Checking quickly a DNS zone: a new variant of check-soa

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

First publication of this article on 26 December 2012. Last update on of 22 November 2014

https://www.bortzmeyer.org/check-soa-go.html

When you want to assert rapidly whether or not a DNS zone works fine, typical exhaustive tools like Zonemaster <https://zonemaster.fr/> may be too slow. There is room for a light-and-fast tool and many people used to rely on the check_soa (with an underscore) program developed by Liu and Albitz and published with their famous book DNS and BIND <http://shop.oreilly.com/product/9780596100575.do>. This original program no longer seems maintained and, the last time I tested it, did not support IPv6. So, I wrote one more variant of "my very own check-soa".

Like the original one, it queries each name server of a zone for the SOA record of the zone :

```
% check-soa fr
d.ext.nic.fr.
192.5.4.2: OK: 2222258998
2001:500:2e::2: OK: 2222258998
d.nic.fr.
194.0.9.1: OK: 2222258998
2001:678:c::1: OK: 2222258998
e.ext.nic.fr.
193.176.144.6: OK: 2222258998
2a00:d78:0:102:193:176:144:6: OK: 2222258998
f.ext.nic.fr.
194.146.106.46: OK: 2222258998
2001:67c:1010:11::53: OK: 2222258998
g.ext.nic.fr.
194.0.36.1: OK: 2222258997
2001:678:4c::1: OK: 2222258997
```

Here, we can see that all name servers of .FR reply properly and their serial number is 2222258998. You have several options (-h to see them all). For instance, -i will display the response time :

```
% check-soa -i nl
nl1.dnsnode.net.
194.146.106.42: OK: 2012122607 (3 ms)
2001:67c:1010:10::53: OK: 2012122607 (40 ms)
ns-nl.nic.fr.
192.93.0.4: OK: 2012122607 (3 ms)
2001:660:3005:1::1:2: OK: 2012122607 (2 ms)
ns1.dns.nl.
193.176.144.5: OK: 2012122607 (25 ms)
2a00:d78:0:102:193:176:144:5: OK: 2012122607 (38 ms)
ns2.dns.nl.
2001:7b8:606::85: OK: 2012122607 (15 ms)
213.154.241.85: OK: 2012122607 (36 ms)
ns3.dns.nl.
194.171.17.10: OK: 2012122607 (24 ms)
2001:610:0:800d::10: OK: 2012122607 (29 ms)
ns4.dns.nl.
95.142.99.212: OK: 2012122607 (25 ms)
2a00:1188:5::212: OK: 2012122607 (34 ms)
ns5.dns.nl.
194.0.28.53: OK: 2012122607 (24 ms)
2001:678:2c:0:194:0:28:53: OK: 2012122607 (33 ms)
sns-pb.isc.org.
2001:500:2e::1: OK: 2012122607 (14 ms)
192.5.4.1: OK: 2012122607 (19 ms)
```

Sometimes, the name servers of the zone are not synchronized (it can be temporary, the DNS being only loosely consistent, or it can be permanent if there is a problem) :

```
% check-soa xname.org
ns0.xname.org.
195.234.42.1: OK: 2012081301
ns1.xname.org.
178.33.255.252: OK: 2009030502
ns2.xname.org.
88.191.64.64: OK: 2012081301
2a01:e0b:1:64:240:63ff:fee8:6155: OK: 2012081301
ns3.xtremeweb.de.
130.185.108.193: OK: 2012081301
2a01:4a0:2002:2198:130:185:108:193: OK: 2012081301
```

Here, `ns1.xname.org` lags behind.

Do note that `check-soa` uses a **zone**, not just any domain :

```
% check-soa gouv.fr
No NS records for "gouv.fr.". It is probably a domain but not a zone
```

Of course, when everything works fine, it is boring. What if there is a problem? `check-soa` will display it and will set the exit code accordingly :

```
% check-soa frnog.org
leeloo.supracrawler.com.
193.178.138.10: OK: 2011112300
mercury.ecp-net.com.
193.178.138.20: OK: 2011112300
ns1.saitis.net.
62.220.128.88: OK: 2011112300
2001:788::88: ERROR: Timeout
ns2.saitis.net.
62.220.128.98: OK: 2011112300
2001:788::98: ERROR: Timeout
```

Here, two name servers failed to reply in time (you can tune the timeout with options `-t` and `-n`). The actual problem was with IPv6 connectivity, so you can try with `-4` :

```
% check-soa -q frnog.org
ns1.saitis.net.
2001:788::88: ERROR: Timeout
ns2.saitis.net.
2001:788::98: ERROR: Timeout
% echo $?
1

% check-soa -q -4 frnog.org
% echo $?
0
```

In this specific case, I tested from several sites. But do note that, quite often, networks problems and the resulting timeout will depend on the site from which you test. `check-soa` sees the Internet from just one point. Other points may be different (this is specially true with IPv6 today.) A good example is a test from Free (it works for every other operator) :

```
% check-soa -i mil
CON1.NIPR.mil.
199.252.157.234: ERROR: Timeout
CON2.NIPR.mil.
199.252.162.234: ERROR: Timeout
EUR1.NIPR.mil.
199.252.154.234: ERROR: Timeout
EUR2.NIPR.mil.
199.252.143.234: OK: 2012122703 (229 ms)
PAC1.NIPR.mil.
199.252.180.234: ERROR: Timeout
PAC2.NIPR.mil.
199.252.155.234: ERROR: Timeout
```

There are of course many other possible errors. For instance, on the TLD of Cambodia :

```
% check-soa kh
admin.authdns.ripe.net.
193.0.0.198: ERROR: REFUSED
2001:67c:2e8:5:53::6: ERROR: REFUSED
dns1.online.com.kh.
203.189.128.1: OK: 2012030124
kh.cctld.authdns.ripe.net.
Cannot get the IPv4 address: NXDOMAIN
ns.camnet.com.kh.
203.223.32.3: OK: 2012030124
ns.telesurf.com.kh.
203.144.66.3: ERROR: Not authoritative
ns1.dns.net.kh.
203.223.32.21: OK: 2012030124
sec3.apnic.net.
2001:dc0:1:0:4777::140: OK: 2012030124
202.12.28.140: OK: 2012030124
```

We see three types of errors, `admin.authdns.ripe.net` refuses to answer for this TLD (it is called a "lame delegation", the TLD is delegated to a server which does not know or does not want to answer about it, probably because of a misunderstanding between operators), `kh.cctld.authdns.ripe.net`

does not exist and `ns.telesurf.com.kh` replies, but is not authoritative (it is actually an open recursive resolver, something which is frowned upon, see RFC 5358 [1]).

By default, `check-soa` uses EDNS. This can create problems with some very old name servers :

```
% check-soa microsoft.com
ns1.msft.net.
65.55.37.62: ERROR: FORMERR
2a01:111:2005::1:1: ERROR: FORMERR
ns2.msft.net.
2a01:111:2006:6::1:1: ERROR: FORMERR
64.4.59.173: ERROR: FORMERR
ns3.msft.net.
213.199.180.53: ERROR: FORMERR
2a01:111:2020::1:1: ERROR: FORMERR
ns4.msft.net.
2404:f800:2003::1:1: ERROR: FORMERR
207.46.75.254: ERROR: FORMERR
ns5.msft.net.
65.55.226.140: ERROR: FORMERR
2a01:111:200f:1::1:1: ERROR: FORMERR
```

All of Microsoft's name servers reply "FORmat ERRor". The `-r` option will force back old DNS :

```
% check-soa -r microsoft.com
ns1.msft.net.
65.55.37.62: OK: 2012122401
2a01:111:2005::1:1: OK: 2012122401
ns2.msft.net.
2a01:111:2006:6::1:1: OK: 2012122401
64.4.59.173: OK: 2012122401
ns3.msft.net.
213.199.180.53: OK: 2012122401
2a01:111:2020::1:1: OK: 2012122401
ns4.msft.net.
2404:f800:2003::1:1: OK: 2012122401
207.46.75.254: OK: 2012122401
ns5.msft.net.
65.55.226.140: OK: 2012122401
2a01:111:200f:1::1:1: OK: 2012122401
```

One of the points where my `check-soa` is an improvment over the original is that it issues DNS requests in parallel. So, the waiting time will depend only on the slowest server, not on the sum of all servers. Let's try it on Sri Lanka TLD :

```
% time check-soa -i lk
c.nic.lk.
203.143.29.3: OK: 2012122601 (268 ms)
2405:5400:3:1:203:143:29:3: OK: 2012122601 (274 ms)
d.nic.lk.
123.231.6.18: OK: 2012122601 (133 ms)
l.nic.lk.
192.248.8.17: OK: 2012122601 (189 ms)
m.nic.lk.
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5358.txt`

```
202.129.235.229: OK: 2012122601 (290 ms)
ns1.ac.lk.
192.248.1.162: OK: 2012122601 (179 ms)
2401:dd00:1::162: OK: 2012122601 (300 ms)
p.nic.lk.
204.61.216.27: OK: 2012122601 (4 ms)
2001:500:14:6027:ad::1: OK: 2012122601 (5 ms)
pendragon.cs.purdue.edu.
128.10.2.5: OK: 2012122601 (142 ms)
ripe.nic.lk.
2001:67c:e0::88: OK: 2012122601 (16 ms)
193.0.9.88: OK: 2012122601 (17 ms)
t.nic.lk.
203.94.66.129: OK: 2012122601 (622 ms)
check-soa -i lk  0.02s user 0.00s system 3% cpu 0.637 total
```

The elapsed time was only 637 ms (a bit more than the slowest server, which was at 622), not the sum of all the delays. Parallelism is specially important when some servers timeout. By default, `check-soa` tries three times, with a waiting time of 1.5 second (other programs have a default of 5 seconds, which is extremely long : a DNS reply never comes back after 5 seconds !). So :

```
% time check-soa -i ml
ciwara.sotelma.ml.
217.64.97.50: ERROR: Timeout
djamako.nic.ml.
217.64.105.136: OK: 2012122100 (115 ms)
dogon.sotelma.ml.
217.64.98.75: OK: 2012122100 (109 ms)
ml.cctld.authdns.ripe.net.
193.0.9.95: ERROR: REFUSED (13 ms)
2001:67c:e0::95: ERROR: REFUSED (14 ms)
ns-ext.isc.org.
2001:4f8:0:2::13: OK: 2012122100 (173 ms)
204.152.184.64: OK: 2012122100 (183 ms)
yeleen.nic.ml.
217.64.100.112: OK: 2012122100 (124 ms)
check-soa -i ml  0.01s user 0.00s system 0% cpu 4.518 total
```

The elapsed time, 4.518 seconds, is mostly because of the timeout (and retries) on `ciwara.sotelma.ml`.

By default, check-soa retrieves the list of name servers to query from the local resolver. If the domain is so broken that it cannot even handle these requests, or simply if you want to test with different name servers (for instance because the zone is not yet delegated), you can use the `-ns` option to indicate explicitely the name servers :

```
% check-soa -ns "a.gtld-servers.net b.gtld-servers.net" com
a.gtld-servers.net.
2001:503:a83e::2:30: OK: 1416671393
192.5.6.30: OK: 1416671393
b.gtld-servers.net.
2001:503:231d::2:30: OK: 1416671393
192.33.14.30: OK: 1416671393
```

Are you convinced ? Do you want to install it ? Then, get the source code <`https://framagit.org/bortzmeyer/check-soa`> and follow the instructions in the file `README.md`. Do note that my `check-soa` is written in Go so you'll need a Go compiler. Also, it depends on the excellent godns <`http://miek.nl/projects/godns/`> library so you need to install it first.

If you read the source code, there is nothing extraordinary : parallelism is very simple in Go, thanks to the **goroutines** so there is little extra effort to make a parallel program (one of the great strengths of Go).

I also wrote a Nagios plugin in Go <https://www.bortzmeyer.org/go-dns-icinga.html> to perform more or less the same tests. But the Nagios plugin does not use parallelism : since it is not an interactive program, it is less important if the elapsed time is longer.

Other versions of `check-soa` (or `check_soa`) :

— The original source code <http://examples.oreilly.com/9781565925120/>.
— The one I prefer is written in Python by Alain Thivillon : (en ligne sur `https://www.bortzmeyer.org/files/soa.py`).
— A Perl one <http://search.cpan.org/~olaf/Net-DNS-0.66/demo/check_soa>, in the excellent Net : :DNS library and a simpler one (en ligne sur `https://www.bortzmeyer.org/files/check_soa.pl`), written by Laurent Frigault.
— There is `check_soa.rb` in dnsruby <http://dnsruby.rubyforge.org/>.

A recent alternative to `check-soa` is the option `nssearch` of dig :

```
% dig +nodnssec +nssearch bortzmeyer.fr
SOA ns3.bortzmeyer.org. hostmaster.bortzmeyer.org. 2012122602 7200 3600 604800 3600 from server 192.134.7.24
SOA ns3.bortzmeyer.org. hostmaster.bortzmeyer.org. 2012122602 7200 3600 604800 3600 from server 2001:67c:221
SOA ns3.bortzmeyer.org. hostmaster.bortzmeyer.org. 2012122602 7200 3600 604800 3600 from server 217.70.190.2
```

It has several limitations : each server is tested only once, even if it has multiple IP addresses (which can belong to different physical machines), if stops immediately for some errors (such as a name server which has no entry in the DNS), it does not react properly to non-EDNS name servers like `microsoft.com` mentioned above, etc.

Thanks to Miek Gieben for godns <http://miek.nl/projects/godns/> and for his debugging of my code. Also, check-soa is available as a package in ArchLinux AUR <https://aur.archlinux.org/packages/check-soa-git/>.