template W. Kumari
Internet-Draft Google
Intended status: Informational R. Arends
Expires: January 5, 2015 Nominet
S. Woolf

D. Migault
Orange
July 4, 2014

Highly Automated Method for Maintaining Expiring Records
draft-wkumari-dnsop-hammer-01

Abstract

This document describes a simple DNS cache optimization which keeps the most popular records in the DNS cache: Highly Automated Method for Maintaining Expiring Records (HAMMER). The principle is that records in the cache are fetched, that is to say resolved before their TTL expires and the record is flushed from the cache. By fetching Records before they are being queried by an end user, HAMMER is expected to improve the quality of experience of the end users as well as to optimize the resources involved in large DNSSEC resolving platforms.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

A recursive DNS resolver may cache a Resource Record (RR) for, at most, the Time To Live (TTL) associated with that record. While the TTL is greater than zero, the resolver may respond to queries from its cache, but once the TTL has reached zero, the resolver flushes the RR. When the resolver gets another query for that resource, it needs to initiate a new query. This is then cached and returned to the querying client. This document discusses an optimization (Highly Automated Method for Maintaining Expiring Records -- (HAMMER), also known as "prefetch") to help keep popular responses in the cache, by fetching new responses before the TTL expires. This behavior is

triggered by an incoming query, and only shortly before the cache entry was due to expire.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Terminology

- HAMMER resolver: A DNS resolver that implements HAMMER mechanism.

- HAMMER FQDN: A FQDN that is a candidate for the HAMMER process.

- HAMMER TIME: TTL Time to consider before triggering the HAMMER mechanism.

3. Motivations

When a recursive resolver responds to a client, it either responds from cache, or it initiates an iterative query to resolve the answer, caches the answer and then responds with that answer.

3.1. Improving browsing Quality of Experience by reducing response time

Any end user querying a fetched FQDN will get the response from the cache of the resolver. This provides faster responses, and thus improves end user experience for browsing and other applications/ activities.

Popular FQDNs are highly queried, and end users have high expectations in terms of application response for these FQDNs. With regular DNS rules, once the FQDN has been flushed from the cache, it waits for the next end user to request the FQDN before initiating a resolution for this given FQDN with iterative queries. This results in at least one end user waiting for this resolution to be performed over the Internet before the response is sent to him. This may provide a poor user experience since DNS response times over the Internet are unpredictable at best and it provides a response time longer then usual.

In some cases, not only the first end user querying that FQDN may be impacted, but also other end users that request the FQDN between the time the FQDN TTL expires and the time the cache is again filled. In this case, the result is impact on multiple end users and possible unnecessary load on the platform. Note that this load is increased

by the use of DNSSEC since DNSSEC may involve additional resolutions, larger payloads, and signature checks.

DNS response time for a resolution over the Internet is highly unpredictable as it depends on network congestion and servers' availability. Links share their bandwidth, so heavily loaded links result in higher response time, regardless of whether the congestion occurs close to the resolver, close to the client, or close to the authoritative servers. Loaded switches or routers may result in packet drop, which requires the resolver to notice the packet has been dropped (usually with a time out) and restart the iterative resolution. These issues are increased by the use of DNSSEC which makes DNS packets larger. Similarly, loaded servers have longer response times.

3.2. Optimize the resources involved in large DNSSEC resolving platforms

Large resolving platforms are often composed of a set of independent resolving nodes. The traffic is usually load balanced based on the query source IP addresses. This results in most popular FQDNs being resolved independently by all nodes. First this increases the number of end users who may experience unnecessary latency. Also, when DNSSEC is used, all nodes independently perform signature check operations, possibly resulting in high loads on the authoritative server.

The challenge these large DNSSEC resolving platforms have to overcome is to provide a uniform distribution of the nodes given that end user and FQDNs do not have a uniform distribution of the resources. More specifically, FQDNs and end users usually present Zipf popularity distributions, which means that most of the traffic is performed by a small set of end users and by a small set of FQDNs.

DNS and large resolving DNS platforms have resulted in uniformly balanced traffic among the nodes. In fact the resolving traffic on the Internet interface was rather small (at least in term of CPU) compared to traffic received from the end users. DNSSEC changed this, as CPU are involved in performing signature checks. One way to reduce the number of DNSSEC resolutions is to fetch the nodes with the most popular FQDNs. This avoids parallel resolutions and overall reduces cost, because signature checks are not performed, while benefiting from the already existing load balancing architecture. This architecture takes advantage of the Zipf distribution of the FQDNs' popularity. In fact, a few number of FQDNs can be cached (a few thousands) to address most of the traffic (up to 70%).

Note that to perform a single resolution for the global platform, nodes may be configured as forwarders for the most popular FQDNs

4. Overview of Operation

When an incoming query is received, and the result is in the cache, the query is answered from the cache. If the remaining TTL of the record is below some threshold, the recursive server will also initiate a cache fill operation in the background to refresh the cache entry.

The fact that the behavior is triggered by an incoming query (and not by periodically scanning the cache and refreshing all entries that are about to expire) allows unpopular names to age out of the cache naturally, while keeping popular entries in the cache.

5. Known implementations

[Ed: Well, this is kinda embarrassing. This idea occurred to us one day while sitting around a pool in New Hampshire. It then took a while before I wrote it down, mostly because I *really* wanted to get "Stop! Hammer Time!" into a draft. Anyway, we presented it in Berlin, and Wouter Wijngaards stood up and mentioned that Unbound already does this (they use a percentage of TTL, instead of a number of seconds). Then we heard from OpenDNS that they *also* implement something similar. Then we had a number of discussions, then got sidetracked into other things. Anyway, BIND as of 9.10, around Feb 2014 now implements something like this (https://deepthought.isc.org/article/AA-01122/0/Early-refresh-of-cache-records-cache-prefetch-in-BIND-9.10.html), and enables it by default. Unfortunately, while BIND uses the times based approach, they named their parameters "trigger" and "eligibility" - and shouting "Eligibility! Trigger time!" simply isn't funny (unless you have a very odd sense of humor... So, we are now documenting implementations that existed before this was published and an impl,entation that we think was based on this. We think that this has value to the community. I'm also leaving in the HAMMER TIME bit, because it makes me giggle. This below section should be filled out with more detail, in collaboration with the implementors, but this is being written *just* before the draft cutoff.].

A number of recursive resolvers implement techniques similar to the techniques described in this document. This section documents some of these and tradeoffs they make in picking their techniques.

5.1. Unbound (NLNet Labs)

The Unbound validating, recursive, and caching DNS resolver implements a HAMMER type feature, called "prefetch". This feature can be enabled or disabled though the configuration option "prefetch: <yes or no>". When enabled, Unbound will fetch expiring records when their remaining TTL is less than 10% of their original TTL.

[Ed: Unbound's "prefetch" function was developed independently, before this draft was written. The authors were unaware of it when writing the document.]

5.2. OpenDNS

The public DNS resolver, OpenDNS implements a prefetch like solution.

[Ed: Will work with OpenDNS to get more details.]

5.3. ISC BIND

As of version 9.10, Internet Systems Consortium's BIND implements the HAMMER functionality. This feature is enabled by default.

The functionality is configured using the "prefetch" options statement, with two parameters:

Trigger This is equivalent to the HAMMER_TIME parameter described below.

Eligibility This is equivalent to the STOP parameter described below.

6. An example / reference implementation

When a recursive resolver that implements HAMMER receives a query for information that it has in the cache, it responds from the cache.

If the queried FQDN is a HAMMER FQDN, the HAMMER resolver compares the TTL value to the HAMMER TIME, as well as if the FQDN has already been fetched.

If the HAMMER FQDN has already been fetched or provisioned) then nothing is done.

If the HAMMER FQDN has not yet been fetched and the TTL is less then the HAMMER_TIME, the HAMMER resolver starts a resolution for the queried FQDN in order to fill the cache, just as if the TTL had expired. During this cache fill operation the resolver continues to

respond from cache (until the TTL expires). When the cache fill query completes, the new response replaces the existing cached information. This ensures the cache has fresh data for subsequent queries.

Since the cache fill query is initiated before the existing cached entry expires (and is flushed), responses will come from the cache more often. This decreases the client resolution latency and improves the user experience.

The cache fill resolution is triggered by an incoming query (and only if that query arrives shortly before the record would expire anyway). This effectively keeps the most popular data uniformly queried in the cache, without having to maintain counters in the cache or proactively resolve responses that are not likely to be needed as often. This is purely an implementation optimization - resolvers always have the option to cache records for less than the TTL (for example, when running low on cache space, etc), this simply triggers a refresh of the RR before it expires.

Note that non-uniformly queried FQDNs may be popular and may not benefit from the HAMMER mechanism. For example, an FQDNs MAY be heavily queried the first 10 minutes of every hour with a 30 minute TTL. In that case DNS queries are not expected to come between TTL - HAMMER_TIME and TTL.

HAMMER FQDNs with small TTL may generate a cache fill process even though they are not so popular. Suppose an end user is setting a specific session which requires multiple DNS resolutions on a given FQDN. These resolutions are necessary for a short period of time, i.e. the necessary time to establish the session. If these FQDNs have been set with a small TTL - in the order of the time session establishment - the multiple queries to a HAMMER resolver may trigger an unnecessary resolution. As a result HAMMER would not scale thousands of these FQDNs. As a result, if the original TTL of the RR is less than (or close to HAMMER_TIME), the described method could cause excessive cache fill queries to occur. In order to prevent this an additional variable named STOP (described below) is introduced. If the original TTL of the RR is less than STOP * HAMMER_TIME then the cache entry should be marked with a "Can't touch this" flag, and the described method should not be used.

6.1. Variables

These are the mandatory variables:

- HAMMER_TIME: is the number of seconds before TTL expiration that a cache fill query should be initiated. This should be a user configurable value. A default of 2 seconds is RECOMMENDED.

- STOP: should be a user configurable variable. A default of 3 is recommended.

Implementations may consider additional variables. These are not mandatory but would address specific use of the HAMMER.

- HAMMER_MATCH: should be a user configurable variable. It defines FQDNs that are expected to implement HAMMER. This rule can be expressed in different ways. It can be a list of FQDNs, or a number indicating the number of most popular FQDNs that needs to be considered. How HAMMER_MATCH is expressed is implementation dependent. Implementations can use a list of FQDNs, others can use a matching rule on the FQDNs, or define the HAMMER_FQDNs as the X most popular FQDNs.

- HAMMER_FORWARDER: should be a user configurable variable. It is optional and designates the DNS server the resolver forwards the request to.

7. IANA Considerations

This document makes no request of the IANA.

8. Security Considerations

This technique leverages existing protocols, and should not introduce any new risks, other than a slight increase in traffic.

By initiating cache fill entries before the existing RR has expired this technique will slightly increase the number of queries seen by authoritative servers. This increase will be inversely proportional to the average TTL of the records that they serve.

It is unlikely, but possible that this increase could cause a denial of service condition.

9. Acknowledgements

The authors wish to thank Tony Finch and MC Hammer. We also wish to thank Brian Somers and Wouter Wijngaards for telling us that they already do this :-) (They should probably be co-authors, but I left this too close to the draft cutoff time to confirm with them that they are willing to have thier names on this).

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

[I-D.ietf-sidr-iana-objects]
Manderson, T., Vegoda, L., and S. Kent, "RPKI Objects issued by IANA", draft-ietf-sidr-iana-objects-03 (work in progress), May 2011.

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

From -00 to 01:

- o Fairly large rewrite.
- o Added text on the fact that there are implmentations that do this.
- o Added the "prefetch" name, cleaned up some readability.
- o Daniel's test (Section 3.2) added.

From -template to -00.

- o Wrote some text.
- o Changed the name.

Authors' Addresses

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

Roy Arends
Nominet
Edmund Halley Road
Oxford OX4 4DQ
United Kingdom

Email: roy@nominet.org.uk

Suzanne Woolf
39 Dodge St. #317
Beverly, MA 01915
US

Email: suzworldwide@gmail.com

Daniel Migault
Orange
38 rue du General Leclerc
92794 Issy-les-Moulineaux Cedex 9
France

Phone: +33 1 45 29 60 52
Email: daniel.migaultf@orange.com