

A Whitepaper

Reducing RTT of DNS Query Resolution using RFC 8806

*Dr. Balaji Rajendran, Gopinath Palaniappan, Sanjay Adiwal,
Shubham Goyal, Bindhumadhava B S*

October 2020



Background

Domain Name System (DNS) is the backbone of the Internet. The DNS service has considerable stake in ensuring genuine and quick Internet availability. The Recursive Resolvers (RR) are an important component of the DNS infrastructure. Most of the DNS query resolutions happen at the RR (its cache) itself without the need to visit all the components in the DNS resolution hierarchy, while some happen after visiting all components of the DNS resolution hierarchy. Some RR servers face the issue of lengthier round-trip-time (RTT) to their closest DNS root server, which happens to be the first step for any DNS query resolution. In this document, we discuss on cutting down the RTT between the RR and its closest DNS root server by implementing RFC 8806. The introduction of RFC 8806 in June 2020 made RFC 7706 obsolete, and here we repeated our experiment to analyze the overall change in the time taken for DNS query resolution.

Approach

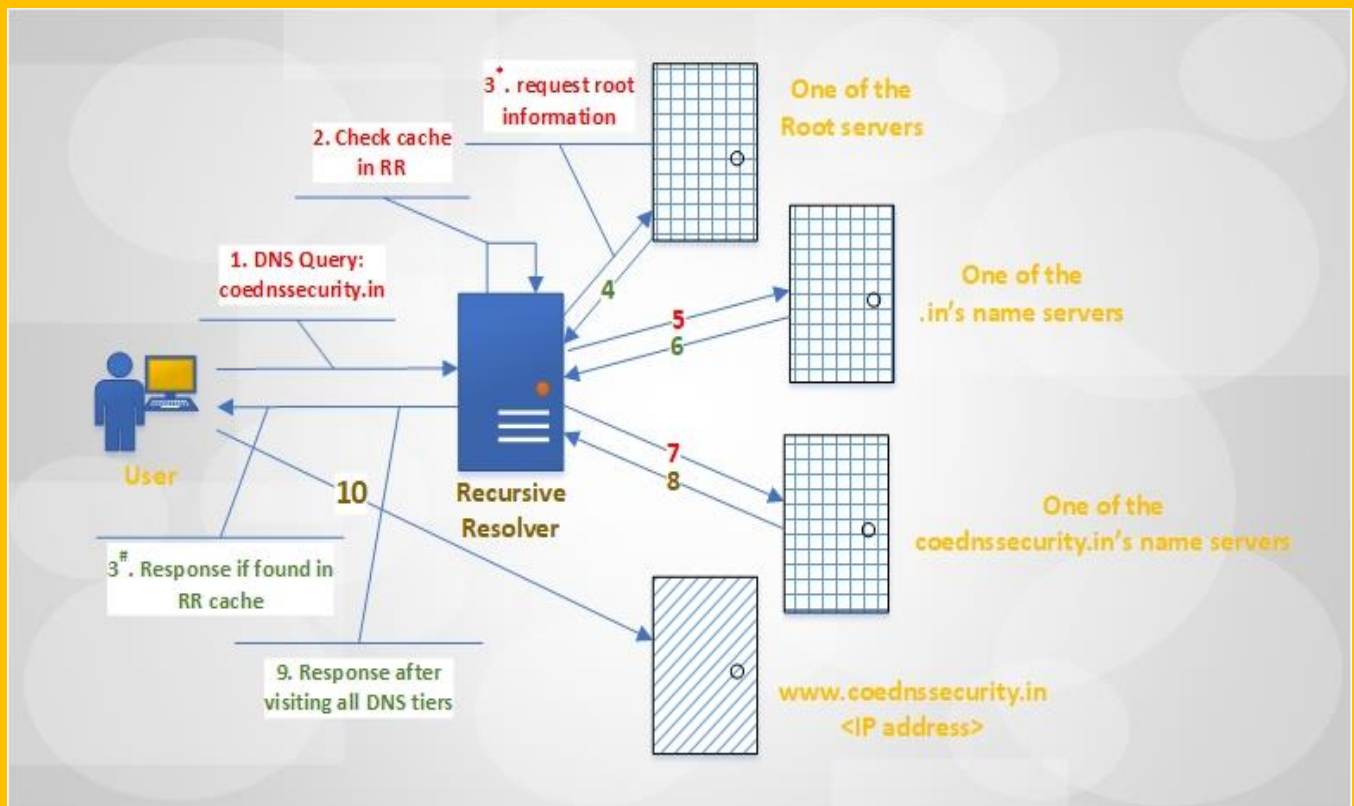


Figure.1: Conventional approach of DNS query resolution

In the **conventional approach**, the RR at the first place tries to resolve a DNS query by checking for the domain name in its own cache, if not found in its cache, it queries the closest root server, and follows it with subsequent queries to the TLD and SLD servers. This conventional approach is represented in Figure.1 above. The domain resolution happens along the steps 1, 2 & 3[#] [[#]resolved from RR cache] or 1, 2, 3^{*}, 4, 5, 6, 7, 8 & 9 [^{*}RR cache doesn't have the domain name and so proceeds for fetching data from root servers] as represented in Figure.1 above, depending on whether the RR server is able to resolve the DNS query from its cache or not.

“In the conventional approach, the RR server spends considerable time to reach out to the closest root server”

In our **proposed approach**, as per **RFC 8806** [1], we recommend creating an up-to-date root zone server as a local service on the same host as that of the RR server, and use that local service when the RR server has to look up for root information. The RR server would validate all responses received from the root server on the loopback address, just as it would do for all responses from a remote root server.

Our proposed approach is represented in Figure.2 below. In this proposed approach, the domain name resolution happens along the steps 1, 2 & 3[#] [[#]resolved from RR cache] or 1, 2, 3^{*}, 4, 5, 6, 7 & 8 [^{*}RR cache doesn't have the domain name and so fetches root data from its own loopback server], depending on whether the RR server is able to resolve the DNS query from its cache or from its loopback server containing root data. One can notice that the hosting of loopback server has reduced the number of steps from 9 to 8, meaning visiting the root servers to fetch root data relevant to the DNS query has been dropped.

“In the proposed approach, the loopback server containing root data is hosted on the RR server itself to avoid visiting the root servers for root data”

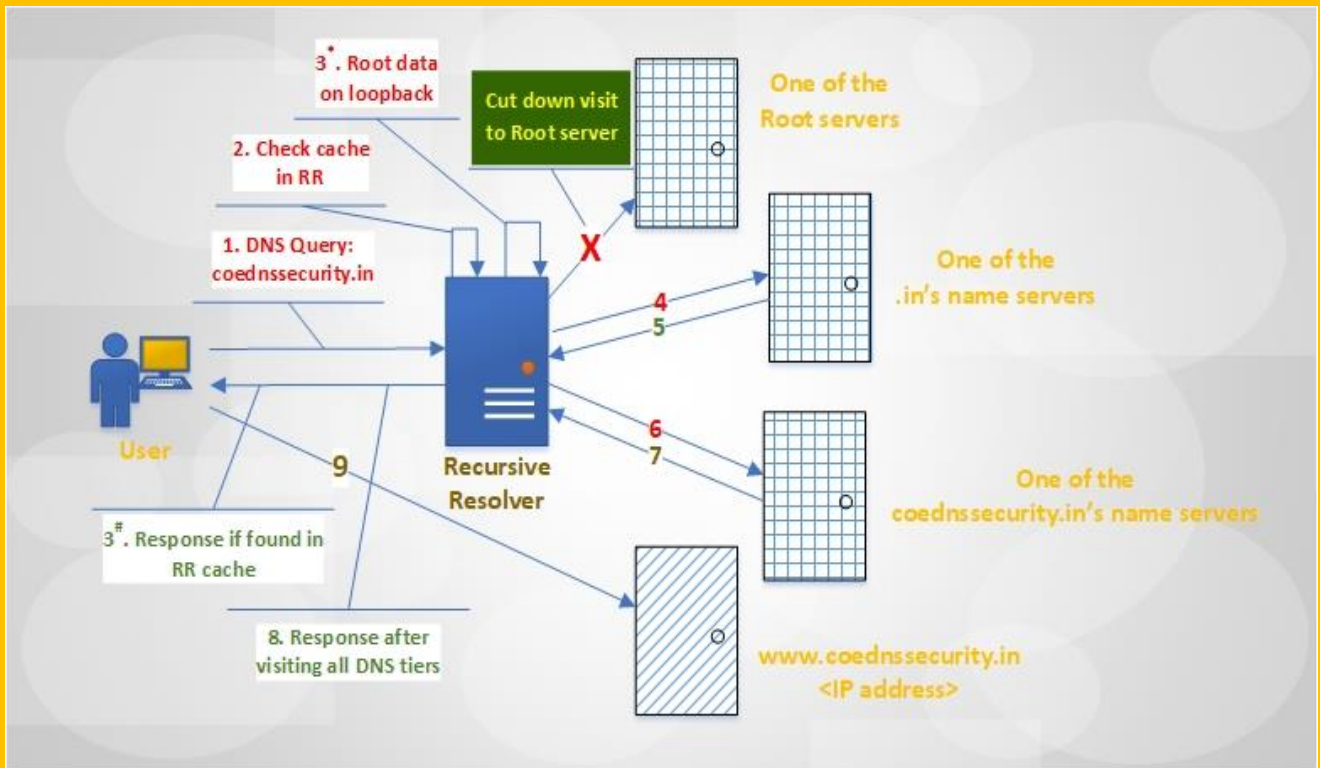


Figure.2: Proposed approach comprising of RFC 8806 implementation

Setting-up root zone at RR loopback

Pre-requisites:

- (a) The RR server must be able to run an authoritative server on one of its loopback addresses (127/8 in IPv4 and ::1 in IPv6).
- (b) The RR server must be able to retrieve a copy of the entire root zone including all DNSSEC-related records, and validate them.
- (c) The RR Server must have a software to do the configuration changes. We use Bind 9.16.6 [2][3][4] for implementation here.

Procedure:

- (a) Get the root zone data with all required DNSSEC records needed for validation. The root zone data should be obtained by AXFR over TCP from anyone of the following mentioned in the Table.1 below:

Root server operators	b.root-servers.net
	c.root-servers.net
	d.root-servers.net
	f.root-servers.net
	g.root-servers.net
	k.root-servers.net
DNS servers from ICANN	xfr.lax.dns.icann.org
	xfr.cjr.dns.icann.org

Table.1: Root Zone Data

```

view root {
    match-destinations { 127.0.0.1; };
    zone "." {
        type mirror;
        file "rootzone.db";
        notify no;
        masters {
            199.9.14.201;           # b.root-servers.net
            192.33.4.12;           # c.rot-servers.net
            199.7.91.13;           # d.root-servers.net
            192.5.5.241;           # f.root-servers.net
            192.112.36.4;          # g.root-servers.net
            193.0.14.129;          # k.root-servers.net
            192.0.47.132;          # xfr.cjr.dns.icann.org
            192.0.32.132;          # xfr.lax.dns.icann.org
            2001:500:200::b;        # b.root-servers.net
            2001:500:2::c;         # c.root-servers.net
            2001:500:2d::d;        # d.root-servers.net
            2001:500:2f::f;        # f.root-servers.net
            2001:500:12::d0d;       # g.root-servers.net
            2001:7fd::1;           # k.root-servers.net
            2620:0:2830:202::132;  # xfr.c jr.dns.icann.org
            2620:0:2d0:202::132;   # xfr.lax.dns.icann.org
        };
    };
};

```

- (b) To configure the Recursive Resolver (RR) on Bind 9.16.6 [2] for implementing a root zone on its loopback address, we need to list the following in its configuration file:

```
view recursive {  
    dnssec-validation auto;  
    allow-recursion { any; };  
    recursion yes;  
    zone "." {  
        type static-stub;  
        server-addresses { 127.0.0.1; };  
    };  
};
```

Our Set-up

We have set-up an **Open, Public Recursive Resolver** for both IPv4 and IPv6 traffic and made it available for Internet users Worldwide, which is configured on Bind 9.16.6 [2][3][4] with RFC 8806 implemented in it and hosted at IPv4 address: **223.31.121.171** and IPv6 address: **2405:8a00:8001::20**.

Our experimental set-up comprised of a dedicated terminal with a public IP address in order to query the RR. We followed the procedure as mentioned below, first with RFC 8806 and next without RFC 8806 on the RR:

- We clear the RR cache before querying.
- We downloaded 100 distinct domain names from IANA, and created 5 files of 20 domains in each.
- We implemented a linux shell script which accepts an input of 20 domain names and queries the RR with them and writes the RTT on to a file upon receiving the response.



Figure.3: Procedure followed in the experiment

Analysis of RTT with and without RFC 8806 implementation

The results of our experiment are listed in the table below and also represented as a graph in Figure.3. Our experiment shows that the RTT for domain name resolution was clearly lesser in the RR with the RFC 8806 implementation.

Description	Time in milliseconds	
	with RFC 8806	without RFC 8806
Minimum RTT	96	127
Maximum RTT	3924	4748
Average RTT	1207.58	1781.32

Table.2: RTT for domain resolution with and without RFC 8806

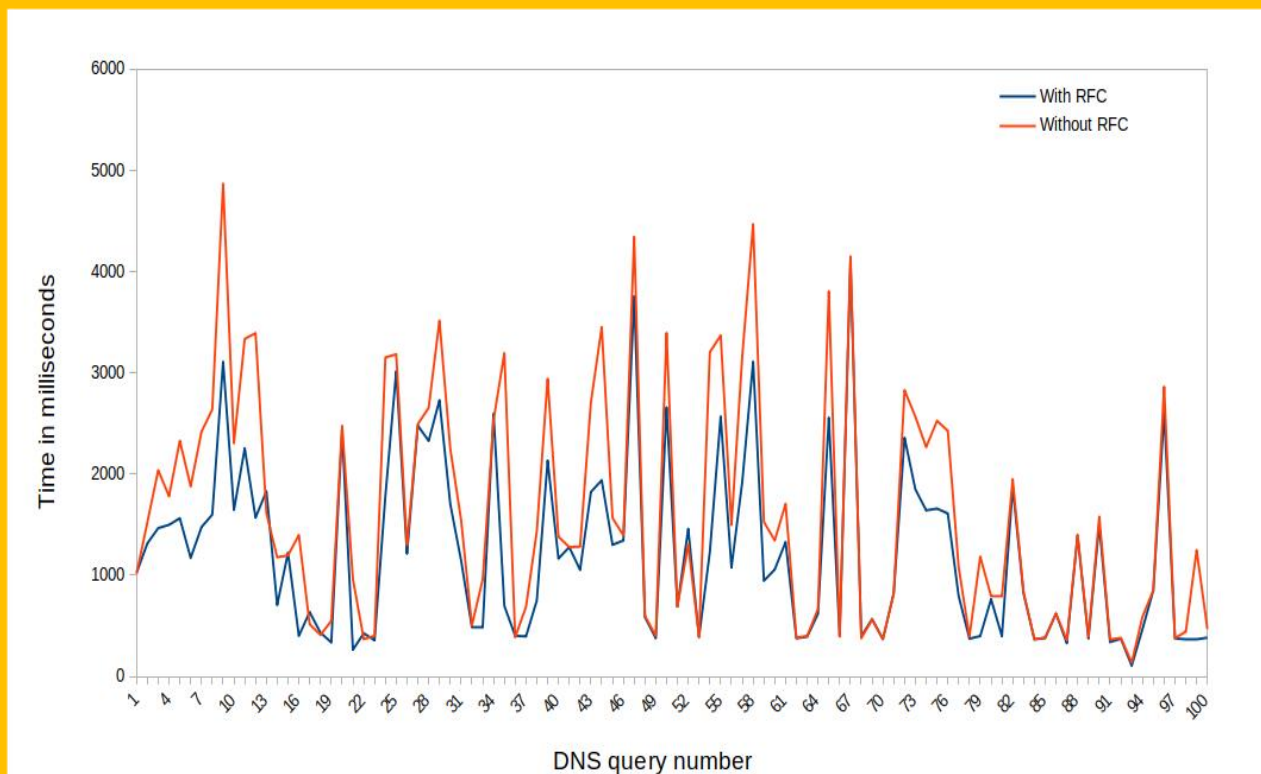


Figure.4: Comparison of domain resolution RTT of with and without RFC 8806

Verifying the implementation of RFC 8806

Having configured the resolver for RFC 8806, the same can be verified using a tool, “dig”, Domain Information Groper. We executed the “dig” command on the terminal with option “+trace” and we could witness that the resolver with RFC 8806 return the result (Figure.5) from the resolver itself whereas if there were no implementation of RFC 8806 then the resolver visits (Figure.6) the complete DNS hierarchy for the query resolution.

```
# dig @223.31.121.171 coednssecurity.in +trace

; <<>> DiG 9.16.6 <<>> @223.31.121.171 coednssecurity.in +trace
; (1 server found)
;; global options: +cmd
;; Received 28 bytes from 223.31.121.171#53 (223.31.121.171) in 1 ms
```

Figure.5: Working of “dig @223.31.121.171 coednssecurity.in +trace” with RFC 8806 implemented on the Recursive Resolver

```
# dig @223.31.121.171 coednssecurity.in +trace

; <<>> DiG 9.16.6 <<>> @223.31.121.171 coednssecurity.in +trace
; (1 server found)
;; global options: +cmd
.          43196 IN      NS      g.root-servers.net.
.          43196 IN      NS      j.root-servers.net.
.          43196 IN      NS      e.root-servers.net.
.          43196 IN      NS      l.root-servers.net.
.          43196 IN      NS      b.root-servers.net.
.          43196 IN      NS      f.root-servers.net.
.          43196 IN      NS      k.root-servers.net.
.          43196 IN      NS      c.root-servers.net.
.          43196 IN      NS      d.root-servers.net.
.          43196 IN      NS      h.root-servers.net.
.          43196 IN      NS      m.root-servers.net.
.          43196 IN      NS      i.root-servers.net.
.          43196 IN      NS      a.root-servers.net.
.          43196 IN      RRSIG  NS 0 518400 20201031170000 202
01018160000 26116 . JYU3zWLUSgQkXWuNH8AJgxeOEgcOfMzq6wI70MSax5aoQlgdolTKiKB2 2r2
40P59wKWZQZSk1/ULY2oCq16Um3/4bUoOoy0sWweA2g1Qa4RVD3 vs5R6dqkU1QAc20APDFsQ02F15
K0NhnA1IsFtWcXVGFpafRGI8V1TZM ldsGWf/m8r7kzD0ZsGt/hFc178KAQXVHoqVQcZsIs5xzaFLAG
90oR0y6 Lf+Z6Sw1HcoYHawKB2bJKPFuhVUw6wul4g1K0BvTs3oCkXzQbJLL5JRv p4hKosXGypfyLQ
MSvSbjKtp1RT613REfqXfH6WFGYnRnHnShs6itb XnsVeA==
;; Received 537 bytes from 223.31.121.171#53 (223.31.121.171) in 1 ms

in.        172800 IN     NS       ns3.registry.in.
in.        172800 IN     NS       ns1.registry.in.
in.        172800 IN     NS       ns5.registry.in.
in.        172800 IN     NS       ns6.registry.in.
in.        172800 IN     NS       ns2.registry.in.
in.        172800 IN     NS       ns4.registry.in.
in.        86400 IN     DS       54739 8 1 2B5CA455A0E65769FF9DF9
E75EC40EE1EC1CDA9
in.        86400 IN     DS       54739 8 2 9F122CFD6604AE6DEDA0FE
09F27BE340A318F06AFAC1174873409D4 3136472C
in.        86400 IN     RRSIG  DS 0 1 86400 20201031170000 2020
1018160000 26116 . Bd2wEgp3CQ22Tc3xfX38CbCWU1jd6WA/amgo3ghle3j17d45jDo1324 juBA
LmCnxI3Zjz2Ro01bRlpTE6zdM3ewCQpWG2f4gPy7sOW1EHaAGo tLo8Z0eW2uD2/Rq1iWuXStpu57n
Oz00nXagEPf+wb0y45UnffPB0rk8cM joYGYow7lr/go9VLUrDhWAgN91jdLy9P1fVnk8LL8/XVL282
r0/CmI yyI9wFdtKwpiTdkKfpo4MT4C3s+B455SRBdph1JHSQSj7WuJkdDF/N3PP sgFincij42WRXqdt
Sr7V7FXzPnH267af7vTg0IINuFwznBxYBh4NgCY C5kXNg==
;; Received 800 bytes from 193.0.14.129#53 (k.root-servers.net) in 152 ms

coednssecurity.in. 86400 IN      NS      dns3.bigrock.in.
coednssecurity.in. 86400 IN      NS      dns4.bigrock.in.
coednssecurity.in. 86400 IN      NS      dns2.bigrock.in.
coednssecurity.in. 86400 IN      NS      dns1.bigrock.in.
bo80loouciino3vfr381rljcrvZucohi.in. 1800 IN NSEC3 1 1 1 00763C64 BOB9S4EEBHFL1V
OTNFEINC9H1UNMULB3 NS SOA RRSIG DNSKEY NSEC3PARAM
bo80loouciino3vfr381rljcrvZucohi.in. 1800 IN RRSIG NSEC3 8 2 1800 20201115131525
20201016124147 65169 in. Wz2bOWkyyhVMp9ASEW8yLJsPGzq+cLSWISS8+Uczagy3ZQ6S3yLCOY
cv xB8bnzD5gJnDLu1lRttIWMED84+spsACuFTQ1/PgyKlaywe6EvZ9+4So EW0E1YPNz8DjVvjrgGMY
FV4a+agaU1RwXN5fWe9Q1V18JELFdJJaUUU1 Jly4B38w33upUZcmG8hNJJPLTJmVuvEgMLBPLTG/sy
wFA==
8vle49nu2a12f914f0sc1flg7cfcp56.in. 1800 IN NSEC3 1 1 1 00763C64 8VMDKND9TRIV7F
DE8030JG982LGFH2L NS DS RRSIG
8vle49nu2a12f914f0sc1flg7cfcp56.in. 1800 IN RRSIG NSEC3 8 2 1800 20201114222640
20201015222542 65169 in. VQMoALB7/dWqf93x2KHw2d5ot5Ta81jQhXU3Sg80ue19itK+dXISOS
nF rMmWZM7Ib8K4qB/jGuOdj6Ldlb9/cHHLJC1AfAkppS3bLm8L4rt+26K vGyPp61kEG2tLAWKSH
A6MLy/fad4sC9aNA5XOUrW17MhR46d6m36R NV+0TmI9yB34Sey1Mw17Wamc8Iy3BNRCzKAaeBHZ7oz
3Kw==
;; Received 713 bytes from 156.154.100.20#53 (ns5.registry.in) in 32 ms

coednssecurity.in. 28800 IN      A       220.156.189.66
;; Received 62 bytes from 162.251.82.119#53 (dns3.bigrock.in) in 251 ms
```

Figure.6: Working of “dig @223.31.121.171 coednssecurity.in +trace” without RFC 8806 implemented on the Recursive Resolver

Conclusion

Using our set-up of open-public RR and querying it from terminal with public IP address helped us get genuine RTT for domain name resolution. By comparing the RTT obtained by querying the RR (*with and without RFC 8806*) with a dataset of 100 domain names, we can clearly understand that implementing RFC 8806, to run an instance of the root zone data in the loopback of RR, saves time, improves throughput and beneficial to the Internet ecosystem.

References:

- (1) RFC 8806: <https://tools.ietf.org/html/rfc8806>
- (2) Procedure for configuring DNS Bind server on CentOS 7:
<https://www.itzgeek.com/how-tos/linux/centos-how-tos/configure-dns-bind-server-on-centos-7-rhel-7.html>
- (3) Download Bind 9.16.6 from: <https://coednssecurity.in/pdf/bind-9.16.6.tar.xz>
- (4) Deploying a RR using Bind 9.16.6: <https://coednssecurity.in/pdf/DNS-Bind-Server-Installation-Configuration.pdf>

Acknowledgements:

We express our sincere thanks to Internet Governance Division of [Ministry of Electronics & Information Technology \(MeitY\)](#) and [National Internet Exchange of India \(NIXI\)](#).

Cite as: Dr. Balaji Rajendran, Gopinath Palaniappan, Sanjay Adiwai, Shubham Goyal, Bindhumadhava B S (2020, October). Reducing RTT of DNS Query Resolution using RFC 8806. DOI: 10.17605/osf.io/t9n5x