PowerDNS dnsdist

UKNOF34

Presentation is on:
https://tinyurl.com/ukdnsdist

http://dnsdist.org/

bert.hubert@powerdns.com / @powerdns_bert

Outline

- PowerDNS/Open-Xchange/Dovecot/Bert
- Importance of DNS (do you measure it? why not?)
- What is dnsdist? why dnsdist?
- Some examples
- Live website, graphics, statistics
- Architecture
- Performance
- Status & Getting dnsdist

PowerDNS, Open-Xchange

- PowerDNS: around since 1999, open source since 2002
  - I’m the founder, these days I’m back to coding!
- Authoritative DNS: 30%-50% of all domains, 75%+ of DNSSEC
  - Few percent of all installs though
- Recursor: 100s of millions of internet users
- Actual open source company, open software, good business
- Since 2015 part of Open-Xchange, together with Dovecot

Importance of DNS

- Everything starts with a DNS lookup
- **DNS lookup slow -> everything slow**
- One of the LEAST MEASURED protocols on the internet though!
  - Nobody keeps ‘access logs’ either (changing though)
- Our observation: DNS is frequently “good enough”, but almost nobody goes for “really good”
  - “works for me!”
- What happens: 500 million users on 8.8.8.8!

Dnsdist: another layer of indirection

- "dnsdist is a highly DNS-, DoS- and abuse-aware loadbalancer. Its goal in life is to route traffic to the best server, delivering top performance to legitimate users while shunting or blocking abusive traffic."
- Swiss army knife of DNS problem solving. Add and remove bits, filter out traffic, inspect traffic live from the console
- Detect infected users, force infected users to other name servers
- Delay and ratelimit bad queries, refuse to do work for certain hosts/ domains
- Loads of statistics
- Open source, vendor neutral - it is not "PowerDNS Dist"
- **And let's not forget: very smart load balancing**

The story of dnsdist

- Started out as a need to do “dnsdist listen-ip destip-1 destip-2”
  - Simple query spreading w/o hassle, also just forwarding
  - Around since 2013
- In 2014 while debugging with a large customer, we found they were willing & able to switch out PowerDNS versions at the drop of a hat since they were comfortable with their loadbalancer
- **Asked around, no one else was happy with their DNS load balancer solution**

Existing load balancers

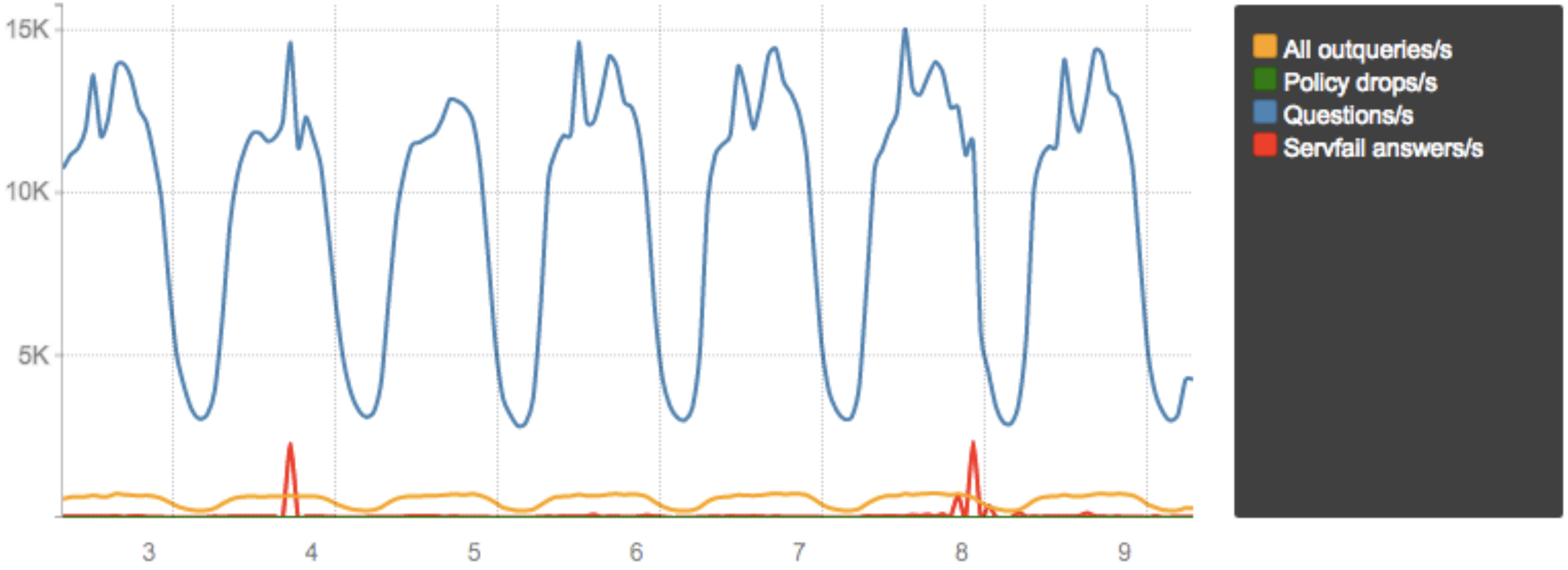
- Most (HW) load balancers know about http, https, imap etc.
  - DNS is sufficiently different that it is hard to treat it as ‘a weird kind of web’, so many vendors mess it up
- Plus the quaint observation that a busy nameserver is a happy name server
  - Caches HOT!
- Leads to need for a ‘concentrating load balancer’: as much traffic on as little servers as possible
  - Exactly the reverse of http etc

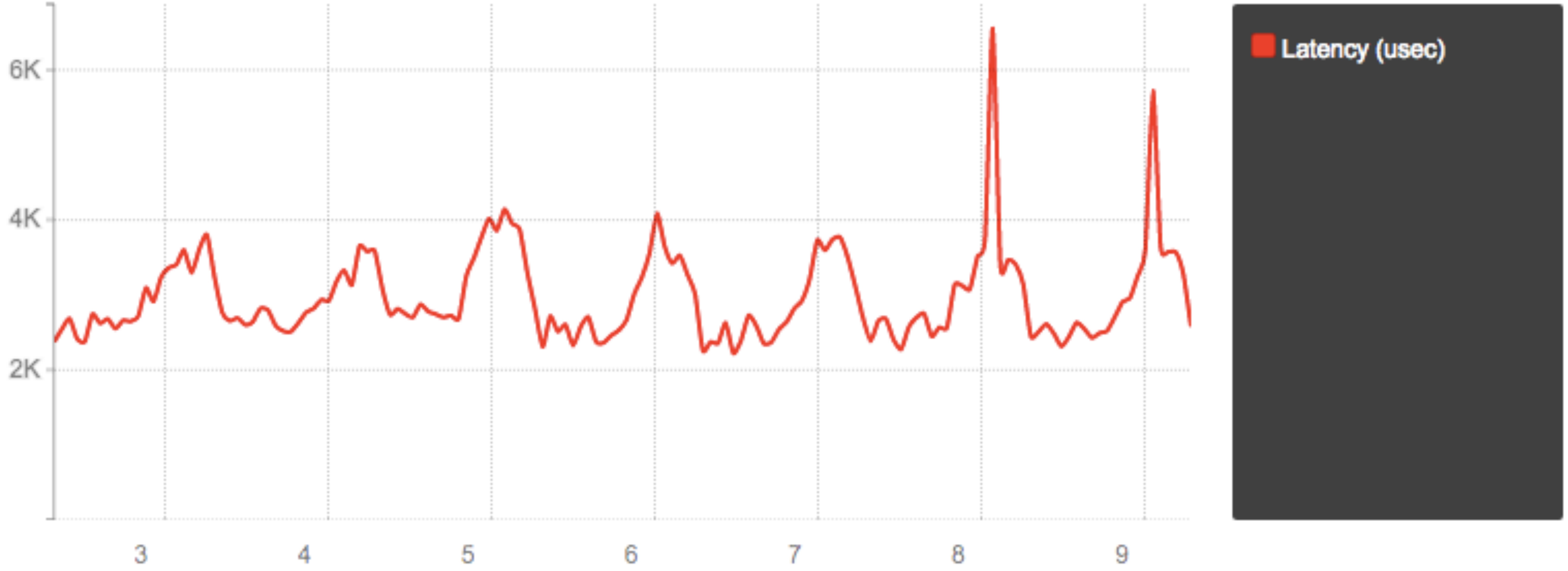
dnsdist: a smart “DNS Delivery Controller”

- Runtime configurable from console (accessible remotely, tab-completing interface)
  - Console & configuration file actually Lua
  - **Most configurations operate Lua-free at runtime**
- Host of built-in load balancing/blocking/shunting/shaping policies (C++), custom policies can be written in Lua (plenty fast)
  - LuaJIT
  - “Million QPS”
- Built-in webserver & API, plus Graphite/Metronome graphs
  - HTTP RESTful: **does not read files from disk ever**
- Provides features ranging from simple round robin load balancing to quarantining of infected customers
- **Vendor-neutral open source - please join in!**

Some real life tests

- With two companies we tested shutting down all their nameservers but a few, leading to lots of traffic going to one server
- In all cases, we observed lower query/response latencies and far lower cache miss rates (±50% lower)
    - Happier customers
- We also observed only minor increases in CPU load, very much sub-linear to the many-fold traffic increase
    - One name server doing millions of cable modems
    - One name server doing 700k domains with online signing
- “We have a winner!”





View all this on https://metronome1.powerdns.com/

Scenarios

- Legacy nameservers, want to get statistics
  - And realtime inspection
- Need to add IPv6 service
  - or: “Just need to move traffic to another server”
- Adding features to legacy DNS: TCP to ANY, Views
- Improve service through query concentration
- Send abusive traffic to “abuse pool”
- Split DNSSEC traffic to DNSSEC servers
  - .. strip DNSSEC when it doesn’t work ..

Scenarios

- Add EDNS Client Subnet tag with original IP address to survive CGNAT and still have per-user settings
- Compensate for bugs in 100k large CPE deployment
- Fix up case sensitive backends / clients
- Use regular expressions to route roaming users to the right EPC data
- Client originated DoS worries: limit each host QPS or per /64
- Large scale DoS worries: absorb & filter at million QPS rates

dnsdist simplest config

- **No config**:
  - # dnsdist -l 8.8.8.8 208.67.222.222 2620:0:ccc::2 2620:0:ccd::2
  - Will listen on port 53, serve for RFC1918, distribute queries to Google & OpenDNS using a sensible load balancing policy
    - Most queries to the most unloaded, fastest server

dnsdist basic config

```
setLocal("130.161.252.29:53")

addACL(“130.161.0.0/24")  — setACL would’ve taken out RFC1918

newServer{address="192.168.1.2", qps=1000, order=1}

newServer{address="192.168.1.79:5300", qps=10000, order=2}

newServer{address="127.0.0.1:5300", order=3}

setServerPolicy(firstAvailable)
```

dnsdist basic config

```
# dnsdist --config=basic.conf
Listening on 130.161.252.29:53
Marking downstream 192.168.1.2:53 as 'up'
Marking downstream 192.168.1.79:5300 as 'up'
Marking downstream 127.0.0.1:5300 as 'down'
> showServers()
#   Name                 Address                       State     Qps    Qlim Ord Wt    Queries   Drops Drate   Lat Pools
0                        192.168.1.2:53                   up     0.0    1000   1  1          0       0   0.0   0.0
1                        192.168.1.79:5300                up     0.0   10000   2  1          0       0   0.0   0.0
2                        127.0.0.1:5300                 down     0.0       0   3  1          0       0   0.0   0.0
All                                                              0.0                         0       0
```

dnsdist basic config

```
> showServers()
#   Name                 Address                       State     Qps    Qlim Ord Wt    Queries   Drops Drate   Lat Pools
0                        192.168.1.2:53                   up     0.0    1000   1  1          0       0   0.0   0.0
1                        192.168.1.79:5300                up     0.0   10000   2  1          0       0   0.0   0.0
2                        127.0.0.1:5300                 down     0.0       0   3  1          0       0   0.0   0.0
All                                                              0.0                         0       0
> getServer(1):setDown()
> showServers()
#   Name                 Address                       State     Qps    Qlim Ord Wt    Queries   Drops Drate   Lat Pools
0                        192.168.1.2:53                   up     0.0    1000   1  1          0       0   0.0   0.0
1                        192.168.1.79:5300              DOWN     0.0   10000   2  1          0       0   0.0   0.0
2                        127.0.0.1:5300                 down     0.0       0   3  1          0       0   0.0   0.0
All                                                              0.0                         0       0
```

dnsdist: pretty stuff

```
controlSocket(“0.0.0.0") – for the console
setKey(“MXNeLFWHUe4363BBKrY06cAsH8NWNb+Se2eXU5+Bb74=") – for crypto
webserver("0.0.0.0:8083", “geheim2”)– instant webserver
carbonServer(“2a02:2770:8::2635:0:1") – send our statistics here
```

POWERDNS
QPS / SERVFAILPS
5
CACHE HITRATE / CPU %
4
2

dnsdist 1.0.0
dnsdist comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it according to the terms of the GPL version 2.

Uptime: 4 minutes, Number of queries: 75 (0 qps), ACL drops: 0, Dynamic drops: 0, Rule drops: 0, Blockfilter drops: 0
Average response time: 0.10 ms, CPU Usage: 2.00%, Cache hitrate: 0%, Server selection policy: leastOutstanding
Listening on: 82.94.213.34:53, [2001:888:2000:1d::2]:53, ACL: 0.0.0.0/0, ::/0

#Name	Address	Status	Latency	Queries	Drops	QPS	Out	Weight	Order	Pools
0	127.0.0.1:53	up	1	34	0	0	0	1	1	
1	45.55.10.200:53	up	5	5	0	0	0	1	1	pdnsauth
2	188.166.104.87:53	up	0	16	0	0	0	1	1	pdnsauth

#	Rule	Action	Matches
0	qname==dnsdist.org., dnsdist.net., dnsdist.com., powerdns.com.	to pool pdnsauth	41

Dyn blocked netmask	Seconds	Blocks	Reason
	No dynamic blocks active		

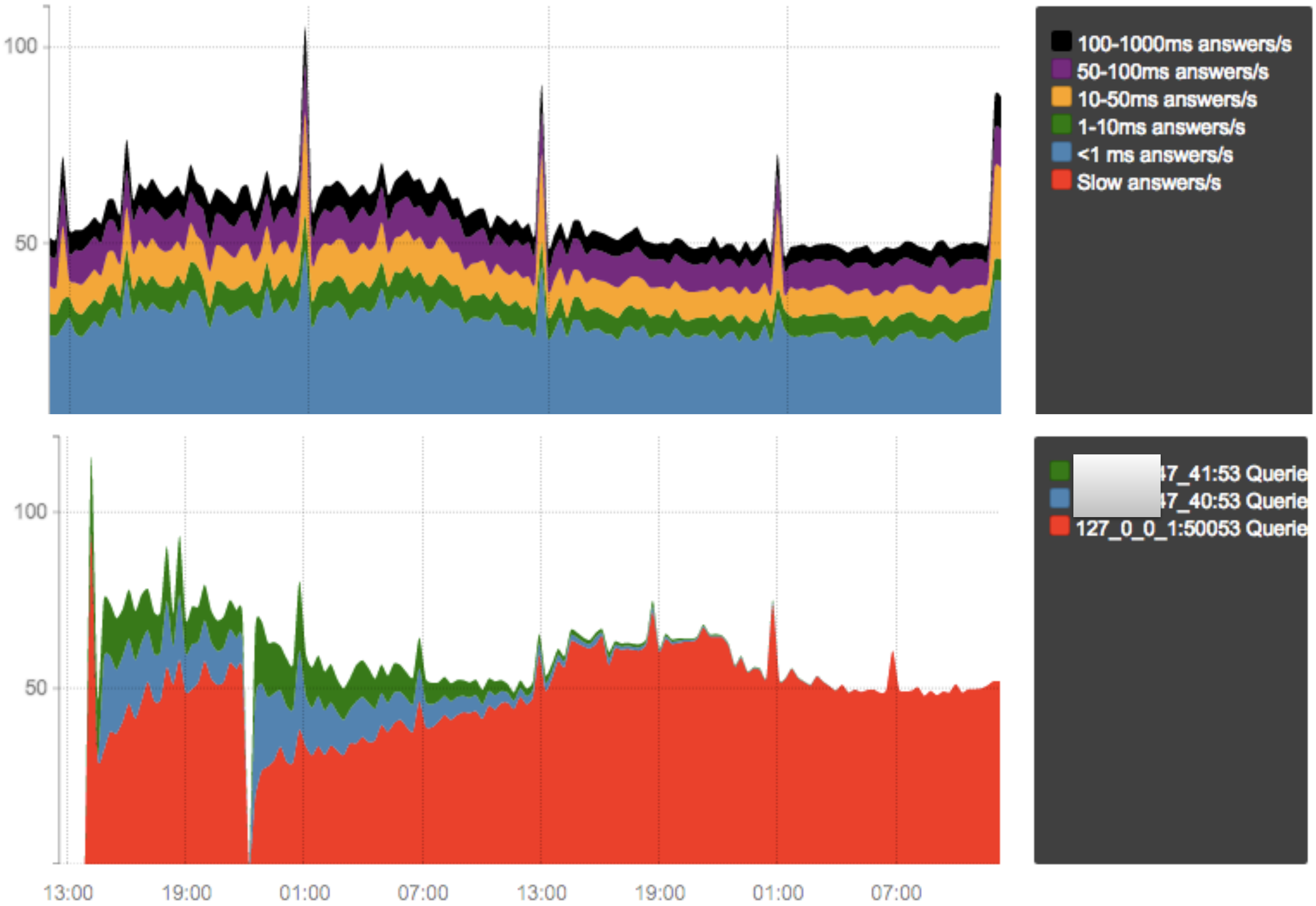
http://i.imgur.com/qoyfJRy.gif

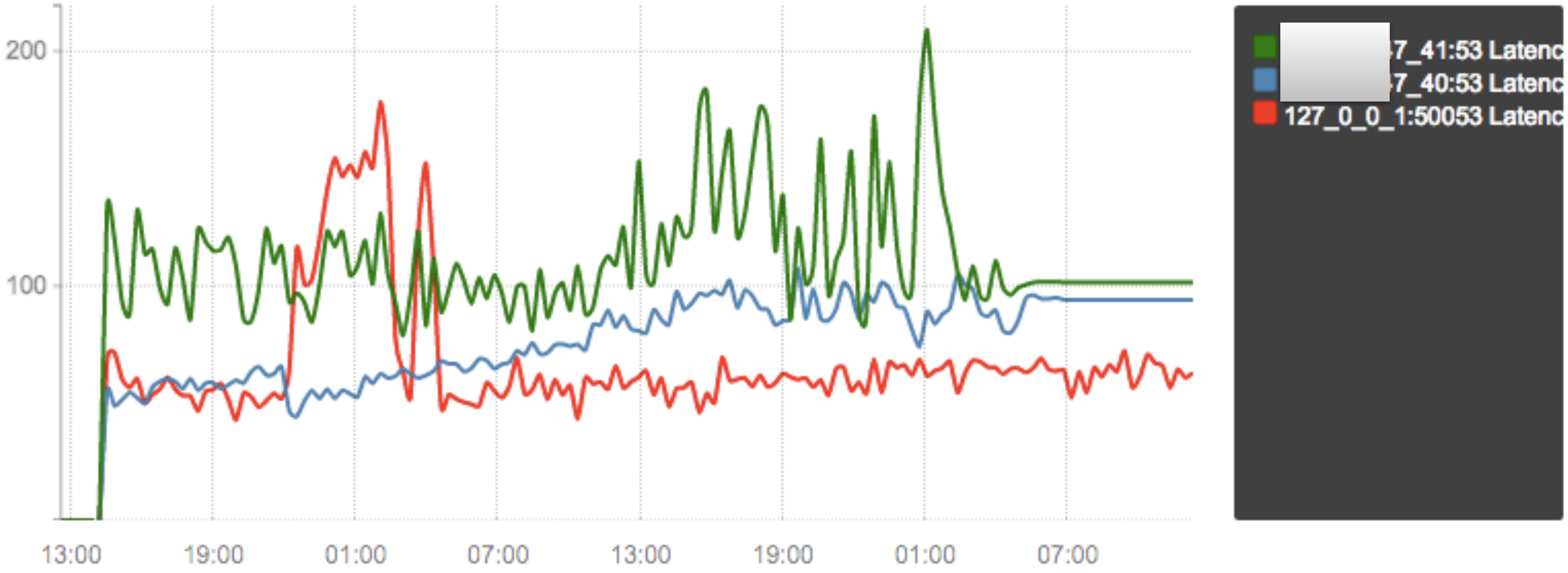
Console & Configuration

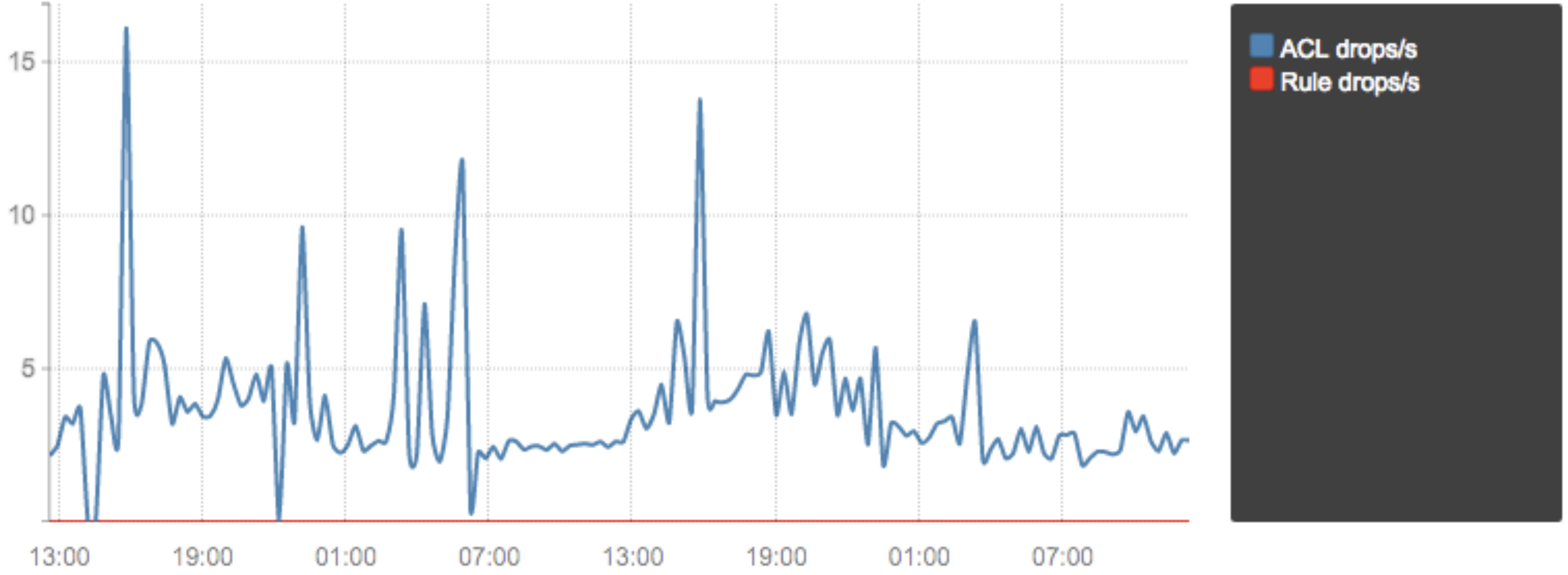
- Connect to the live console over an encrypted connection
  - NaCl/libsodium
- Can also execute commands with ‘dnsdist -e’
- Any commands with side-effects get stored
- Run ‘delta()’ at any time to figure out what changed compared to the original configuration
  - Output can be added to end of dnsdist.conf!

Statistics

- We output statistics in the ‘carbon’ format as used by Graphite
- Sometimes we can’t help ourselves and reimplement the wheel
- “Metronome” is our very simple “works out of the box” mini-Graphite
  - Support Auth, Recursor, dnsdist & “System”
- **Public instance available so we can debug you!**

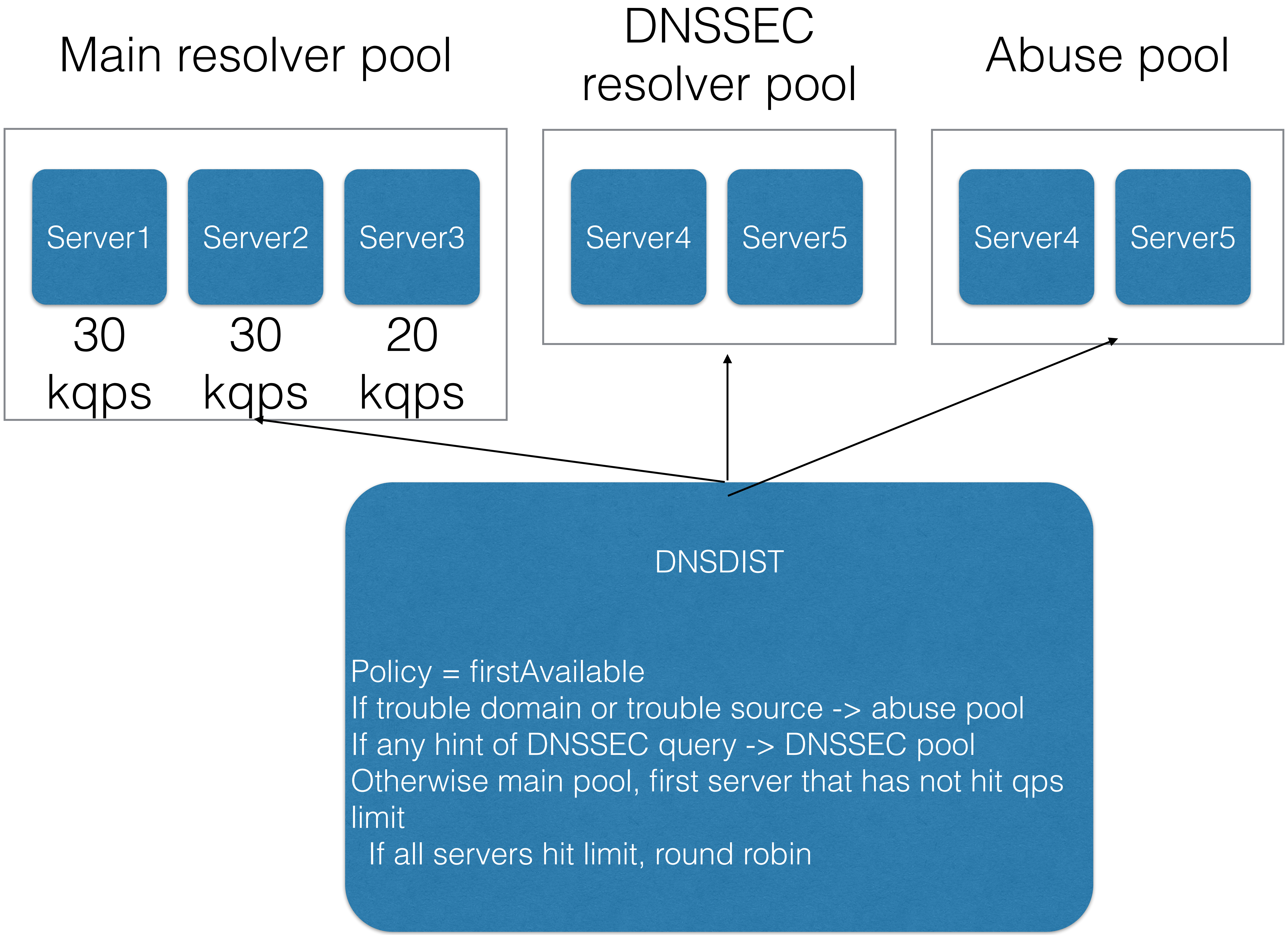

100
50
🟫 100-1000ms answers/s
🟪 50-100ms answers/s
🟧 10-50ms answers/s
🟩 1-10ms answers/s
🟦 <1 ms answers/s
🟥 Slow answers/s
100
50
🟩 [redacted]47_41:53 Querie
🟦 [redacted]47_40:53 Querie
🟥 127_0_0_1:50053 Querie
13:00
19:00
01:00
07:00
13:00
19:00
01:00
07:00


200
100
13:00
19:00
01:00
07:00
13:00
19:00
01:00
07:00
🟩 [redacted]7_41:53 Latenc
🟦 [redacted]7_40:53 Latenc
🟥 127_0_0_1:50053 Latenc


15
10
5
13:00
19:00
01:00
07:00
13:00
19:00
01:00
07:00
🟦 ACL drops/s
🟥 Rule drops/s

Policies

- firstAvailable: Pick first server that has not exceeded its QPS limit, ordered by the server 'order' parameter
- wrandom: Weighted random over available servers, based on the server 'weight' parameter
  - whashed: same thing but ‘sticky’
- roundrobin: Simple round robin over available servers
- **leastOutstanding: Send traffic to downstream server with least outstanding queries, with the lowest 'order', and within that the lowest recent latency**
- Lua: go wild!



Main resolver pool

Server1 30 kqps

Server2 30 kqps

Server3 20 kqps

DNSSEC resolver pool

Server4

Server5

Abuse pool

Server4

Server5

DNSDIST

Policy = firstAvailable
If trouble domain or trouble source -> abuse pool
If any hint of DNSSEC query -> DNSSEC pool
Otherwise main pool, first server that has not hit qps limit
  If all servers hit limit, round robin

DoS undisconnectable subscriber usecase

- DoS attacks of the algorithmic kind - don’t kill you with bandwidth, do cause outgoing traffic that does, do cause degraded performance
- Frequently blocked with complicated iptables rules, or deployed custom zones within name servers
- Option in dnsdist: move senders of harmful DNS traffic to dedicated servers
  - Where they only ‘soil their own nest’

Campus usecase: QPS Limit map

- Global QPS limits, per-server QPS limit
- Sometimes attacks come from single users
- MaxQPSLimit():
  - addAction(MaxQPSIPRule(5, 24, 64), DropAction())
- 5 queries/second, grouped by /24 on IPv4 and by /64 on IPv6

protobuf: logging all or part of traffic

- DNS queries are extremely valuable for security research
    - Either in-house
    - Or for finding infected ISP subscribers
- “Various other reasons”
- dnsdist logs protobuf messages for each query & response over TCP/IP through a very light-weight mechanism
- Receive traffic using ‘xinetd’ or PowerDNS Platform
    - For long term storage & search on commodity hardware

Cache: Performance & Uptime

- High-performance cache, delivering hundreds of thousands of answers/s per core
- Can be used to scale up poor overloaded backend
- Legitimate strategy to use more processing power by splitting the work
    - Backend 100% focused on cache misses
    - Frontend dnsdist .. just keeps on serving
    - Keeps you alive even under DoS since good answers continue to come out
- Optional: persistent mode if backends truly down

Generic rule/action engine

- Full Lua access to all packets
  - Upside: 100% flexibility,
  - downside: slowdown at very high query rates. Also, need to program
- In C++:
  - Less dynamic, extremely fast
  - Actions: drop traffic, change traffic, redirect etc

Examples

- addPoolRule({"ezdns.it.", "xxx."}, “abuse")
- addQPSPoolRule("com.", 100, “abuse")
- addDomainBlock("powerdns.org.")
- addLuaAction("192.168.1.0/24", luarule)
- **addDisableValidationRule({“servfail.nl”, “1.0.0.0/8”}**
- showRules()

Rules

- Source address, query type, query class, query domain
- QPS Limit total, QPS limit per source IP or netmask
- Regular Expression, RE2
- DNSSEC on/off
- Protocol selector
- And, Or, Not
- Lua Rule

Actions

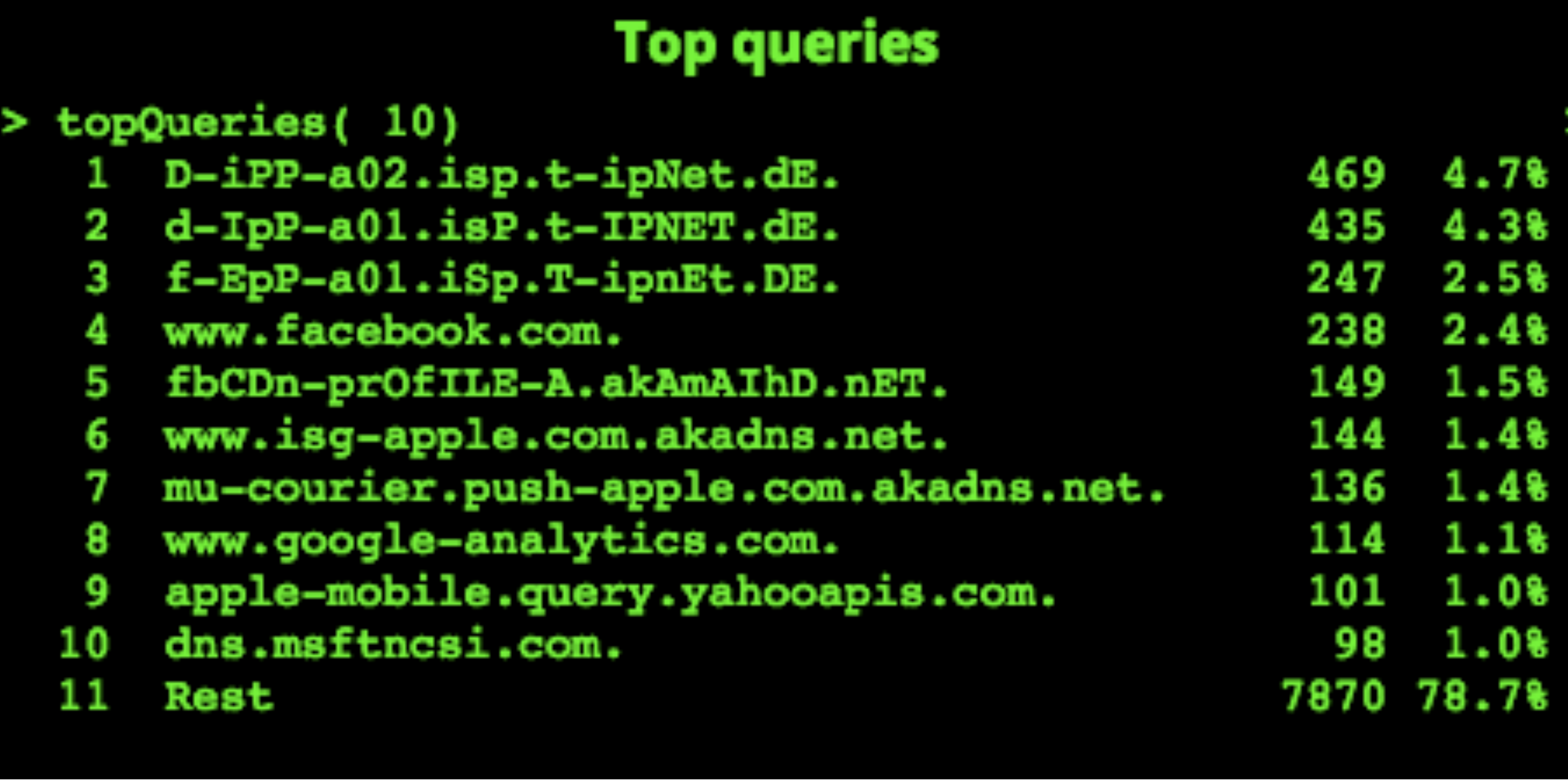
- Drop
- Route to pool
- Truncate (TC=1)
- Issue Servfail, Notimp, Refused answer
- Custom answer generation, including ‘real’ NXDOMAIN, CNAMEs etc
- Delay response by n milliseconds
- Drop RD or CD or DNSSEC bits
- Add MAC address for per-device settings
- Log query to TCP/IP Protobuf host

Dynamic rules

- If defined, every second dnsdist will call the **maintenance()** function
- This function has access to query ringbuffers & helpers that provide statistical summaries of ringsbuffers
- Can institute dynamic blocks which expire automatically
  - Excessive queries, timeouts, servfails, NXDOMAINS
- Can (re)configure shaping and abuse pools
  - “Your abusive traffic goes -> there”

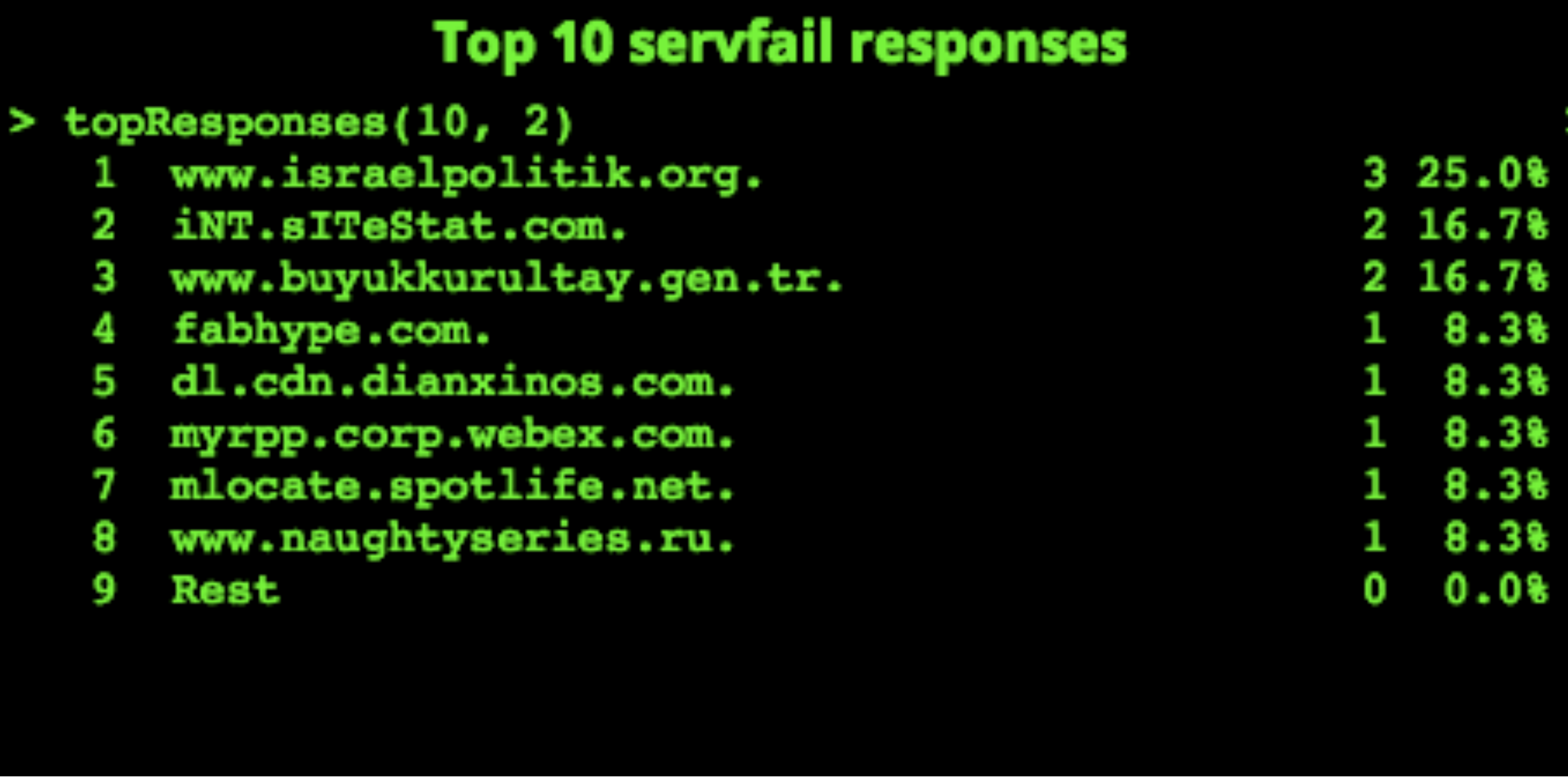
Other things we added

- Live traffic inspection: Top-N queries, top-N clients, top-N servfail generating queries, top-N **servfail** generating domains & **clients**
- Latency distribution histogram
- A substantial Lua runtime which should facilitate ‘everything’ for those that need flexibility
- You can do “everything”
  - Want to block traffic from prime number domains? GO! (don’t)



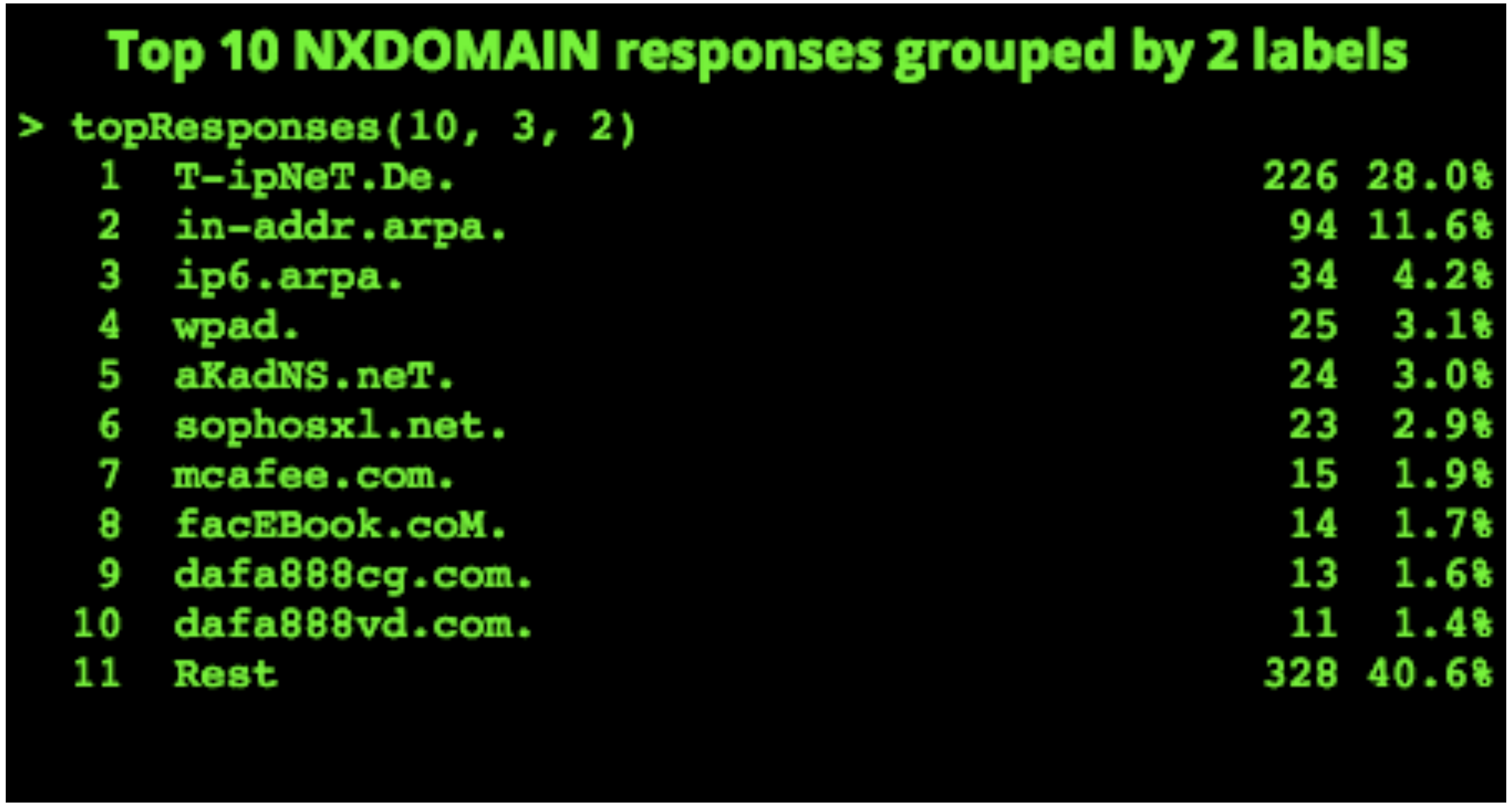
Top queries

```
> topQueries( 10)
   1  D-iPP-a02.isp.t-ipNet.dE.                  469   4.7%
   2  d-IpP-a01.isP.t-IPNET.dE.                  435   4.3%
   3  f-EpP-a01.iSp.T-ipnEt.DE.                  247   2.5%
   4  www.facebook.com.                          238   2.4%
   5  fbCDn-prOfILE-A.akAmAIhD.nET.              149   1.5%
   6  www.isg-apple.com.akadns.net.              144   1.4%
   7  mu-courier.push-apple.com.akadns.net.      136   1.4%
   8  www.google-analytics.com.                  114   1.1%
   9  apple-mobile.query.yahooapis.com.          101   1.0%
  10  dns.msftncsi.com.                           98   1.0%
  11  Rest                                      7870  78.7%
```



Top 10 servfail responses

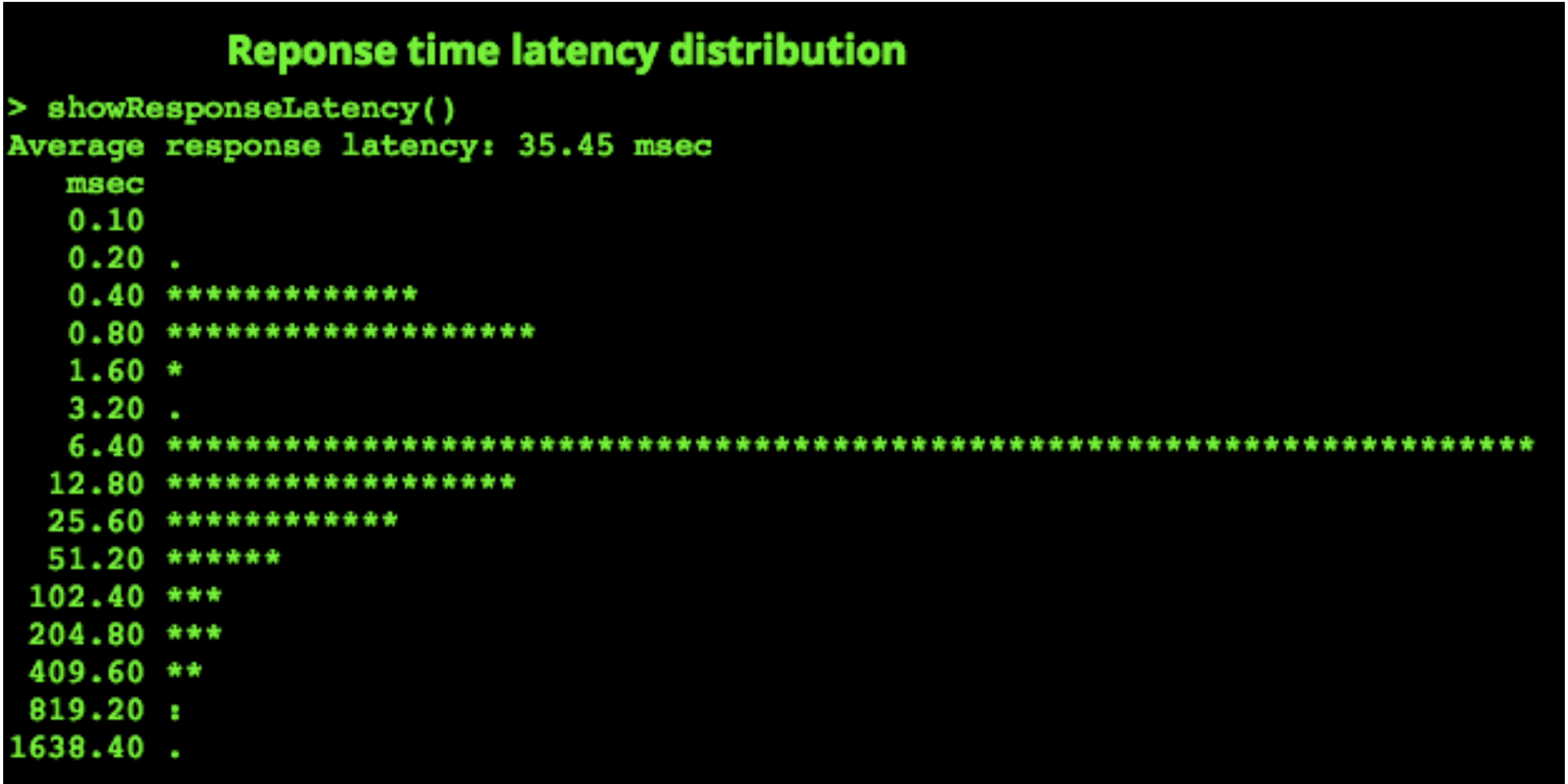
```
> topResponses(10, 2)
   1  www.israelpolitik.org.                                  3 25.0%
   2  iNT.sITeStat.com.                                       2 16.7%
   3  www.buyukkurultay.gen.tr.                               2 16.7%
   4  fabhype.com.                                            1  8.3%
   5  dl.cdn.dianxinos.com.                                   1  8.3%
   6  myrpp.corp.webex.com.                                   1  8.3%
   7  mlocate.spotlife.net.                                   1  8.3%
   8  www.naughtyseries.ru.                                   1  8.3%
   9  Rest                                                    0  0.0%
```



Top 10 NXDOMAIN responses grouped by 2 labels

```
> topResponses(10, 3, 2)
   1  T-ipNeT.De.                          226  28.0%
   2  in-addr.arpa.                         94  11.6%
   3  ip6.arpa.                             34   4.2%
   4  wpad.                                 25   3.1%
   5  aKadNS.neT.                           24   3.0%
   6  sophosxl.net.                         23   2.9%
   7  mcafee.com.                           15   1.9%
   8  facEBook.coM.                         14   1.7%
   9  dafa888cg.com.                        13   1.6%
  10  dafa888vd.com.                        11   1.4%
  11  Rest                                 328  40.6%
```

I LOVE STATISTICS

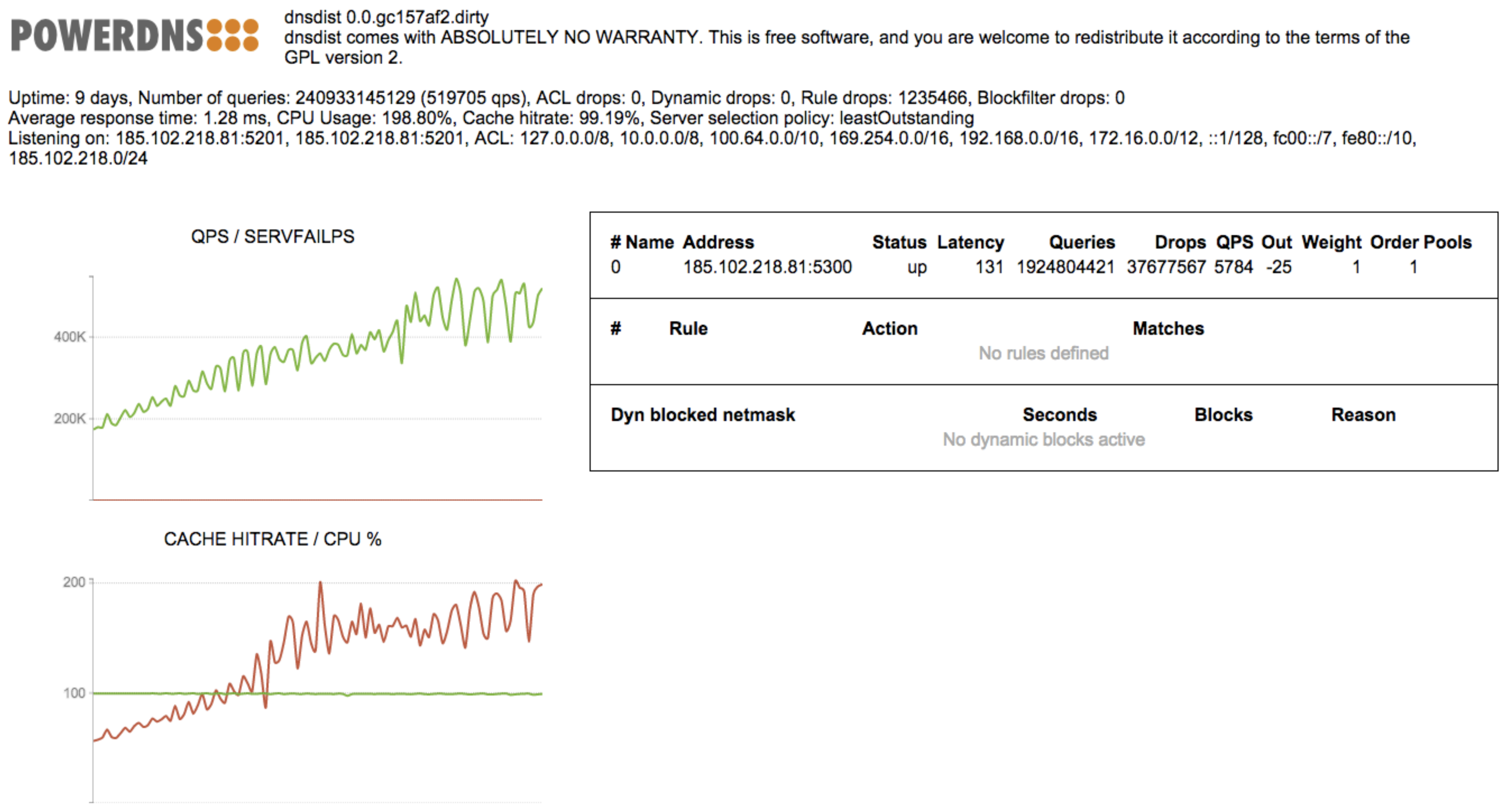


Reponse time latency distribution

```
> showResponseLatency()
Average response latency: 35.45 msec
   msec
   0.10
   0.20 .
   0.40 *************
   0.80 *******************
   1.60 *
   3.20 .
   6.40 ***********************************************************************
  12.80 ******************
  25.60 ************
  51.20 ******
 102.40 ***
 204.80 ***
 409.60 **
 819.20 :
1638.40 .
```

Performance

- Depends on configuration of course
- Typical: several hundred thousands queries/core
  - Linear scaling with SO_REUSEPORT
  - Million QPS on single server has been measured
- Usually more than you need
- **Has displaced Arbor at one deployment**



dnsdist 0.0.gc157af2.dirty
dnsdist comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it according to the terms of the GPL version 2.

Uptime: 9 days, Number of queries: 240933145129 (519705 qps), ACL drops: 0, Dynamic drops: 0, Rule drops: 1235466, Blockfilter drops: 0
Average response time: 1.28 ms, CPU Usage: 198.80%, Cache hitrate: 99.19%, Server selection policy: leastOutstanding
Listening on: 185.102.218.81:5201, 185.102.218.81:5201, ACL: 127.0.0.0/8, 10.0.0.0/8, 100.64.0.0/10, 169.254.0.0/16, 192.168.0.0/16, 172.16.0.0/12, ::1/128, fc00::/7, fe80::/10, 185.102.218.0/24

#	Name	Address	Status	Latency	Queries	Drops	QPS	Out	Weight	Order	Pools
0		185.102.218.81:5300	up	131	1924804421	37677567	5784	-25	1	1	

#	Rule	Action	Matches

No rules defined

Dyn blocked netmask	Seconds	Blocks	Reason

No dynamic blocks active

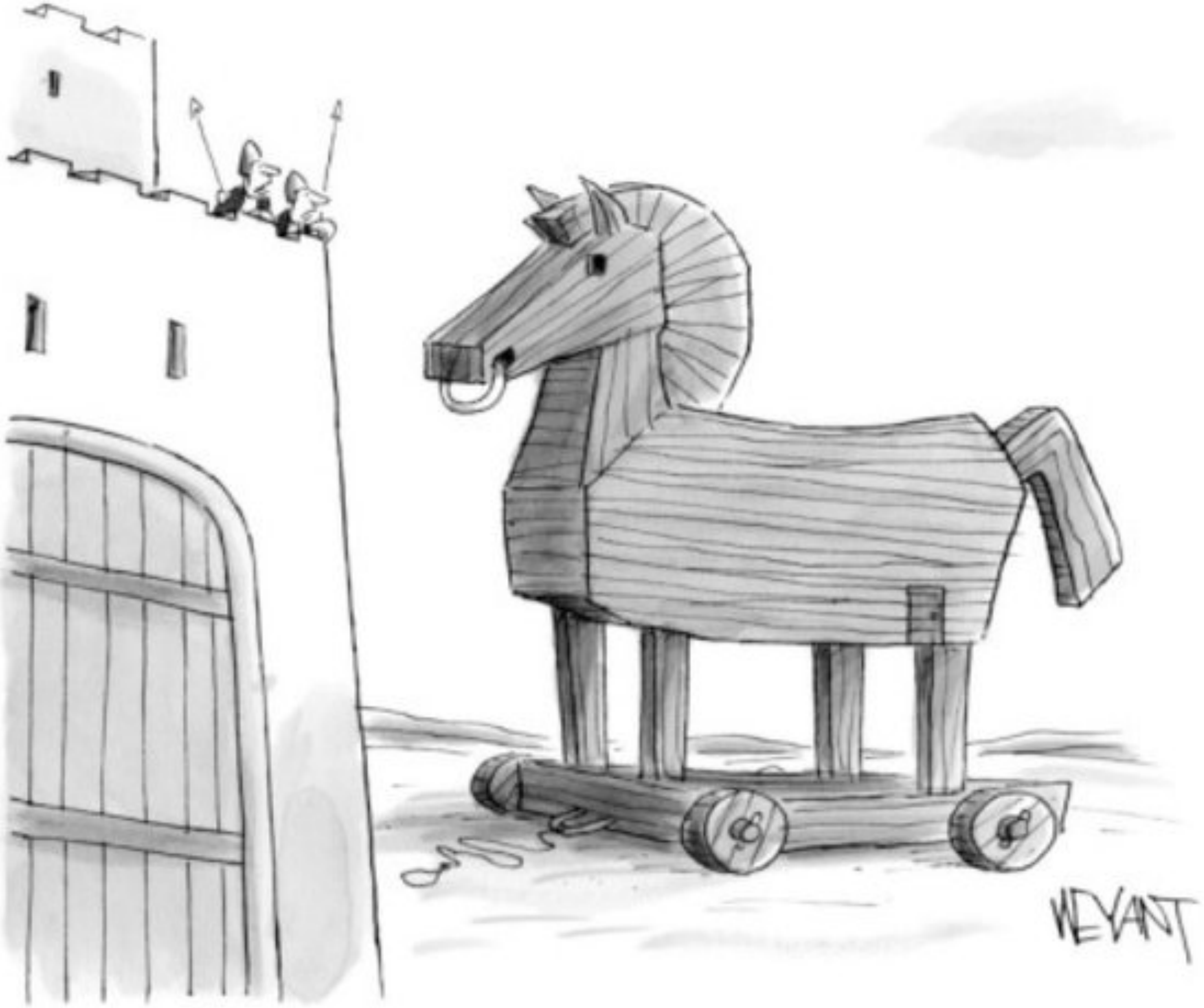
Status & Getting it

- When will dnsdist 1.0 be released?
  - **NOW!**
- Already powers several ISPs, a nation wide cell phone carrier, a bunch of ccTLDs/newTLDs
  - Please raise hands!
  - Made possible by our telemetric monitoring & heavy support promise
- Latest & greatest: https://repo.powerdns.com/ (tar, deb, yum, various platforms)

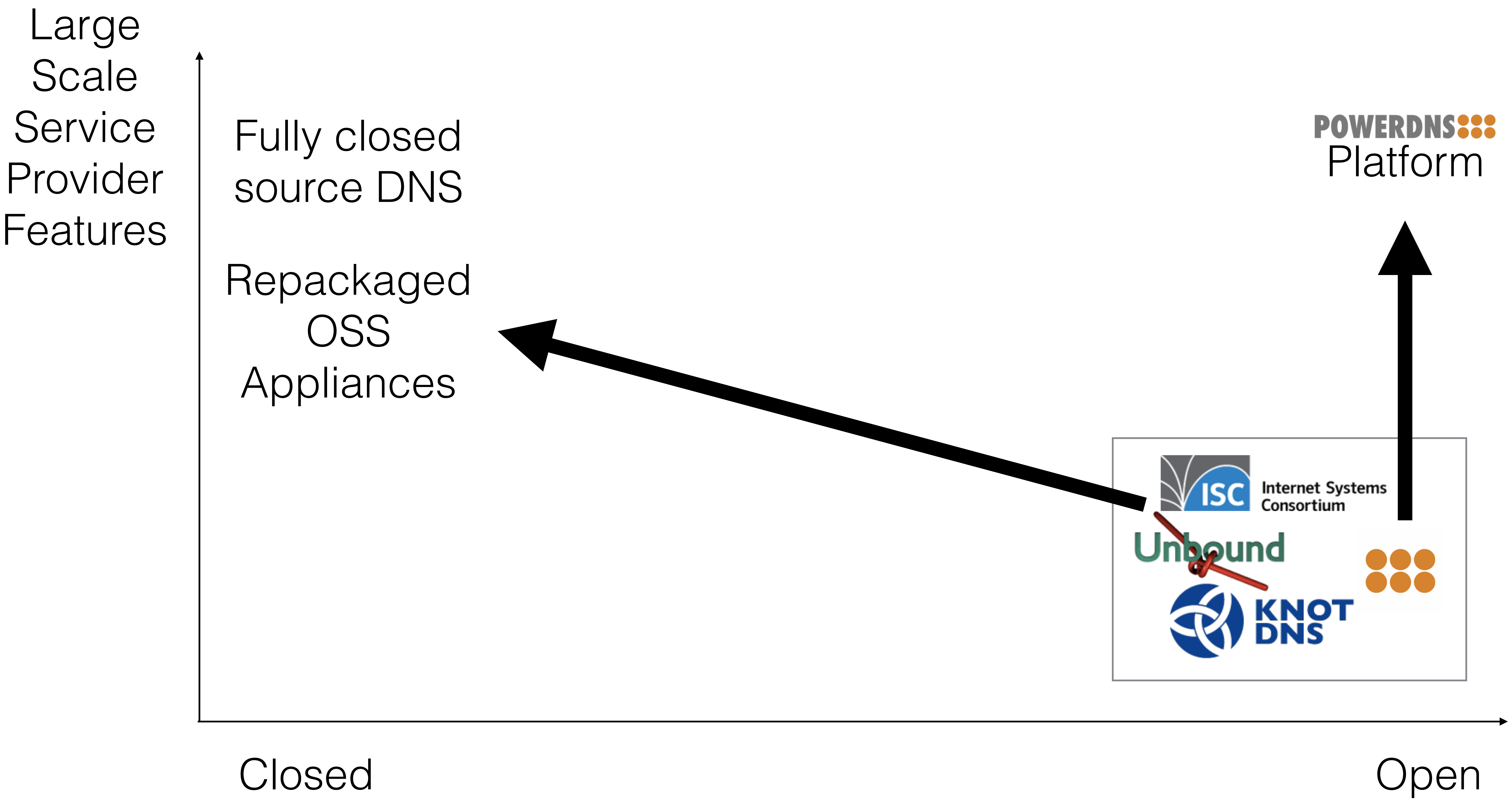
Summarizing: dnsdist

- Modern UNIX daemon
  - C++ 2011 for speed, Lua(JIT) for flexibility
- Runtime & realtime console
- RESTful HTTP based API, built-in webserver
- Very complete telemetry / statistics
  - About experienced service level
  - Resource utilisation (file descriptors, “real” memory use, CPU use)
  - Downstream health
- “Does everything with your DNS”

Beware of geeks bearing gifts



"It runs on an Arduino."



Large Scale Service Provider Features

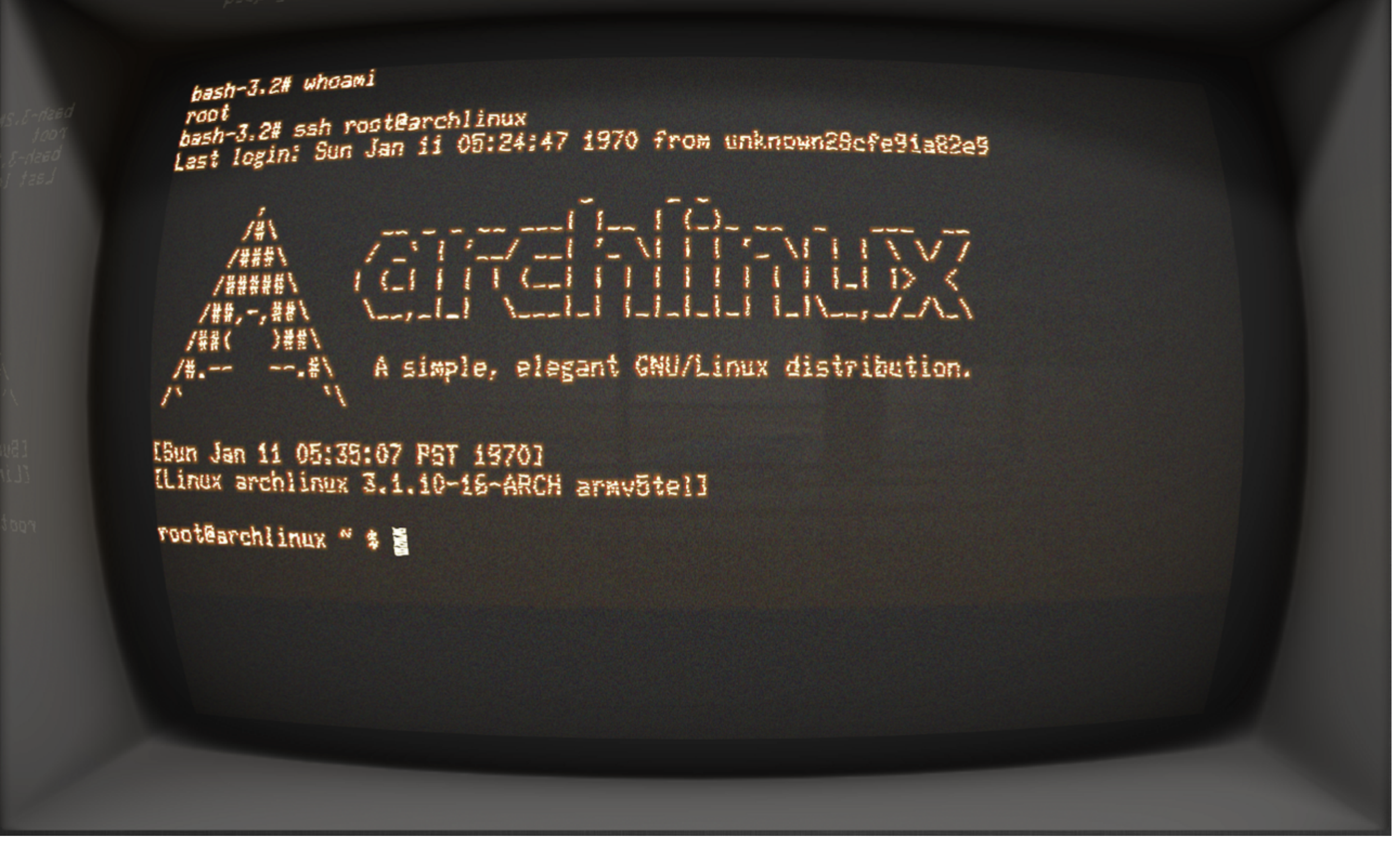
Fully closed source DNS

Repackaged OSS Appliances

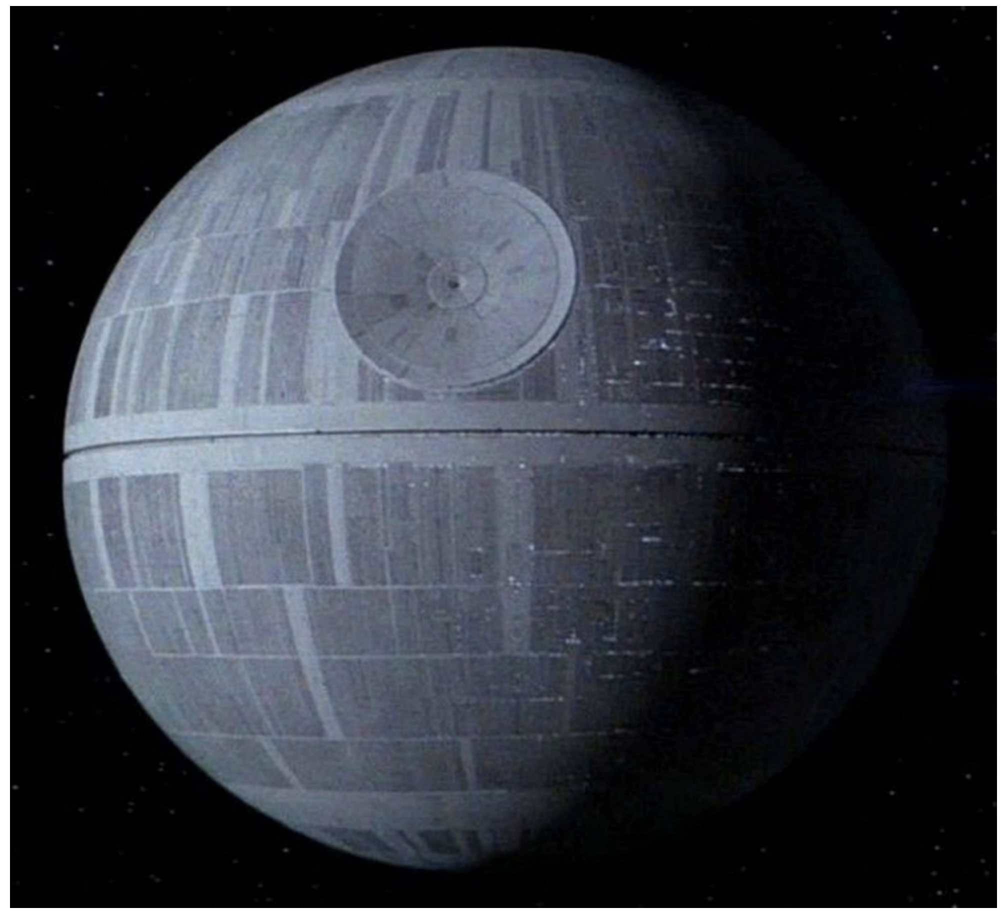
POWERDNS Platform

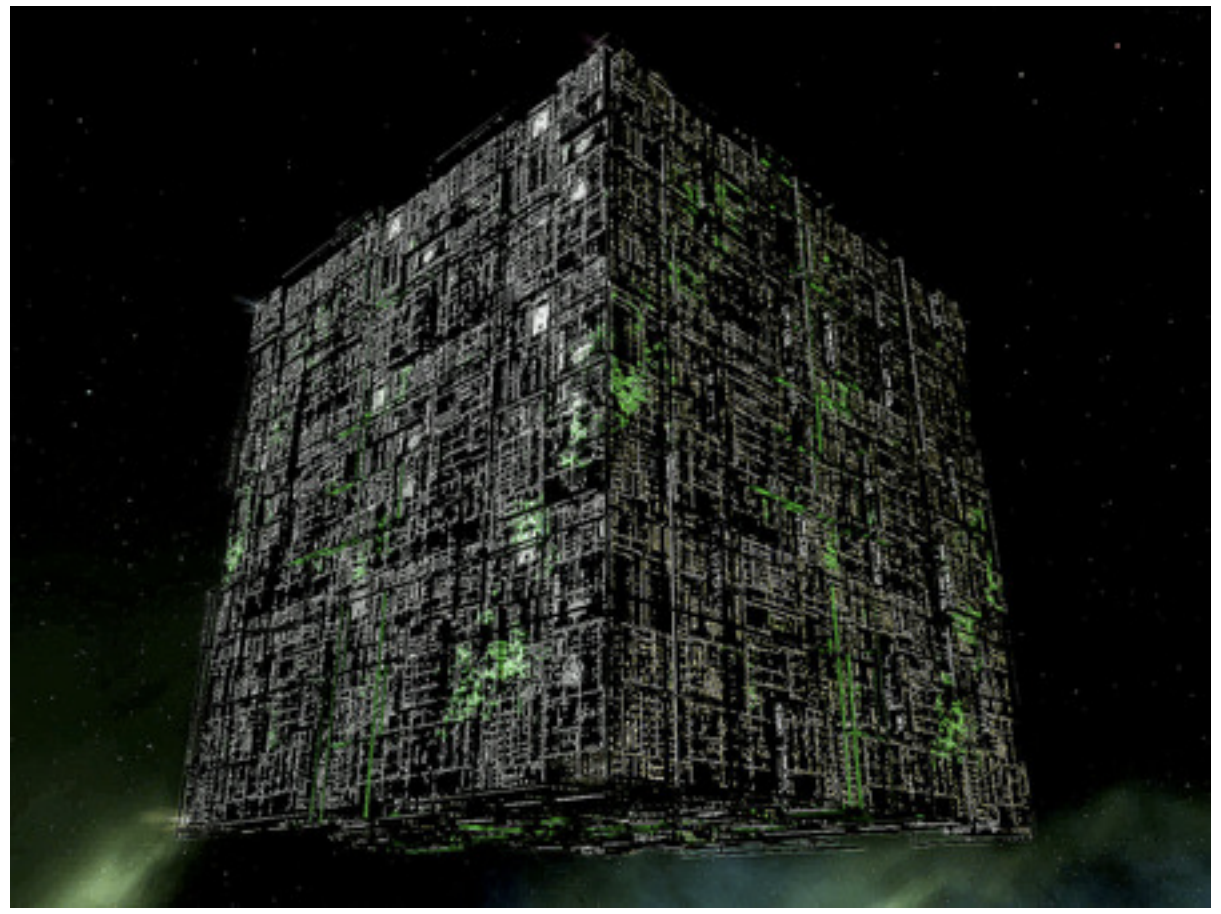
Closed

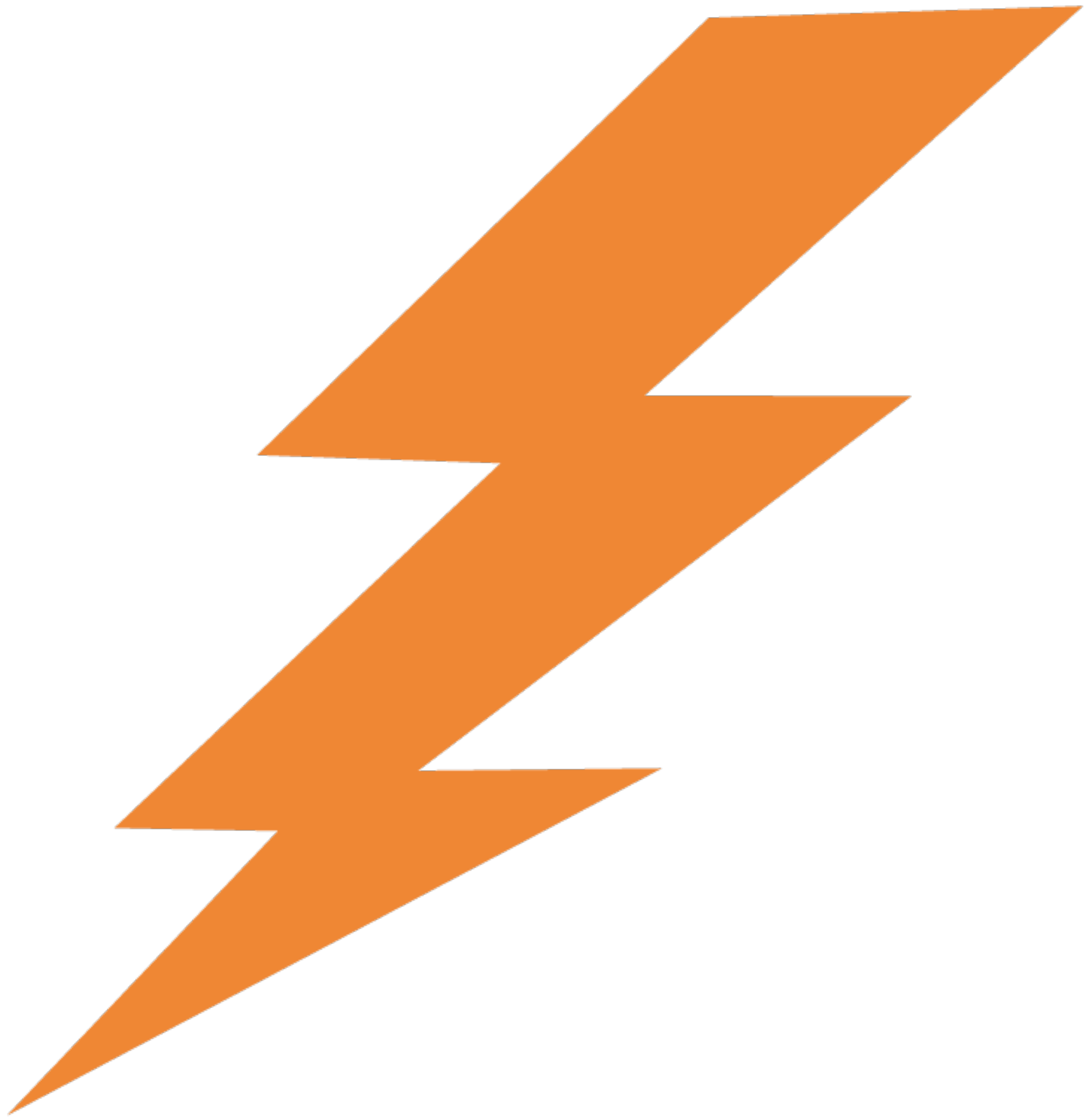
Open

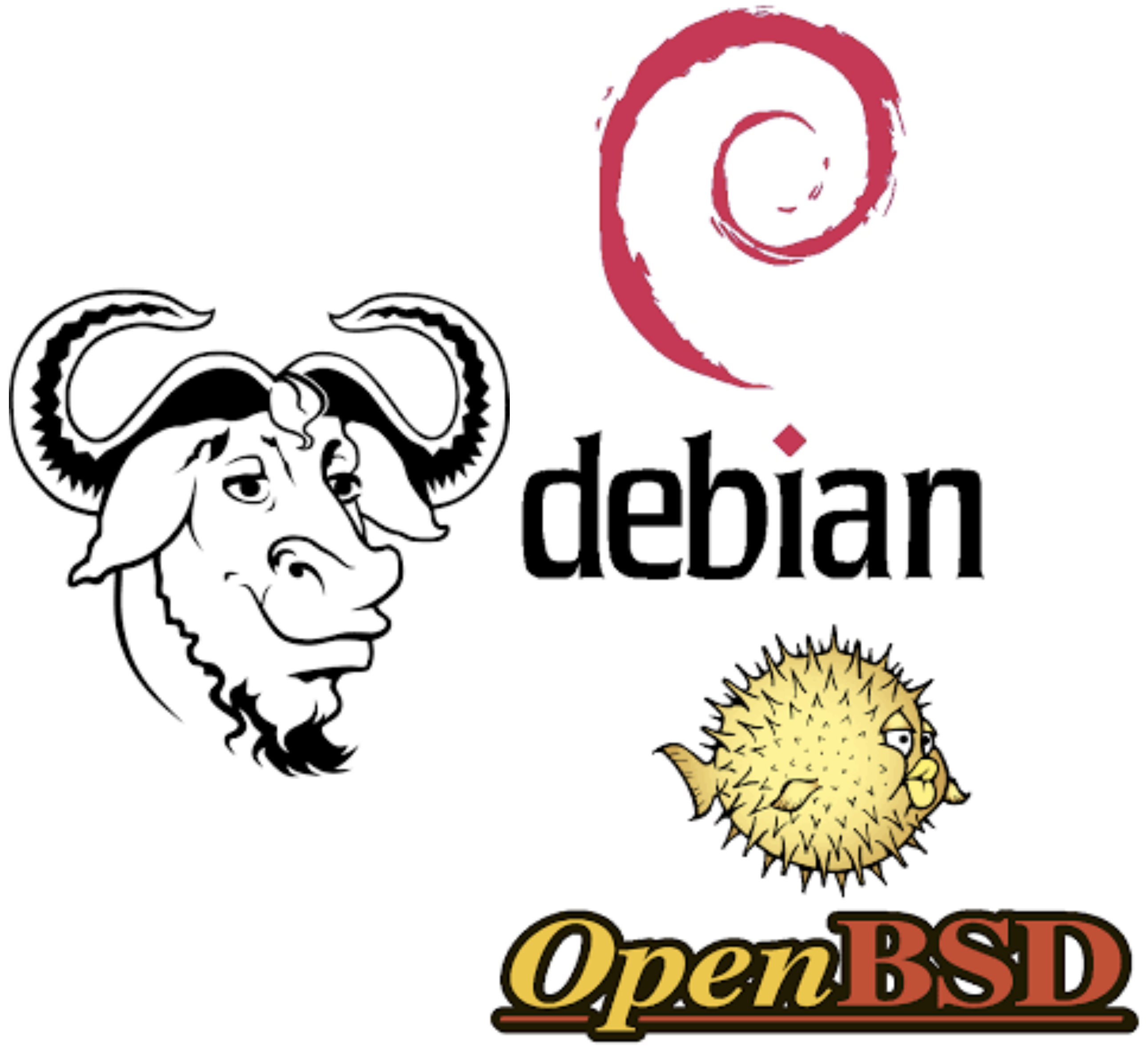


```
bash-3.2# whoami
root
bash-3.2# ssh root@archlinux
Last login: Sun Jan 11 05:24:47 1970 from unknown28cfe91a82e5

       ,                       _     _ _
      /#\      __ _ _ __ ___| |__ | (_)_ __  _   ___  __
     /###\    / _` | '__/ __| '_ \| | | '_ \| | | \ \/ /
    /#####\  | (_| | | | (__| | | | | | | | | |_| |>  <
   /##,-,##\  \__,_|_|  \___|_| |_|_|_|_| |_|\__,_/_/\_\
  /##(   )##\
 /#.--   --.#\  A simple, elegant GNU/Linux distribution.
/`           `\

[Sun Jan 11 05:35:07 PST 1970]
[Linux archlinux 3.1.10-16-ARCH armv5tel]

root@archlinux ~ $
```









PowerDNS dnsdist

UKNOF34

Presentation is on: https://tinyurl.com/ukdnsdist

http://dnsdist.org/

bert.hubert@powerdns.com / @powerdns_bert