dnsop
Internet-Draft
Intended status: Standards Track
Expires: March 31, 2016

J. Yao
N. Kong
X. Li
CNNIC

September 28, 2015

Decreasing Fetch time of Root Data by Improving the Mechanism of Root Data Cacheing
draft-yao-dnsop-root-cache-00

Abstract

Many DNS recursive resolvers have long round trip times to the DNS root server. It has been an obstacle to increse the performance of DNS query. In order to decrease fetch time of DNS root data, this document proposes a new mechanism by improving the mechanism of root data cacheing.

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 March 31, 2016.

Table of Contents

1. Introduction

Many DNS recursive resolvers have long round trip times to the DNS root server, which sometimes become the big burden to increase the DNS resolving performance. The document [Root-loopback] allows the new root zone server to be run on a loopback address to decrease access time to root servers, but it also points out " this design being described here is not considered a "best practice." This document provides a new method by improving the mechanism of root data cacheing.

The cache provides an efficient way for DNS to efficiently improve its resolving performance. The resolver can eliminate network delay and name server load from most requests by answering them from its cache of prior results.

TTL is a time limit on how long an RR can be kept in a cache. Root zone data normally is kept stable except the SOA record which the root administrator may intentionaly change. Long TTLs can be used to maximize caching. Recursive resolvers currently send queries for all TLDs that are not in their caches to root servers. If there is a mechanism which can increase the valid time of root data in the cache without using the outdated root data information, it will help to decrease access time to root servers generally or help to decrease the fetch time of root data. When any RR is updated in a zone, there will have a new version of zone and have a different serial No. in SOA record.

The root zone normally updates its data not frequently as other zones such as the TLD zone. The data for the root zone is usually long-lived. While TTLs can be used to dynamically adjust caching, and a zero TTL prohibits caching, the realities of root zone features suggest that these times should be on the order of days for the typical resolver. 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.

2. Terminology

The basic key words such as "MUST", "MUST NOT", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "MAY", and "MAYNOT" are to be interpreted as described in [RFC2119].

The basic DNS terms used in this specification are defined in the documents [RFC1034] and [RFC1035].

3. Design Considerations

The following features will be used to design the cache based mechanism to decrease fetch time of root data:

- All resolvers have to connect the root servers for root data. The root data is a common one for all resolvers.

- The TLD contents of the root zone is not changed frequently.

- DNSSEC[RFC4033][RFC4034][RFC4035] protects the data origin authentication and data integrity.

The goal of our design is that the cached root data can be valid as long as the expire value of the root SOA or longger while the data still remains fresh. The mechanism proposed by this dcoument will decrease fetch time of DNS root data efficently.

4. Mechanism for decreasing fetch time of DNS root data

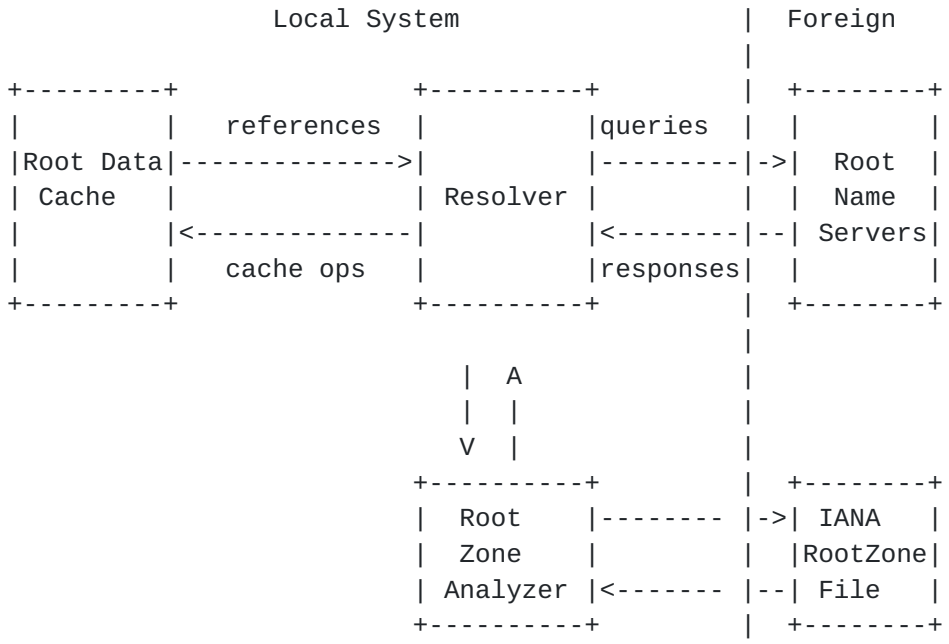

Figure 1. The structure of root data caching mechanism

The figure above shows the structure of root data caching mechanism. There will have a separated cache specially designed for root data called root data cache in the resolver. There will have a Root Zone Analyzer which will get the SOA record and zone file of root, and analyze the data. The design is shown in the figure above. The detailed function of the three components in the local system is shown below.

1)Root Data Cache:
It will cache the root TLD data information from the root name server queried by the resolver. It has three parameters, respectively, version No., fingerprint and refresh timer.

Version No:
The value is same to SOA serial No. of the current version of root zone.

Fingerprint:
The value is the digest value by the same digest operation (MD5 [RFC 1321]) performed by the root zone analyzer to the root zone (excluding the SOA and its RRSIG)

Refresh Timer:

The contents of the root data cache MUST be refreshed using the timer. That is that every RR will be assigned a refresh timer with the default value when it is put into the root data cache. The default timer value is the expire value from the root SOA record in [RFC1035]. The administrators may set their own timer value to RRs in the root data cache.

2)Resolver:
When the resolver needs to query the TLD information in the root, it should check its cache (specified in RFC 1035) first. If it fails, it then should check the root data cache. If it can not find the answer queried, it then asks the question to root name servers. If it is a DNSSEC query to the root, the resolver should verify the response from the root with DNSSEC. If it passes the DNSSEC validation, the resolver should put these TLD RRs into the root data cache. If it is not a DNSSEC query, the resolver should send another query with the same question with DNSSEC to the root. If the response from the root passes the DNSSEC validation, the resolver should put the TLD data in the response into the root data cache. If these TLD RRs can not pass the DNSSEC validation, it should discard them without caching. The basic implication is that all sanity checks on any TLD RR should be performed before any of it is cached into the root data cache. When a resolver finds a set of RRs for some TLD name in a response from the root, and wants to cache the RRs into the root data cache, it should check its root data cache for already existing RRs. If there is one, the resolver should replace it with the new one. When several RRs of the same type are available for a TLD, the resolver should either cache them all or none at all. The resolver should not cache a possibly partial set of RRs.

3)Root Zone Analyzer:
It will check the root SOA record and analyze the root zone data frequently. Currently, the root SOA is changed every day even if the contents of the root zone are unchanged. So in most cases, the difference between the new zone and old zone is that SOA record and its RRSIG have the new version while the other parts will be kept same. When the root zone analyzer finds that the root SOA is changed, it should download the root zone file. The root zone analyzer will perform the same digest operation (such as MD5), and compare the result to fingerprint of the root zone (excluding the SOA and its RRSIG) stored in the root data cache. If the SOA is changed and the fingerprint is not changed, the version No. will be changed to the new SOA serial No.. If both the SOA and the fingerprint are changed, the version No. must be changed to the new SOA serial No., and the fingerprint must be changed to the new digest value of the root zone (excluding the SOA and its RRSIG), and flush the root data cache. It means that all root data cache information may be outdated. The system should discard all data in the root data cache.

The detailed steps for using this system is shown below:

step 1, When the resolver needs to query the TLD name information in the root, it should check its cache (specified in RFC 1035) first. If it finds the information, go to step 4; otherwise go to step 2.

step 2, Check the root data cache for the TLD name information. If it finds the answer queried, go to step 4; otherwise go to step 3.

step 3, Ask the question to the root name servers. If it finds the answer queried, go to step 5 and step 4 simultaneously; otherwise go to step 4.

step 4, The resolver has the necessary information and followed the steps specified in RFC 1035. End.

step 5, If it is a DNSSEC query to the root, the resolver should validate the response from the root with DNSSEC; if it passes the DNSSEC validation, the resolver should put these TLD RRs into the root data cache, and set the refresh timer with the default value for these RRs; if it does not pass the DNSSEC validation, these RRs will not be put into the root data cache; end. If it is not a DNSSEC query, go to step 6.

step 6, The resolver should send another query with the same question with DNSSEC to the root. If the response from the root passes the DNSSEC validation, the resolver should put the TLD data in the response into the root data cache, and set the refresh timer with the default value for these RRs. If these TLD RRs can not pass the DNSSEC validation, it should discard them without caching. End.

Note: The basic implication is that all sanity checks on any TLD RR should be performed before any of it is cached into the root data cache.

5. System Requirements

In order to implement the mechanism described in this document:

- The system MUST be able to validate DNSSEC resource records.
- The system MUST have an up-to-date copy of the DNS root key.
- Only root TLD data from root zone should be cached.

The requirement above is to be sure that authoritative data in the root data cache MUST be identical to the same data in the root zone

for the DNS. It is possible to change the unsigned data in the root data cache, but that operation MUST not be allowed.

6. Requirement of the Root Data Cache

Root cached data will be discarded by a timeout mechanism with refresh timer value or by the root data anlayzer when the data of the root zone is changed. The requirements to run the root data cache are shown below:

- Inside the root data cache, refresh timers for cached root data conceptually "count down", while TTL in the RR will be kept in the constant value.
- When the RRs are exported from root data cache, the refresh timer will be removed and the TTL in RR will start to "count down".
- Every RR will be assigned a refresh timer with the default value when it is put into the root data cache.
- The root data cache should discard old RRs whose timer has expired.

The resolver may make the queries to several different root name servers to answer a particular user query. Since all the root servers serve the same root data, it will not impact the data accurance.

7. Requirement and Operation of the Root Zone Analyzer

The resolver expects to cache root data which it receives in responses from the root since it is useful in answering future client requests. However, there are several types of data which should not be cached:

- any data not from the root
- a possibly partial set of the RRs of the same type for a particular TLD name
- TLD data that does not pass the DNSSEC validation
- Any TLD data from the root data cache itself

The system should periodically checks to make sure that its root data cache are up to date. It MUST discard the cached root data if the data is outdated. The steps to run the Root Zone Analyzer are shown below:

step 1, Get the SOA value from the root name server query, set root data cache's version No. = serial of SOA; refresh timer value = expire value of SOA.

step 2, Get the full root zone, and do the same digest operation (such as MD5 [RFC 1321]) to the root zone data excluding the SOA and its RRSIG. Set the root data cache fingerprint = the digest value.

step 3, Check the SOA record from the root name server, and compare the serail of the SOA with the version No. of root data cache. If the value is same, wait for fixed time (the suggested time is the refresh value of SOA of the root zone or every 15 minutes) and go to step 3; If the value is not same, go to step 4.

step 4, Notify that the root data cache should stop response to the resolver temporarily and give a message to the resolver that no answer has been found for the question. Get the full root zone, and do the same digest operation (MD5 [RFC 1321]) to the whole root zone data excluding the SOA and its RRSIG. Compare the digest value with the fingerprint of root data cache. If the value is same, set root data cache's version No. = current serial of SOA; if the RRs of root SOA and its RRSIG are in the root data cache, they should be updated to the new one; notify that the root data cache should start response to the resolver immediately and go to step 3. If the value is not same, the system MUST discard all data in the root data cache, notify that the root data cache should start response to the resolver immediately and go to step 1.

8. IANA Considerations

This document requires no action from the IANA.

9. Security Considerations

A DNS cache may become poisoned when unauthorized RRs are inserted into it. A non-DNSSEC query may suffer from this problem in the same way as any resolver that does not do DNSSEC validation on responses from a remote root server. The resolver with a DNSSEC query will know how to deal with it.

DDOS attackers may aim to the root data cache. They may query random invalid root TLD. In our current design, the root data cache will not cache any negative answers from the root.

10. Change History

RFC Editor: Please remove this section.

10.1. draft-yao-dnsop-root-cache: Version 00

- Decreasing fetch time of Root Data by Improving the Mechanism of Root Data Cacheing

11. References

11.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <http://www.rfc-editor.org/info/rfc1321>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <http://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <http://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <http://www.rfc-editor.org/info/rfc4035>.

11.2. Informative References

[Root-loopback]
Kumari, W. and P. Hoffman, "Decreasing Access Time to Root Servers by Running One on Loopback", May 2015, <https://datatracker.ietf.org/doc/draft-ietf-dnsop-root-loopback/>.

Authors' Addresses

Jiankang Yao
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3007
Email: yaojk@cnnic.cn

Ning Kong
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3147
Email: nkong@cnnic.cn

Xiaodong Li
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3020
Email: xl@cnnic.cn