DNS 'Root-less'

Running a 'root zone' less DNS Service on OmniOS
126th FRAOSUG meeting

Dr. Erwin Hoffmann

[feh@fehcom.de]

22.4.2026

Open Task on my GUUG FFG Workshop 2026:

9 Die Kür Teil 2: Root-less DNS-Dienste

Die Zonendatei der DNS Root-Nameserver ist öffentlich und kann kopiert werden: `https://www.iana.org/domains/root/files`.

- Die Root-Server sind chronisch unter Druck, weil jeder *resolving* Nameserver erstmals diese befragen muss.
- Waren 'früher' nur die bekannten Top-Level-Domain (TLD) (also `.com`, `.net`, `.de`, `.ua`) hierin enthalten, so finden sich aktuell hierin > 14k Nameserver-Einträge.
- Zudem werden hierin noch die DNSSec-Informationen hinterlegt (`KEY`, `NSEC` Records ...) und es müssen signierte Antworten generiert werden.

Dies führt zu hoher Belastung der Root-Zonen Nameserver, obwohl diese bereits IP-Anycasting benutzen. Ist man unter Kontrolle eines eigenen Netzwerks mit einem DNS-Cacheserver für die downstream Rechner, so ergibt sich die Möglichkeit, die Root-Zone *lokal* zu halten und damit den Verkehr zu diesen zu reduzieren und die Antwortzeiten für die DNS-Clients zu verbessern (die Root-Nameserver droppen schon mal gerne Anfragen).

Aufgaben:

- Herunterladen der *Zonen-Datei*.
- Umwandlung in das **tinydns**-Format.
- Aufsetzen eines *tinydns-root* Services.
- Anbindung des *dnscache* Services an diesen.
- Analyse der Arbeitsweise und Antwortzeiten.
- Blick in `https://datatracker.ietf.org/doc/rfc8806/`.

Figure: Task 9: Setting up a 'root-less' DNS server

The task was to setup a 'local' copy of the `root zone` on **tinydns** a part of my *djbdnscurve6* package.

Recap DNS

In the Domain Name System (DNS) we have different actors:

- The DNS *stub resolver* (as part of the *C-Lib* or my *fehQlibs*) starting an *recursive query*.
- Some strange DNS *local resolvers* (like **systemd-resolvd**).
- Potentially a DNS *forwarder* and DNS *cache* in your network (could be your Internet router).
- The DNS *full resolvers* and caches at your ISP.
- The local *authoritative name servers* (content) with their `zone files`, either provided by the given domain or outsources to some DNS provider (eg. BuddyNS).
- In particular the 13 *root name servers* (`a-m.root-server.net`) which are set up in *anycast* fashion on the Internet with in total > 1800 physical name server.

The root Name Servers are operated by different organizations (to be elected) and different underlying authoritative Name Servers (Bind, PowerDNS, ...).

Top Level Domains in the DNS Root Zone

In the 'good old days' of the Internet, we just had a small number of *Top Level Domains* (TLD):

- The GTLD like `.net`, `.org.`, `.com`, and `.gov`.
- Regional TLDs, like `.eu`, `.de`, `.nl` and other country specific ones, in particular `.tv`.

These days are gone, and by today the DNS Root Zone comprises of $>$ 7500 top level domain names.

From the technical point they include:

- The name of the zone (eg. `.xxx`) as **NS** record together with the responsible Name Server (mostly outsourced now).
- The **A** and **AAAA** records for those Name Servers, called their Glue.
- The adjacent **RRSIG**, **NSEC** and **DNSKEY** records.

$\hookrightarrow$ In the well-known BIND Zone format this information includes roughly 25k lines describing those 7500 top level domains.

The Root Zone is *DNSSEC signed* with seven key stubs distributed among seven different people.

Full Resolvers and the Root Zone

In oder to provide a local DNS name resolution, the full resolvers needs at first to have access to the Root name servers:

- Each full resolver has a copy of the *IP addresses* of the a-z.root-server.net) typically `dnsroots.global` (as file name).
- Since 16 root servers are included and IP anycasting is used, only 16 A and 16 AAAA references are required.

> Once, the name server of the top level domain is fetched by a A and/or AAAA query, it is cached until the domain's (SOA) TTL or the individual TTL of the records is expired.

$\hookrightarrow$ A query for a not-yet existing entry in the cache is called a *cold query* requiring full name resolution given its cold cache.

Unqualified DNS Names

Each DNS name has several labels, separated by dots (as given in the zone file and used for description; but not in the *wire format*).

A DNS lookup has to undergo a qualification: A bare 'hostname' needs to be appended with a domain name and a TLD. In the file `resolv.conf` this is given by the `domain` or search `parameter`.

However, things can go wrong, unqualified DNS names need to be treated by the full resolver as well: The *unqualified* name is subject of a full name resolution (as well).

As a result, the DNS Root Name servers are flooded by garbage queries from broken DNS clients: $\rightarrow$ More than 70% of the DNS queries reaching the Root Name Servers are junk.

> Authoritative name servers often drop queries because of overload conditions. Typically the full resolver tries a query to a (randomly chosen) different name server.

Standard (full) Name Resolution

A standard full name resolution by a recursive name server given a *cold cache* looks like the following:

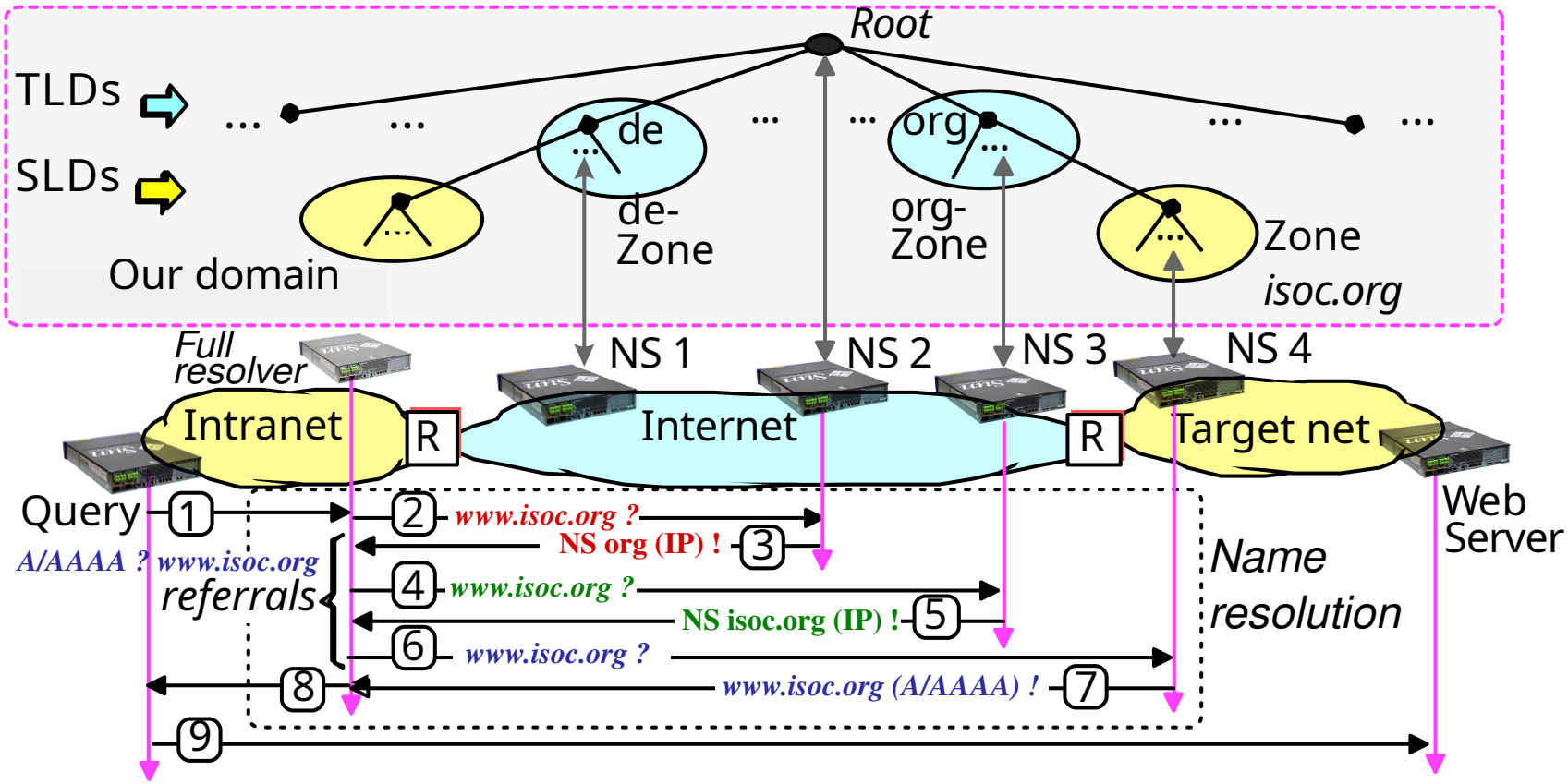

Figure: IP-Query and name resolution for '`www.isoc.org`'; R: Router, NS: Name Server

↪ On our DNS resolution path, we leak the information of our query even to the Root Name Servers. DNSSEC does not protect.

Qname Minimization for Name Resolution

This undesired behavior can be eliminated by means of *Qname minimization* (RFC 7816):

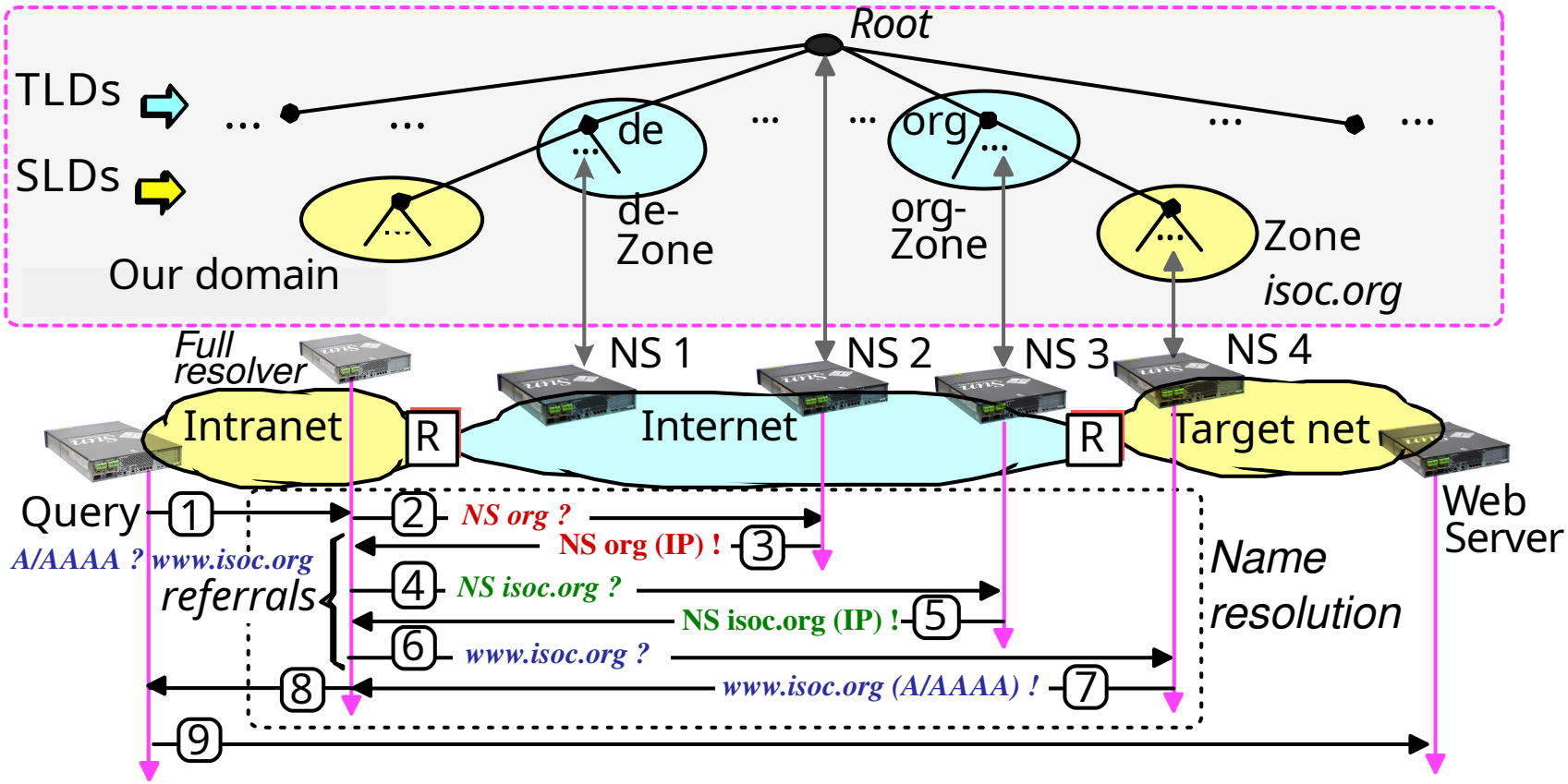

Figure: DNS Query with Qname minimization for '`www.isoc.org`'; R: Router, NS: Name Server

↪ Most of modern DNS full resolvers will do this (but not my **dnscache**; need to be done).

Alternate DNS Root Zone

Since the Root Zone is public, it can be downloaded:

- Link: `https://www.iana.org/domains/root/files`
- Unlike the root server, it changes frequently and needs to be refreshed (every week?).

Given the 'Task 9' as defined in my DNS Workshop, I have the following setting:

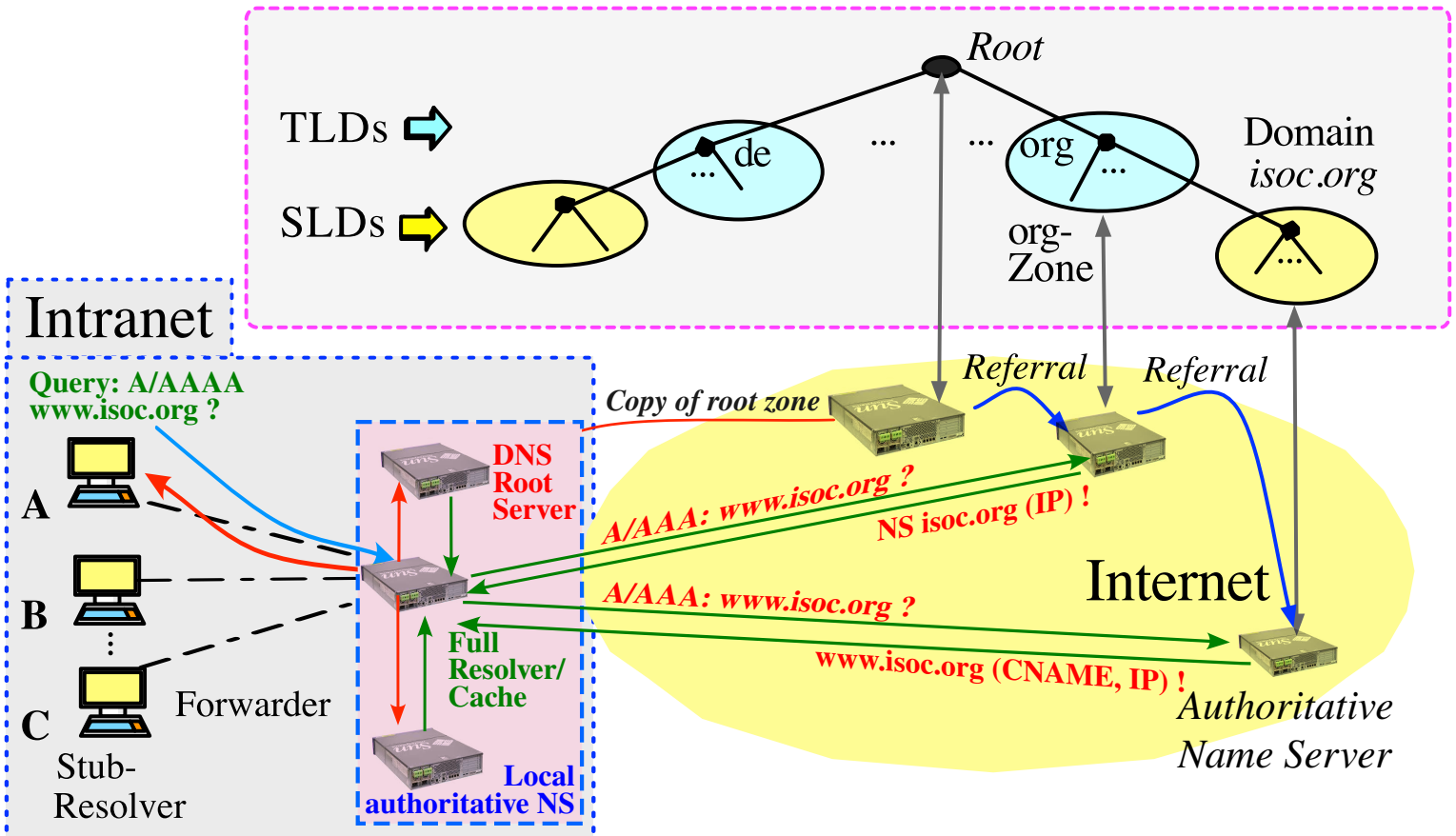

Figure: 'Root-less' DNS setting

Alternate DNS Root Zone Infrastructure

Given in alternate DNS Root zone infrastructure, this RFC describes the scenario:

- RFC 7706: *Decreasing Access Time to Root Servers by Running One on Loopback*

In my case I have:

- My own authoritative name server (**tinydns**) bound to IP `127.0.0.1`.
- A copy of the zone file running on **tinydns** available via IP `127.0.0.53`.
- My (internally public) full resolver/cache server (**dnscache**) serving all DNS queries from all allowed IPv4 and/or IPv6 addresses.
  - Access to the my own zone/domain is given via a hint: `fehnet.dnl` → `127.0.0.1`.
  - The standard dnsroot.global file (called '@') is substituted by: `@` → `127.0.0.53`.

Challenges:

- Cleanup the downloaded 'root zone' file and transfer it into the **tinydns** data format (not BIND like!).
- Coupling the three services providing DNS services such, they have a *correct binding* on OmniOS.
- Making OmniOS aware af *additional loopback* IP addresses.

Alternate DNS Root Zone Infrastructure: Solutions Part One

Daniel Bernstein has made an attempt to provide 'dnsroot' (`https://cr.yp.to/dnsroot.html`. But this was in the glory old times of the Internet with just a few GTLDs/TLDs. However, he called his attempt 'vaporware'.

Today, the procedure '*Extract/Transform/Load*' is done based on IANA's *root zone* `file` and not via Zone transfer. For **tinydns**, the entire DNSSEC garbage has to be filtered out. A straightened **tinydns** compatible root-zone file needs to be generated:

```
.                       86400   IN      SOA     a.root-servers.net. nstld.verisign-grs.com. 2026021201 1800 900 604800 864
.                       86400   IN      RRSIG   SOA 8 0 86400 20260225170000 20260212160000 21831 . qNM9zoRiJzioe9eoWlmVVc
Oe9Q4cngBiWlUd63y7ytc8RBs0xw2S+dLHSouyx6YGsaIKMDhUBFko4ltL3PNICJ6lkUdu4WWDMTxxf9uqe7BfLxD0bcR8Npnml3qKc9bG6H1iiG5B1TSbUhD/
XsyxUggk7BTEl/k5ShP/HmEzErIzNtltEYMlotb/Q+SOSzhrqarTQ1K3jImTnQvVk1Q2R+owWhhEzqhUytboF09H0epvz3TIqOoq4eWdNZEXGNgb2JrmEfmtLY
yGFp+HhIYO4aedZNuxrB9XWWyjQYOAvZcRb8RJDJL7b5l+CleQZAPqh64KfoTb4w==
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.
.                       518400  IN      RRSIG   NS 8 0 518400 20260225170000 20260212160000 21831 . dZfebmH7nrJSn8XU+iC+2x
u882NfDhv3UeI8XKAQHKw2g9+lM9NBn0h8W0xzw5tiGuJEoXKT9qazNXnYdDFDa4Es0IaRN1J8VqB6qO3kEtjWJev23J6QVVFlO4pvSvpwFlY7Sm794bxLfjw
ZLDtz80xnxb3UFoACaHtHKPvpqGhTYXkaQK0xQ4A7VEXE2DmiCzvfr7/K+/rZ5eoekk8FwXThyvz6uMRkAKPf8FaJIoG7DMHAlLKGRPXyAboyKpHdD0vN1Nc8
CQwIb7qFBMjPfjiz8179FpDjBTnfIcfzT/v/qJw88x7ncMBqPfLIAGLYrcJF6LdA==
.                       86400   IN      NSEC    aaa. NS SOA RRSIG NSEC DNSKEY ZONEMD
.                       86400   IN      RRSIG   NSEC 8 0 86400 20260225170000 20260212160000 21831 . GtQ2wEBLcRT5qmMYegvgW
xELb3uGO7p2vroykEAXuvMKqtX/DsRrSRWZfdLIx86T6RNclJz872CHDGwgPLMPAuMiaUGwXHSg+4sBhDnrpNDhCjJIKfdH9GA2P95a0uE8Z79/vlJCDwUzZAT
E55JPmVVOTnWkYQmiPfhY4qKgsmnJR/odgDpMbqKjlTrDzn3skJMf1bL5cLPko9c+YUihRci6x4K8FIXHjYfNnxisBcHybdxLzQeKL209g0YN8DP5ghsqpnlzC
XTQmEUcq0NW9UWmcxSUAgzXAJyEZRPl1KFW7e/XfR0fNFpAG58M4j3eneRJawjd2w==
```

Figure: IANA root zone file

```
.||127.0.0.53
&aaa|37.209.192.9|a.nic.aaa|172800
&aaa|2001:dcd:1::9|a.nic.aaa|172800
&aaa|37.209.194.9|b.nic.aaa|172800
&aaa|2001:dcd:2::9|b.nic.aaa|172800
&aaa|37.209.196.9|c.nic.aaa|172800
&aaa|2001:dcd:3::9|c.nic.aaa|172800
&aaa|156.154.144.2|ns1.dns.nic.aaa|172800
&aaa|2610:a1:1071::2|ns1.dns.nic.aaa|172800
&aaa|156.154.145.2|ns2.dns.nic.aaa|172800
&aaa|2610:a1:1072::2|ns2.dns.nic.aaa|172800
&aaa|156.154.159.2|ns3.dns.nic.aaa|172800
&aaa|2610:a1:1073::2|ns3.dns.nic.aaa|172800
&aarp|37.209.192.9|a.nic.aarp|172800
&aarp|2001:dcd:1::9|a.nic.aarp|172800
&aarp|37.209.194.9|b.nic.aarp|172800
&aarp|2001:dcd:2::9|b.nic.aarp|172800
&aarp|37.209.196.9|c.nic.aarp|172800
&aarp|2001:dcd:3::9|c.nic.aarp|172800
&aarp|156.154.172.82|x.nic.aarp|172800
&aarp|2610:a1:1074::1:82|x.nic.aarp|172800
&aarp|156.154.173.82|y.nic.aarp|172800
&aarp|2610:a1:1075::1:82|y.nic.aarp|172800
&aarp|156.154.174.82|z.nic.aarp|172800
&aarp|2610:a1:1076::1:82|z.nic.aarp|172800
&abb|65.22.112.41|a0.nic.abb|172800
&abb|2a01:8840:6e::41|a0.nic.abb|172800
&abb|65.22.115.41|a2.nic.abb|172800
&abb|2a01:8840:71::41|a2.nic.abb|172800
&abb|65.22.113.41|b0.nic.abb|172800
&abb|2a01:8840:6f::41|b0.nic.abb|172800
&abb|65.22.114.41|c0.nic.abb|172800
&abb|2a01:8840:70::41|c0.nic.abb|172800
&abbott|65.22.156.41|a0.nic.abbott|172800
&abbott|2a01:8840:9a::41|a0.nic.abbott|172800
&abbott|65.22.159.41|a2.nic.abbott|172800
```

Figure: tinydns data file; to be transferred into binary

Alternate DNS Root Zone Infrastructure: Solutions Part Two

Trying to query the own DNS Root zone failed miserably:

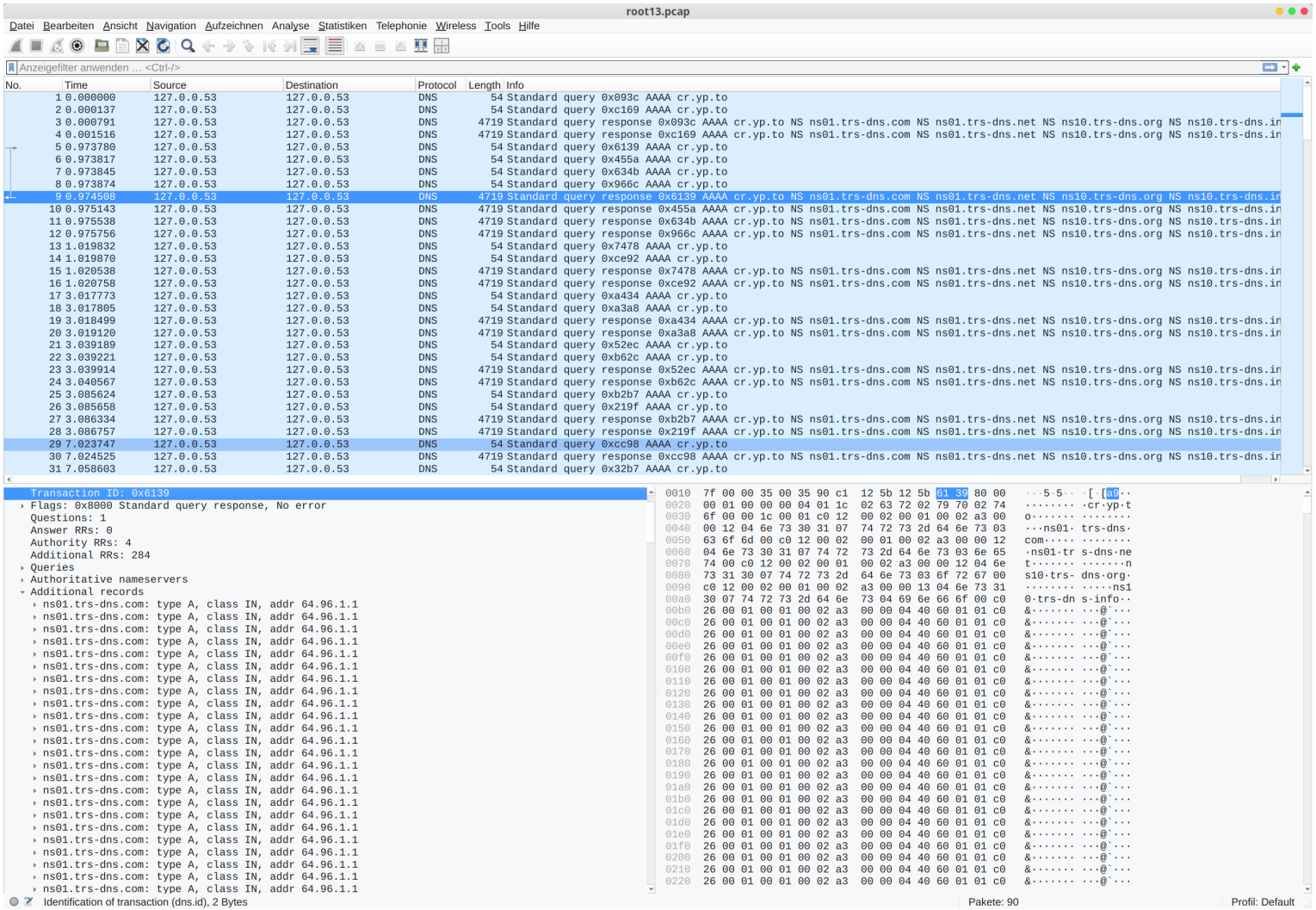

Figure: Gigantic Additional Section in DNS Responses

Alternate DNS Root Zone Infrastructure: Solutions Part Two

Trying to query the own DNS Root zone failed miserably:

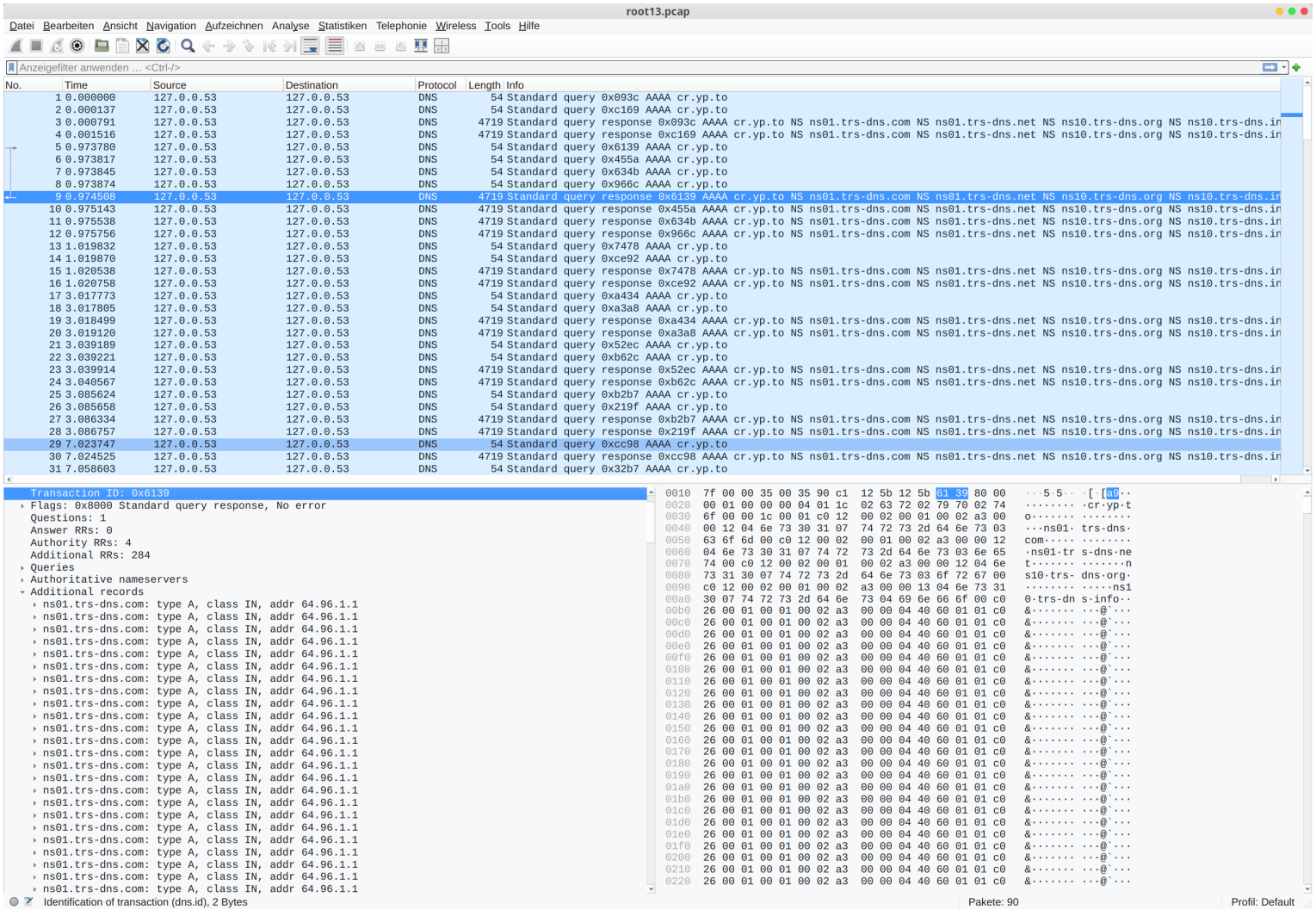

Figure: Gigantic Additional Section in DNS Responses

→ Need to enhance **tinydns** to understand multiple identical IP entries!

Alternate DNS Root Zone Infrastructure: Solutions Part Three

Given IP loopback addresses und providing dual-stack operation the text books (including my own 'Technik der IP-Netze') provide the following solutions:

1. In IPv4 with have the whole range `127.0.0.0/8` or `127.0.0.0` mask `255.0.0.0`. Thus any address starting from `127.0.0.1` up to `127.255.255.254` is allowed; a complete class **A** network!
2. Classical IPv6 provides us with `::1/128`. This is a single logical but persistant host entry.
3. Since we have a virtual interface, in OmniOS given as '`lo0`', we also have a logical LLU address assigned to it: `fe80::1%lo0`.

Obiously, the SunOS and Illumos developer have never read my book and don't provide the convenient solution.

↪ Rather, on OmniOS use '**ipadm**'!

```
$ ipadm create-addr -T static -a 127.0.0.53 lo0/v53

$ ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
lo0/v53           static   ok           127.0.0.53/8
```

→ This works until ...

Alternate DNS Root Zone Infrastructure: Solutions Part Three (not yet)

After reboot, the logical loopback address becomes 'disabled':

```
# ipadm
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
lo0/v53           static   disabled     127.0.0.53
```

In OmniOS, **ipadm** seems to be broken significantly:

a) The address postfix has to be provided always in the form '/v...'.

```
# ipadm create-addr -T static -a 127.53.0.1 lo0/_dummy
ipadm: Error in creating address object: Invalid argument provided
```

b) Deletion of an **addrobj** is not possible:

```
# ipadm delete-addr lo0/v53
ipadm: could not delete address: Object not found
```

Alternate DNS Root Zone Infrastructure: Solutions Part Three (not yet)

After reboot, the logical loopback address becomes 'disabled':

```
# ipadm
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
lo0/v53           static   disabled     127.0.0.53
```

In OmniOS, **ipadm** seems to be broken significantly:

a) The address postfix has to be provided always in the form '/v...'.

```
# ipadm create-addr -T static -a 127.53.0.1 lo0/_dummy
ipadm: Error in creating address object: Invalid argument provided
```

b) Deletion of an **addrobj** is not possible:

```
# ipadm delete-addr lo0/v53
ipadm: could not delete address: Object not found
```

After discussing the issue with Andy on the Illumos mailing list, an 'old-fashioned' work-around was given:

```
cat /etc/hostname.lo0
addif 127.0.0.53/8
```

Alternate DNS Root Zone Infrastructure: Solutions Part Three (there)

Now, I get:

```
$ ipadm
ADDROBJ           TYPE     STATE        ADDR
lo0/_a            static   ok           127.0.0.1/8
lo0/_c            static   ok           127.0.0.53/8
lo0/_b            static   ok           ::1/64
lo0/v53           static   disabled     127.0.0.53
```

Comments:

- Certainly, **ipadm** needs to be fixed and its kernel interface has to be straigthend.
- Currently, **ipadm** does not work as anticipated from the man page.
- The term '`addrobj`' shall be omitted and replaced by something like '*communication endpoint*' `CEP`, what it is.

Alternate DNS Root Zone Infrastructure: Solutions Part Four

Finally, given that scenario everything is in place, except for the order of binding to IP addresses:

- Here, I use **daemontools** from Dan Berstein, because **djbdns** (and my **djbdnscurve6**) is based on those.
- The order, **daemontools** starts a *service* (located under `/service`), is strictly alphabetic.
- For this Use Case – and any comparable – *service dependencies* (like given under **systemd**) would be a great achievement.

↪ Thus, I enhanced daemontools(-0.76) to become daemontoolsx(-1.06) in order to provide this kind of capability:

```
# svstat /service/*
/service/bincimaps: up (pid 742) 17357 seconds
/service/dnscache: up (pid 783) 17355 seconds,
                   trigger: /service/tinydns/supervise/status (live)
/service/fcgi: up (pid 746) 17357 seconds
/service/minidlnad: up (pid 745) 17357 seconds
/service/qmail-pop3d: down 17357 seconds
/service/qmail-pop3sd: down 17357 seconds
/service/qmail-send: up (pid 725) 17357 seconds
/service/qmail-smtpd: down 17357 seconds
/service/qmail-smtpsd: up (pid 723) 17357 seconds
/service/qmail-smtpsub: down 17357 seconds
/service/tinydns: up (pid 782) 17356 seconds,
                  trigger: /service/tinydns-root/supervise/status (live)
/service/tinydns-root: up (pid 773) 17357 seconds
```

Advantages of an own local Root Zone

What is it good for?

- Become independent of un-responsive Root Name Servers.
- Don't leak information about queries (remember: all cleartext).
- Faster lookup, since one query omitted.
- Don't spam Internet Root servers with own misguided DNS lookup.

Additional observation:

- Using NFS, publish a TXT records to speed up operation:
- `'_nfsv4idmapdomain.fehnet.dnl|fehnet.dnl|86400` (**tinydns-data** format)

RFCs and Sources

RFCs:

- DNS Basics: `https://datatracker.ietf.org/doc/html/rfc1035`
- EDNS0: `https://datatracker.ietf.org/doc/html/rfc2671`, `https://datatracker.ietf.org/doc/html/rfc6891`
- DNSRoot: `https://datatracker.ietf.org/doc/html/rfc7706`
- Qname Minimization: `https://datatracker.ietf.org/doc/html/rfc7816`
- Service Bindings: `https://datatracker.ietf.org/doc/html/rfc9460`

Sources:

- djbdns: `https://cr.yp.to/djbdns.html` (DJB)
- DNSRoot: `https://cr.yp.to/dnsroot.html` (DJB)
- Wiki Root Server: `https://en.wikipedia.org/wiki/Root_name_server`
- IANA Root Zone Repository: `https://www.iana.org/domains/root/files` (for download)
- Alternative Root Server: `https://en.wikipedia.org/wiki/Alternative_DNS_root`
- ORSN: `https://en.wikipedia.org/wiki/Open_Root_Server_Network`
- Root Server Junk: `https://pulse.internetsociety.org/blog/more-than-70-of-dns-root-queries-are-junk`
- djbdnscurve6: `https://www.fehcom.de/ipnet/djbdnscurve6.html`
- daemontoolsx: `https://www.fehcom.de/ipnet/daemontoolx.html`