Unbound & FreeBSD

A true love story
(at the end of November ’2013)

Presentation for EUROBSDCON 2019 Conference

September 19-22, 2019

Lillehammer, Norway

About me:

Pablo Carboni (42), from Buenos Aires, Argentina.

Worked as **Unix Admin, DNS Admin,** Net Admin, etc, the last 2 decades.

“Passionate” for **DNS, FreeBSD, Network, RFC**, and development stuff related.

My contacts:

 @pcarboni /  @pcarboni@bsd.network

 linkedin.com/in/pcarboni

Disclaimer: “Sensitive info has been renamed/removed intentionally from this story”.

How did this story start?

This adventure began almost 6 years ago, by taking KPIs from some DNS hardware appliances, when I've detected a performance bottleneck with the CPU usage and QPS from those **DNS** servers …

(HW/Infra upgrade - 'capacity planning' was planned in the meantime)

The "***not-so-funny detail***": Those boxes were used by more than **2.5M(!)** customers connected at the same time, for resolving internet addresses.

The awful truth - #1/2 (“the numbers”)

- 2.8 M of internet subscribers at the same time (customers).
- A pair of DNS Appliances
- **A plateau line graphic, from 12pm to 8pm on both devices, reaching 60% of cpu avg usage** during the whole range (the line got stuck there, no curves, no peaks).
- QPS Summary: 20 kqps per physical box (40 kqps total)

Again, it’s worth to note that the HW/Infra upgrade was planned in the meantime.

The awful truth - #2/2 (making it WORSE)

- Furthermore, the firewalls didn't help so much, because the DNS traffic was traversing them (high resource consumption because of high volume of UDP packets, including CPU and other KPIs).

… yes, the DNS service was degraded!

(It's worth to note, in parallel, - just for "fun" -, I began to test Unbound under FreeBSD, by the means of my little lab environment - This was motivated because some people gave me good comments about it)

Next steps - Planned actions

- First step: **A huge DNS traffic re-engineering was needed**.

  ⇒ It was done in less than 2 months, by rerouting it, and avoiding firewalls in the middle of the paths. ✅

- Second step: **Deploy planned HW, load balancers plus physical servers.**

  ⇒ This last step wasn't so 'easy' as I really wanted. (*Unexpected issues* appeared in the meantime!) ❌

When the local problems hits hard...

- **Argentina's economical facts (2013):** There were many (bureaucratic) impediments to import hardware to Argentina because of economical crisis, triggering delays for its local reception.
- **HW planned (bought) versus (received):** Enough physical servers + Enough Load Balancers (LB) were bought.
- However, **only Load Balancers arrived to the datacenters**!

In the meantime, the stuff (lab infra, part #1/2)

- **Hardware:** Dell PowerEdge 1950 double Quadcore (2,0 Gigahertz)
- **OS:** FreeBSD 8.4 RELEASE/AMD64
- **DNS software:** Unbound 1.4.21 [NLNet labs], installed from ports directory -tree updated-, compiled with Libevent [Niels Provos].

  Just in case, I've used Libevent 1.4.14b (proven stable)

  (No **DNSSEC** support was used at that time just to avoid making things worse at that critical moment)

- **Measurement tools:** dnstop, from Measurement factory.

In the meantime, stuff+reading (lab infra, part #2/2)

- **Stress testing tools:** dnsperf package, in particular ***resperf*** (plus query file sample) [Nominum - Now Akamai]

  Query files taken from:
  ftp://ftp.nominum.com/pub/nominum/dnsperf/data

- **A depth-in reading (*essential, do not skip it!*)** from the site: https://calomel.org

  (In particular, ***Unbound DNS tutorial*** *and* ***FreeBSD Network performance tuning***)

  Note: The site is **highly recommended** for tasks like fine tuning services, and *BSD OSes.

So...what should we do now? (Master plan, #1/5)

Because the **service** became **degraded** more and more, this was the plan:

- Install the needed infrastructure, **both load balancers, and replacement for missing servers** behind the LBs.

My boss: Hey Pablo, **because you were testing Unbound on your lab, do you want to try it on production? (yes/yes)** :-)

Me: Ok, let's recover/recycle some (old) hardware server boxes from the own stock**, and try to get the most of that.**

To make it short: hands on!

A (tmp) network/service diagram (Masterplan, #2/5)

The following were the premises for the **(temp) low level design, *some of them based on own needs, and others on the hardware supplier/consultancy***:

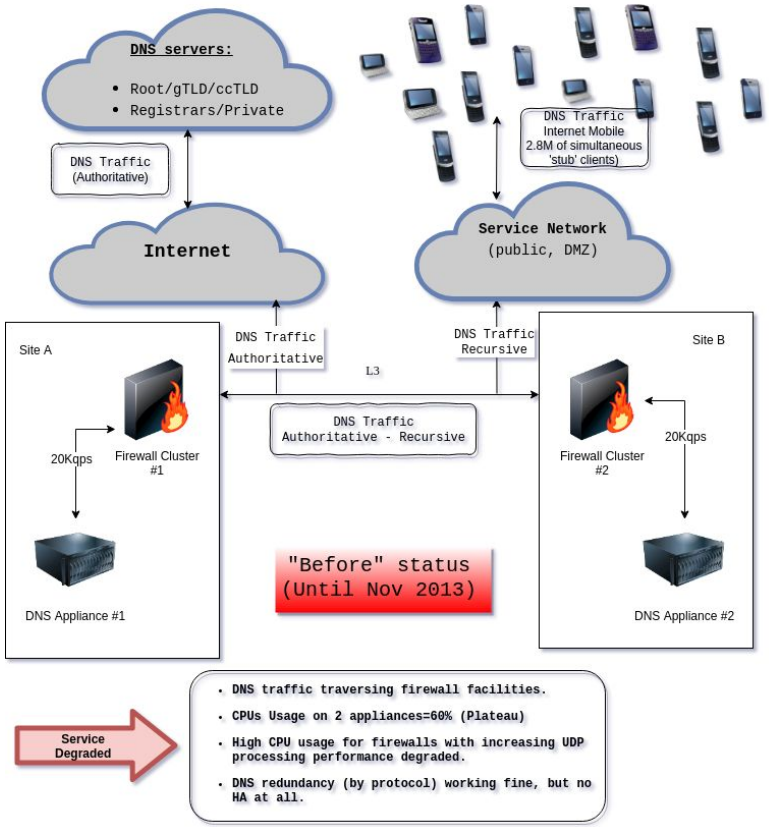
- A **cluster** of **load balancers**, one per site

  One VIP every 50k udp ports.

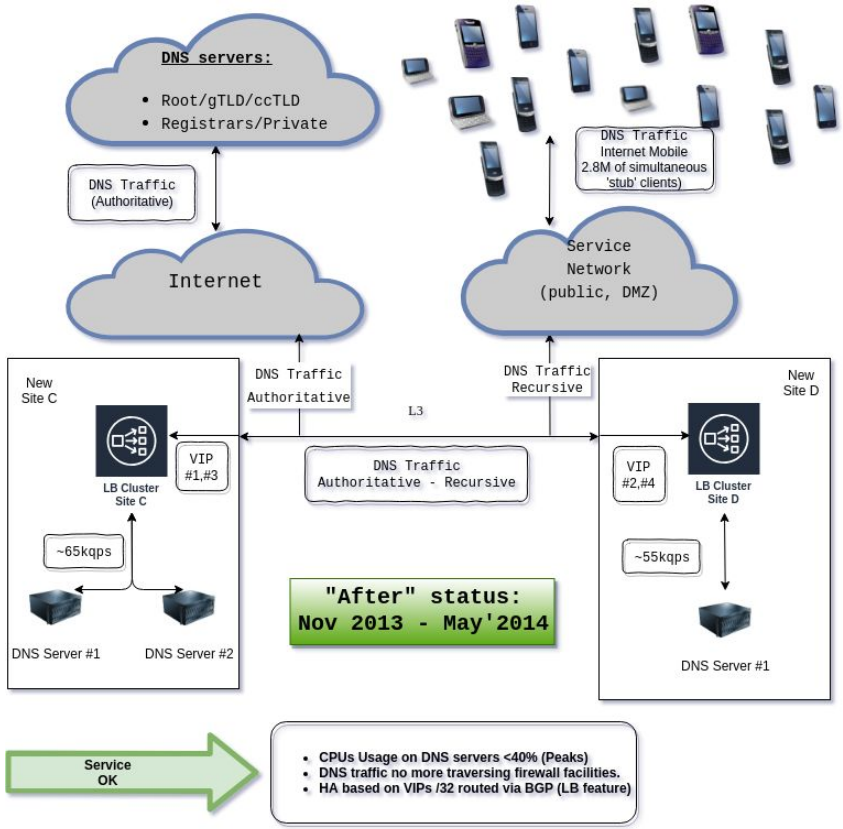
- Several servers behind those LB (remember the lack of those ones). Unbound + FreeBSD would be used (tmp).

- The **VIP** should be ‘easy’ to move between sites (HA).
  BGP was the choice. **No anycast network at all**.

The big picture - Before re-engineering



The big picture, final - After re-engineering.



OS fine tuning (Masterplan, #1/6)

After FreeBSD was installed, **fine tuning** was applied based on lab:

At Operating System level (FreeBSD):

- Available UDP sockets, port range, and backlog.
- **NIC** drivers / timings / buffers / interrupt modes (**Net I/O**)
- **Logs** (Yes, **I/O on disk** is very important, right? ;-)

At DNS Service level (Unbound):

- DNS **instances** providing service (Enabling **more than 1 core/thread**)
- UDP fine tuning, **queries per core**, etc.

OS fine tuning - The details (Masterplan, #2/6)

The following knobs are available (very incomplete list - Sample values provided):

Operating System (file: /boot/loader.conf):

```
net.isr.maxthreads=3              # Increases potential packet
                                  # processing concurrency
kern.ipc.nmbclusters=492680       # Increase network mbufs
net.isr.dispatch="direct"         # Int. handling via multiple CPU
net.isr.maxqlimit=”10240”         # Limit per workstream-queues.
net.link.ifqmaxlen="10240"        # Increase interface send queue
                                  # length
```

OS fine tuning - The details (Masterplan, #3/6)

Operating System (file: /etc/sysctl.conf):

```
kern.ipc.maxsockbuf=16777216     #Combined socket buffer size
net.inet.tcp.sendbuf_max=16777216    # Network buffer (send)
net.inet.tcp.recvbuf_max=16777216    # Network buffer (recv)
net.inet.ip.forwarding=1             # Fast forwarding between
net.inet.ip.fastforwarding=1         # interfaces
net.inet.tcp.sendspace=262144    # TCP buffers(sendspace)
                                 # default 65536
net.inet.tcp.recvbuf_inc=524288  # TCP buffers(recv).
                                 # Default 16384 default
kern.ipc.somaxconn=1024 # backlog queue (incoming TCP conn.)
```

OS fine tuning - The details (Masterplan, #4/6)

Some knobs available for **Unbound** (samples provided)
File: /usr/local/etc/unbound.conf (very incomplete list)

num-threads: 4 (number of cores)
msg-cache-slabs/rrset-cache-slabs: 4 (memory lock contention)
infra-cache-slabs/key-cache-slabs: 4 (memory lock contention)
rrset-cache-size: 512m (resouce Record Set memory cache size)
msg-cache-size: 256m (msg memory cache size)
Outgoing-range: 32768 (number of ports to open)
Num-queries-per-thread: 4096 (Queries server per core)
so-rcvbuf/so-sndbuf: 4m (socket receive/send buffer)

Stress testing - Using dnstop (Masterplan, #5/6)

A text terminal was opened with dnstop. Another terminal was running resperf.

Why did I use dnstop?

- It's a powerful tool for debugging queries and gathering dns stats.
- *When queries quantity was almost the same as the answers,* it shows that maximum capacity was not reached (yet).
- It doesn't interfere with any DNS service.
- It's very lightweight, available for several OSes

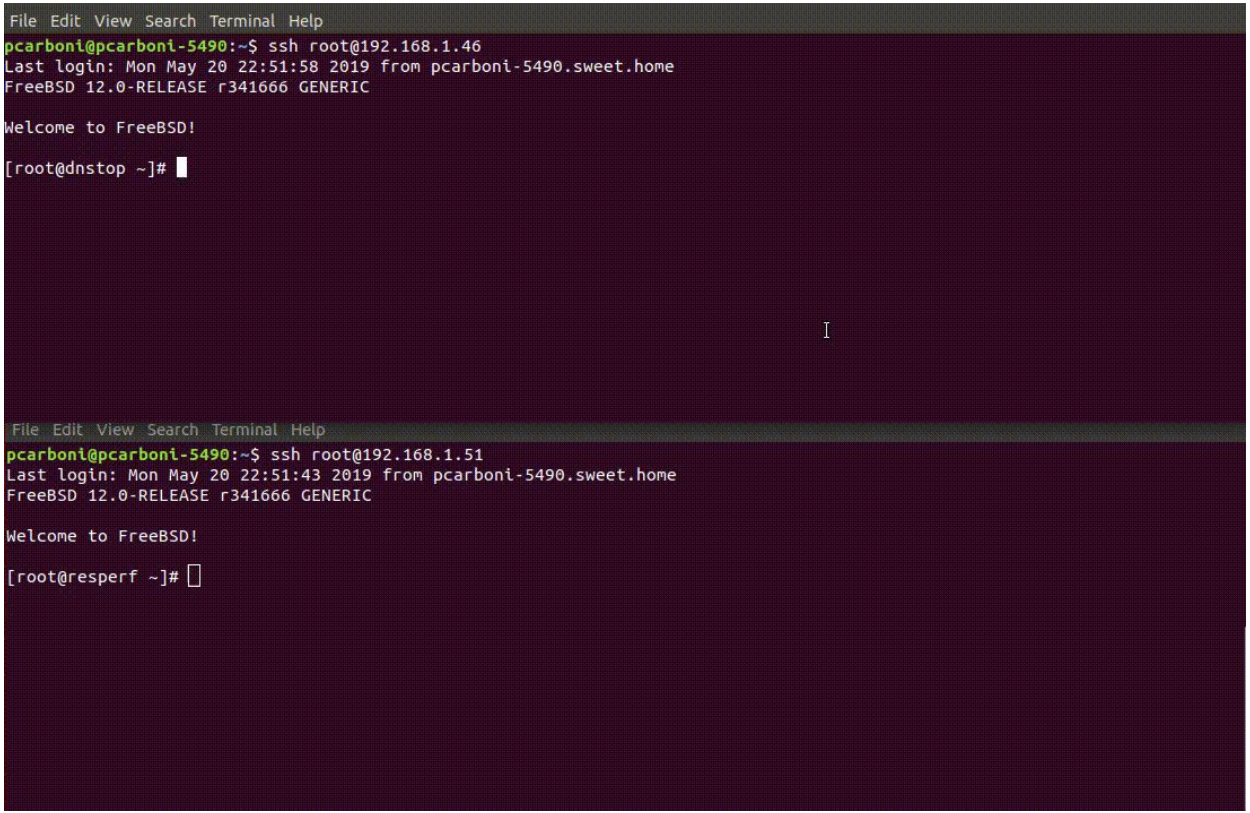
Stress testing - Using resperf (Masterplan, #6/6)

Why did I use resperf? (Seems that current dnsperf was enhanced)

- It gave me the **maximum qps allowed by random queries** by simulating a cache resolver and increasing queries quantity
- At least at that time, it had **better(objective) results vs dnsperf**.

Note that resperf is an interesting tool for simulating random queries from a desired source file with certain maximum desired.

Little demo: dnstop / dnstop in action



Initial conclusions from the lab infrastructure

- First tests were promising. Without tuning, I've got 10-15kqps
- By following **Calomel's hints about Unbound and FreeBSD**, I've ended up by doing fine tuning on network card, OS (udp, sockets, ports range, etc), and Unbound config. (**However, no DNSSEC was used**)
- My dry (but real) tests were incredible: I've got > 54kqps!
- Yes, DNS service -with high load in mind- was under control! :-)

Firing up the new DNS service

- The DNS assignment to the subscribers was (is) relatively easy.

  (Just replace the desired IP addresses into the profile and wait for the sessions until reconnect to the internet service).

- It was a matter of time (a very few hours) until the whole migration was completed successfully.

- KPIs graphics monitoring was done with a customized Cacti.

- The **dnstop** tool was my best friend while monitoring ‘live’ DNS traffic.

Conclusions (#1/3)

It should be noted that **a rapid deployment based on the lab took place** because of several factors.
(Including dns performance bottleneck).

- **Main conclusion:** Unbound running on FreeBSD provided an excellent performance without suffering any kind of stability/performance issues (kernel, tcp ip stack, process, etc)

unbound FreeBSD

More conclusions (#2/3 - Raw numbers)

- **Final deployment lasted for more than 6 months** until definitive hardware/propietary software arrived

- **Queries received** started from 80kqps, **ended up with 120kqps** distributed on 3 physical servers.

- **DNS response times** for non-cached queries were lowered to < 0.1s!)

FreeBSD®

Conclusions (#3/3 - End of “love” story)

“It’s worth to note that the queries were made from mobile subscribers to the internet!”

In summary:

The impact on the DNS service provided to customers was incredible good, and the “quick and not-so-dirty” solution was well received!

unbound FreeBSD®

Lessons learned #1/2 (Dont's)

- **Don't route your DNS traffic through a general purpose firewall** while having really high DNS traffic volume. (It didn't scale well - with NAT, timers, sockets)
- **Don't trust blindly** on the appliance datasheet values. (Make sure your KPI's have normal values).
- **Don't avoid HA DNS infrastracture.** DNS redundancy behaviour works fine, but sometimes it's better to have an DNS HA deployment due high speed requirements needs.

FreeBSD®

Lessons learned #2/2 (Do's)

- Have your **KPIs well defined (QPS, traffic, UDP traffic).** Use tools like dnstop. Stress testing is recommended too.
- Put a **dedicated LB (HW) in front of your DNS servers.** It helps with HA by reducing possible timeouts. If possible, 2 or more sites.
- **Physical servers are better**, by leveraging the whole HW resources.
- Use **scalable OS / DNS software.** It allows to do fine tuning easly while levaraging CPU cores, network HW, and optimizing DNS resolution times and protection by hardening the service.

Acknowledgements

- FreeBD project (https://www.freebsd.org)
- NLNet labs (https://www.nlnetlabs.nl/)
- Nominum (now part of Akamai) (https://ww.akamai.com)
- The Measurement Factory (http://dns.measurement-factory.com/tools/)

Special **acknowledgements to Mariusz Zaborski (@oshogbovx)** because he motivated me to send the talk to this event!

… Also a big **“thank you” to Allan Jude (@allanjude)** for corrections, suggestions, over these slides.

Questions?

Thank you!