to have much lower TTL values, so that cached DNS values do not become stale as system changes occur, e.g. changes to end-system location information to support new methods of mobility. However, current conventions for DNS configuration normally use conservatively high TTL values. We have conducted an empirical study of a live DNS deployment where we have reduced to zero the TTL values of records for the entire School of Computer Science at the University of St Andrews. Our results show that the increase in DNS load is much lower than might be expected, following a highly non-linear decrease with respect to the TTL value of the DNS records.**

I. Introduction

Naming services provide mappings from user-friendly or application-friendly names to numerical addresses. The Domain Name Service (DNS) maps fully qualified domain names (FQDNs), e.g. marston.cs.st-andrews.ac.uk, to other values such as Internet Protocol (IP) addresses, e.g. 138.251.195.61. There is increased interest in the use of naming techniques for enabling functionality in future Internet services and systems. For example; for enabling mobility (e.g. [1]–[3]); mapping services for hierarchical routing (e.g. [4]); and the authors' Identifier Locator Network Protocol (ILNP)[1] [5], [6] which will use naming to implement functionality such as mobility and multi-homing.

A. An example – using DNS for mobility

The existing Secure DNS Dynamic Update standard [7] would permit a mobile node to update its location information when the node moves. Widely used systems software, such as Microsoft Windows and the BIND software used with UNIX, already include support for Secure Dynamic DNS Update [8]. Previous studies indicate the use of Secure DNS Dynamic Update for mobility is entirely feasible [1], [2]. However, these previous studies do not consider the potential increase in traffic and load on the DNS as the time-to-live (TTL) of the relevant resource records is decreased. The TTL specifies the time for which the record can be cached before it must be retrieved from the authoritative servers. Lower TTL values mean less caching and so more DNS name lookups. A previous study stated that DNS caching is not as effective as is commonly thought [9], with the trace-driven emulations suggested that it should be possible to have TTL values as low as ∼1000s (or even a few 100s) without adverse impact on DNS.

[1] http://ilnp.cs.st-andrews.ac.uk/

B. Outline of this paper

In this paper, we present results that support the assertion in [9], that reducing the TTL of DNS resource records to low values has relatively little impact on DNS load. Indeed, we take the assertion to the limit and implement zero TTL. We have conducted an empirical study in which we have used different TTL values for DNS A records for the operational DNS server in the School of Computer Science at the University of St Andrews. In Section II we give the relevant background, including a summary of the findings in [9]. In Section III, we describe our experimental configuration and provide descriptions of the data-sets we have collected. In Section IV, we provide a basic statistical analysis and a simple 'spectral' analysis of the data-sets. After a short discussion in Section V, we conclude with a summary and indications of our (ongoing and) future work in Section VI.

II. DNS usage

The current DNS resolution process forms a spatio-temporal caching hierarchy, the cache time controlled by the use of TTL value for the DNS records. We provide here a simple example to illustrate the use of caching and TTL values within the DNS system: a fuller description can be found in [8].

A. DNS operation

The DNS provides distributed lookups for fully qualified domain names (FQDNs). Queries to the DNS system result in the return of DNS Resource Records (or just *records*). There are different *record types*. Each record has a *Time-To-Live (TTL)* value in seconds: the time for which it is valid and can be cached. After the TTL for a record has expired, it must be looked up again and should not be served from cache.

In Figure 1, h1.blob.org will resolve "h2.flump.com". The query initially goes to the DNS resolving server local to h1, n1.blob.org (1). This query results in information about the generic top-level-domain (gTLD) .com (2), the NS record returned in step (3). From a .com gTLD server, n1.blob.org requests the name server for the domain flump.com (4), receiving the NS record for ns1.flump.com. (5). Then, n1.blob.org can query for the name "h2.flump.com" (6) (7), returning an

978-1-4244-9920-5/11/$26.00 ©2011 IEEE

A record in step (8). There can be different levels of caching at different points in the name-space lookup. Normally, the responses returned at steps (3) and (5) include 'glue' records – A records with IP addresses for the nameserver. Those NS records and associated A records typically have large TTLs (perhaps several days). Records can have different TTL values, e.g. the A records returned in steps (3), (5) and (7) can have different TTL values. The information returned at step (8) could also be cached at the end-system (see Section III-B, however). Fewer DNS queries will result in less DNS traffic and faster name resolution. This is achieved by having results cached for longer periods of time, i.e. having large values for TTLs on DNS records. Certainly, NS records and corresponding A records should have a long TTL to reduce the load on root and gTLD servers.

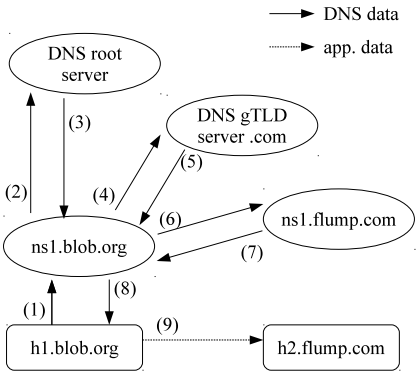

Fig. 1. A simplified DNS lookup, assuming no information is cached as yet. h1.blob.org needs to resolve the name "h2.flump.com". (1) h1 makes a query for "h2.flump.com" to its local DNS resolving server, n1.blob.org. (2) n1 makes a query to the root server. (3) The root server responds with a NS record (including a 'glue' A record) for generic top-level-domain (gTLD) for .com. (4) ns1.blob.org sends a query to a .com server and (5) receives an NS record for ns1.flump.com (including its IP address in a 'glue' A record). (6) ns1.blob.org then sends a query for "h2.flump.com" and (7) receives an A record with the relevant IP address(es), which (8) are then passed back to h1. Then (9) h1 uses the IP address to send application packets to h2.

B. DNS server-side caching

The intuition would be that cache times for DNS records should be kept as high as possible in order to keep low both DNS server load and DNS traffic. A detailed study using a set of DNS traffic traces from 2001 [9] reported that, based on analyses of the traces they used: ∼23% of lookups received no answer, accounting for more than half of all DNS packets in the wide area; ∼13% of all lookups receive a negative response, i.e. they are malformed or "bad" queries (typos etc.); and over 25% of all lookups sent to the root servers are also "bad". In summary, a substantial proportion of DNS traffic and associated DNS server load consists of "bad" queries, and so DNS servers already deal with a relatively high load: additional "good" queries may not necessarily introduce significant new load, in comparison.

The most interesting claim, however, was based on a trace-driven emulation to determine the effectiveness of per-client or shared caches for DNS. The conclusions stated that, " ... reducing the TTLs of address (A) records to as low as a few hundred seconds has little adverse effect on hit rates.", and that, " ... widespread use of dynamic, low-TTL A-record bindings should not degrade DNS performance." However, at the time of writing, with some notable exceptions (see later), most system administrators continue to use TTL values that are conservatively high, e.g. the order of hours or days.

III. Experimental configuration

We have tested experimentally the assertion for the ineffectiveness of caching made above. We used the operational DNS system of the School of Computer Science at the University of St Andrews. We captured the DNS traffic while varying the TTL value of the DNS records on the School DNS server(s).

A. Experimental procedure

A summary of the experimental environment is given in Table I. No modifications were made to the DNS servers except to the TTL values of the DNS records. The *tcpdump* application was used to capture the traffic seen on a port mirror of the switch ports to which the DNS servers were connected. All DNS queries, both from internal sources and external sources, were captured. The changes to the TTL values were made during the normal semester period, with the usual mix of School activities (teaching, research and administration) being undertaken during the measurement period. School were modified to honour TTL values of DNS records.

TABLE I
DNS operational information for data capture

Time frame	Q4 2009 (during normal semester)
DNS servers	DNS server version 6.0 6002 (0x1772) on ActiveDirectory / Windows Server 2008 R1 (standard configuration of 4 servers)
TTL values	1800s, 30s, 0s
Duration	601,200s (7-days less 1 hour) at each TTL value
Capture method	port-mirror of DNS server ethernet port(s) and capture with *tcpdump*

B. Client-side caching

For IPv4, very few applications do anything with the TTL value for a DNS record. Applications typically make DNS queries each time they wish to resolve a name (e.g. using `gethostbyname()` in C, or via other similar APIs). This programming behaviour is beneficial if we wish to exploit low TTL values, as it means that most applications do not cache DNS records. However, we find that some applications and end-systems do cache DNS results:

- *Windows* [2] : Different versions of Windows have different DNS cache behaviours, and might not honour the TTL. However, the Windows DNS cache does not keep stale DNS entries, ensuring that entries are deleted before their DNS TTL expires. This DNS cache can be disabled.
- *Linux*[3]: The Name Server Caching Daemon (*ncsd*) may be enabled on some distributions of Linux, and may

[2]http://support.microsoft.com/kb/318803
[3]http://linux-documentation.com/en/man/man8/nscd.html

have a default cache time of 15 minutes, ignoring TTL values. It can be disabled through normal configuration mechanisms (/etc/nscd.conf).

- *MacOS X & Safari*[4]: MacOSX 10.5.6 has a system-wide DNS cache that honours DNS TTLs. Safari uses that system-wide cache, instead of having its own cache.
- *Internet Explorer (IE)*[5]: IE caches DNS records for 30 minutes (24 hours in versions of IE before v4.x). This can be disabled through the Windows Registry.
- *Firefox*[6]: Firefox has a DNS cache which ignores TTL values, but can be disabled through the normal FireFox configuration mechanisms.

For our experiments, the client-side caches of School-managed end-systems were modified so there was no caching unless the TTL value was honoured.

C. Data-sets captured

We examine only the numbers of A record queries for the servers internal to the School, and infer from that the additional traffic load being placed on the DNS server by reduced TTL values producing additional lookups. A previous study has shown that A record queries can account for more than half of all DNS queries [10]. However, DNS data can also be very noisy: the selection of A queries for a set of well-known end-systems (School servers) allows us to filter malformed requests and look at the additional load generated by "real" queries only, and make comparisons across data-sets.

A summary of our data-sets is given in Table II. The "No. of pkts captured" (2nd column) includes all packets seen on port 53 including errors or "bad" queries, but does not include DNS responses. The "No. of A queries" (3rd column) counted are for the School servers, some 67 machines at the time of the measurement. The School had approximately 500 client systems (student labs, student's personal laptops, academic staff, research staff, technical staff and admin staff), with a mix of *MS Windows* and *Linux* machines in roughly equal numbers, and around 70 *MacOSX* machines.

D. Data processing

The data-sets were processed using scripts created originally in *python* v2.5.2, with the use of standard functions from the modules *math* and *numpy*[7]. The *tcpdump* captures were filtered only for those DNS A record queries for the names of 67 known servers within the School. These A record requests were counted into 1 second buckets. For the Figures 2, 3, 4, 5 and 6: the left column of the page is for TTL=1800s; the middle column for TTL=30s; and the right column for TTL=0s. The time domain plot for these 1s buckets is shown in Figure 2, with the CDFs in Figure 3.

Following the example presented in [11], we have performed a simple 'spectral' analysis on the data, in order to observe the change in the query rate volumes and distributions.

[4]Private communication with Stuart Cheshire, Apple.
[5]http://support.microsoft.com/kb/263558
[6]http://www.techiecorner.com/225/how-to-disable-firefox-dns-cache/
[7]http://numpy.scipy.org/

TABLE II
META-DATA FOR DNS TRAFFIC DATA-SETS

Data set name[a]	No. of DNS pkts captured[b]	No. of A record queries[c]	
all queries			
2009-1800	41,868,522	2,004,133	(4.79%)
2009-0030	71,105,247	2,648,769	(3.73%)
2009-0000	55,868,573	4,501,590	(8.06%)
internal: queries from end-systems within the School			
2009-1800-i	29,486,362	792,339	(2.69%)
2009-0030-i	54,097,231	951,485	(1.76%)
2009-0000-i	30,555,305	1,419,782	(4.65%)
external: queries from end-systems outside the School			
2009-1800-e	12,382,160	1,211,794	(9.79%)
2009-0030-e	17,008,016	1,697,311	(9.98%)
2009-0000-e	25,313,268	3,081,808	(12.17%)

[a] Format of dataset name is YYYY-TTTT, YYYY = year, TTTT = TTL value in seconds. Each data-set holds 601,200s of packets (7 days less one hour temporal guard-band to allow for changes in TTL for the DNS records).
[b] *All* packets to port 53 (UDP and TCP), including errors.
[c] Queries for 67 servers active during the period of measurement.

This is presented in Figures 4, 5 and 6: we have used a line between points as a visual aid, but these are discrete spectra, defined to be valid only at the marked points. For the simple spectra of Figure 4, we counted the number of queries per second (per 1s bucket) as our *query rate*, recorded the occurence of these query rates (the number of 1s buckets containing that value), across all 7 days, and then normalised. The CDF plots in Figure 5 are then produced for the corresponding graph of Figure 4. The plot of Figure 6 summarises the daily spectra across the 7 days: the points mark the mean requests per second, with minimum and maximum bars to show the variation of the A record query rates.

IV. ANALYSES

Our aim is to examine the general overall effect on DNS offered load as the TTL changes, rather than provide detailed statistics for specific DNS server or client behaviour. As we were not able to control client-side behaviour on external nodes, and due to space limitations, we choose to focus on internal nodes (which also produced more traffic than external nodes). We use the overall effects observed as indications of the scale of overall behaviour.

A. Basic statistical analysis

Figure 2 plots the queries/s for each data set over the period of each measurement. In Figure 3, we see the corresponding CDF plots for each plot of Figure 2, showing that requests were fairly evenly distributed across each of the 7-day periods.

From Table II (third column), we see that the number of queries does increase, as might be expected, as the TTL decreases. As a proportion of overall DNS traffic, we see A record query rates increase from $\sim$4.8% to $\sim$8.1%. In Table III, we see that the mean query rate increases, again as expected. However, in both cases, the increase in query rate is relatively low, and certainly not in proportion to the decrease in TTL value. In moving from TTL=1800s to TTL=30s, the mean query rate increases by $\sim$1/3 for all queries, and $\sim$1/5 for internal queries. The mean value at TTL=0s reaches a rather

modest 7.49 queries/s. Maximum query rates also remain modest, reaching 369 queries/s at TTL=0s. Overall, the mean and the maximum query rates remain manageable by today's off-the-shelf server systems and DNS software.

TABLE III
BASIC STATISTICS FOR A RECORD QUERY RATES

Data set	mean	std dev	max	$\sim$95%[d]	$\sim$99%[e]
all queries					
2009-1800	3.33	3.47	183	8	14
2009-0030	4.41	4.27	261	10	16
2009-0000	7.49	4.93	369	15	22
internal: queries from end-systems within the School					
2009-1800-i	1.31	2.98	176	8	22
2009-0030-i	1.58	3.57	168	8	24
2009-0000-i	2.36	3.48	68	8	15
external: queries from end-systems outside the School					
2009-1800-e	2.02	1.76	66	5	7
2009-0030-e	2.82	2.28	259	7	9
2009-0000-e	5.13	3.40	368	11	14

[d]The value of query rate at which we first see (to 3 s.f.) $\geq$95% of queries.
[e]The value of query rate at which we first see (to 3 s.f.) $\geq$99% of queries.

B. Simple spectral analysis

In Figure 4, we see the normalised frequency of query rates. If we look at the counts of query requests from the 1s buckets, we can build a picture of the relative spread of query rates. As the TTL changes, we see the greatest change in the query rate counts below 10 queries/s, and relatively few changes in the query rate profile above 10 queries/s. We see this very clearly in the respective CDF plots in Figure 5: most queries occur as bursts of 10 queries/s or less. From the data used to create the plots of Figure 5, we can find the approximate 95^{th}-percentile and 99^{th}-percentile of the query rates, as given in the final two columns of Table III. We see that for the internal traffic, the (approximate) 95^{th}-percentile is the same (to 3 s.f.), and is a low value (8 queries/s). So, while the mean request rate has more than doubled, most of the requests arrive in relatively small bursts. We can also see in Figure 6 the reasons for the increase in standard deviation values in Table III. The lower TTL values show increasing variation across the query rate profiles: note the size of the bars for maximum and minimum values at each query rate across the 7-day period.

On the log-log plots of Figures 4 and 6, we might reasonably fit a straight line for the upper portion of the plot (upto $\sim$20 queries/s), and so there is likely to be a power-law relationship for the distribution of query rates.

V. DISCUSSION

It is clear that decreasing the TTL value for DNS A records does increase traffic load, but the increase does not have a significant impact on DNS traffic overall, even with TTLs as low as zero. We do not propose that sites should set TTL to zero – our experiment was to assess DNS load to learn about the impact of using reduced DNS caching, and so we chose an extreme scenario: setting a whole site's A records to use zero TTL and measuring DNS load. Others have already explored the possibility for DNS-enabled mobility in various different ways [1], [2], [12].

A. Mobility revisited

For our example of mobility, a purely analytical study of the use of zero TTL is presented in [12], and extrapolates from one of the same data-sets used in [9]. The main finding was that a TTL of zero for A records in supporting mobile nodes is unlikely to have significant impact on DNS traffic load, but, of course, will be dependent on a number of factors, including the number of mobile nodes and rate of mobility. Our experiments support that finding. However, to perform a complete assessment of the impact of using DNS for mobility, we would also need to assess the additional DNS server load introduced by Secure DNS Dynamic Update (ala RFC3007).

B. Operational issues

We did not analyse the impact on application level performance due to the decrease in TTL values. Anecdotally, there were no reports of additional application or service latency from either our users, or other system and network administrators from other parts of the University with which our systems interact.

Our Systems Admin Group did observe a potentially serious problem, not with end-systems, but with the mail service records. MX records and associated A records are used by the mail service to resolve the address(es) of mail servers for a domain. The School's primary DNS servers are within the School; a secondary DNS server is in a different building nearby. In a preliminary experiment in 2008 (with BIND8), we also had reduced the DNS TTL value for many record types, including MX records. At one point, there was a network outage affecting the entire University. This meant that all of the DNS servers were unavailable externally, so MX record queries would not be answered. Different mail server implementations would probably interpret this differently. It could be interpreted simply as a temporarily unavailable mail server, or more drastically as the mail domain being a spam site. So, it is desirable for DNS records for services, such as MX and SRV records, and associated A/AAAA records, to avoid using low DNS TTL values. Also, this experience reinforces the well known importance of topologically diverse secondary DNS servers.

Other operational issues with the behaviour of DNS servers and clients under various network conditions have been documented in detail [10], [13]. While we have not yet analysed fully the implications of low TTL on possible operation of applications using DNS, it seems clear that low TTL values will have an impact on the behaviour of some systems.

Meanwhile, at the time of writing, Yahoo has servers with A record TTL values of 60 seconds, while Akamai has servers with A record TTL values of 20s. So low DNS TTL values are already in use in commercial environments today.

Local *address resolution* protocols may need to be reconfigured. RFC4861 does specify some configurable caching

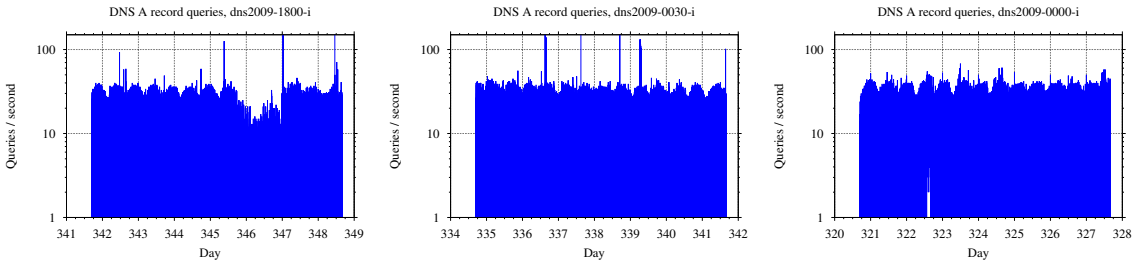

Fig. 2. DNS A record queries. The horizontal axis shows time in units of days, 01 Jan 2009 is day 1.

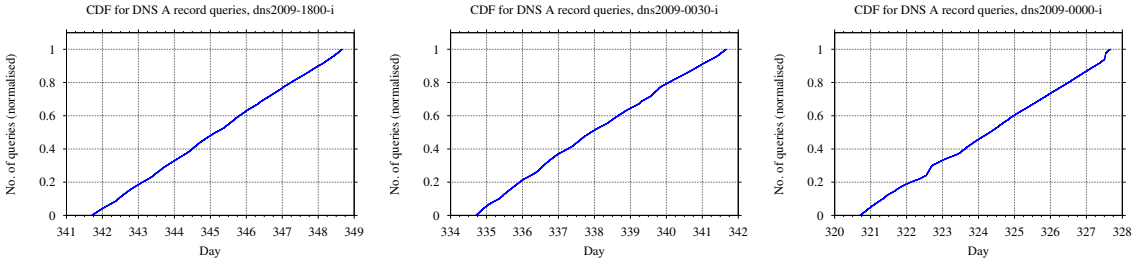

Fig. 3. CDF plots for DNS A record queries. The horizontal axis shows time in units of days, 01 Jan 2009 is day 1.

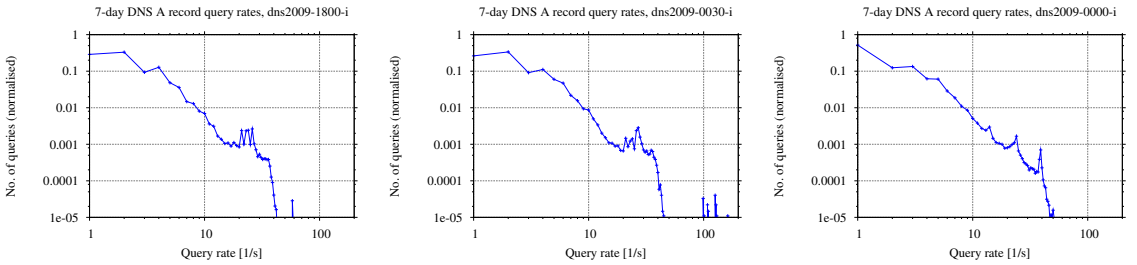

Fig. 4. DNS A record query rates – 7-days, totals, normalised (log-log axes).

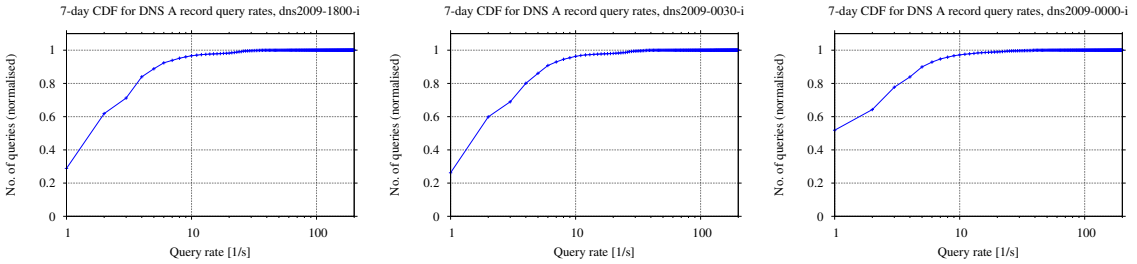

Fig. 5. CDF plots for DNS A record query rates – 7 days, totals (CDFs of plots in Figure 4, log-linear axes).

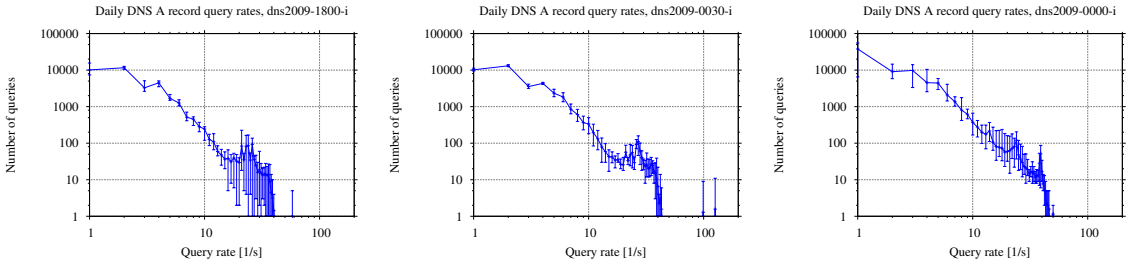

Fig. 6. DNS A record query rates – mean daily query counts for 7-day period (with minimum and maximum, log-log axes).

behaviour for IPv6: Neighbour Discovery (ND) can be configured for a site through Router Advertisements. For IPv4, *ad hoc* mechanisms may be required, especially if ARP cache times are hard-wired into system code and difficult to change.

C. Exploiting zero TTL for DNS records

Our original motivation for examining the caching performance of DNS with low TTL was in relation to our ongoing work on ILNP., including mobility. We believe that zero TTL provides other possibilities worth exploring for improved network functionality.

A potential area is load balancing and management of virtual machines (including migration) in data centres and cloud computing. Indeed, content and service providers have been using for some time, DNS with low TTL values for load balancing [14] and content distribution, but there can be issues with DNS behaviour [15].

We are exploring possibilities for ILNP, including mobility [16]. New DNS records could hold multiple network Locator (address prefix) values for a site, with preference values. This could be used to offer options for path-control and lightweight traffic-engineering from edge sites, as well as possibilities for network defence [17].

However, it may be beneficial to configure TTL=1s instead of zero for performance benefits of some applications. For example, this would permit web browser code loops which open multiple connections in parallel to the same site to set-up those connections without having to make separate DNS lookups for each connection.

Low TTL values could help to alleviate the effects of DNS cache poisoning attacks [18]. However, spoofing attacks could be more successful due to the increased number of queries. Authenticated DNS records would help in both cases.

VI. Conclusion and Future Work

We have investigated the use of low TTL values in DNS A records. We have made 7-day measurements of DNS queries for the School of Computer Science at the University of St Andrews for TTL values of 1800s, 30s, and 0s.

We have found good agreement with the conclusions drawn in previous analyses, including a trace-driven emulation [9] which suggested that TTL values of a few 100s for DNS A records should not impact significantly on DNS. However, we found that much lower values than suggested are possible, and zero TTL was used in our experiments without adverse impact on DNS load. Tables II and III (internal) show that decreasing TTL from 1800s to 0s caused mean query rates for A records to almost double, while absolute query rates remained modest (mean of 2.36 queries/s, maximum of 68 queries/s), and at a relatively low proportion of overall DNS load ($\sim$4.7%).

We propose, therefore, that the use of naming within the DNS could be exploited to offer functionality, such as mobility, by allowing DNS A record values to have low cache times.

For future work, we aim to run additional experiments at other sites, and to test the impact of Secure DNS Dynamic Update coupled with low TTL values. We are confident that naming and the DNS can be used for supporting network layer functionality, and we intend to implement and investigate mobility, multi-homing, VM migration and other functionality for our ongoing work in ILNP, by exploiting low TTL values in DNS records. Security issues also need further investigation.

Acknowledgments

Our thanks to the SysAdmin Group (School of Computer Science, University of St Andrews) for enabling the DNS experiments on the operational DNS system. Some of this work was presented in invited talks at NANOG50[8] and IETF79[9]: we are grateful for the discussion from the attendees, as well as specific comments by Stuart Cheshire (Apple), Alan Clegg (ISC), Aaron Falk (BBN), Peter Koch (DENIC eG), Dave Thaler (Microsoft), Christian Vogt (Ericsson), Scott Whyte (Google). Dr Cheshire and Dr Thaler provided information on DNS behaviour of commercial products. Dr Cheshire also initiated the discussion about why TTL=1s would be better than TTL=0s. The GI2011 reviewers gave feedback which has helped to improve the presentation of this paper.

References

[1] A. C. Snoeren and H. Balakrishnan. An end-to-end approach to host mobility. In *Proc. MobiCom 2000*, pages 155–166, 2000.

[2] A. Pappas, S. Hailes, and R. Giaffreda. Mobile Host Location Tracking Through DNS. In *LCS2002: 2002 IEEE London Communications Symposium*, Sep 2007.

[3] R. Atkinson, S. Bhatti, and S. Hailes. Mobility through naming: impact on DNS. In *Proc. MobiArch 2008*, pages 7–12, 2008.

[4] L. Jakab, A. Cabellos-Aparicio, F. Coras, D. Saucez, and O. Bonaventure. LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System. *IEEE JSAC*, 28(8):1332–1343, Oct 2010.

[5] R. Atkinson, S. Bhatti, and S. Hailes. ILNP: mobility, multi-homing, localised addressing and security through naming. *Telecommunication Systems*, 42:273–291, 2009.

[6] R. Atkinson, S. Bhatti, and S. Hailes. Evolving the Internet Architecture Through Naming. *IEEE JSAC*, 28(8):1319–1325, Oct 2010.

[7] B. Wellington. Secure Domain Name System (DNS) Dynamic Update. RFC 3007, Nov 2000.

[8] C. Liu and P.Albitz. *DNS and BIND, 5th Edition*. O'Reilly and Associates, Sebastopol, CA, USA, May 2006.

[9] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS performance and the effectiveness of caching. *IEEE/ACM Trans. Netw.*, 10(5):589–603, 2002.

[10] D. Wessels and M. Fomenkov. Wow, That's a Lot of Packets. In *Proc PAM2003*, Apr 2003.

[11] A. Broido, E. Nemeth, and kc Claffy. Spectroscopy of DNS update traffic. *SIGMETRICS Perform. Eval. Rev.*, 31(1):320–321, 2003.

[12] Yi Wu, J. Tuononen, and M. Latvala. An Analytical Model for DNS Performance with TTL Value 0 in Mobile Internet. In *TENCON 2006: 2006 IEEE Region 10 Conference*, pages 1–4, Nov 2006.

[13] D. Wessels, M. Fomenkov, N. Brownlee, and kc Claffy. Measurements and Laboratory Simulations of the Upper DNS Hierarchy. In *Proc. PAM2004*, Apr 2004.

[14] T. Briscoe. DNS Support for Load Balancing. RFC 1794, Apr 1995.

[15] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig. Comparing DNS resolvers in the wild. In *Proc. IMC 2010*, pages 15–21, 2010.

[16] D. Rehunathan and S. Bhatti. A Comparative Assessment of Routing for Mobile Networks. In *Proc. WiMob2010*, pages 434–441, Oct 2010.

[17] R. Atkinson and S. Bhatti. Site-Controlled Secure Multi-homing and Traffic Engineering for IP. In *Proc. MILCOM2009*, Oct 2009.

[18] A. Klein. BIND 9 DNS Cache Poisoning, Mar 2007. http://www.trusteer.com/bind9dns.

[8] http://www.nanog.org/meetings/nanog50/presentations/Tuesday/NANOG50.Talk64.bhatti-nanog50_dns.pdf

[9] http://www.ietf.org/proceedings/79/slides/nbs-9.pdf