net: allow custom Resolver method implementation(s) #12503

New issue

Open **sajal** opened this issue on Sep 5, 2015 · 23 comments

sajal commented on Sep 5, 2015

I mentioned in #12476 that I would like to detect the time it took for DNS resolution phase of the Dial process in `Dialer.Dial` . The solution posted there was very hacky and adds unnecessarily to the API.

@bradfitz suggested

> Perhaps `net.Dialer` could have an optional `Resolver` , akin to how `http.Client` has an optional `Transport` , or `http.Server` has an optional `ErrorLog` , etc.

This seems like an excellent idea. Here is how I propose we go about it by adding minimal complexity and preserving code compatibility.

I propose net package adds a new `Resolver` interface.

```
type Resolver interface {
    Resolve(host string, deadline time.Time) (addrs []IPAddr, err error)
}
```

The signature of `Resolver.Resolve` is same as `lookupIPDeadline` which `Dial` eventually uses. `Dialer` gets an optional field `CustomResolver` of type `Resolver` .

The `Resolver` object (or `nil`) gets passed around thru the resolution process.

`Dialer.Dial -> resolveAddrList -> internetAddrList` .

`internetAddrList` currently always uses `lookupIPDeadline` , it would need to be changed such that if the passed custom resolver is not nil then use it, otherwise use `lookupIPDeadline` .

Other functions calling `resolveAddrList` or `internetAddrList` would need to be modified to add an extra `nil` argument . This does not break code compatibility because they are unexported functions.

Benefits of allowing a custom `Resolver`

- Allowing me to collect timing information as mentioned in #12476
- Allowing users to implement their own DNS logic. Failovers, etc.
- Mocking for tests.
- Client side caching, pre-fetching, etc.
- In time, other packages (like the superb github.com/miekg/dns) could provide their own `Resolver` implementations.
- Great for people like me who rely on Go to write network debugging tools.

14

sajal changed the title from **proposal: Allow passing of custom Resolver to Dialer.dial in package net** to **proposal: Allow passing of custom Resolver to Dialer.Dial in package net** on Sep 5, 2015

sajal changed the title from **proposal: Allow passing of custom Resolver to Dialer.Dial in package net** to **proposal: Allow passing of custom Resolver to net.Dialer** on Sep 5, 2015

ianlancetaylor added the **Proposal** label on Sep 5, 2015

Assignees

No one assigned

Labels

HelpWanted

Proposal

Proposal-Accepted

Projects

None yet

Milestone

Go1.9

Notifications

Subscribe

You're not receiving notifications from this thread.

14 participants

ianlancetaylor added this to the **Unplanned** milestone on Sep 5, 2015

sajal commented on Sep 10, 2015

Should I implement and submit the change for codereview? Or wait for some comments here?

bradfitz commented on Sep 10, 2015 Owner

No need to prototype it yet. The code will be relatively easy compared to getting the design right.

I suspect that signature isn't general enough. Maybe it's good enough for a dialer, but perhaps it needs a different name.

I bet we don't want to define an interface in the net package. If anything, it could just be an optional func type on the Dialer, similar to funcs on http://golang.org/pkg/net/http/#Transport

sajal commented on Sep 10, 2015

Perhaps call it `Lookupfunc` (or better name) and deadline-ing is handled inside net package. It might mirror signature of `net.LookupIP` which is used by default if `Lookupfunc` is nil.

Anything that does a lookup could ask for optional field for `Lookupfunc` to allow user to provide their own implementation.

bradfitz referenced this issue on Sep 11, 2015

net/http: Transport analytics #12580 Closed

adg added **Proposal** and removed **Proposal** labels on Sep 25, 2015

rsc modified the milestone: **Proposal**, **Unplanned** on Oct 24, 2015

rsc changed the title from **proposal: Allow passing of custom Resolver to net.Dialer** to **proposal: allow net.Dialer to use custom resolver** on Oct 24, 2015

rsc changed the title from **proposal: allow net.Dialer to use custom resolver** to **proposal: net: allow Dialer to use custom resolver** on Oct 24, 2015

benburkert commented on Nov 6, 2015 Contributor

I would also like to see a `Resolver` interface but with multiple methods that match the `net.Lookup*` funcs.

```
type Resolver interface {
  LookupAddr(addr string) (names []string, err error)
  LookupCNAME(name string) (cname string, err error)
  LookupHost(host string) (addrs []string, err error)
  LookupIP(host string) (ips []IP, err error)
  LookupMX(name string) (mxs []*MX, err error)
  LookupNS(name string) (nss []*NS, err error)
  LookupPort(network, service string) (port int, err error)
  LookupSRV(service, proto, name string) (cname string, addrs []*SRV, err error)
  LookupTXT(name string) (txts []string, err error)
}
```

The timeout & deadline functionality could be configured when the resolver is created:

```
func NewResolver(options ResolverOption...) (Resolver, error)

type ResolverOption func(*resolver) error

func ResolverTimeout(duration time.Duration) ResolverOption
func ResolverDeadline(deadline time.Time) ResolverOption
```

bradfitz commented on Nov 6, 2015 Owner

@benburkert, that is not a Go-style (small) interface. Once you have 9 methods, surely somebody would want to add a tenth later, but they can't for compatibility reasons. 9 methods is also hard to implement. We'd probably have to add some sort of EmptyResolver type that people could embed which just returned errors for everything.

I'd start with looking at which interfaces are actually needed by the things this bug is about. Maybe you'd have 9 interfaces instead (maybe starting with 3?) and combine them as needed like io.ReadWriteCloser? I don't know. I haven't given this much thought yet.

davecheney commented on Nov 6, 2015 Contributor

What about Lookup(recordtype, query string) ...

It's similar to our Dial(network, address string) function, and would permit wildcard, ANY, and lookups for types not yet added to the dns spec.

Just spitballing...

theckman commented on Jan 14, 2016

I just ran in to this issue myself, except a little bit abstracted away from `net.Dialer`. My use-case may be a little weird, but this would come in extremely handy for me if it was also exposed within the `net/http` package.

I'm writing a utility that's going to talk over TLS to backend systems (HTTP + JSON) and I'm using Consul to discover the individual backend nodes. The biggest issue is that I don't have all of my system's DNS requests being serviced by Consul, so pulling a configuration from `/etc/resolv.conf` won't really work. I plan on using their port `8600` DNS interface.

So I'll end up needing to first obtain a list of IP addresses from the Consul DNS endpoint and then use that IP address in the URL. Following that, I'll need to set the `Host` field on the request so that the TLS validation works. The only downside here is that I end up having to do a network operation at the creation time of the `http.Request` struct instead of when actually invoking the request.

If the `http.Transport` struct was modified to support a custom DNS resolver code path, it would make the code cleaner and avoid the upfront network call.

mikioh commented on Mar 11, 2016 Contributor

At the moment, Dial runs the following processes serially for simplicity:

1. multiple host and service discovery racers
2. making a short list of target addresses
3. multiple connection setup racers, and picking a single winner

In near future, when we want more performance on some circumstances, we probably run:

1. multiple host/service discovery+connection setup racers
   - per address family, per service {name,port}, etc
2. picking a single winner

For both cases, the Resolver interface needs to take information for host and service filters. Moreover, it would probably need certificates for supporting upcoming DNS over TLS and DANE.

Looks like this proposal makes it possible to place complicated DNS-related packages at x/net repository. I'm happy if we have fancy name/service discovery functionality without replumbing of packages in standard library.

anatol commented on Mar 30, 2016

I am trying to run a network application at Android arm64 system and `http.Get` fails because of DNS resolution failed. It turned out Android uses custom dns resolver interface. https://android.googlesource.com/platform/bionic/+/master/libc/dns/net/gethnamaddr.c#564 An

application opens `/dev/socket/dnsproxyd` socket and uses it to resolve names. I tried `GODEBUG=netdns=cgo` and for some reason it does not work on Android.

It would be nice if I can implement a custom dns resolver and tell my application to use it. Here is similar issue from another project [1].

[1] syncthing/syncthing-android#412 (comment)

sajal commented on Mar 30, 2016

Valid use-case for custom resolver, but have you tried using the Android NDK to build your binary?

jabley commented on May 12, 2016

I'd similarly be interested in having timing information available, similar to `time_*` variables in curl. A monitoring tool that can periodically probe networks would be very handy.

Happy to open up a separate proposal if it's felt to be off-topic for this one?

bradfitz commented on May 12, 2016 Owner

@jabley, that already happened for Go 1.7. See #12580

adg commented on Jul 20, 2016 Contributor

This needs a proper proposal document to move forward.

bradfitz referenced this issue on Aug 30, 2016
net: no way to set timeout for LookupAddr #16672 Closed

bradfitz self-assigned this on Aug 30, 2016

adg added the **HelpWanted** label on Oct 4, 2016

bradfitz was unassigned by **adg** on Oct 4, 2016

adg commented on Oct 4, 2016 Contributor

An extension to the work done in #16672

bradfitz commented on Oct 4, 2016 Owner

In particular, this got submitted: https://go-review.googlesource.com/29440

kayrus referenced this issue in **kubernetes/kubernetes** on Oct 18, 2016
kubelet: http prober DNS resolver #35032 Open

adg modified the milestone: **Go1.9**, **Proposal** on Nov 1, 2016

adg commented on Nov 1, 2016 Contributor

Need design work, but should be good for Go 1.9.

1

bradfitz changed the title from **proposal: net: allow Dialer to use custom resolver** to **net: allow custom Resolver method implementation(s)** on Nov 1, 2016

bradfitz added the Proposal-Accepted label on Nov 1, 2016

polomsky commented on Dec 5, 2016

I have another use case for which this feature would be usable. I need to make some app firewall. I have proxy which uses `http.Request` . Urls are specified by user and I want to block access to several ip addresses, e.g. localhost. Currently I do not know any way how to do it except usage of Host header. But it is not sufficient, because I can not specify multiple ip addresses for request (in case when DNS server returns more addresses for same name). So custom dns resolver (with removing wrong ips) would be very great.

sajal commented on Dec 6, 2016

@polomsky I am doing something similar. In my `Transport` I am using my own dialer which is basically clone of the default dialer but applies the IP policy before making the connection usable. The drawback is a TCP connection was made but the client cant do anything with it if `islocalip` returns true.

sajal commented on Dec 16, 2016

So, currently `net.Dialer` accepts a `net.Resolver` which is a struct and not an interface. As I understand it, there is no way to override the Lookup* methods. Are there any plans to make it an interface? Just like how `http.Client` uses `http.RoundTripper` interface instead of explicit `http.Transport` struct.

bradfitz commented on Dec 17, 2016 Owner

@sajal, yes, we left room for in the design for that:
https://github.com/golang/go/blob/ecc4474341504f5893c8333dbb68c520dbe93ca5/src/net/lookup.go#L100

2

This was referenced on Jan 11

net: ability to replace Resolver impl #17554 Closed

Make integrationtests run without changing /etc/hosts lucas-clemente/quic-go#553 Open

sheerun commented on May 26 • edited

@bradfitz I'd like to point out that it would be nice if LookupPort could accept a hostname to connect to as well. For now it seems you expect port number as `service` argument, as shown for example by ReverseProxy implementation.

Use Case: Resolving to custom port with ReverseProxy (e.g. example.com should go to 127.0.0.1:3456)

mikegleasonjr commented 26 days ago

Hi guys, just wondering if go 1.9 beta drops tomorrow would it means this feature won't make its way into go 1.9?

ianlancetaylor commented 26 days ago Contributor

@mikegleasonjr As far as I know nobody has written the code for this. At this point I think it is unlikely to get into Go 1.9, but it is still possible if somebody sends in a patch very soon.

5

CAFxX referenced this issue in **cloudfoundry/cli** 26 days ago

cf cli fails if the first dns from /etc/resolv.conf is not responding #1089 Open

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Comment