

The inconspicuous role of **conntrack** in Kubernetes networking

Ricard Bejarano

Lead Site Reliability Engineer



ЯДЕРНОЕ
ДЕЛЕНИЕ

СТЕРЖНИ
УПРАВЛЕНИЯ

ВОДА

НЕГАТИВНЫЙ
КОЭФФИЦИЕНТ
ТЕМПЕРАТУРЫ

ОТРАВЛЕНИЕ
КСЕНОНОМ

ОТРАВЛЕНИЕ
КСЕНОНОМ



Before we begin...

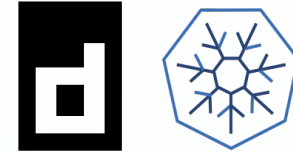
Let's set a bit of a **system prompt**



Kubernetes
(upstream)



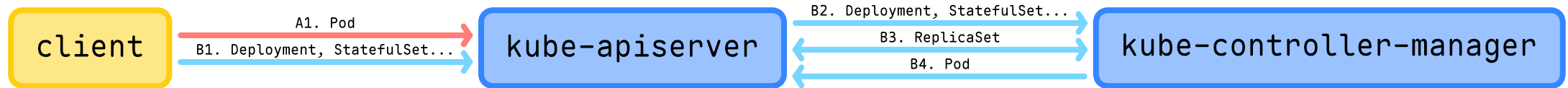
Linux
(5.x or above)



Common runtimes
(like containerd or CRI-O)

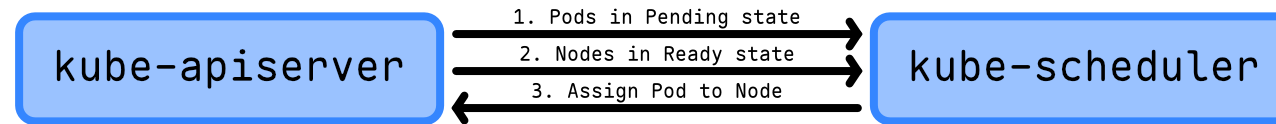
Before we begin...

Requesting a Pod



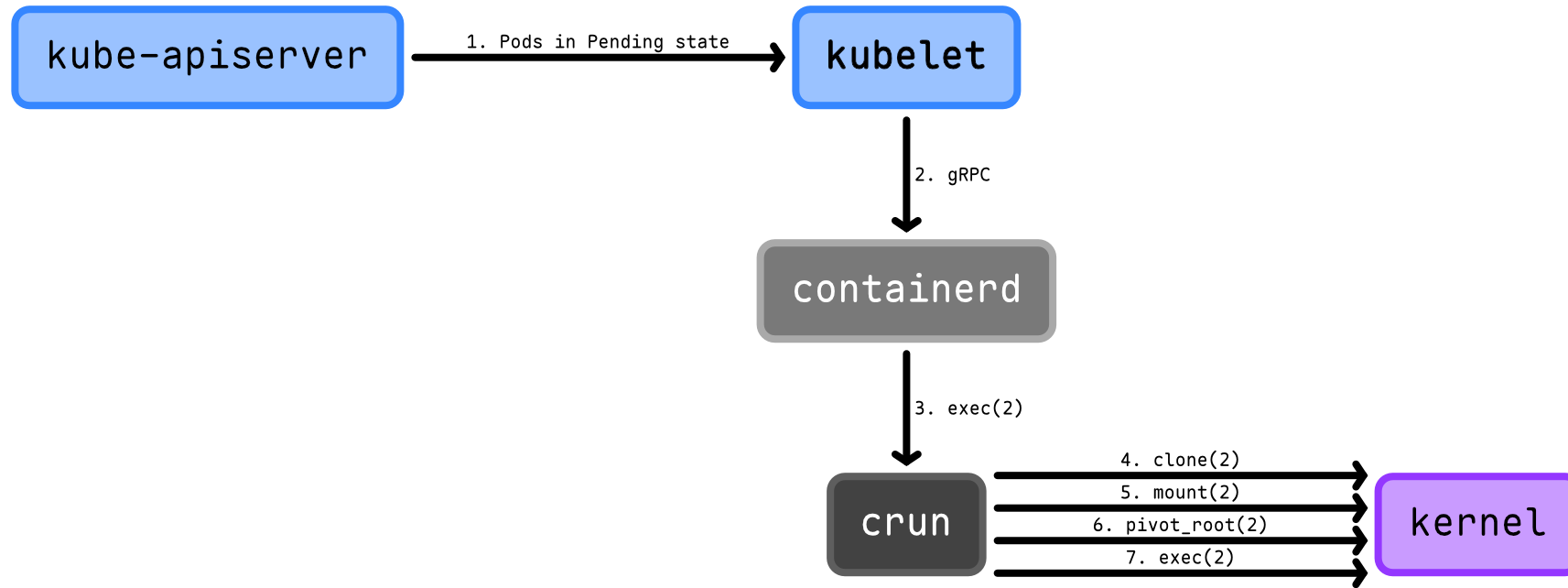
Before we begin...

Scheduling a Pod



Before we begin...

Provisioning a Pod



Our pod is Running

```
$ kubectl get pod
```

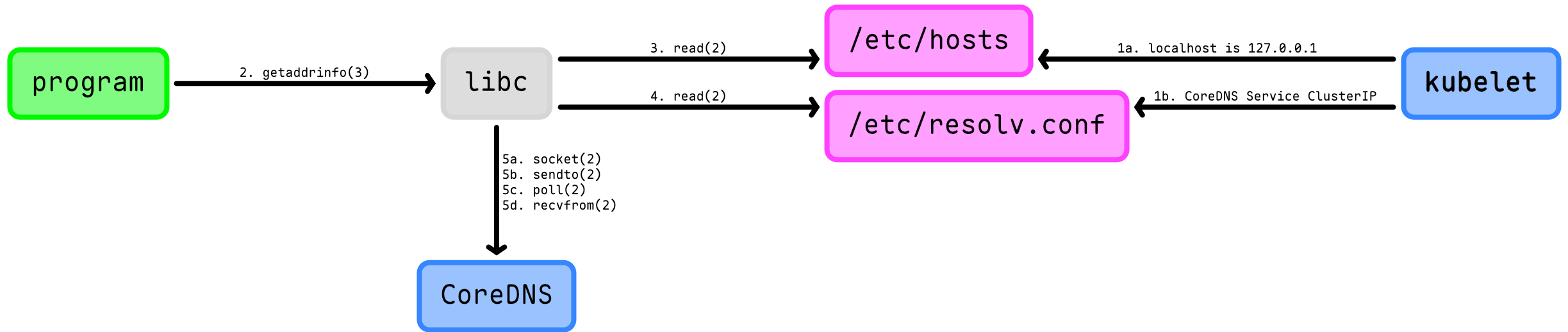
NAME	READY	STATUS	RESTARTS	AGE
foo-6f9c7d8b6f-xyz12	1/1	Running	0	12s

```
2025-09-04T14:12:01Z [ERROR] read udp REDACTED:42999->REDACTED:53: i/o timeout
2025-09-04T14:12:01Z [ERROR] read udp REDACTED:59401->REDACTED:53: i/o timeout
2025-09-04T14:12:02Z [ERROR] read udp REDACTED:40757->REDACTED:53: i/o timeout
2025-09-04T14:12:03Z [ERROR] read udp REDACTED:18235->REDACTED:53: i/o timeout
2025-09-04T14:12:03Z [ERROR] read udp REDACTED:21225->REDACTED:53: i/o timeout
2025-09-04T14:12:03Z [ERROR] read udp REDACTED:43711->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:42821->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:43706->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:19688->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:48784->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:30583->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:25345->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:53407->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:41013->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:23084->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:40908->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:23099->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:11909->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:44784->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:38213->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:19995->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:55447->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:43100->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:43512->REDACTED:53: i/o timeout
2025-09-04T14:12:06Z [ERROR] read udp REDACTED:25845->REDACTED:53: i/o timeout
```

September 4th, 2025

How does name resolution work?

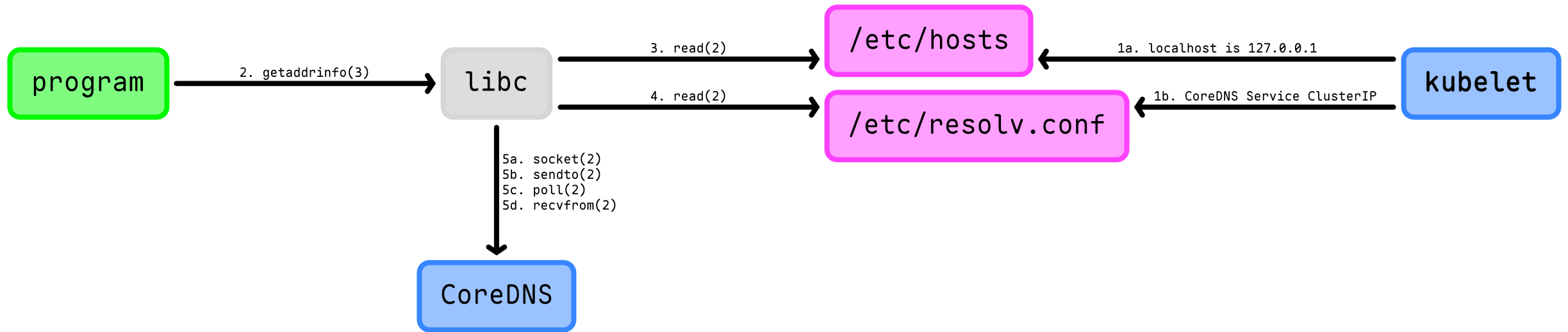
getaddrinfo(3)



**Non-standard behavior
hurts my feelings**

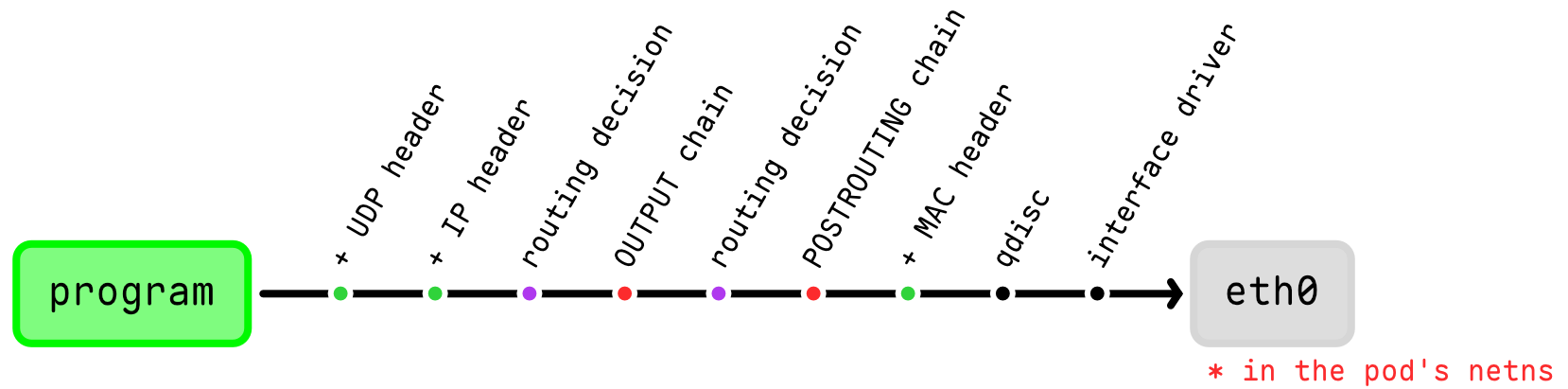
How does name resolution work?

getaddrinfo(3)



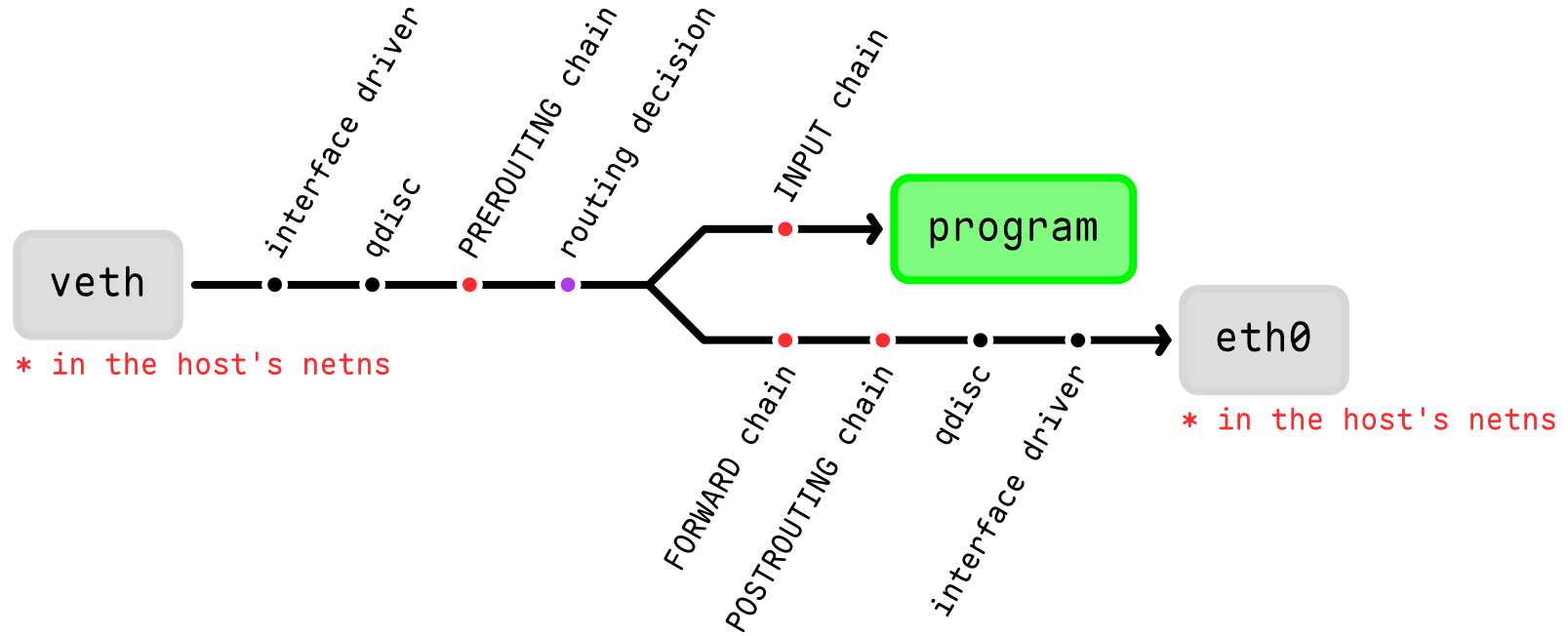
How does routing work?

Egress path



How does routing work?

Ingress path



Chain KUBE-SERVICES

target	prot	opt	in	out	source	destination	
KUBE-SVC-64172328	udp	--	*	*	0.0.0.0/0	10.96.0.10	udp dpt:53

Chain KUBE-SVC-64172328

target	prot	opt	in	out	source	destination	
KUBE-SEP-04AD2884	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	random probability 0.3333
KUBE-SEP-E8CE23BB	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	random probability 0.5000
KUBE-SEP-152964BC	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	random probability 1.0000

Chain KUBE-SEP-04AD2884

target	prot	opt	in	out	source	destination	
DNAT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	to:10.0.12.34:53

Chain KUBE-SEP-E8CE23BB

target	prot	opt	in	out	source	destination	
DNAT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	to:10.0.51.16:53

Chain KUBE-SEP-152964BC

target	prot	opt	in	out	source	destination	
DNAT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	to:10.0.3.7:53

The KUBE-SERVICES iptables chain

Chain KUBE-SERVICES

target	prot	opt	in	out	source	destination	
KUBE-SVC-64172328	udp	--	*	*	0.0.0.0/0	10.96.0.10	udp dpt:53

Chain KUBE-SVC-64172328

target	prot	opt	in	out	source	destination	
KUBE-SEP-04AD2884	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	random probability 0.3333
KUBE-SEP-E8CE23BB	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	random probability 0.5000
KUBE-SEP-152964BC	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	random probability 1.0000

Chain KUBE-SEP-04AD2884

target	prot	opt	in	out	source	destination	
DNAT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	to:10.0.12.34:53

Chain KUBE-SEP-E8CE23BB

target	prot	opt	in	out	source	destination	
DNAT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	to:10.0.51.16:53

Chain KUBE-SEP-152964BC

target	prot	opt	in	out	source	destination	
DNAT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	to:10.0.3.7:53

The KUBE-SVC iptables chain

The KUBE-SEP iptables chains

```
Chain KUBE-SERVICES
target
KUBE-SVC-64172328  udp  --  *  *  0.0.0.0/0  10.96.0.10  udp dpt:53
```

```
Chain KUBE-SVC-64172328
target          prot  opt  in  out  source          destination
KUBE-SEP-04AD2884  udp  --  *  *  0.0.0.0/0  0.0.0.0/0  random probability 0.3333
KUBE-SEP-E8CE23BB  udp  --  *  *  0.0.0.0/0  0.0.0.0/0  random probability 0.5000
KUBE-SEP-152964BC  udp  --  *  *  0.0.0.0/0  0.0.0.0/0  random probability 1.0000
```

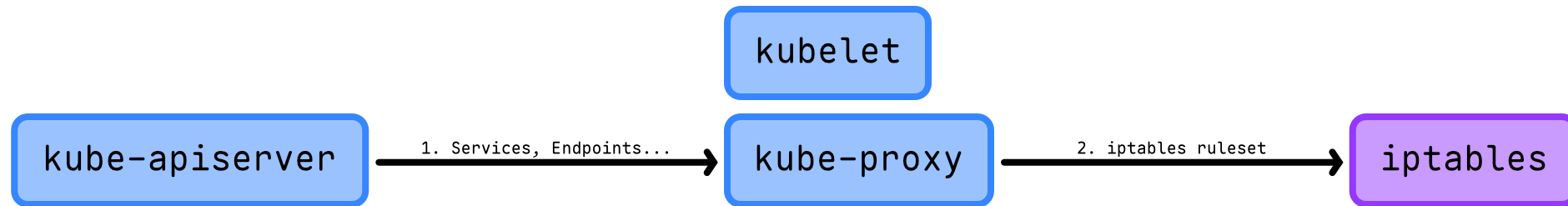
```
Chain KUBE-SEP-04AD2884
target          prot  opt  in  out  source          destination
DNAT            udp  --  *  *  0.0.0.0/0  0.0.0.0/0  to:10.0.12.34:53
```

```
Chain KUBE-SEP-E8CE23BB
target          prot  opt  in  out  source          destination
DNAT            udp  --  *  *  0.0.0.0/0  0.0.0.0/0  to:10.0.51.16:53
```

```
Chain KUBE-SEP-152964BC
target          prot  opt  in  out  source          destination
DNAT            udp  --  *  *  0.0.0.0/0  0.0.0.0/0  to:10.0.3.7:53
```

Our first agent of chaos: kube-proxy

Introducing kube-proxy



How does routing work?

Cilium



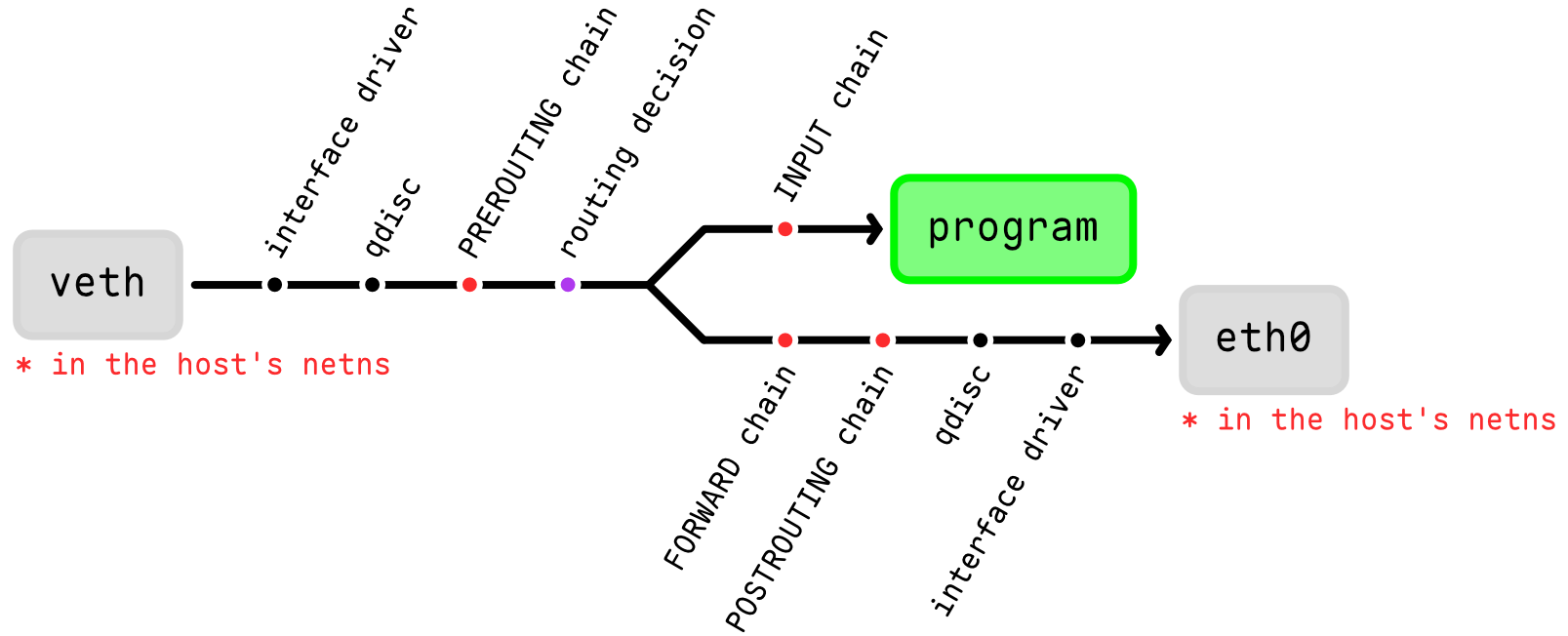
cilium

Map-based lookups. Instead of matching packets' destination IP addresses to a Service sequentially in the KUBE-SERVICES iptables chain.

eBPF data path. Can be configured to bypass iptables and replace kube-proxy completely.

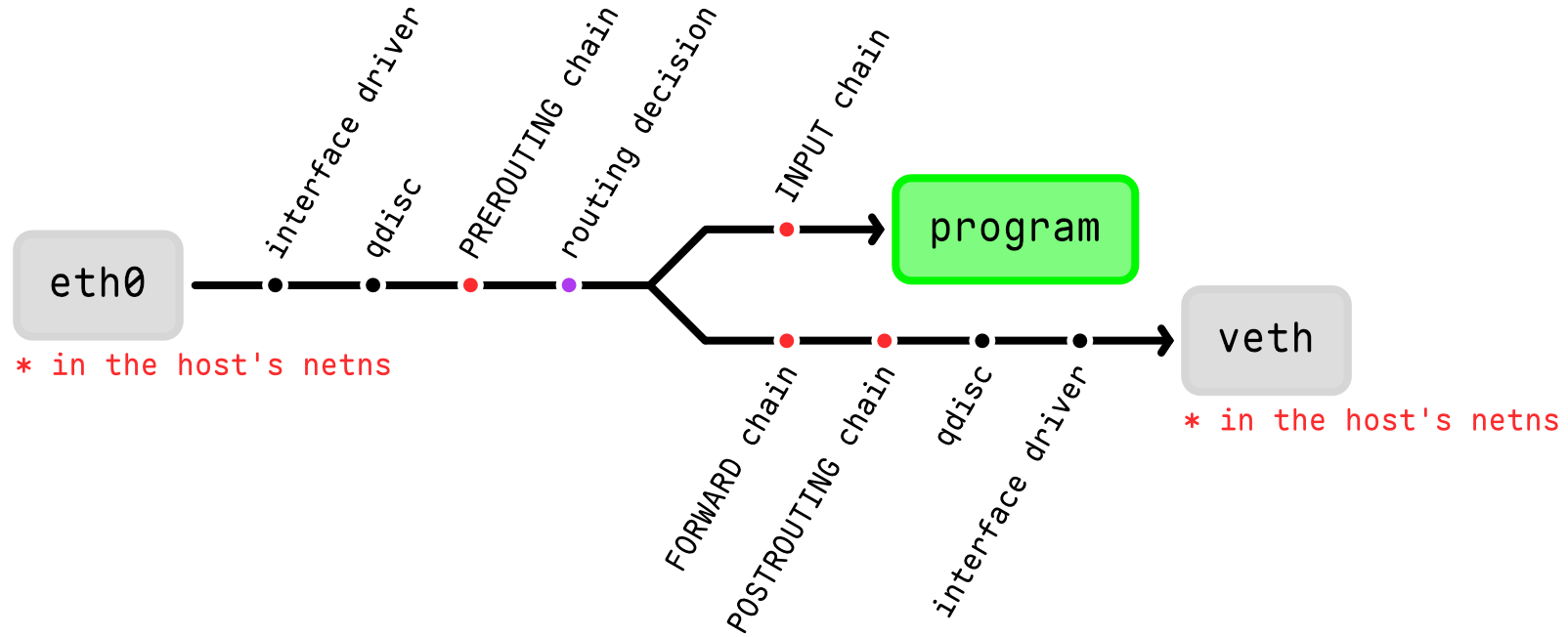
How does routing work?

Ingress path



How does routing work?

Ingress path (again)



```
2025-09-04T14:12:01Z [ERROR] read udp REDACTED:42999->REDACTED:53: i/o timeout
2025-09-04T14:12:01Z [ERROR] read udp REDACTED:59401->REDACTED:53: i/o timeout
2025-09-04T14:12:02Z [ERROR] read udp REDACTED:40757->REDACTED:53: i/o timeout
2025-09-04T14:12:03Z [ERROR] read udp REDACTED:18235->REDACTED:53: i/o timeout
2025-09-04T14:12:03Z [ERROR] read udp REDACTED:21225->REDACTED:53: i/o timeout
2025-09-04T14:12:03Z [ERROR] read udp REDACTED:43711->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:42821->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:43706->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:19688->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:48784->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:30583->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:25345->REDACTED:53: i/o timeout
2025-09-04T14:12:04Z [ERROR] read udp REDACTED:53407->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:41013->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:23084->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:40908->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:23099->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:11909->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:44784->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:38213->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:19995->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:55447->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:43100->REDACTED:53: i/o timeout
2025-09-04T14:12:05Z [ERROR] read udp REDACTED:43512->REDACTED:53: i/o timeout
2025-09-04T14:12:06Z [ERROR] read udp REDACTED:25845->REDACTED:53: i/o timeout
```

September 4th, 2025

```
$ kubectl describe pod coredns-6f9c7d8b6f-xyz12
```

```
Name: coredns-6f9c7d8b6f-xyz12
Namespace: kube-system
Node: ip-10-0-12-34.ec2.internal/10.0.12.34
Start Time: Thu, 4 Sep 2025 14:22:11 +0000
Labels: app=coredns
        pod-template-hash=6f9c7d8b6f
Status: Running
IP: 10.244.3.45
Controlled By: ReplicaSet/coredns-6f9c7d8b6f
```

A CoreDNS pod was OOMKilled

```
Containers:
  coredns:
    Container ID: docker://a1b2c3d4e5f67890123456789abcdef
    Image: docker.io/coredns:latest
    Port: 53/UDP
    State: Running
      Started: Thu, 4 Sep 2025 14:25:03 +0000
    Last State: Terminated
      Reason: OOMKilled
      Exit Code: 137
      Started: Thu, 4 Sep 2025 14:23:10 +0000
      Finished: Thu, 4 Sep 2025 14:24:58 +0000
    Ready: True
```

Our second agent of chaos

Istio

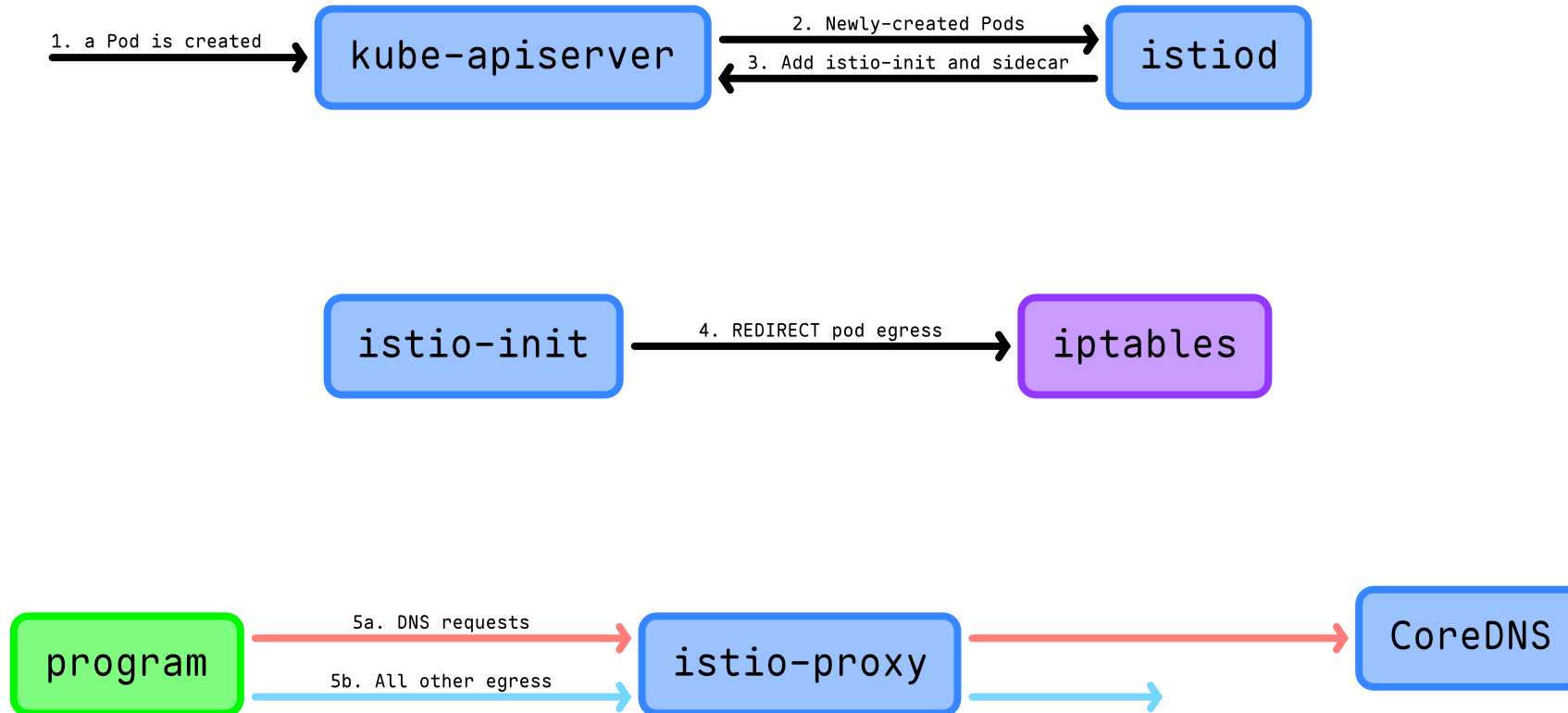


Istio

Our service mesh of choice. Makes some very complex things simple, but some very simple things complex.

Sidecar mode. Pods get an Istio sidecar container run alongside them.


How does Istio work?




Istio Service Entries

The screenshot shows a web browser window with the URL `istio.io/latest/docs/reference/config/networking/service-entry`. The page features the Istio logo and a navigation menu with links for About, Blog, News, Get involved, and Documentation. A search bar is present in the top right, along with a 'Try Istio' button. The main content area is titled 'Service Entry' and includes a breadcrumb trail: 'Documentation > Reference > Configuration > Traffic Management > Service Entry'. A search box is located on the left side of the page. The main text explains that ServiceEntry enables adding additional entries into Istio's internal service registry, allowing access to external services like web APIs or mesh-internal services. It also mentions that endpoints can be dynamically selected using the `workloadSelector` field. A code block at the bottom shows a YAML configuration for a ServiceEntry that routes traffic to external APIs like `api.dropboxapi.com` and `www.googleapis.com`. A sidebar on the right lists related topics: ServiceEntry, Location, Resolution, ServicePort, ServiceEntryStatus, and ServiceEntryAddress.

istio.io/latest/docs/reference/config/networking/service-entry

 Istio

About ▾ Blog News Get involved Documentation ▾  [Try Istio](#)

Documentation > Reference > Configuration > Traffic Management > Service Entry

Service Entry

🕒 13 minute read

`ServiceEntry` enables adding additional entries into Istio's internal service registry, so that auto-discovered services in the mesh can access/route to these manually specified services. A service entry describes the properties of a service (DNS name, VIPs, ports, protocols, endpoints). These services could be external to the mesh (e.g., web APIs) or mesh-internal services that are not part of the platform's service registry (e.g., a set of VMs talking to services in Kubernetes). In addition, the endpoints of a service entry can also be dynamically selected by using the `workloadSelector` field. These endpoints can be VM workloads declared using the `WorkloadEntry` object or Kubernetes pods. The ability to select both pods and VMs under a single service allows for migration of services from VMs to Kubernetes without having to change the existing DNS names associated with the services.

The following example declares a few external APIs accessed by internal applications over HTTPS. The sidecar inspects the SNI value in the ClientHello message to route to the appropriate external service.

```
apiVersion: networking.istio.io/v1
kind: ServiceEntry
metadata:
  name: external-svc-https
spec:
  hosts:
  - api.dropboxapi.com
  - www.googleapis.com
```

ServiceEntry

- Location
- Resolution
- ServicePort
- ServiceEntryStatus
- ServiceEntryAddress

Overview

- What is Istio?
- Why choose Istio?
- Sidecar or ambient?
- Quickstart

Concepts

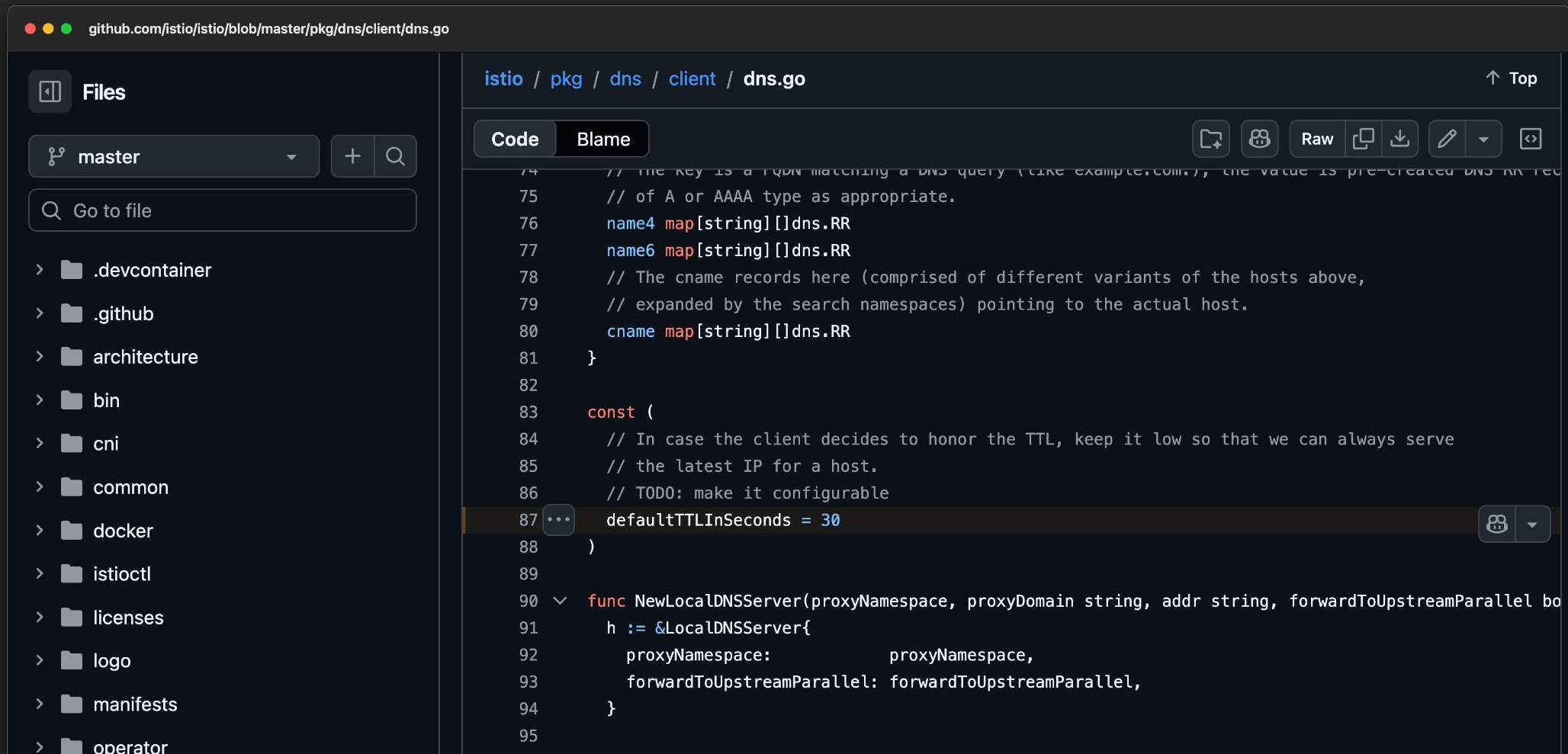
- Traffic Management
- Security
- Observability
- Extensibility

Sidecar Mode

- Getting Started
- Platform Setup
- Install
- Upgrade
- More Guides

© 2026 Cisco and/or its affiliates. All rights reserved.

This interval is hard-coded



The screenshot shows a GitHub repository view for the file `istio/pkg/dns/client/dns.go`. The left sidebar displays the file tree with folders like `.devcontainer`, `.github`, `architecture`, `bin`, `cni`, `common`, `docker`, `istioctl`, `licenses`, `logo`, `manifests`, and `operator`. The main content area shows the Go code with the following lines highlighted:

```
74 // The key is a fqdn matching a dns query (like example.com.), the value is pre-created dns RR rec
75 // of A or AAAA type as appropriate.
76 name4 map[string][]dns.RR
77 name6 map[string][]dns.RR
78 // The cname records here (comprised of different variants of the hosts above,
79 // expanded by the search namespaces) pointing to the actual host.
80 cname map[string][]dns.RR
81 }
82
83 const (
84 // In case the client decides to honor the TTL, keep it low so that we can always serve
85 // the latest IP for a host.
86 // TODO: make it configurable
87 defaultTTLInSeconds = 30
88 )
89
90 func NewLocalDNSServer(proxyNamespace, proxyDomain string, addr string, forwardToUpstreamParallel bo
91     h := &LocalDNSServer{
92         proxyNamespace:      proxyNamespace,
93         forwardToUpstreamParallel: forwardToUpstreamParallel,
94     }
95
```

The value `30` in line 87 is highlighted, indicating it is hard-coded. The Cisco logo is visible in the bottom right corner.

Our third agent of chaos

conntrack

Sets a *connection state* attribute. Matches packets to active connections based on other packets it has seen.

Prevents full iptables re-evaluation for packets of known established connections, saving CPU time waste.

```
$ sudo conntrack -L
tcp 6 431999 ESTABLISHED src=192.168.1.10 dst=172.17.0.2
tcp 6 120 ESTABLISHED src=192.168.1.11 dst=172.17.0.2
tcp 6 98 TIME_WAIT src=192.168.1.12 dst=151.101.1.1
udp 17 27 src=192.168.1.13 dst=8.8.8.8 sport=55442
tcp 6 300 ESTABLISHED src=10.0.0.5 dst=93.184.216.14
icmp 1 25 src=192.168.1.14 dst=1.1.1.1 type=8 code=0
tcp 6 250 ESTABLISHED src=192.168.1.15 dst=104.154.149.14
tcp 6 200 ESTABLISHED src=192.168.1.16 dst=151.101.1.1
udp 17 28 src=192.168.1.17 dst=8.8.4.4 sport=55442
tcp 6 180 TIME_WAIT src=192.168.1.18 dst=151.101.1.1
tcp 6 175 ESTABLISHED src=192.168.1.19 dst=172.17.0.2
udp 17 30 src=192.168.1.20 dst=1.1.1.1 sport=6080
tcp 6 160 ESTABLISHED src=192.168.1.21 dst=93.184.216.14
tcp 6 155 ESTABLISHED src=192.168.1.22 dst=104.154.149.14
udp 17 29 src=192.168.1.23 dst=9.9.9.9 sport=55442
tcp 6 150 TIME_WAIT src=192.168.1.24 dst=151.101.1.1
tcp 6 145 ESTABLISHED src=192.168.1.25 dst=172.17.0.2
udp 17 28 src=192.168.1.26 dst=8.26.56.26 sport=55442
tcp 6 140 ESTABLISHED src=192.168.1.27 dst=151.101.1.1
tcp 6 135 ESTABLISHED src=192.168.1.28 dst=13.107.141.101
udp 17 27 src=192.168.1.29 dst=208.67.222.222 sport=55442
tcp 6 130 TIME_WAIT src=192.168.1.30 dst=151.101.1.1
tcp 6 125 ESTABLISHED src=192.168.1.31 dst=104.154.149.14
```

Kubernetes issue #126130

github.com/kubernetes/kubernetes/issues/126130

kubernetes / kubernetes

Q Type / to search

Code Issues 1.8k Pull requests 794 Actions Projects Security Insights

Implement a kube-proxy conntrack reconciler #126130

New issue

Closed #127318

aojea opened on Jul 16, 2024 · edited by aojea

Edits Member

What would you like to be added?

We'd like to have conntrack reconciler for Services in kube-proxy, for each Service or Endpoint change it should reconcile the Service/Endpoint table with the existing conntrack table and remove stale UDP entries (we have concluded in other issues that UDP is the only protocol that requires this)

Why is this needed?

Despite a lot of work and testing was added to solve the problem with stale UDP entries that can blackhole traffic, we still have reports of issues related to UDP stale entries

Assignees

aroradaman


Labels

kind/bug kind/feature sig/network triage/accepted

Type

No type

Projects





Istio

Extremely **aggressive** cache refresh mechanism.

Reuse of the same client port for all DNS requests.



kube-proxy

Lack of a **cleanup mechanism** for **stale contrack entries** of terminated pods.

Fixed in v1.32.



contrack

Default bidirectional UDP stream **timeout** setting of **2 minutes**.

The inconspicuous role of contrack in Kubernetes networking

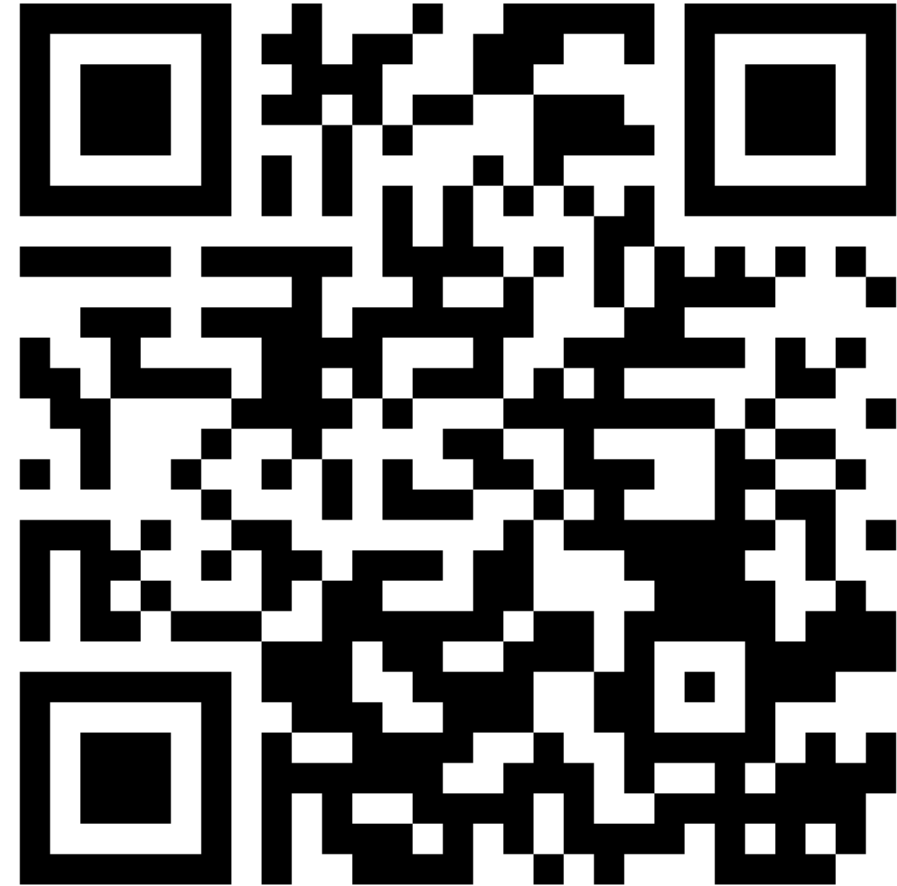
Ricard Bejarano

Lead Site Reliability Engineer
Cisco ThousandEyes



**I'm writing a
book about
homelabbing!**

O'REILLY®



bejarano.io/the-homelab-handbook

Meet me if you have a homelab!