Stub Resolvers

And how they interact with your cluster DNS

KubeCon

miek@{coredns.io, google.com}

Stub What? Helpful answer in RFC 7719:

Stub resolver:

A resolver that cannot perform all resolution itself. Stub resolvers generally depend on a recursive resolver to undertake the actual resolution function.

Translation: convert to/from DNS wireformat, send, wait for reply.

C: getaddrinfo (and older gethostbyname)
Go: net.LookupXXX

How do implementations differ?

1. GNU C Library: glibc - standard libc on Linux
2. musl libc - libc on Alpine Linux (small size)
3. Go std lib - when 100% statically compiled (CGO_ENABLED=0)

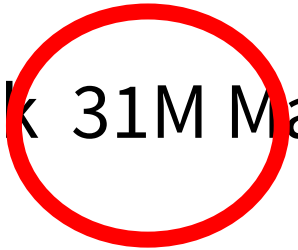
Testing:
Run `ping example.org` in docker where CoreDNS is the upstream (A only)
Use `net.LookupHost` in Go (A and AAAA)

Code and config https://github.com/miekg/stubs

```
% docker images|grep miek/stub
miek/stub-go        latest   24 seconds ago   74.3MB
miek/stub-debian    latest   40 seconds ago   72.9MB
miek/stub-alpine    latest   26 hours ago     13.2MB
```

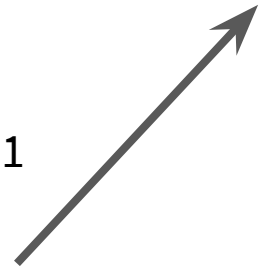
Yes, I like small images as well - I also like images that mimic my laptop/workstation (Debian)

```
-rwxrwxr-x  1 miek miek  31M Mar  7 12:47 coredns*
```



CoreDNS that use the *erratic* plugin (coredns.io/plugins/erratic)

1 in 3 replies will get Truncated back



Corefile

```
. {
   bind 127.0.0.1
   erratic {
      truncate 3
   }
   log . "{name} {type} {class} - {proto} buf:{>bufsize} do:{>do} {>rflags}"
}
```

log . "{name} {type} {class} - {proto} buf:{>bufsize} do:{>do} {>rflags}"

plugin

example.org. A IN - udp buf:4096 do:false qr,aa,rd

- *Q*uery *R*esponse
- *A*uthoritative *A*nswer
- *R*ecursion *D*esired
- *T*run*C*ation

Test: Retry on Truncation

glibc:

example.org. A IN - udp buf:512 do:false qr,aa**,tc,**rd
example.org. A IN - **tcp** buf:65535 do:false qr,aa,rd

musl:

example.org. A IN - udp buf:512 do:false qr,aa**,tc,**rd
...

musl doesn't support TCP in DNS?

Go:

example.org. A IN - udp buf:512 do:false qr,aa,rd
example.org. AAAA IN - udp buf:512 do:false qr,aa**,tc,**rd
example.org. AAAA IN - **tcp** buf:65535 do:false qr,aa,rd

Adhering to “DNS Specification” for 100% is nearly impossible.

https://datatracker.ietf.org/meeting/101/materials/slides-101-dnsop-sessa-the-dns-camel-01 185(!) RFCs

2781 pages / 166891 lines / 888233 words

This is **2** times “The C++ Programming Language” (4th ed)

Not doing TCP is a pretty big omission though… What does the “spec” say?

So, only UDP is mandatory?

Ok. SHOULD it is! (predates RFC 2119)

Ok. I will not implement TCP then.

Ok. I see. **MUST** support **TCP** (RFC 2119)

1987: RFC 1034, Section 5.3.1. Stub Resolvers:

Use of TCP may be an answer, but TCP may well place burdens on the host's capabilities which are similar to those of a real resolver.

1989: RFC 1123 Section 6.1.3.2. Transport Protocols

*DNS resolvers and recursive servers MUST support UDP, and **SHOULD** support TCP, for sending (non-zone-transfer) queries.*

1997: RFC 2181, Section 9. The TC (truncated) header bit

*... it **should** ignore that response, and query again, using a mechanism, such as a TCP connection, that will permit larger replies.*

2010: RFC 5966, 4. Transport Protocol Selection

*Stub resolver implementations (...) **MUST** support TCP since to do otherwise would limit their interoperability with their own clients and with upstream servers.*

So, only UDP is mandatory?

Ok. SHOULD it is! (predates RFC 2119)

Ok. I will not implement TCP then.

Ok. I see. **MUST** support **TCP** (RFC 2119)

23 years!

1987: RFC 1034, Section 5.3.1. Stub Resolvers:

Use of TCP may be an answer, but TCP may well place burdens on the host's capabilities which are similar to those of a real resolver.

1989: RFC 1123 Section 6.1.3.2. Transport Protocols

*DNS resolvers and recursive servers MUST support UDP, and **SHOULD** support TCP, for sending (non-zone-transfer) queries.*

1997: RFC 2181, Section 9. The TC (truncated) header bit

*... it **should** ignore that response, and query again, using a mechanism, such as a TCP connection, that will permit larger replies.*

2010: RFC 5966, 4. Transport Protocol Selection

*Stub resolver implementations (...) **MUST** support TCP since to do otherwise would limit their interoperability with their own clients and with upstream servers.*

1 RR	80	69
2 RRs	131	105
...	...	...
9	488	357
10	**539**	393
11	590	429
12	641	465
13	692	501
14	743	**537**
15	...	...
16	...	...

;; ANSWER SECTION:
www.example.org. 3600 IN SRV 1000 80 10 web1.example.com.

} 9 RRs - no compression

} 13 RRs - compression

13 root name servers? Partly because of same reason.

Configuring stubs: /etc/resolv.conf

- The domain and search keywords are *mutually exclusive*. If more than one instance of these keywords is present, the last instance wins.

- The search keyword of a system's resolv.conf file can be overridden on a per-process basis by setting the environment variable LOCALDOMAIN to a space-separated list of search domains

POSIX search:
Preferences set for this query: Query was "resolv.conf". Substring matching turned on.
Nothing found.

Keywords

- nameserver (max 3, rest silently ignored (glibc, musl, Go)
- domain - just us search
- search- the search list is currently limited to 6 domains with a **total of 256 chars**(?) (glibc).
- options
  - Tweak *some* things for *some* implementations

- ndots - when to do “initial absolute query”
- timeout - defaults to 5s in glibc
- attempts - 2 retries by default (attempts * timeout = 10s!)
- edns0 - glibc only, larger UDP buffer size
- use-vc - code word for “use TCP”
- rotate - don’t always use the first NS listed
    - not in musl libc (parallel!)

https://wiki.musl-libc.org/functional-differences-from-glibc.html

Test: options edns0

glibc:

example.org. A IN - udp **buf:1024** do:false qr,aa,rd

musl:

example.org. A IN - udp **buf:512** do:false qr,aa,rd

Go:

example.org. A IN - udp **buf:512** do:false qr,aa,rd
example.org. AAAA IN - udp **buf:512** do:false qr,aa,rd

Test: options use-vc

glibc:

example.org. A IN - **tcp buf:65535** do:false qr,aa,rd

musl:

example.org. A IN - udp buf:512 do:false qr,aa,rd

Go:

example.org. A IN - udp buf:512 do:false qr,aa,rd
example.org. AAAA IN - udp buf:512 do:false qr,aa,rd

Note miekg/dns also doesn't parse this option, nor option edns0

Search

1 NS: no parallel querying from musl!

nameserver 10.96.0.10
search *default.svc.cluster.local* svc.cluster.local *cluster.local*
options ndots:5

(silently capped at 15)

Search

ndots:5 ~ all “Internet” queries will go through this entire search path

`a.b.www.google.com.` → initial absolute query

`www.google.com.` → goes through path:

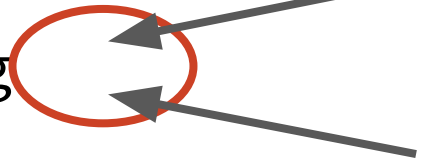
1. `www.google.com.default.svc.cluster.local` (NXDOMAIN)
2. `www.google.com.svc.cluster.local` (NXDOMAIN)
3. `www.google.com.cluster.local` (NXDOMAIN)
4. `www.google.com`

Amplification of #search for *every non-local* name!

(partial [k8s] solution coredns.io/plugins/autopath)

Search

```
$ ping example.org
```



Relative name
1 dots (< 5)

Glibc and musl

```
example.org.default.svc.cluster.local. A IN - udp buf:512 do:false qr,aa,rd
example.org.svc.cluster.local. A IN - udp buf:512 do:false qr,aa,rd
example.org.cluster.local. A IN - udp buf:512 do:false qr,aa,rd
example.org. A IN - udp buf:512 do:false qr,aa,rd
```

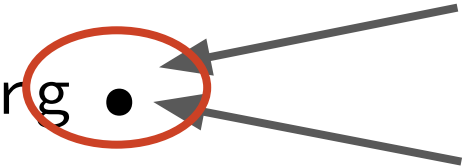
Go

```
example.org.default.svc.cluster.local. AAAA IN - udp buf:512 do:false qr,aa,rd
example.org.default.svc.cluster.local. A IN - udp buf:512 do:false qr,aa,rd
example.org.svc.cluster.local. AAAA IN - udp buf:512 do:false qr,aa,rd
example.org.svc.cluster.local. A IN - udp buf:512 do:false qr,aa,rd
example.org.cluster.local. AAAA IN - udp buf:512 do:false qr,aa,rd
example.org.cluster.local. A IN - udp buf:512 do:false qr,aa,rd
example.org. AAAA IN - udp buf:512 do:false qr,aa,rd
example.org. A IN - udp buf:512 do:false qr,aa,rd
```

silly little plugin, called *searchpath*
(github.com/miekg/searchpath)

Search

```
$ ping example.org .
```



Fully Qualified Domain Name
(FQDN)

example.org. A IN - udp buf:512 do:false qr,aa,rd

example.org. AAAA IN - udp buf:512 do:false qr,aa,rd
example.org. A IN - udp buf:512 do:false qr,aa,rd

Feature table

	glibc	*musl*	*Go*
Transports	UDP + TCP	UDP	UDP+TCP
Nameserver order	In order	Parallel	In order
Requery on TC	Yes	No	Yes
Largest packet size	65336	512	65536
Largest UDP size (options edns0)	1024	512	512
Name Service Switch	yes	no	yes

Yes, I also love Alpine Linux for its small size

Be aware of the differences

Author switched to Debian-slim images for ~everything, unless I bypass libc completely (100% static Go binaries).

Read it! https://wiki.musl-libc.org/functional-differences-from-glibc.html