CoreDNS is misconfigured leading to unexpected healthcheck behaviour #64

Closed **bentasker** opened this issue on Nov 6, 2021 · 11 comments

bentasker commented on Nov 6, 2021 • edited

CoreDNS is configured to healthcheck the Cloudflare fallback every 5 minutes, however in practice, a check is performed once a minute (and retries are generated when it fails).

The `fallback` directive also causes healthchecks at startup, which can create substantial query rates.

This is also why users have reported seeing small packet storms when Cloudflare is not reachable (by **@lialosiu** here and **@tescophil** here).

The intended behaviour appears to be to check once every 5 minutes

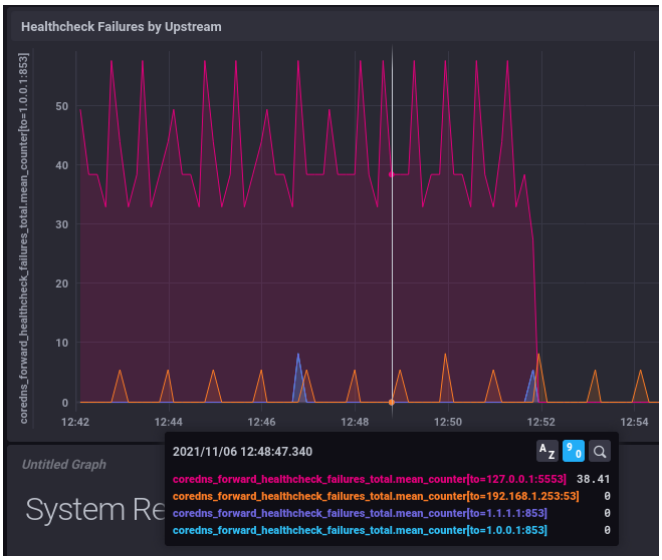
```
forward . tls://1.1.1.1 tls://1.0.0.1 {
    tls_servername cloudflare-dns.com
    except local.hass.io
    health_check 5m
}
```

However, that is not the only check being performed, because the encompassing server block is referenced elsewhere

```
forward . {{ join " " .servers }} {{ if len .locals | eq 0 }}dns://127.0.0.11{{ else }}{{ join " " .locals }}{{ end }} dns://127.0.0.1:5553 {
    except local.hass.io
    policy sequential
    health_check 1m
}
```

A check will be run once a minute against `127.0.0.1:5553` as well as the locals (if present) - from further testing it *appears* those will only begin once you've had an initial failure which leads `coredns` to move onto the next forward host.

We can see this is the case by enabling coredns' prometheus endpoints and pointing telegraf at them



That's failures per minute - each failure represents a single query (where healthchecks are concerned for `example.org`) sent to `127.0.0.1:5553`.

However, to `127.0.0.1:5553` (and any other upstream for that matter) it's just another query, so when it's query to it's upstream (one of `1.1.1.1` or `1.0.0.1`) fails, it retries and we end up with new packets hitting the wire, one after the other.

~~In terms of the fix, it's not clear *why* the fallback behaviour is implemented/hardcoded in the first place (I couldn't find any architecture discussions on it in that repo, perhaps I missed them), but the correct way to have implemented this would be one of the following options~~

~~Option 1: not include `127.0.0.1:5553` in the forwards statement at all (as it's handled by the fallback).~~

```
forward . {{ join " " .servers }} {{ if len .locals | eq 0 }}dns://127.0.0.11{{ else }}{{ join " " .locals }}{{ end }} {
    except local.hass.io
    policy sequential
    health_check 1m
}
fallback REFUSED,SERVFAIL,NXDOMAIN . dns://127.0.0.1:5553
```

~~(It'd need some logic to handle empty locals)~~

~~Option 2: Not use a separate server block~~

```
forward . {{ join " " .servers }} {{ if len .locals | eq 0 }}dns://127.0.0.11{{ else }}{{ join " " .locals }}{{ end }} tls://1.1.1.1 tls://1.0.0.1 {
    except local.hass.io
    policy sequential
    health_check 1m
```

~~`tls_servername cloudflare-dns.com`~~
~~`}`~~
~~`fallback REFUSED,SERVFAIL,NXDOMAIN . dns://127.0.0.1:5553`~~

~~(perhaps there some other reason an entire separate server block was stood up, but I don't see any reference to it).~~

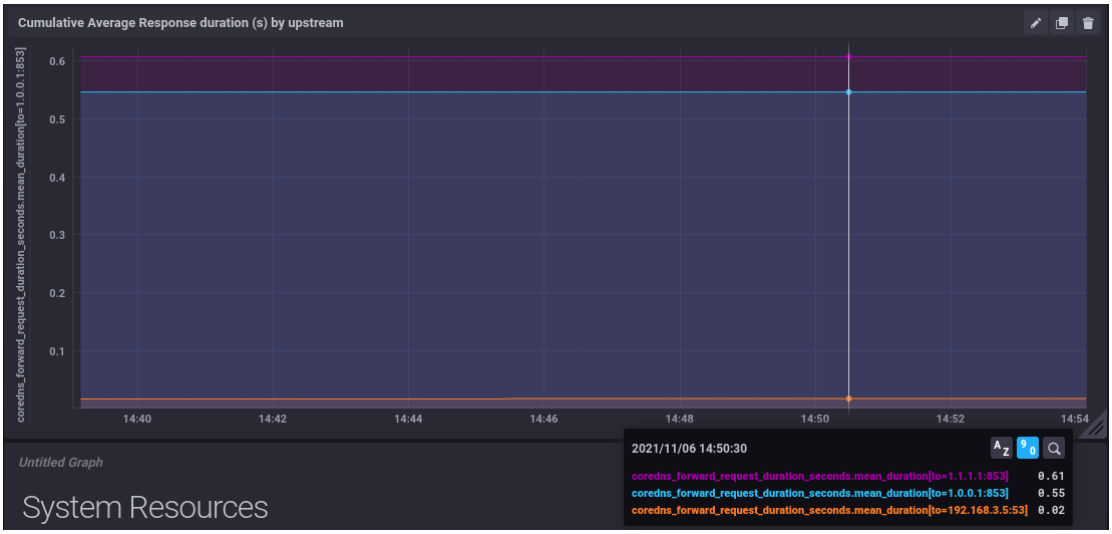
- ~~You could also add some config to handle~~ `example.org` ~~locally (so the healthcheck against~~ `:5553` ~~isn't passed upstream), but that's more horrid than the current setup.~~
- ~~Turning off healthchecks against the locals is likely to be undesirable due to them having to wait for~~ `coredns` ~~'s timeouts if a local does go down, so I've not included that~~

~~The reason this isn't a PR is because it's blocked by a decision on approach.~~
Correction: The much bigger issue is actually the `fallback` statement, see #64 (comment)

Additional Observations

Whilst capturing telemetry there were a few things I noticed which might help inform a decision on the above

When in use, the Cloudflare fallback introduces a significant level of latency:



At the network level, Cloudflare is only 10-15ms away, but the average query duration for CF upstreams is half a second. The presumption is that's due to DoT overheads, but unfortunately `coredns` doesn't currently expose metrics that can help verify this.

I'd posit therefore that as well as fixing the healthcheck issue

- The choice to have cloudflare enabled/disabled should be available to the user
- If mandatory, the choice to use DoT should be open to the user

But, realistically, if this issue is fixed so that healthchecks aren't amplified onto the network, then users who want to block CF DoT at the perimeter will be able to do so without HA gradually attempting to flood the network.

I like `coredns`, but it does feel rather out of place in an appliance - it's approach to dynamic timeouts isn't really very well tuned to the foibles of domestic connections/networks.

👍 5

bentasker commented on Nov 8, 2021 • edited — Author

Attaching a packet capture of the first 17 seconds after `coredns` startup with CF blocked - there are 160,000 packets (though this includes the RSTs coming back from the firewall).

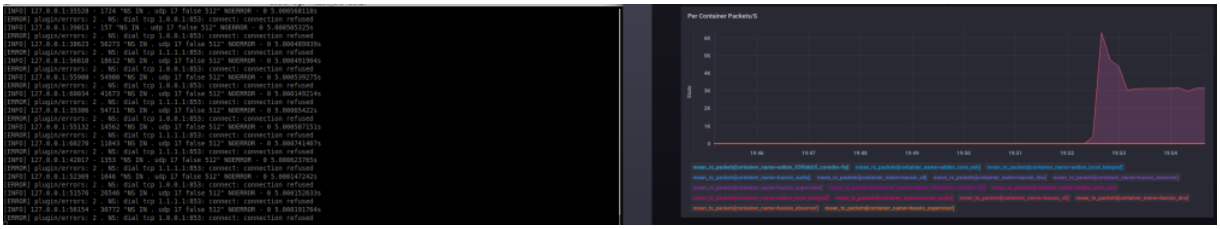
dns.pcap.gz

If the firewall is set to drop rather than reject, then `coredns` still generates packets (too) regularly, but we see a much, much lower rate - 10s or 100s of packets in the same period.

I think this misconfig is exacerbated by `coredns`'s own behaviour - it *looks* like coredns's response to getting a RST is to try and open a new connection, whilst you also get the periodic retries, ultimately building up into a storm (that's really not good behaviour, *particularly* for something on a domestic connect).

bentasker commented on Nov 8, 2021 • edited — Author

What's interesting is that once triggered, the rate spikes and then remains fairly stable, dumping 3000 packets/sec onto the network



Left to run, the rate fluctuates a bit but stays around that level

Experimenting a bit, it seems the issue is not in fact the use of `127.0.0.1:5553` in the `forward` section but in the `fallback` statement. If we remove

```
fallback REFUSED,SERVFAIL,NXDOMAIN . dns://127.0.0.1:5553
```

Then the packet storm never occurs, even with `127.0.0.1:5553` set in the `forward` declaration.

So actually, it seems the correct fix here is not to have that fallback statement at all. It also brings the behaviour inline with that expected of DNS servers - if you don't trust responses from your upstream then why query it in the first place?

bentasker commented on Nov 8, 2021 • edited — Author

The reason this happens is that CoreDNS will loop over the configured upstreams until it reaches it's own timeout - https://github.com/coredns/proxy/blob/master/proxy.go#L78 - (FAOD the `fallback` plugin uses `proxy` under the hood).

Because the RSTs can come back fast from the firewall, there's plenty of time to fling packets onto the network.

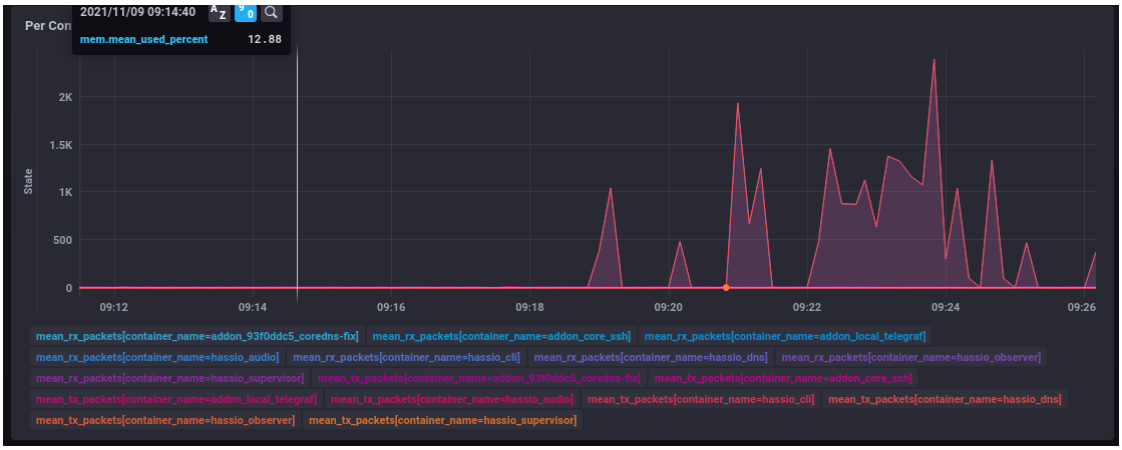
Once CoreDNS reaches that timeout, it'll consider the backend down, but only for 2 seconds. In practice this doesn't matter much, because if all backends are considered down (which they will be here) then default behaviour is to spray randomly against one of the hosts.

TL:DR forcing a hardcoded `fallback` is potentially harmful to the network hosting HomeAssistant.

bentasker commented on Nov 9, 2021 — Author

The storm can also trigger sometime after startup - if `coredns` moves onto it's next `forward`, then the issue will also be triggered.

We can verify this by adding a firewall rule to drop traffic from HomeAssistant on the local DNS service. What we get is intermittent storms - the "recoveries" are `coredns` moving back to the local DNS and having to wait for timeout - notice that `coredns` 's dynamic timeouts lead to shorter and shorter recoveries



At this point, you could theorise that if we change our rule to REJECT rather than DROP, we should cause a storm against the local DNS server. However, that's not the case (at least where the local DNS is contacted using UDP).

In fact, if we rewrite the `coredns` config so that the fallback uses UDP instead of DoT

```
forward . dns://1.1.1.1 dns://1.0.0.1  {
    tls_servername cloudflare-dns.com
    except local.hass.io
    health_check 5m
}
```

We also do not get a storm. Manually querying against the server block doesn't cause a storm either. This suggests that the underlying issue is in `coredns` 's handling of TCP upstreams.

This means there's an alternative fix/mitigation available here - if the devs are overly attached to the existence of the `fallback` statement, then this issue can be mitigated by using UDP, not DoT to reach Cloudflare.

bentasker commented on Nov 9, 2021 • edited — Author

OK, so pulling this altogether, this is how to Repro and verify the above.

Metrics collection

Optional, you could also just run a packet capture if you don't want graphs

- Stand up an InfluxDB instance (or sign up for a free account at https://cloud2.influxdata.com)
- Create a telegraf config in `/config/telegraf.conf` (Config I used is here)
- Install and start my Telegraf addon
- Exec into `hassio_dns` and edit `/etc/corefile` to add `prometheus 0.0.0.0:9153` to each of the server blocks
- Run `pkill coredns` on `hassio_dns` to force a config reload
- Stats should start appearing in InfluxDB (If you're using Chronograf, you can import HomeAssistant DNS.json.gz dashboard as a starting point - you'll prob need to edit the DB name if you're writing into a different one than me)

Repro

On your network firewall, add two rules

- Dest: `1.0.0.1` Proto: any, dport: `853` `REJECT`
- Dest: `1.1.1.1` Proto: any, dport: `853` `REJECT`

Exec into `hassio_dns` and

- `cp /etc/corefile /root/`
- `pkill coredns` to trigger a restart

You should see thousands of packets hit the network. If you're using some other metrics+graphing solution, be aware that you may not see them in graphs for `eth0` (or whatever your main interface is) because the container uses an aliased interface.

Now exec into `hassio_dns` and edit `/etc/corefile` to remove the `fallback` line. Run `pkill coredns` to force a restart - you should not see significant packet rates on the network after this.

Restore the original config `cp /root/corefile /etc/` and then edit it to make the upstream use plain DNS

```
forward . dns://1.1.1.1 dns://1.0.0.1  {
    tls_servername cloudflare-dns.com
    except local.hass.io
    health_check 5m
}
```

On your firewall, add two more rules

- Dest: `1.0.0.1` Proto: any, dport: `53` `REJECT`
- Dest: `1.1.1.1` Proto: any, dport: `53` `REJECT`

`pkill coredns` should not elicit a packet storm.

bentasker mentioned this issue on Nov 9, 2021

Use UDP instead of TLS for failover DNS servers #58

Closed

stale bot commented on Jan 8

This issue has been automatically marked as stale because it has not had recent activity. It will be closed if no further activity occurs. Thank you for your contributions.

stale bot added the stale label on Jan 8

mtdcr commented on Jan 15

Nothing changed since the report, dear stale bot.

Dear developers, please add an option to disable Cloudflare servers or just disable them by default, if any other server is configured. If a server was configured by the user, the user has had a reason to do so. Please respect that and behave like a good netizen.

14

stale bot removed the stale label on Jan 15

Strohhutpat commented on Jan 27

OK, so pulling this altogether, this is how to Repro and verify the above.

Metrics collection

Optional, you could also just run a packet capture if you don't want graphs

- Stand up an InfluxDB instance (or sign up for a free account at https://cloud2.influxdata.com)
- Create a telegraf config in `/config/telegraf.conf` (Config I used is here)
- Install and start my Telegraf addon
- Exec into `hassio_dns` and edit `/etc/corefile` to add `prometheus 0.0.0.0:9153` to each of the server blocks
- Run `pkill coredns` on `hassio_dns` to force a config reload
- Stats should start appearing in InfluxDB (If you're using Chronograf, you can import HomeAssistant DNS.json.gz dashboard as a starting point - you'll prob need to edit the DB name if you're writing into a different one than me)

Repro

On your network firewall, add two rules

- Dest: `1.0.0.1` Proto: any, dport: `853` `REJECT`
- Dest: `1.1.1.1` Proto: any, dport: `853` `REJECT`

Exec into `hassio_dns` and

- `cp /etc/corefile /root/`
- `pkill coredns` to trigger a restart

You should see thousands of packets hit the network. If you're using some other metrics+graphing solution, be aware that you may not see them in graphs for `eth0` (or whatever your main interface is) because the container uses an aliased interface.

Now exec into `hassio_dns` and edit `/etc/corefile` to remove the `fallback` line. Run `pkill coredns` to force a restart - you should not see significant packet rates on the network after this.

Restore the original config `cp /root/corefile /etc/` and then edit it to make the upstream use plain DNS

```
forward . dns://1.1.1.1 dns://1.0.0.1  {
    tls_servername cloudflare-dns.com
    except local.hass.io
    health_check 5m
}
```

On your firewall, add two more rules

> - Dest: `1.0.0.1` Proto: any, dport: `53` `REJECT`
> - Dest: `1.1.1.1` Proto: any, dport: `53` `REJECT`
>
> `pkill coredns` should not elicit a packet storm.

works for me

This was referenced on Jan 27

.local named devices (i.e. ESPHOME devices) only resolve for a few minutes after booting HA #54

Closed

Dns stops resolving within hours #51

Closed

Strohhutpat commented on Feb 17

> > OK, so pulling this altogether, this is how to Repro and verify the above.
> >
> > ### Metrics collection
> >
> > Optional, you could also just run a packet capture if you don't want graphs
> >
> > - Stand up an InfluxDB instance (or sign up for a free account at https://cloud2.influxdata.com)
> > - Create a telegraf config in `/config/telegraf.conf` (Config I used is here)
> > - Install and start my Telegraf addon
> > - Exec into `hassio_dns` and edit `/etc/corefile` to add `prometheus 0.0.0.0:9153` to each of the server blocks
> > - Run `pkill coredns` on `hassio_dns` to force a config reload
> > - Stats should start appearing in InfluxDB (If you're using Chronograf, you can import HomeAssistant DNS.json.gz dashboard as a starting point - you'll prob need to edit the DB name if you're writing into a different one than me)
> >
> > ### Repro
> >
> > On your network firewall, add two rules
> >
> > - Dest: `1.0.0.1` Proto: any, dport: `853` `REJECT`
> > - Dest: `1.1.1.1` Proto: any, dport: `853` `REJECT`
> >
> > Exec into `hassio_dns` and
> >
> > - `cp /etc/corefile /root/`
> > - `pkill coredns` to trigger a restart
> >
> > You should see thousands of packets hit the network. If you're using some other metrics+graphing solution, be aware that you may not see them in graphs for `eth0` (or whatever your main interface is) because the container uses an aliased interface.
> > Now exec into `hassio_dns` and edit `/etc/corefile` to remove the `fallback` line. Run `pkill coredns` to force a restart - you should not see significant packet rates on the network after this.
> > Restore the original config `cp /root/corefile /etc/` and then edit it to make the upstream use plain DNS
> >
> > ```
> > forward . dns://1.1.1.1 dns://1.0.0.1 {
> > tls_servername cloudflare-dns.com
> > except local.hass.io
> > health_check 5m
> > }
> > ```
> >
> > On your firewall, add two more rules
> >
> > - Dest: `1.0.0.1` Proto: any, dport: `53` `REJECT`
> > - Dest: `1.1.1.1` Proto: any, dport: `53` `REJECT`
> >
> > `pkill coredns` should not elicit a packet storm.
>
> works for me

for long uptime it doesnt work

alexdelprete mentioned this issue on Mar 22

Not resolving local host names #20

Closed

mdegat01 mentioned this issue on Apr 4

Cloudflare as fallback only and no healthcheck #82

Merged

mdegat01 commented on Apr 25 Contributor

Fixed by #82

mdegat01 closed this as completed on Apr 25

mdegat01 commented on Apr 25 Contributor

Also note that there is a new option to disable the fallback dns added here: home-assistant/supervisor#3586 as I would guess a number of users on here would be interested in that.

🎉 1 ❤️ 1

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

4 participants