# Using DNAME for reverses of inter-RIRs transferred resources

*Design document*

*Author: Hugo Salgado <hsalgado@nic.cl> / NIC Chile*

*Version 0.7 / October 7, 2021*

Table of Contents:

# Introduction

The "Regional Internet Registries" (RIRs) are the organizations in charge of manage and delegate resources of IP numbers (IPv4 and IPv6) and Autonomous Numbers (ASN) in all the world. There are 5 in total, each in charge of a particular region. In the case of Latin America, the RIR is LACNIC, installed in Montevideo, Uruguay; and that's how it is too we have APNIC in Asia Pacific, AFRINIC in Africa, RIPE in Europe and ARIN in North America.

Within the tasks of each RIR there is to maintain the DNS sub-tree of the reverse of the IP addresses that are delegated to an end user. It is as if, for example, NIC Chile receives the IPv4 prefix 200.7.7.0/24 from LACNIC, its reverse names must be kept under 7.7.200.in-addr.arpa, which is a child of the parent zone 200.in-addr.arpa, administered by LACNIC. And so each assignee of an IP prefix in LACNIC can request the delegation of their segment.

To do this, LACNIC has a control panel where each organization can declare its name servers (NS), and thus obtain the delegation in the DNS.

## The problem with inter-RIR transfers

So far so good, but what happens when an organization registered in LACNIC sub-delegates in turn a chunk of its assignment to an organization that wants to register with another RIR? This is what is called "inter-RIR transfer". It occurs when, for example, an organization in European Union, which has a shortage of IPv4 addresses, reaches an agreement with some organization in Latinamerica that had free addresses (with no use), and agree to transfer a segment. In this case, both the entity that transfers a segment, as well as the recipient, go to LACNIC and RIPE to record the transfer, and update whois data, geolocation, and especially administration of the reverse of the segment, which from this transfer will appear in the user panel in RIPE of the new organisation, and will no longer appear in the LACNIC user panel.

However, one problem that arises is how to just delegate correctly the reverse of the resource in DNS.

In a case like the one in the example, the new assignee will define in RIPE the NSs for what he wants to delegate the segment, but that DNS sub-tree does not belong to RIPE but to LACNIC, so some kind of coordination is necessary to communicate the data.

## Zonelets

The mechanism that was defined in these cases among all the RIRs was the use of "zonelets", which are chunks of DNS configuration that each RIR communicates to the RIR corresponding to the reverse delegation of the resource, through an automated mechanism that shares daily this information.

Returning to the example, when this organization defines its NSs in RIPE, it is RIPE that build a "zonelet" with this data, put it in a private repository shared between the different RIRs, and it is

LACNIC that collects this piece of configuration and places it on the correspondent reverse tree. In this way, the DNS query that starts at in-addr.arpa, is then delegated to the prefix under LACNIC's control, and going down the tree there comes a time when it is delegated finally to the NS of the end customer.

## The problem with zonelets

This mechanism has worked well for years, despite some sporadic problems that have been due to lack of maintenance of systems and verification bugs, systems that have been improved over time. However, the system suffers from a problem more intrinsic to the solution, which is the time it takes for a change to be carried over to DNS. A mechanism of the style of the zonelets requires batch processing and checks that are not necessarily the same fast enough.

On the other hand, in the current reality of IPv4 depletion it is expected that inter-RIR transfers become more and more frequent, subjecting the system to greater loads.

This is why our proposal is to modify the re-delegation mechanism with a relatively new technique in DNS that makes the process much simpler, leaving everything within the normal DNS protocol: the use of DNAMEs.

# Project description

## What is DNAME

DNAME is a DNS extension technique originally defined in 1999, but updated in RFC6672 of the year 2012, which defines a registry that allows delegating a sub-whole tree to another DNS node. The name refers to the famous CNAME record (which means "canonical name"), which makes an alias but for an "end node" of the DNS tree. DNAME does it for an entire branch.

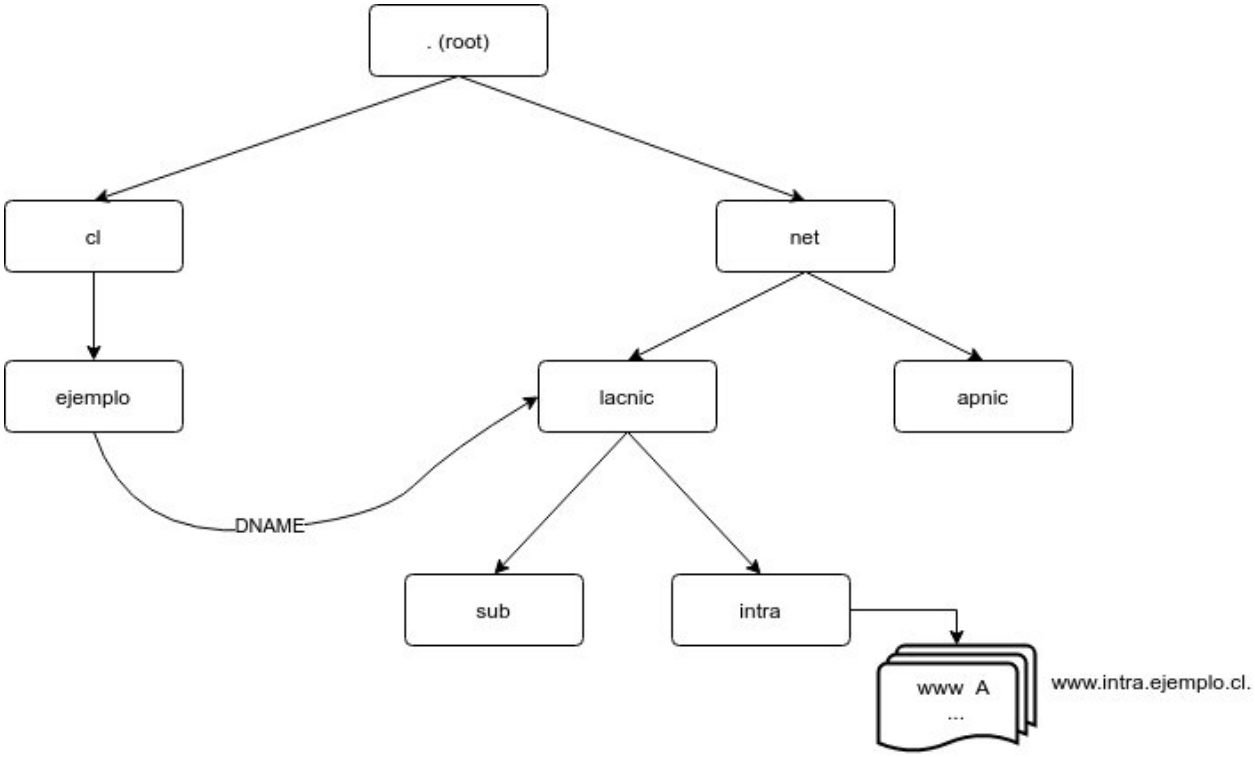


Figure 1, DNAME record use

## How to use it for reverses

Thus, returning to our case, the only thing that LACNIC should do at the moment to assign one of its resources via transfer to another RIR, is to define a DNAME record pointing to a new subtree in the target RIR, which in turn can redefine NS for the inferior names.

This new subtree should be a space under the control of the destination RIR, which is agreed previously. We will use as an example **in-addr.transfer.<RIR SPACE>**.

In this way any change of the NS of the controlling body of the prefix makes it directly in RIPE, which in turn modifies it under the tree in-addr.transfer.ripe.net directly under its control, without further involving the former RIR LACNIC.

The end customer must also take care to change the name of his end zone to respect the new subtree under control of its RIR, and not directly in in-addr.arpa as it is normal.

The changes in the procedure would be:

1. a RIR that gives up a resource must add a DNAME in place of the zone-cut, pointing to a new name that is under the control of the obtaining RIR;
2. the obtaining RIR puts in this new name the NS of the end customer,
3. the end customer becomes authoritative of that new name within the RIR space obtaining.

In the case of needing to “cancel” a transfer, simply remove the DNAME and return to the original normal delegation, with the original customer's NS.

In the case of “deep” delegations the procedure is the same. If a parent delegates directly to a grandchild, without referring to the child (empty-non-terminal); it is in this same point that the NS are replaced by the DNAME to the external tree.

# Example of use with DNAME

Let us suppose that the organization "Ejemplo Sociedad Anónima" domiciled in Latin America, has the resource 200.7.24.0/23 delegated by LACNIC. In this case, prior to transfer, we have that LACNIC defines the following records in its subtree:

```
24.7.200.in-addr.arpa.      NS   ns1.ejemplo.lat.
                            NS   ns2.ejemplo.lat.

25.7.200.in-addr.arpa.      NS   ns1.ejemplo.lat.
                            NS   ns2.ejemplo.lat.
```

If “Ejemplo SA” decides to transfer one of its subsegments, the 200.7.25.0/24 to the organization "Exemple Société" under the RIPE RIR, LACNIC must define a DNAME only once for this reverse, signaling the delegation to a sub-tree under RIPE control, which does not never change again:

```
25.7.200.in-addr.arpa       DNAME  25.7.200.in-addr.transfer.ripe.net.
```

And in turn RIPE defines the NS of the end customer there:

```
25.7.200.in-addr.transfer.ripe.net.   NS   ns1.exemple.eu.
                                      NS   ns2.exemple.eu.
```

In order for “Exemple Société” to finally define its reversals of final IPs, taking care to define correctly the sub-tree:

1.25.7.200.in-addr.transfer.ripe.net.  PTR  gateway.exemple.eu.

The following diagrams explain the initial situation, with the resource originally delegated in LACNIC, and RIPE with its sub-tree for receiving transfers:

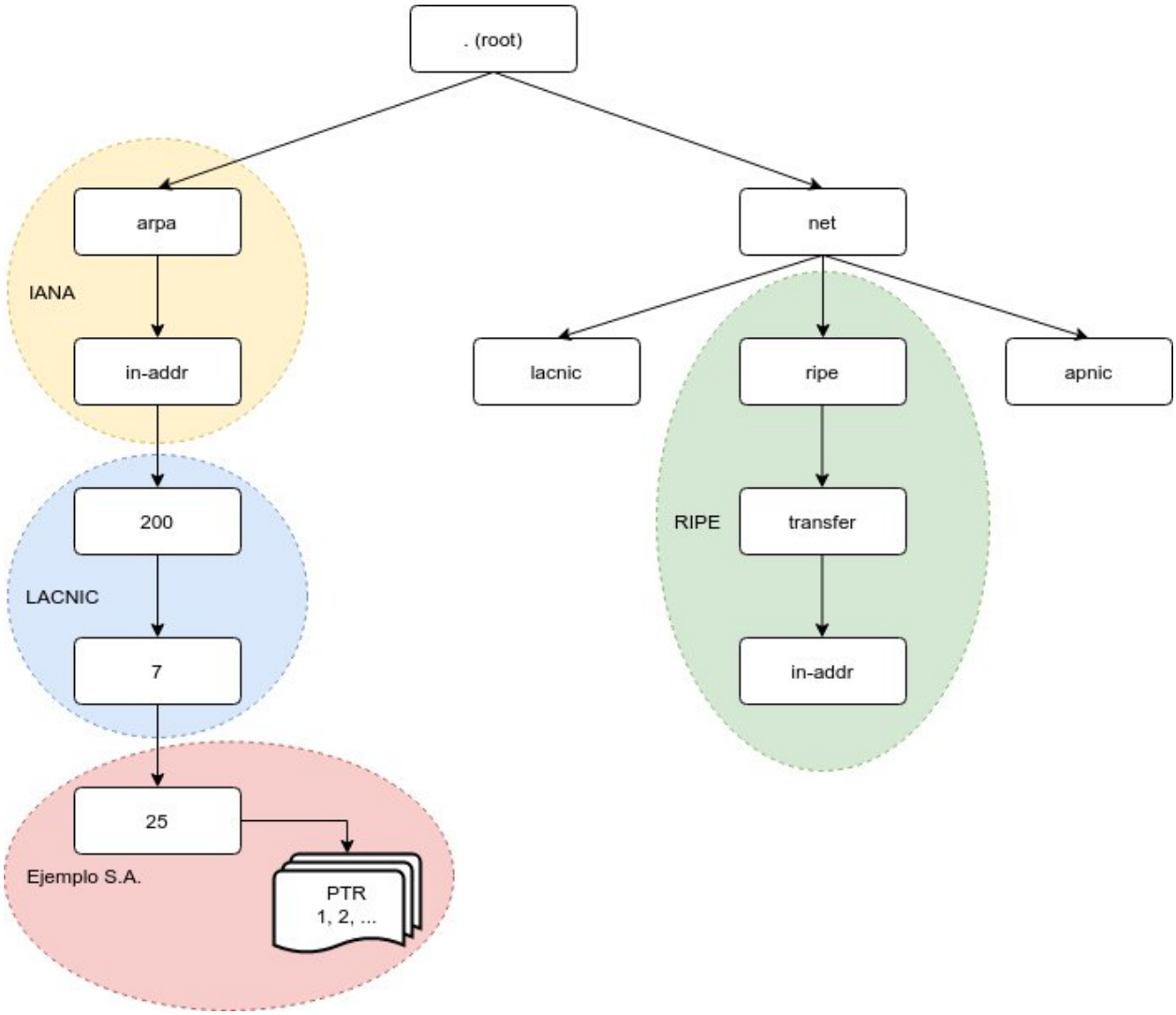


Figure 2, example of a normal delegation

And the situation after the transfer, where RIPE expands its sub-tree, LACNIC change the delegation of the resource by a DNAME, and the new organization taking control:

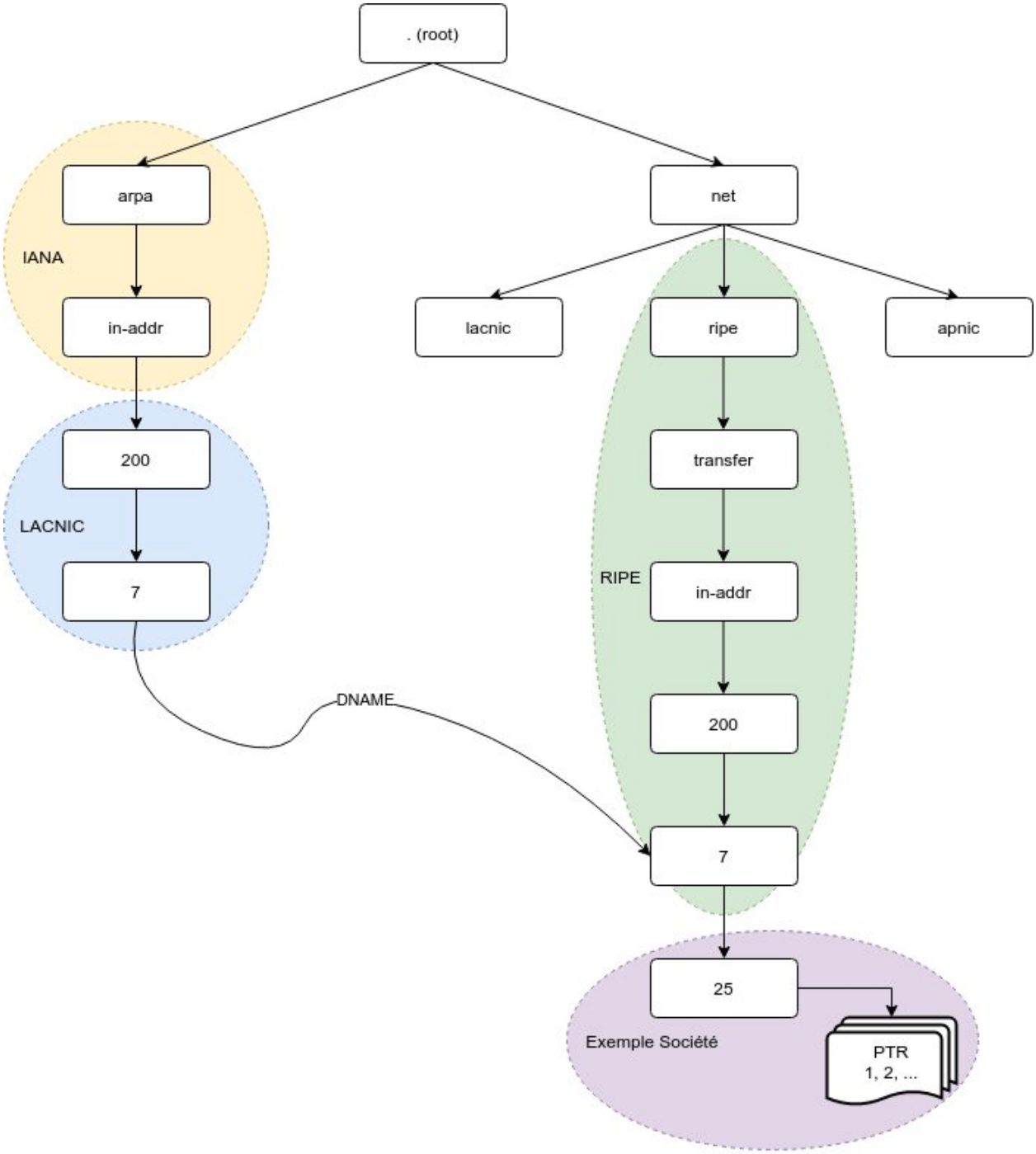


Figure 3, example of a DNAME delegation

If we now carry out a query for the reverse, we have the following result:

```
 $ dig 1.25.7.200.in-addr.arpa ptr
[ ... ]
;; ANSWER SECTION:

25.7.200.in-addr.arpa.                  DNAME   25.7.200.in-addr.transfer.ripe.net.
1.25.7.200.in-addr.arpa.                CNAME   1.25.7.200.in-addr.transfer.ripe.net.
1.25.7.200.in-addr.transfer.ripe.net. PTR     gateway.exemple.eu.
```

which correctly synthesizes the final answer and automatically delivers a CNAME in case of resolvers that do not know how to interpret the DNAME.

This mechanism even allows a design that allows delegating in sub-delegations beyond current "octet" limits. At the moment the minimum prefix that can be transferred between RIRs it is a /24, which matches just the last octet of the IP address, but perhaps this requirement could be relaxed in the future and it will be possible to transfer (and route) in larger prefix lengths, that could be handled in reverse such as:

1.0.1.1.1.0.1.25.7.200.sub24.in-addr.transfer.ripe.net

which would be a delegation from 200.7.25.186/31.

The namespace of each RIR could also be derived from .arpa, if that is considers it to give it greater stability, ease of configuration, and "relevance". For example, IANA could delegate to each RIR a space under in-addr.arpa of the style:

afrinic.in-addr.arpa
arin.in-addr.arpa
apnic.in-addr.arpa
lacnic.in-addr.arpa
ripe.in-addr.arpa

and that way in practice the client setup only consists of adding 1 extra label to the reverse zone:

25.7.200.ripe.in-addr.arpa

# Experiment and measurements

A structure was assembled in a controlled environment but in real production, simulating an architecture of delegations, all under lab.nic.cl .

For this, the following domain structure was established:

| Organisation / Actor | Domain / Zone | Name servers |
|---|---|---|
| Root / IANA | arpa.lab.nic.cl<br>in-addr.arpa.lab.nic.cl | ns1.root.net.lab.nic.cl<br>ns2.root.net.lab.nic.cl |
| LACNIC | 200.in-addr.arpa.lab.nic.cl<br>7.200.in-addr.arpa.lab.nic.cl | ns1.lacnic.lab.nic.cl<br>ns2.lacnic.lab.nic.cl |
| Ejemplo S.A. (former customer delegation) | 24.7.200.in-addr.arpa.lab.nic.cl | ns1.ejemplo.lab.nic.cl<br>ns2.ejemplo.lab.nic.cl |
| RIPE, transfer space | ripe.in-addr.arpa.lab.nic.cl<br>200.ripe.in-addr.arpa.lab.nic.cl<br>7.200.ripe.in-addr.arpa.lab.nic.cl | ns1.ripe.lab.nic.cl<br>ns2.ripe.lab.nic.cl |
| Exemple Soc. (new customer delegation) | 24.7.200.ripe.in-addr.arpa.lab.nic.cl | ns1.exemple.lab.nic.cl<br>ns2.exemple.lab.nic.cl |

Each of the nameservers (ns1, ns2) were served by separate authoritative instances, to more accurately simulate the separation of zones and authority space of each one (avoiding delegations that cover more than the labels that are the responsibility of each organization), each with an independent IPv4 and IPv6 addresses.

The names of the Name Servers are put directly in the meta-root (lab.nic.cl), because they are not part of the experiment.

A base situation was established, prior to the transfer, with the original delegation normal to “Ejemplo SA”. On this basis the first public measurements were made, using different platforms.

Then the redelegation was carried out through the DNAME registry to “Exemple Sociètè”. Measurements where carried out at the time of delegation, to be certain of the expected downtime.

Once the redelegation was carried out, new measurements were made from platforms as diverse as possible, to ensure the correct interpretation of the DNAME, of the synthesized CNAMEs, and taking into account issues such as TTLs, intermediate caches (forwarders) and various responses (SERVFAIL, NXDOMAIN, NODATA).

# Results

## Support in public open resolvers

The following open resolvers that gives public service (also called quad-N known resolvers) were reviewed for the support for this solution:

| Resolver | IPs (v4 only) | Query for a PTR with DNAME delegation |
|---|---|---|
| Verisign | 64.6.64.6 64.6.65.6 | OK |
| Google Public DNS | 8.8.8.8 8.8.4.4 | OK |
| Cloudflare | 1.1.1.1 1.0.0.1 | OK |
| Comodo Secure DNS | 8.26.56.26 8.20.247.20 | OK |
| Norton ConnectSafe | 199.85.126.10 199.85.127.10 | OK |
| SafeDNS | 195.46.39.39 195.46.39.40 | OK |
| Dyn | 216.146.35.35 216.146.36.36 | OK |
| UncensoredDNS | 89.233.43.71 | OK |
| puntCAT | 109.69.8.51 | OK |
| CNNIC SDNS | 1.2.4.8 210.2.4.8 | OK |
| AliDNS | 223.5.5.5 223.6.6.6 | OK |
| OneDNS | 117.50.11.11 117.50.22.22 | OK |
| OpenDNS | 208.67.222.222 208.67.220.220 | OK |
| Level 3 | 209.244.0.3 209.244.0.4 | OK |
| Quad9 | 9.9.9.9 149.112.112.112 | OK |
| DNS.WATCH | 84.200.69.80 84.200.70.40 | OK |
| OpenNIC | 185.121.177.177 | OK |
| Freenom World | 80.80.80.80 80.80.81.81 | OK |
| FreeDNS | 37.235.1.177 | OK |
| Yandex.DNS | 77.88.8.8 77.88.8.1 | OK |
| Hurricane Electric | 74.82.42.42 | OK |
| Neustar | 156.154.70.1 156.154.71.1 | OK |
| Baidu Public DNS | 180.76.76.76 | OK |
| 114DNS | 114.114.114.114 114.114.115.115 | OK |
| DNSpai | 101.226.4.6 218.30.118.6 | OK |
| CleanBrowsing | 185.228.168.9 185.228.169.9 | OK |
| AdGuard DNS | 94.140.14.14 94.140.15.15 | OK |
| CIRA Canadian Shield Family | 149.112.121.30 149.112.122.30 | OK |

## Using open resolvers in-the-wild

We used a list of open recursive resolvers from the "Public DNS Server List ”(https://public-dns.info/), obtained by scanning the address space IPv4. It is important to note that these open resolvers do not deliver an official public service of DNS, so many times they are broken and misconfigured machines, which causes a high error rate.

A list of 979 IPv4 addresses in total was obtained, of which 329 (33.6%) effectively answered DNS queries.

Of these, 286 (86.9%) responded correctly with NOERROR status, and the answer it was as expected, following the DNAME delegation (nofwd.exemple.lab.nic.cl). 28 IP addresses (8.5%) delivered SERVFAIL results, and the remaining 4.6% delivered other types of errors (FORMERR, REFUSED, NXDOMAIN).

A detailed analysis of the errors resulted in different problems:

- erratic behaviors, giving a response 1 out of every 3 or every 5 queries. Those resolvers are suspected to be behind balancers against a backend farm, and some have problems.
- don't do the full recursion. Some delivered the synthesized CNAME, without following the target. When querying the explicit target, it did return the complete response.
- they do not respond when activating EDNS. Disabling EDNS does get results.
- REFUSED answers intermittency
- interception of results (block lists), throwing NXDOMAIN.

Of all the cases, there was zero occasion where a correct answer was obtained with the control domain (normal PTR) and SERVFAIL with the domain under DNAME; which is the behavior that one would expect from a resolver who is not able to understand the technique.

## RIPE Atlas probes

A permanent measurement was left for two weeks on 500 RIPE Atlas probes in all over the world (see Appendix 1 for a distribution illustration).

Each of these probes performed a DNS query every 3 hours for 1.24.7.200.in-addr.arpa.lab.nic.cl/IN/PTR directed to the particular external resolver of each network (obtained by DHCP or manual guest configuration).

Initially, this name was under a normal delegation representing a situation prior to a transfer. The PTR pointed to the name “nofwd.example.lab.nic.cl”. Subsequently, the delegation was modified simulating a transfer to another RIR, by means of a DNAME record in parent zone 7.200.in-addr.arpa.lab.nic.cl. The destination name corresponded to 1.24.7.200. ripe .in-addr.arpa.lab.nic.cl, with rdata "nofwd.exemple.lab.nic.cl".

After this change, the results of each measurement were analyzed, comparing the behavior in general. Atlas probes show considerable inconsistency and variability in measurements, representing the realities of the different scenarios where they are hosted (intermittent home networks, connectivity and power, behind CPEs with varied behavior, and stubs / resolvers with filters); so the objective was to analyze trends to big scale.

With these first results, RCODEs were analyzed comparing the rate between NOERROR and SERVFAIL, which is the code to expect if there were problems with DNAME delegation. In the Initial situation, this average value was 2.53% failures. When passing to the DNAME scheme, this value doubled to an average of 5.14%. Analyzing case by case, it was seen that there was a lot of failure due to the choice of a low TTL (1 hour) which allowed flexibility to test changes, but that affected the increase in the chain of resolutions to reach the final result.

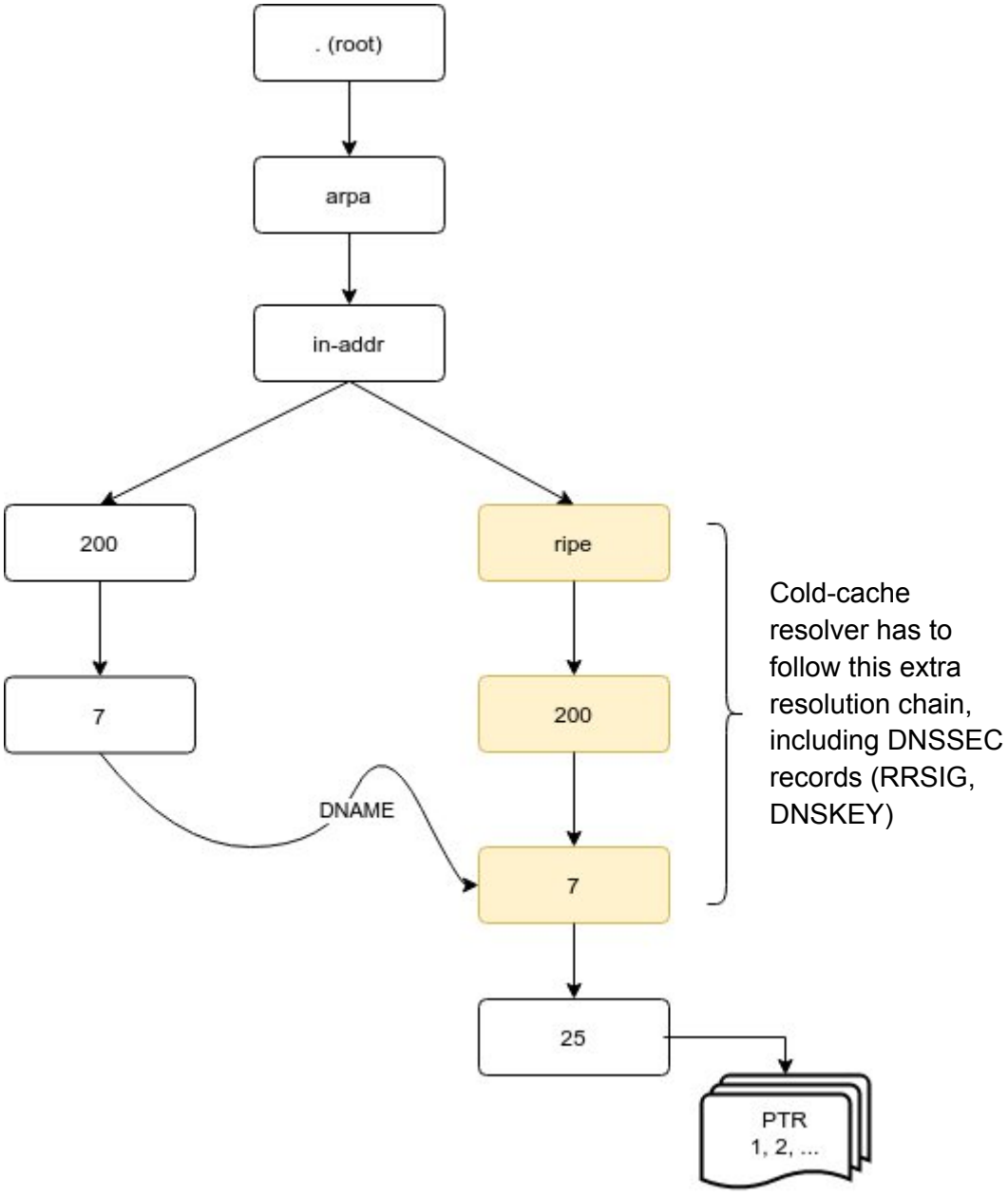


Figure 4, extra lookups using DNAME

Indeed, compared to the case of a directly delegated PTR record, the case DNAME could have more than 9 extra resolutions in the case of signed registration, in an empty cache.

Therefore, the testing environment was modified by increasing the TTL of all records to 1 day, in such a way as to take advantage of the caches in the probes for a longer time (the measurements are repeated every 3 hours).

On the other hand, analyzing other failures in detail, quite a few cases of resolvers that returned NODATA for any PTR query (rcode NOERROR), including the original situation. For this reason, it was decided in a second analysis not to trust the RCODEs but also analyze the ANSWER, which should contain a valid value.

Finally, it was identified that the Unbound software delivers TTL 0 for the CNAME synthesized from a DNAME, even though the authoritative upstream responds correctly with CNAME TTL equal to DNAME. This behavior does not seem to go against the standard (in RFC .... only the authoritative is mentioned) and it could be considered that it allows a quick reaction against changes (TTL zero allows its use only 1 time, without putting it in cache), but although not we have evidence that it may be a cause of failures, it could cause problems in stubs or forwarders with suboptimal behavior.

After these adjustments, ultimately an average of 4.12 % failure was reached in the case direct delegation, and 4.52 % using DNAME (measuring equivalent days and hours, after stabilized change).

Again analyzing the detail case by case, ruling out cases of intermittency expected in each probe, at least one probe was detected (id 52315) that was effectively encounters a resolver that stops working with the new schema. This resolver responds correctly to a query by PTR in the case of normal delegation, but gives SERVFAIL in the case of DNAME. This probe underwent further analysis, detecting that it resolved correctly normal PTR records and CNAMEs, but fails in cases of DNAME to PTR types and even A. Apparently it is the presence of a DNAME record that makes it fail. Of all forms, it was not possible to obtain more information on the type of resolving or other characteristics of the particular network that could cause it to fail (firewalls, traffic inspectors).

## Open source resolvers support

ISC Bind has supported DNAME since at least 2008. Knot Resolver since its first version in 2016. Unbound at least since 2007. PowerDNS Recursor since May 2019.

A curious case was the people from "DNS Institute" who managed to raise a Bind named 4.8.3 on a NetBSD (June 1990 release). In their tests they failed to load an area with DNAME record, but yet, acting as caching resolver, was able to follow the CNAME synthesized, and is able to return the final answer. More information in http://dnsinstitute.com/research/2021/ancient-1990-bind-4.8.3.html

# Final conclusions

We believe that this architecture can be coordinated between RIRs and replace the current scheme of zonelets. The improvements include:

- Simpler solution as a concept, using DNS standards;
- stop depending on ad-hoc solutions subject to other types of failures, and move to a solution within DNS;
- greater control over the end customer, who once again has authoritative control of the zone using the DNS protocol;
- changes propagate instantaneously, within normal DNS ranges.

Of course there are considerations that must be taken into account, where there will be changes in the way of coordinating between each RIR, but we believe that they are less than those that exist currently and therefore can be carried out with planning and internal coordination.

The change in the configuration of the end customer should also be considered, which now will need to add a tag in the parent, to <rir> .in-addr.arpa.

Finally, it is important to take into account the TTL of the records. The rise in the chain resolution can lead to timeouts, so it is important to make optimal use of the cache.

On the solution itself, experiments show that an increase could be expected failure rate of 1.1% using the technique described in this document, product of resolvers they are not able to follow the DNAME. However it could be ventured that these failures they would be from very old resolvers, poorly configured, or within networks that filter excessive. This failure rate could be considered to be to be expected in such a diverse environment like the Internet, where it is virtually impossible to expect zero error rates. Anyway, there is also the possibility that the RIRs themselves carry out tests closer to the world real, using “canary” delegations within the real tree, and using other success metrics (such as having a real mail server in the delegation IPs and measuring the rate of failures of errors in post office that could be due to resolvers unable to solve the reverse, something typical in the dispatch of emails).

# Acknowledgements

To Mauricio Vergara-Ereche (ICANN) and Carlos Martínez (LACNIC) for their suggestions and comments. To Celsa Sánchez (NIC Chile) for her corrections and comments.

This work was carried out in its entirety by the author, who is especially grateful to NIC Chile for the support, and the delivery of this report to the community.

# Appendix 1: RIPE Atlas measurement details:

The measurement was made publicly, with the number of “measurement id” 32002160, and all its detail, including downloading of results are available on https://atlas.ripe.net/measurements/32002160/#general

The epoch moment of each of the changes in the delegation corresponds to:

- 1629918000: change to DNAME
- 1630004400: 1 day parent TTL increase
- 1630072263: change to normal NS delegation
- 1630091400: change to DNAME, and all TTLs at 1 day

An image of the global distribution of the probes, provided by the same website from RIPE Atlas:

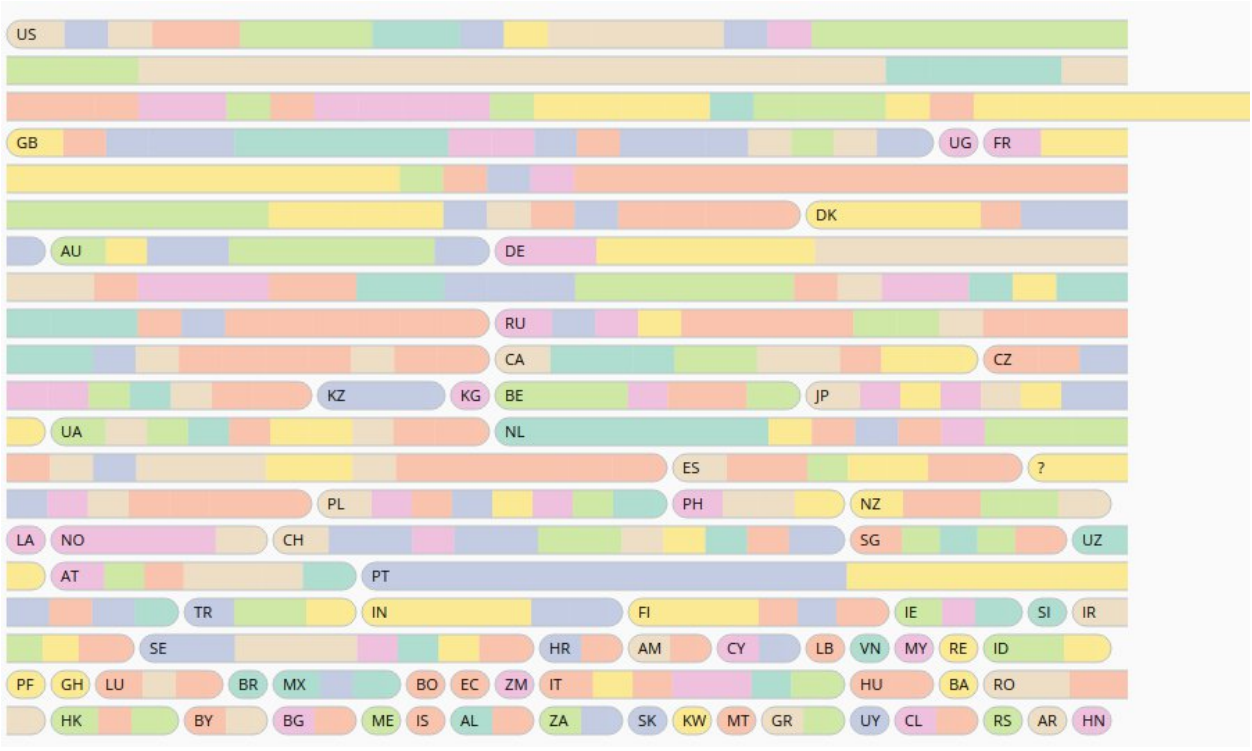


Figure 5, RIPE Atlas probes distribution