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 ratelimt 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()

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Address</th>
<th>State</th>
<th>Qps</th>
<th>Qlim</th>
<th>Ord</th>
<th>Wt</th>
<th>Queries</th>
<th>Drops</th>
<th>Drate</th>
<th>Lat</th>
<th>Pools</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>192.168.1.2:53</td>
<td>up</td>
<td>0.0</td>
<td>1000</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>192.168.1.79:5300</td>
<td>up</td>
<td>0.0</td>
<td>10000</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>127.0.0.1:5300</td>
<td>down</td>
<td>0.0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>All</td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### dnsdist basic config

```plaintext
> showServers()

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Address</th>
<th>State</th>
<th>Qps</th>
<th>Qlim</th>
<th>Ord</th>
<th>Wt</th>
<th>Queries</th>
<th>Drops</th>
<th>Drate</th>
<th>Lat</th>
<th>Pools</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>192.168.1.2:53</td>
<td>up</td>
<td>0.0</td>
<td>1000</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>192.168.1.79:5300</td>
<td>up</td>
<td>0.0</td>
<td>10000</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>127.0.0.1:5300</td>
<td>down</td>
<td>0.0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>All</td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

> getServer(1):setDown()

> showServers()

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Address</th>
<th>State</th>
<th>Qps</th>
<th>Qlim</th>
<th>Ord</th>
<th>Wt</th>
<th>Queries</th>
<th>Drops</th>
<th>Drate</th>
<th>Lat</th>
<th>Pools</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>192.168.1.2:53</td>
<td>up</td>
<td>0.0</td>
<td>1000</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>192.168.1.79:5300</td>
<td>DOWN</td>
<td>0.0</td>
<td>10000</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>127.0.0.1:5300</td>
<td>down</td>
<td>0.0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>All</td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
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
POWERNOS

```
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
```

```plaintext
<table>
<thead>
<tr>
<th>#Name</th>
<th>Address</th>
<th>Status</th>
<th>Latency</th>
<th>Queries</th>
<th>Drops</th>
<th>QPS</th>
<th>Out</th>
<th>Weight</th>
<th>Order Pools</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>127.0.0.1:53</td>
<td>up</td>
<td>1</td>
<td>34</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 pdnsauth</td>
</tr>
<tr>
<td>1</td>
<td>45.55.10.200:53</td>
<td>up</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 pdnsauth</td>
</tr>
<tr>
<td>2</td>
<td>188.166.104.87:53</td>
<td>up</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 pdnsauth</td>
</tr>
</tbody>
</table>

# Rule
0 qname==dnsdist.org, dnsdist.net, dnsdist.com, powerdns.com.
Action to pool pdnsauth Matches 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!**
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!
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 dnssdist: 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 dnsmst .. 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

<table>
<thead>
<tr>
<th>Query</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-iPP-a02.isp.t-ipNet.de.</td>
<td>469</td>
<td>4.7%</td>
</tr>
<tr>
<td>d-iP-a01.isP.t-IPNET.de.</td>
<td>435</td>
<td>4.3%</td>
</tr>
<tr>
<td>f-Ep-a01.iSp.T-ipnEt.DE.</td>
<td>247</td>
<td>2.5%</td>
</tr>
<tr>
<td><a href="http://www.facebook.com">www.facebook.com</a>.</td>
<td>238</td>
<td>2.4%</td>
</tr>
<tr>
<td>fbCDn-prOfILE-A.akAmAIhD.net.</td>
<td>149</td>
<td>1.5%</td>
</tr>
<tr>
<td><a href="http://www.isg-apple.com.akadns.net.">www.isg-apple.com.akadns.net.</a></td>
<td>144</td>
<td>1.4%</td>
</tr>
<tr>
<td>mu-courier.push-apple.com.akadns.net</td>
<td>136</td>
<td>1.4%</td>
</tr>
<tr>
<td><a href="http://www.google-analytics.com">www.google-analytics.com</a>.</td>
<td>114</td>
<td>1.1%</td>
</tr>
<tr>
<td>apple-mobile.query.yahooapis.com</td>
<td>101</td>
<td>1.0%</td>
</tr>
<tr>
<td>dns.msftncsi.com.</td>
<td>98</td>
<td>1.0%</td>
</tr>
<tr>
<td>Rest</td>
<td>7870</td>
<td>78.7%</td>
</tr>
</tbody>
</table>

Top 10 servfail responses

<table>
<thead>
<tr>
<th>Response</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.israelpolitis.org">www.israelpolitis.org</a>.</td>
<td>3</td>
<td>25.0%</td>
</tr>
<tr>
<td>iNT.sITeStat.com.</td>
<td>2</td>
<td>16.7%</td>
</tr>
<tr>
<td><a href="http://www.buyukkurultay.gen.tr">www.buyukkurultay.gen.tr</a>.</td>
<td>2</td>
<td>16.7%</td>
</tr>
<tr>
<td>fabhype.com.</td>
<td>1</td>
<td>8.3%</td>
</tr>
<tr>
<td>dl.cdn.dianxinos.com.</td>
<td>1</td>
<td>8.3%</td>
</tr>
<tr>
<td>myrpp.corp.webex.com.</td>
<td>1</td>
<td>8.3%</td>
</tr>
<tr>
<td>mlocate.spotlife.net.</td>
<td>1</td>
<td>8.3%</td>
</tr>
<tr>
<td><a href="http://www.naughtyseries.ru">www.naughtyseries.ru</a>.</td>
<td>1</td>
<td>8.3%</td>
</tr>
<tr>
<td>Rest</td>
<td>0</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Top 10 NXDOMAIN responses grouped by 2 labels

<table>
<thead>
<tr>
<th>Response</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-ipNet.De.</td>
<td>226</td>
<td>28.0%</td>
</tr>
<tr>
<td>in-addr.arpa.</td>
<td>94</td>
<td>11.6%</td>
</tr>
<tr>
<td>ip6.arpa.</td>
<td>34</td>
<td>4.2%</td>
</tr>
<tr>
<td>wpad.</td>
<td>25</td>
<td>3.1%</td>
</tr>
<tr>
<td>aKadNS.net.</td>
<td>24</td>
<td>3.0%</td>
</tr>
<tr>
<td>sophosxl.net.</td>
<td>23</td>
<td>2.9%</td>
</tr>
<tr>
<td>mcafee.com.</td>
<td>15</td>
<td>1.9%</td>
</tr>
<tr>
<td>facEBook.com.</td>
<td>14</td>
<td>1.7%</td>
</tr>
<tr>
<td>dafa888cg.com.</td>
<td>13</td>
<td>1.6%</td>
</tr>
<tr>
<td>dafa888vd.com.</td>
<td>11</td>
<td>1.4%</td>
</tr>
<tr>
<td>Rest</td>
<td>328</td>
<td>40.6%</td>
</tr>
</tbody>
</table>
I LOVE STATISTICS

Response 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
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

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Address</th>
<th>Status</th>
<th>Latency</th>
<th>Queries</th>
<th>Drops</th>
<th>QPS Out</th>
<th>Weight</th>
<th>Order</th>
<th>Pools</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>185.102.218.81:5300</td>
<td>up</td>
<td>131</td>
<td>1924804421</td>
<td>37677567</td>
<td>5784</td>
<td>-25</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Rule</th>
<th>Action</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dyn blocked netmask</td>
<td>Seconds</td>
<td>Blocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reason</td>
</tr>
</tbody>
</table>

No rules defined
No dynamic blocks active
Status & Getting it

• When will dnstdist 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

“How runs on an Arduino.”
PowerDNS dnsdist

UKNOF34

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

http://dnsdist.org/

bert.hubert@powerdns.com / @powerdns_bert