- unwind(8) - florian@OpenBSD.org

- A recursive name server for every laptop

- Opportunistic DNSSEC validation

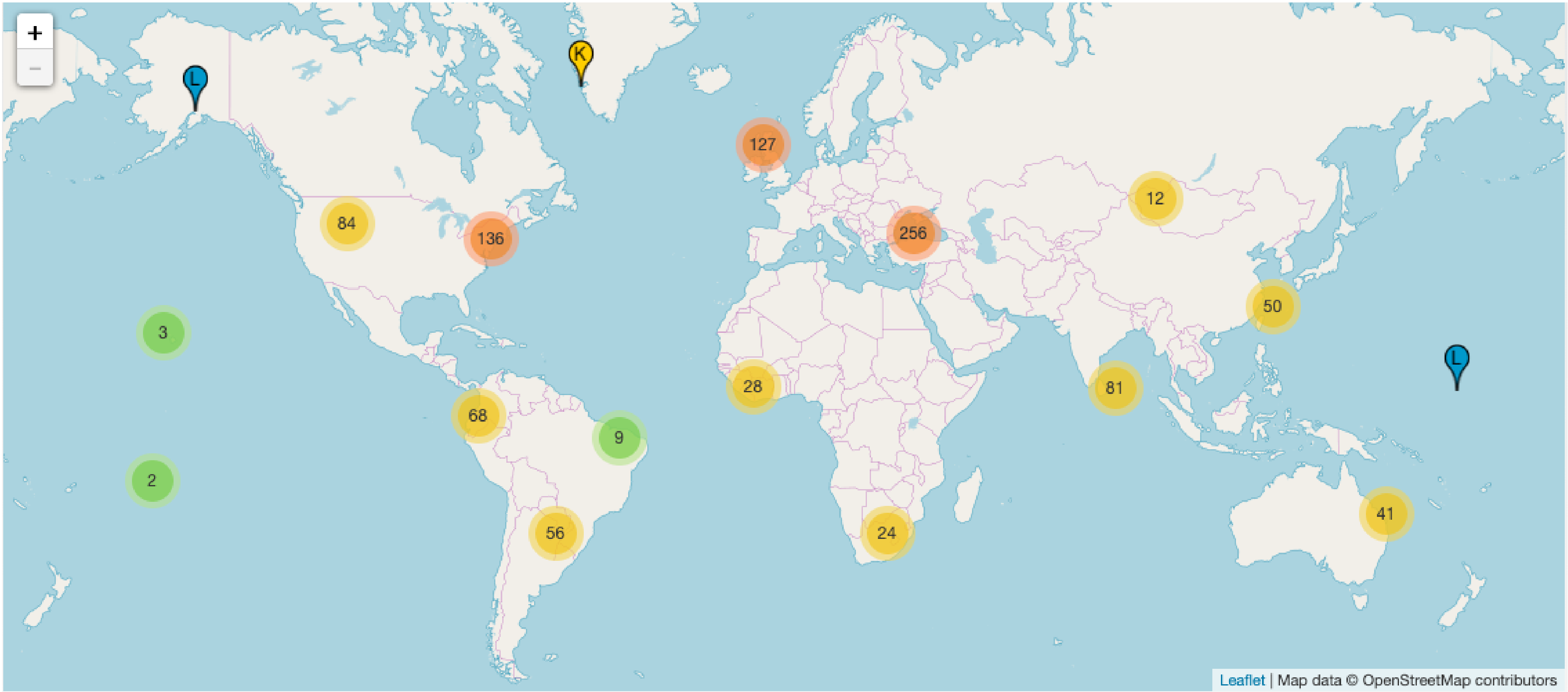
- Captive-portal detection

- Adapt to local conditions...

- … no matter how harsh.

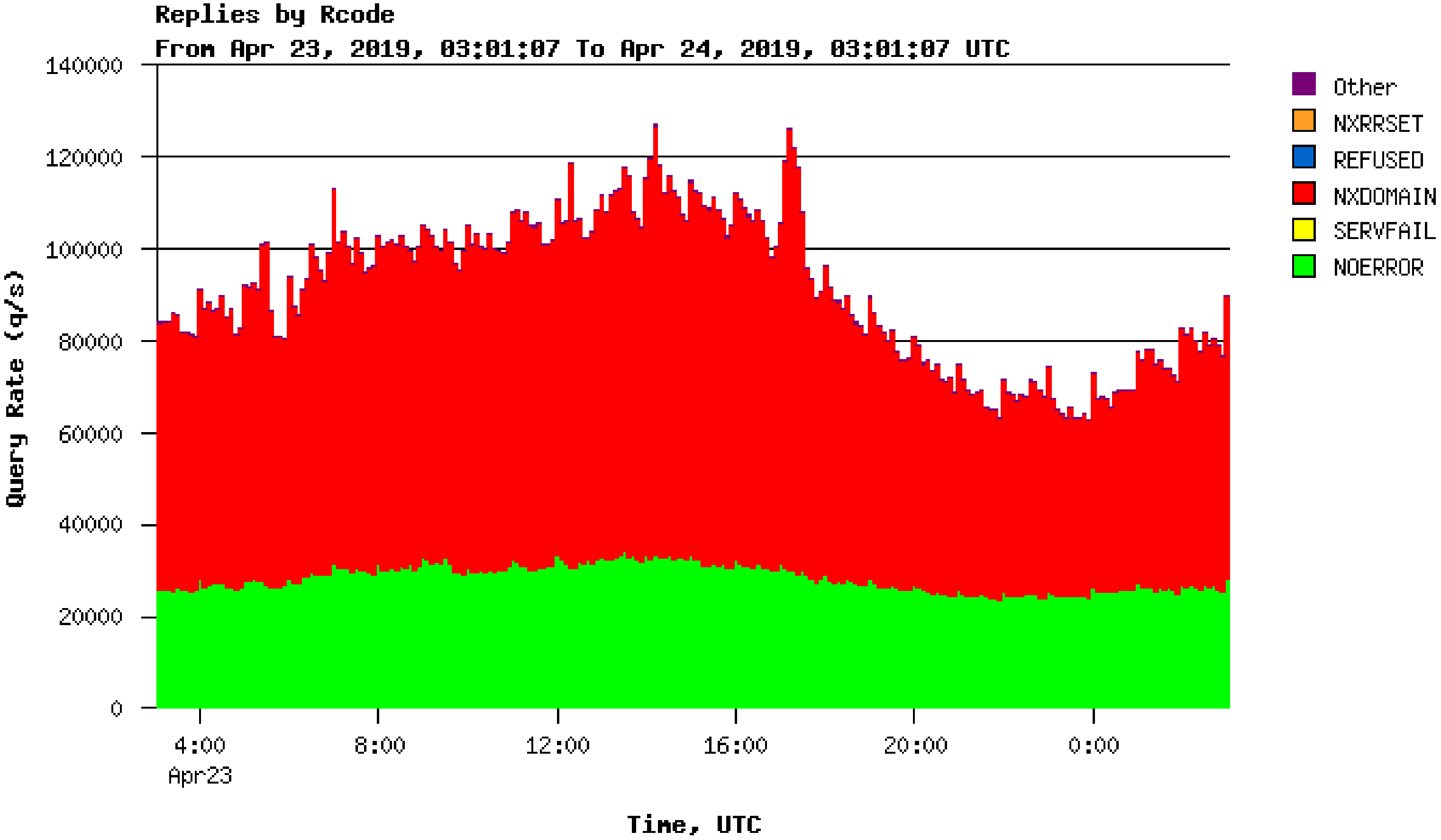
- OpenBSD developer since 2012
  - author of slowcgi(8), slaacd(8) (cf. BSDCan 2018), rad(8), unwind(8), sysupgrade(8), ...
  - poked at things in the network stack
- Senior Systems Engineer @ RIPE NCC
  - BGP, DNS, ...
  - k.root-servers.net, pri.authdns.ripe.net

root name servers

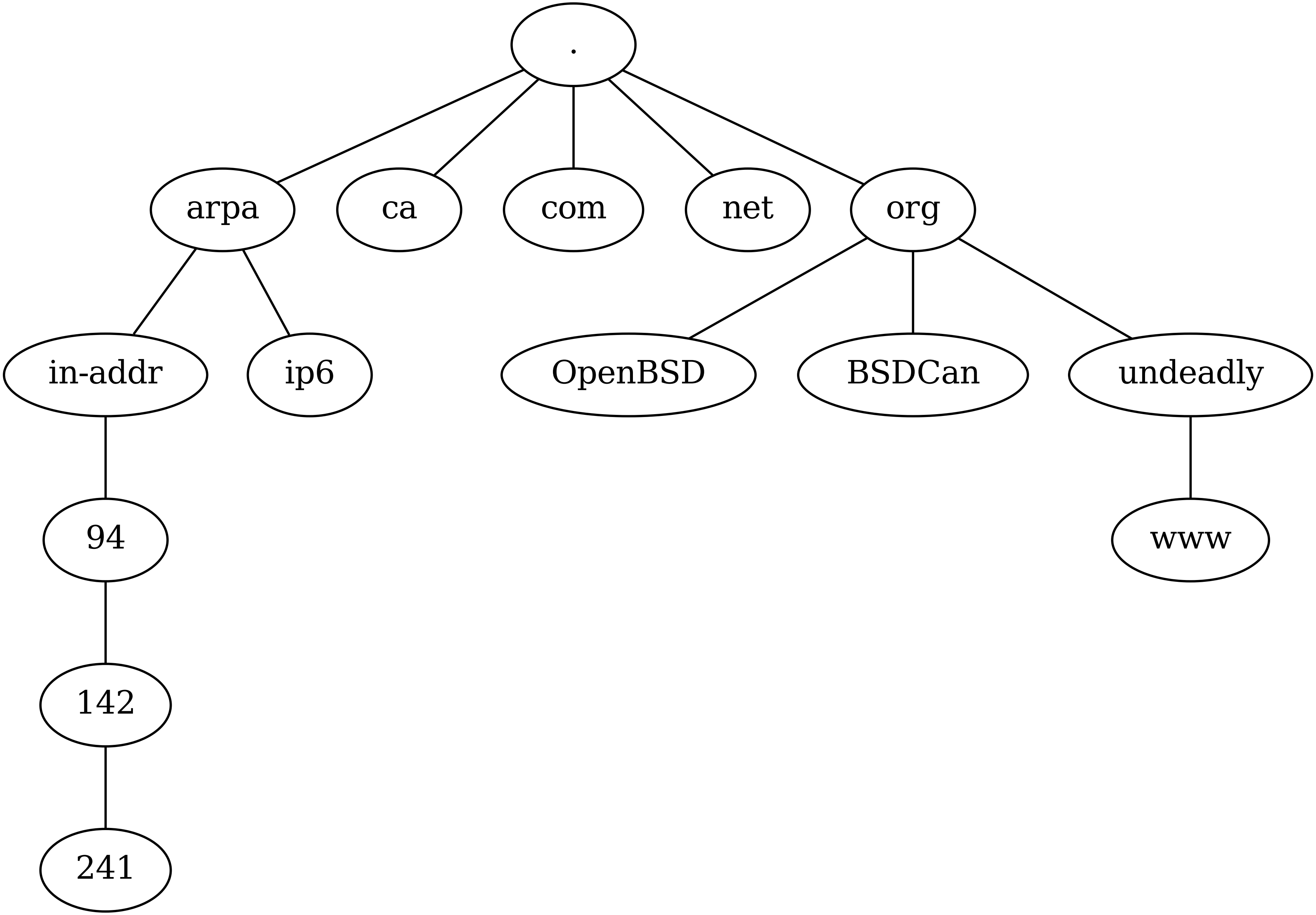


- 13 servers ([a..m].root-servers.net)
- ~ 1000 instances
- run by 12 independent root server operators
  (cf. root-servers.org)

A Day in the Life of a Root Name Server



- quick introduction to DNS
  - ~2k - 3k pages of RFCs; there will be inaccuracies, lies and omissions (cf. powerdns.org/dns-camel)
  - distributed hierarchical key-value database
  - (www.undeadly.org, A) → 94.142.241.173
  - (173.241.142.94.in-addr.arpa, PTR) → www.undeadly.org


.
arpa
ca
com
net
org
in-addr
ip6
OpenBSD
BSDCan
undeadly
94
www
142
241

- Authoritative Name Server
  - **The** source of truth for part of the hierarchy (root (.), org, undeadly.org)
  - knows the answer (NOERROR)
  - knows that there is no answer (NXDOMAIN)
  - knows who else to ask (NOERROR, delegation)
  - key is outside the name servers hierarchy (REFUSED)

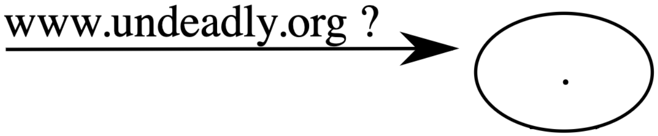
- Recursive Name Server
    - navigates the DNS tree
    - most of the complexity and smarts of DNS

- lib C resolver
  - getaddrinfo(3) / getnameinfo(3)
  - talks to a recursive name sever
  - configured in /etc/resolv.conf

What is the IPv4 address of www.undeadly.org?

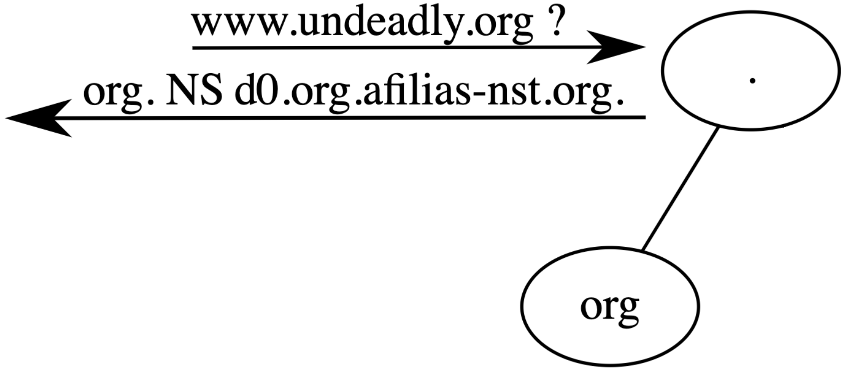
```
struct addrinfo hints, *res0;

memset(&hints, 0, sizeof(hints));
hints.ai_family = AF_INET;
getaddrinfo("www.undeadly.org", "www", &hints, &res0);
```



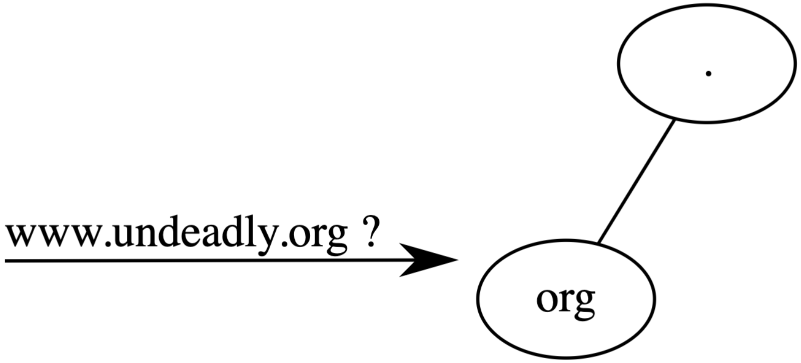
www.undeadly.org ?

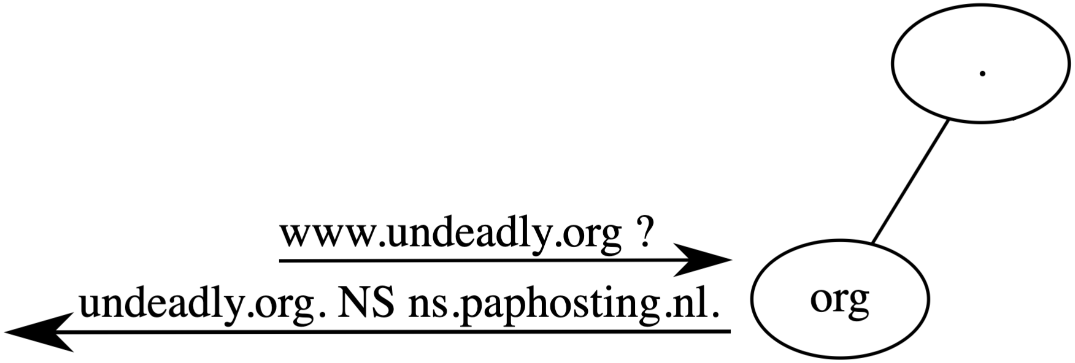
.



www.undeadly.org ?

org. NS d0.org.afilias-nst.org.


.
www.undeadly.org ?
org

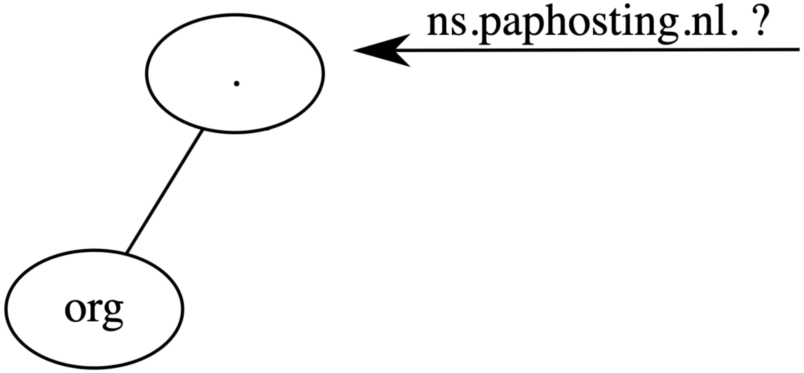


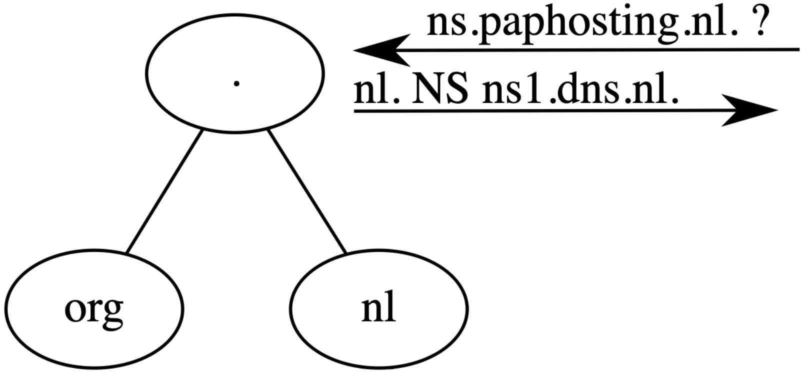
.

org

www.undeadly.org ?

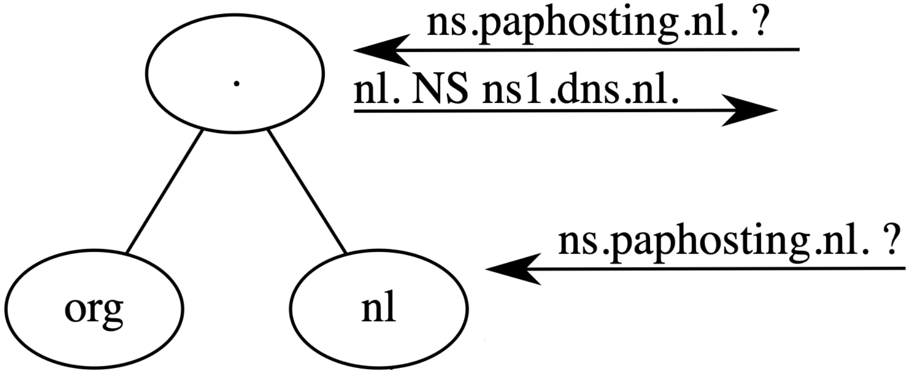
undeadly.org. NS ns.paphosting.nl.

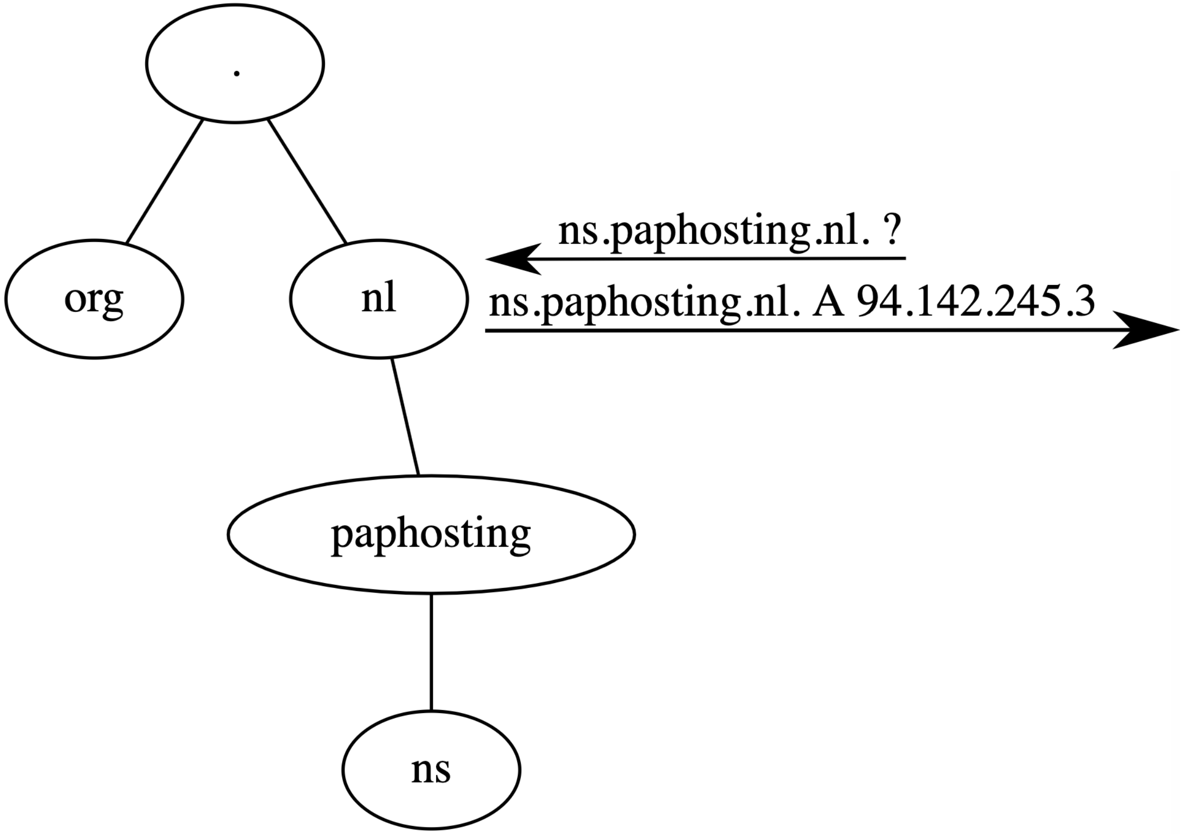

ns.paphosting.nl. ?
.
org

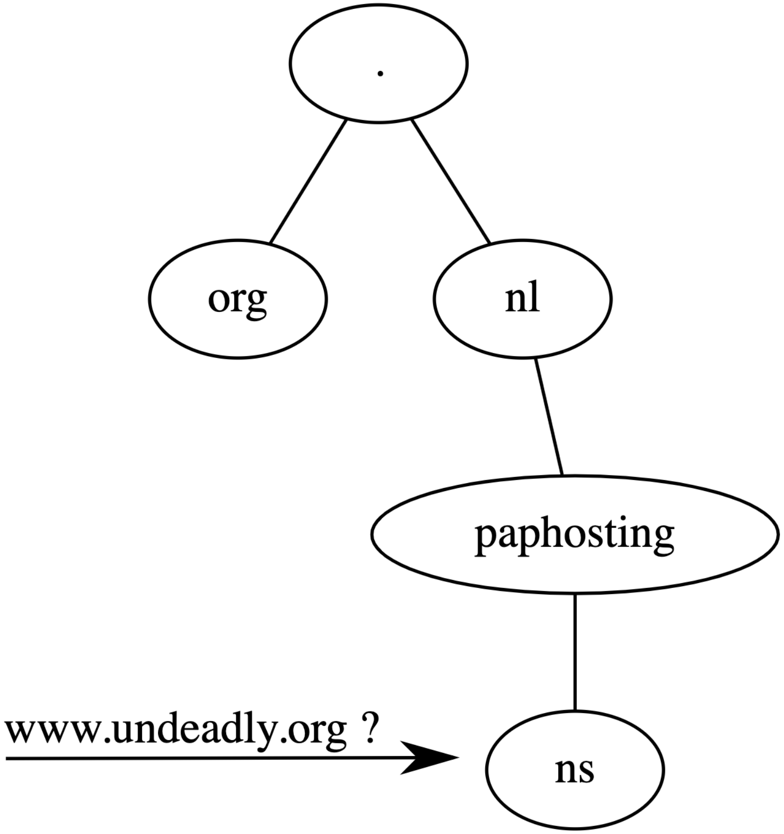


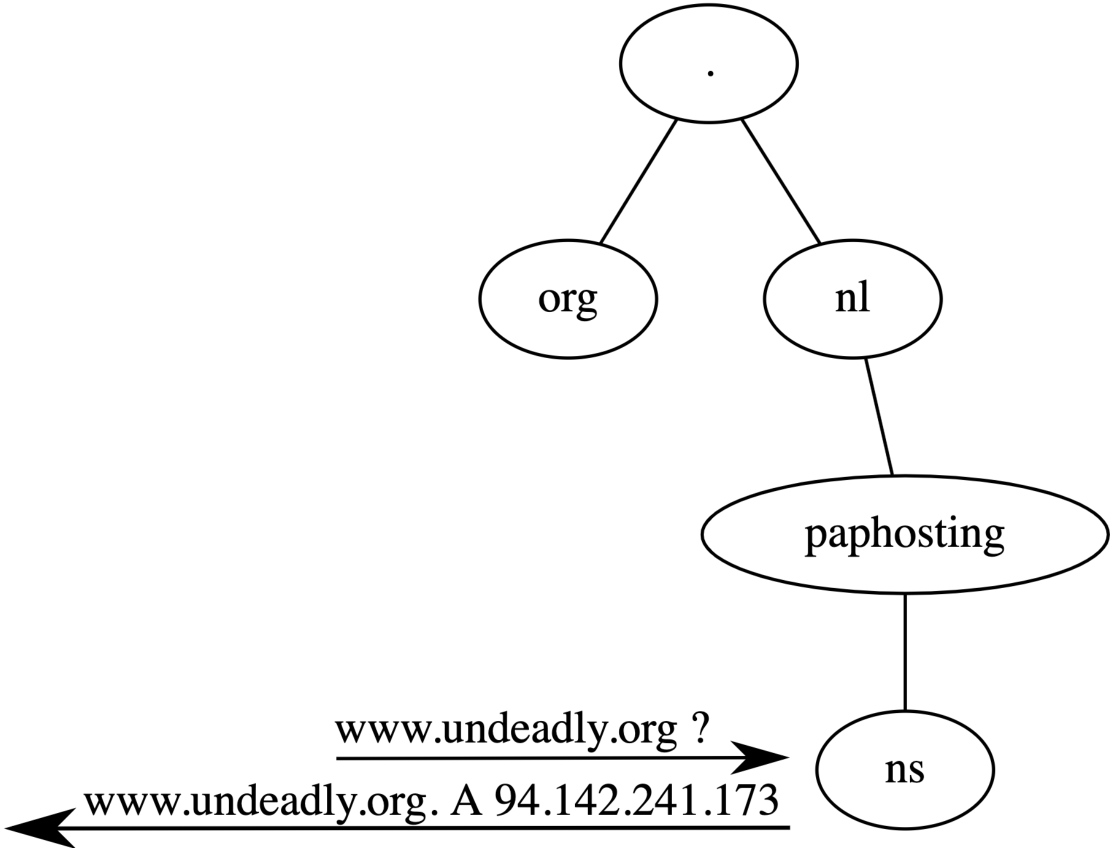
ns.paphosting.nl. ?

nl. NS ns1.dns.nl.

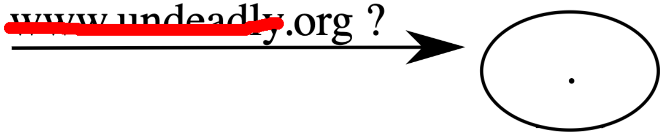

.
org
nl
ns.paphosting.nl. ?
nl. NS ns1.dns.nl.
ns.paphosting.nl. ?


.
org
nl
paphosting
ns
ns.paphosting.nl. ?
ns.paphosting.nl. A 94.142.245.3

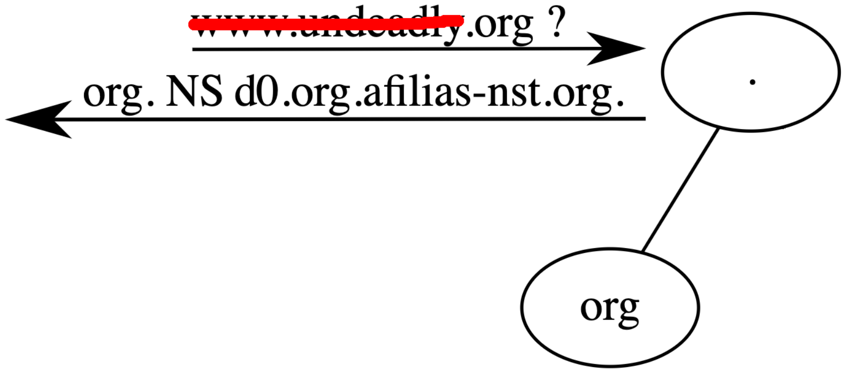

.
org
nl
paphosting
ns
www.undeadly.org ?


.
org
nl
paphosting
ns
www.undeadly.org ?
www.undeadly.org. A 94.142.241.173

- query name minimization (qname minimization)
  - only send required parts to authoritative servers
  - improves privacy
  - needs a few quirks in recursive name servers but works well enough

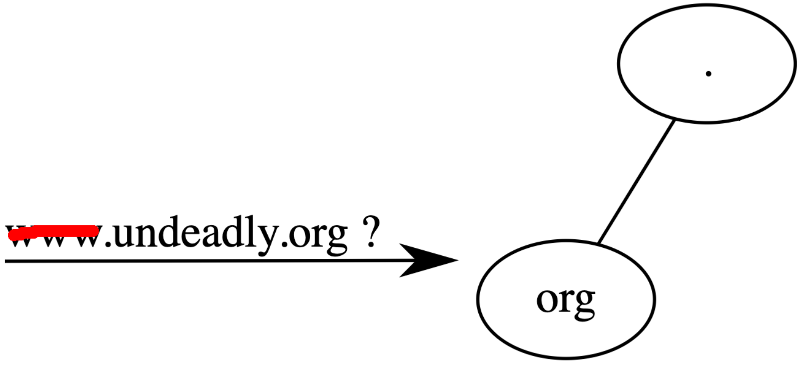


~~www.undeadly~~.org ?



~~www.undeadly~~.org ?

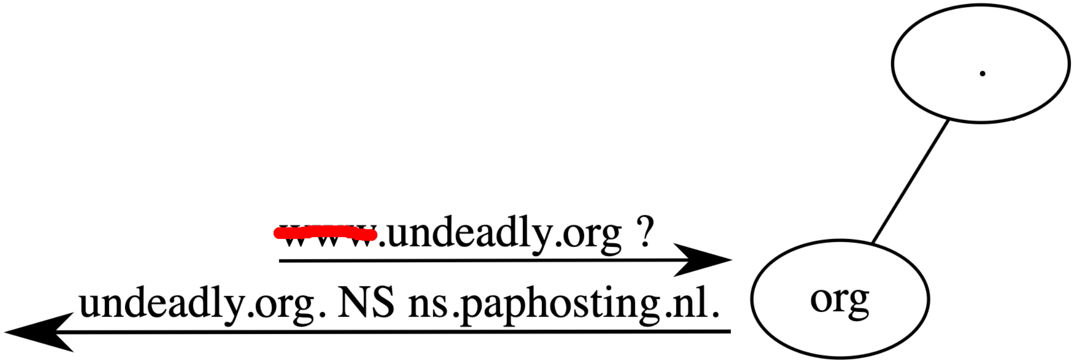
org. NS d0.org.afilias-nst.org.



~~www~~.undeadly.org ?

.

org



~~www~~.undeadly.org ?

undeadly.org. NS ns.paphosting.nl.

DNSSEC

Relax..., breathe!

- DNSSEC
  - origin authentication
  - integrity
  - denial of existence
  - no confidentiality

- DNSSEC can do some neat things
  - follows the DNS hierarchy, so not everyone can sign everything like in TLS / X509
  - DANE binds X509 certificates to domain names
- validation (kinda) must run on the local machine

- DNSSEC has some problems on a laptop
    - needs accurate clock (how unwind(8) started!)
    - network middle boxes filtering DNSSEC
    - recursive name server doesn't support DNSSEC

That wasn't too bad.

Two more things...

- Where to send DNS queries
  - do your own recursion
  - configure a name server (quad-X, maybe w/ DNS over TLS (DoT))
  - name server learned via DHCP or router advertisements
  - WiFi or 4G?
  - all fighting over /etc/resolv.conf

- Where to send DNS queries (cont'd): privacy - who can see the queries
  - dhcp / quad-X
    - server operator
    - Person In The Middle (pitm)
  - DoT
    - DoT server operator
    - pitm DoT → auth correlate queries to origin(?)
  - recursion with qname minimization
    - pitm near laptop but generally not near auths

- captive-portals
  - "Click here to accept Terms of Service"
  - plays evil tricks with DNS, blocks Internet access
  - must use DHCP provided name servers

Let's get cracking!

- previous approaches: dhclient
    - just owns /etc/resolv.conf
    - will get you past captive-portals
    - at the mercy of recursive name server operator
    - no DNSSEC

- previous approaches: static configuration
  - tell dhclient to leave /etc/resolv.conf alone
  - will likely not get you past captive-portals
  - will not work in places where DNS is filtered
  - no DNSSEC

- previous approaches: run unbound(8) on localhost
  - tell dhclient to leave /etc/resolv.conf alone
  - can use DNS over TLS (DoT)
  - DNSSEC validation
  - will likely not get you past captive-portals
  - will not work in places where DNS is filtered

- previous approaches: FreeBSD's resolvconf(8) / openresolv
  - framework to handle multiple sources for /etc/resolv.conf
  - very powerful: controllable by scripts, executes scripts as event handlers
  - supports local recursive name servers
  - does not seem to come with batteries included

Welcome unwind(8).

- unwind(8) introduction
  - a validating name server for every laptop
  - should always run
  - must be at least as good as using DHCP provided name servers

- unwind(8) introduction (cont'd)
  - uses libunbound for the heavy DNS lifting:
    - DNSSEC
    - recursion
    - forwarding to recursive name servers
    - DNS over TLS

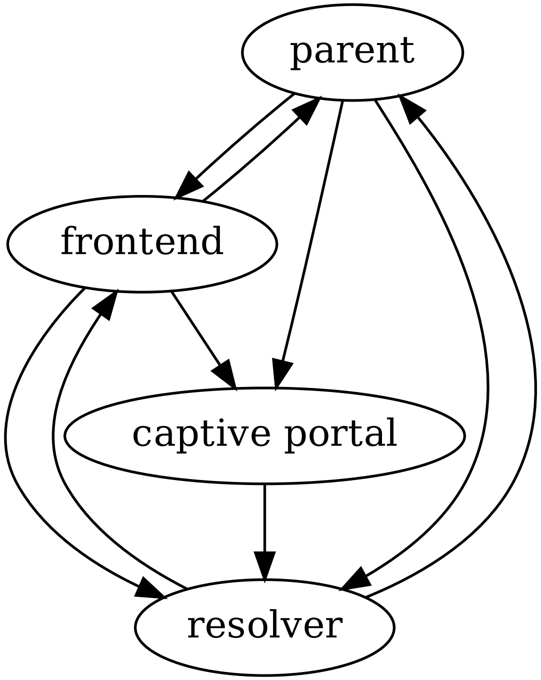
- unwind(8) introduction (cont'd)
  - privilege separated daemon
  - processes run in a restricted-service operating mode (pledge(2))
  - processes have a restricted filesystem view (unveil(2))

- unwind(8) introduction (cont'd)
  - looks out for network changes
  - actively monitors network quality

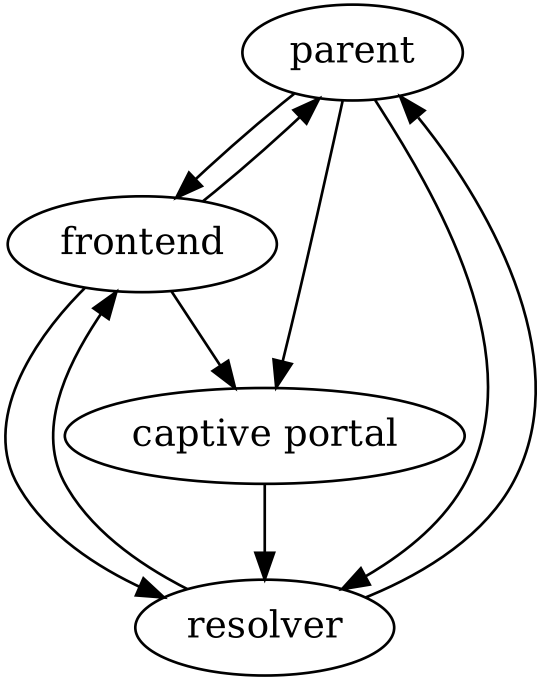
Let's check out some details.

- libunbound
  - developed by NLnet Labs as part of unbound(8)
  - unwind has a local copy, but no changes
    - → updates are easy, whenever we update unbound in base, copy files over
  - upstream is receptive to diffs

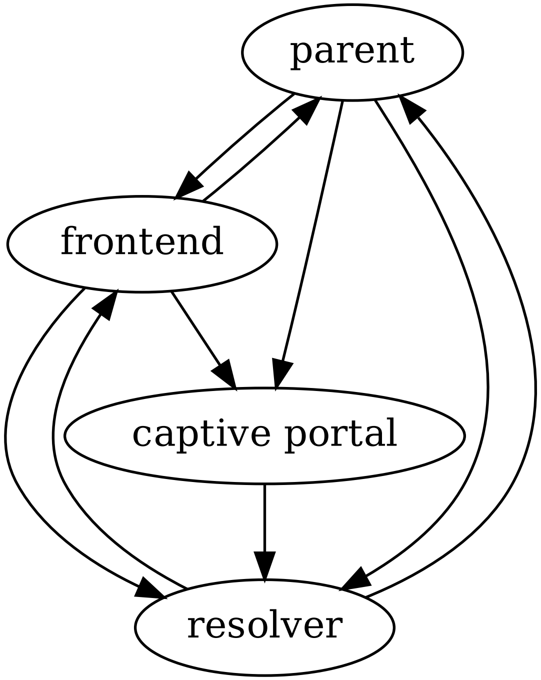
- privilege separation, pledge(2) & unveil(2)
    - standard for all network daemons in OpenBSD
    - easiest way to get a new one:
        - transmogrify an existing one (~ 1 - 2h)
    - automatically has all the security benefits, a config parser, config reload, a logging framework, and a control tool



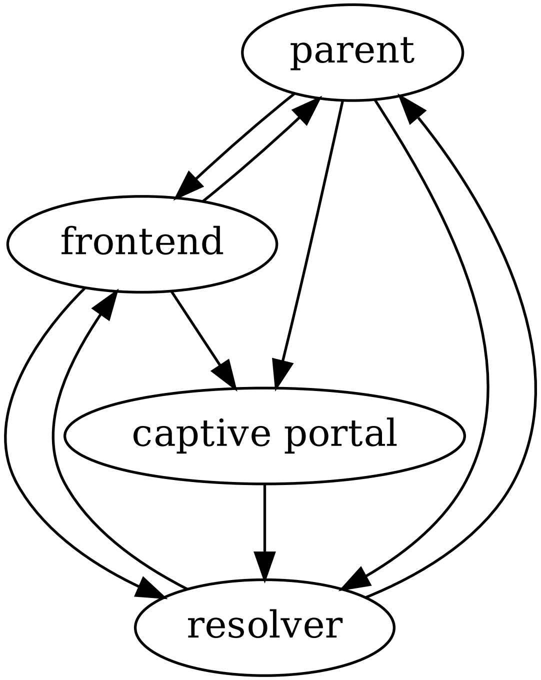
- priv'sep (cont'd), parent:
  - parse config, send to children
  - → frontend:
    - route socket
    - listen control socket
    - trust anchor file (rw)
    - listen udp/53
    - dhcp lease file (r)
  - → captive-portal:
    - connect check host tcp/80



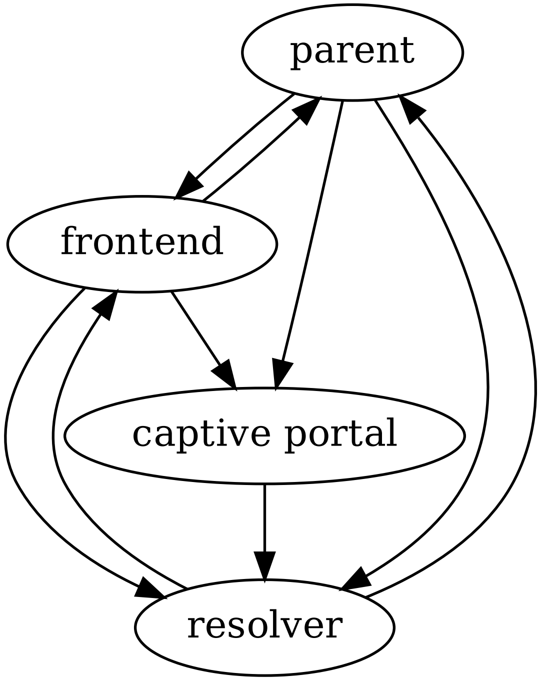
- priv'sep (cont'd), frontend:
  - handle service port (53/udp)
    - read query, pass on to resolver, send answer
    - ask parent to open 53/udp when resolver indicates DNS working
    - close udp/53 when resolver indicates that DNS stopped working



- priv'sep (cont'd), frontend:
  - handle control socket
    - set log level in all procs
    - ask parent to config reload
    - pass status request on
  - handle route socket
    - on interface change ask parent to open DHCP lease file, parse it, and pass name servers on to resolver process



- priv'sep (cont'd), resolver:
  - DNS heavy lifting
    - receives query from frontend, sends answer to frontend
    - checks quality of different resolving strategy, decides on best
    - initiates captive-portal check via parent
    - periodically check DNS for new TAs



- priv'sep (cont'd), captive-portal:
  - HTTP speaker
    - receives connected socket from parent
    - sends GET request
    - parses response and compares to expected response from config file
    - informs resolver

- priv'sep, pledge(2) & unveil(2) (cont'd)
  - pledge(2): restricted-service operating mode
    - stdio: operate on open FDs only
    - inet: talk to Internet
    - rpath: open files for reading
    - ...
  - unveil(2): restricted filesystem view

- priv'sep, pledge(2) & unveil(2) (cont'd)
    - parent: stdio, inet, dns, rpath, sendfd
    - frontend: stdio, unix, recvfd
    - resolver: stdio, inet, dns, rpath
        - unveil: /etc/ssl/cert.pem
    - captive-portal: stdio, recvfd

- monitoring network quality
  - multiple resolving strategies:
    - recursion
    - dhcp
    - forwarder
    - DoT

- monitoring network quality (cont'd)
  - periodically sends "SOA" queries for the root zone
    - known to exist
    - known to be signed
  - resolving strategy quality
    1. validating
    2. resolving
    3. unknown
    4. dead

- monitoring network quality (cont'd)
  - keeps a histogram of response time
    - aggregates by buckets
    - could be used to switch resolving strategies

```
[florian@x1:~]$ unwindctl status recursor
selected             type status
       *         recursor validating

                           histogram[ms]
   <10   <20   <40   <60   <80  <100  <200  <400  <600  <800 <1000
  1021    63   380   444   283   123   190    99    25    17    16
```

- misc
  - captive-portal detection
    - configure URL and expected HTTP status code and / or body
    - prefer dhcp name servers
    - re-probe continuously

```
# Running a connectivity test provider with httpd(8)
# httpd.conf:
#server "c.example.com" {
#       listen on * port 80
#       location "*" { block return 204 }
#}
captive portal {
        url "http://c.example.com/"
        expected status 204
}
```

- misc (cont'd)
  - config file
    - works well without one!
    - but no built-in captive-portal url :(

```
captive portal { ... }

# default
# preference { DoT forwarder recursor dhcp }

forwarder 208.67.222.222        # resolver1.opendns.com

forwarder "9.9.9.9" port 853 authentication name "dns.quad9.net" Do
```

- misc (cont'd)
  - must be as good as dhcp
    - if all strategies fail, close listen 53/udp socket → lib C resolver will fall over to dhcp provided name servers immediately

```
$ cat /etc/resolv.conf
# Generated by vio0 dhclient
search home
nameserver 127.0.0.1
nameserver 84.116.46.21
nameserver 84.116.46.20
```

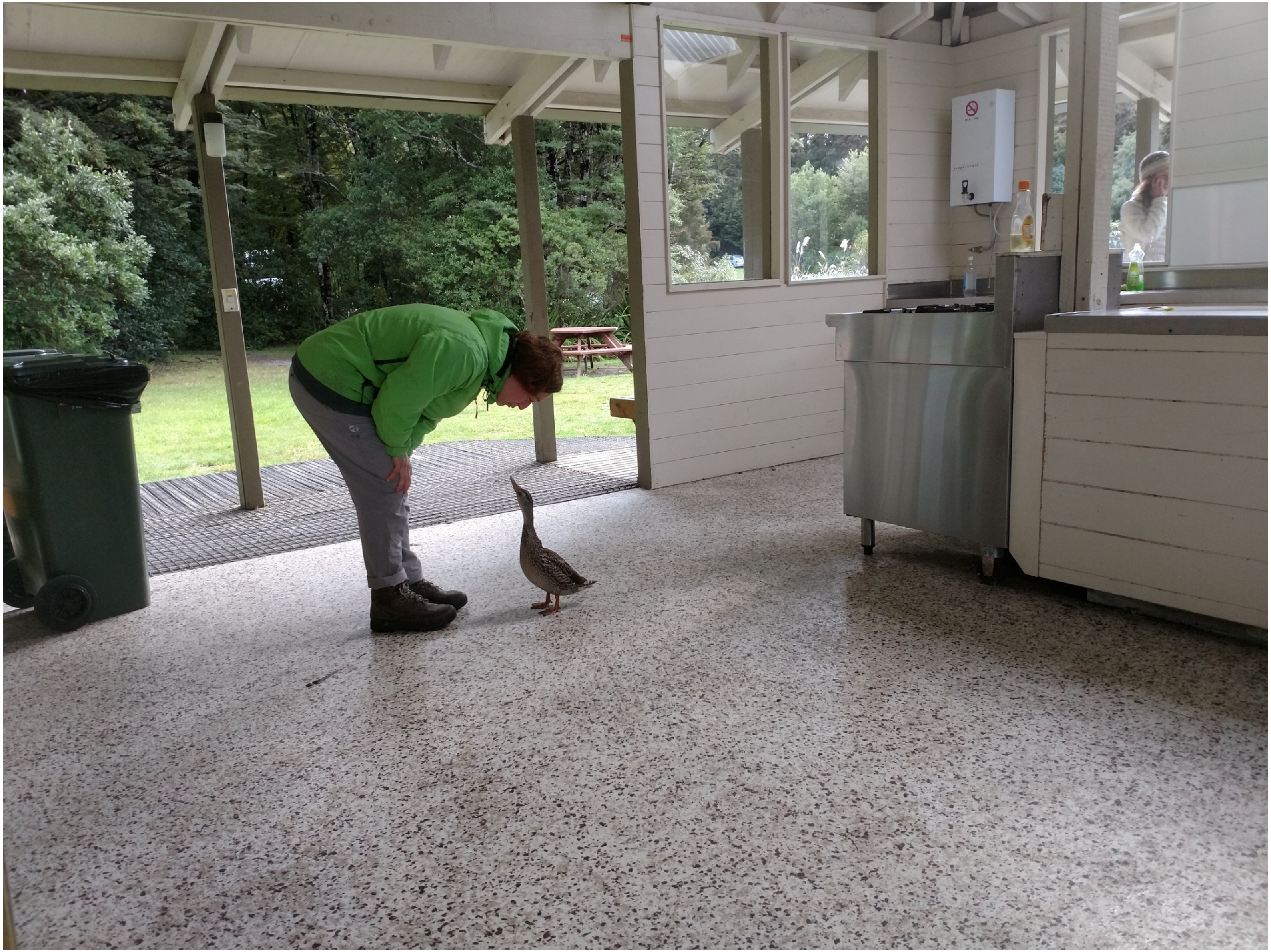
```
$ cat /etc/dhclient.conf
prepend domain-name-servers 127.0.0.1;
```

- portable notes
  - RTM_IFINFO, dhclient lease file: extend unwindctl(8)
  - pledge(2) & unveil(2): #define 0, add chroot(2), arrange access to cert.pem
  - treat pledge(2) & unveil(2) as annotations for your sandboxing facility

Future work

- Future work
  - stop parsing lease files; switch to RTM_PROPOSAL
  - get name servers from router advertisements
  - per-network config for split horizon DNS, VPNs, ...
  - switch strategy if current one is "too slow"
  - built-in captive-portal detection
  - DNSSEC validation too opportunistic

Questions?



Come on! Don't be shy!