Blackholes & Wormholes:

Understand and Troubleshoot the “Magic” of k8s Networking

KubeCon Europe
May 3 2018

Minhan Xia <github.com/freehan>
Rohit Ramkumar <github.com/rramkumar1>

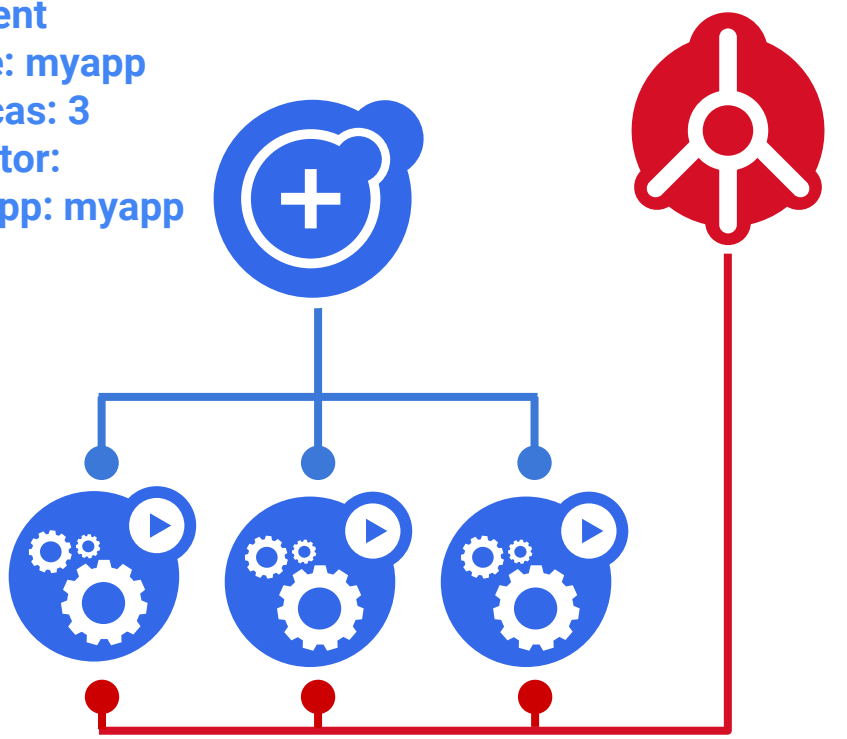
Overview

- Case Studies
- Lessons Learned
- Best Practices

Case Study: Blackhole

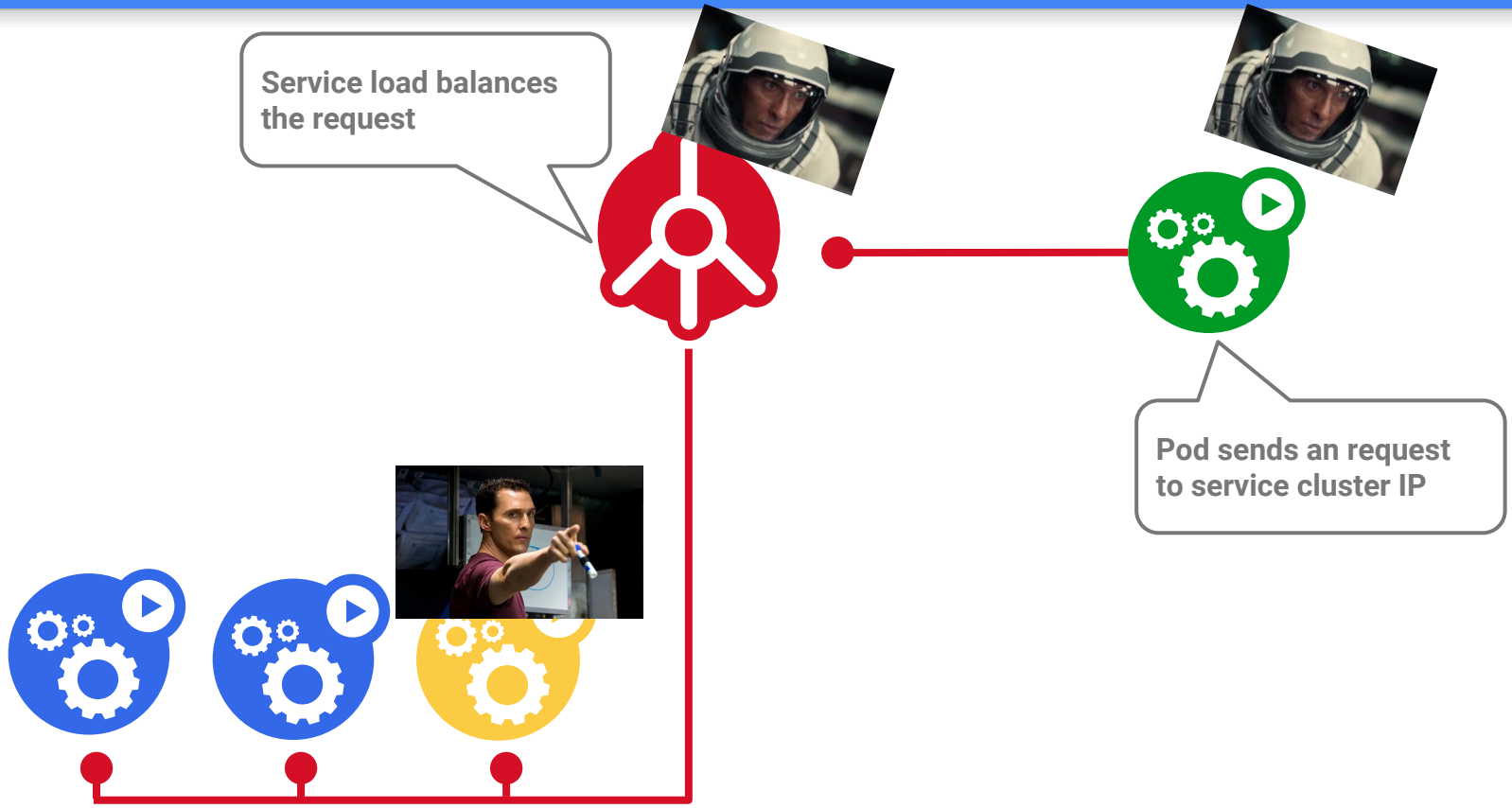
Blackhole - Set Up

Deployment
- name: myapp
- replicas: 3
- selector:
  - app: myapp

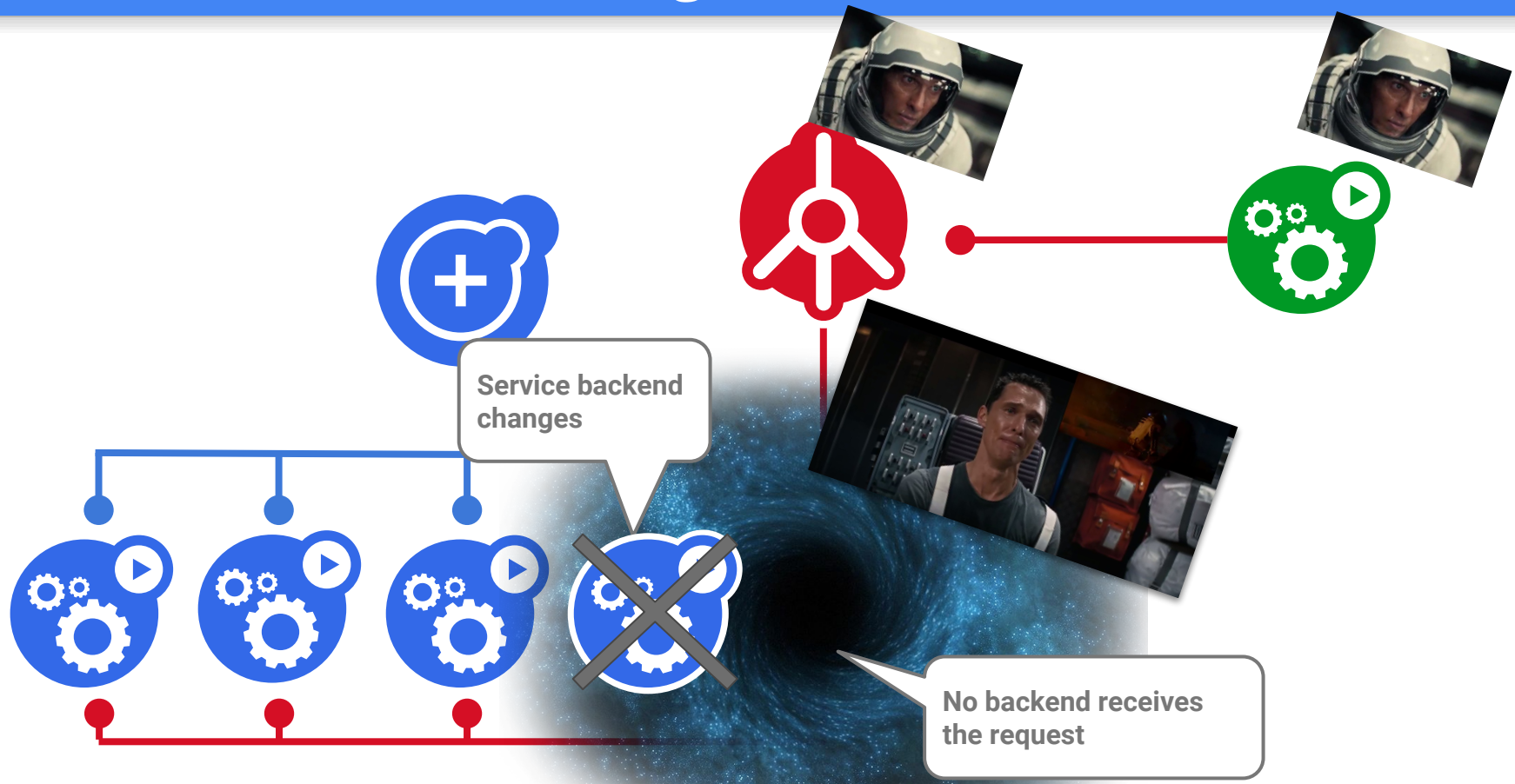


Service
- app: myapp
- type: ClusterIP
- ports:
  - port: 53
  - protocol: udp

Blackhole - Happy Ending



Blackhole - Sad Ending



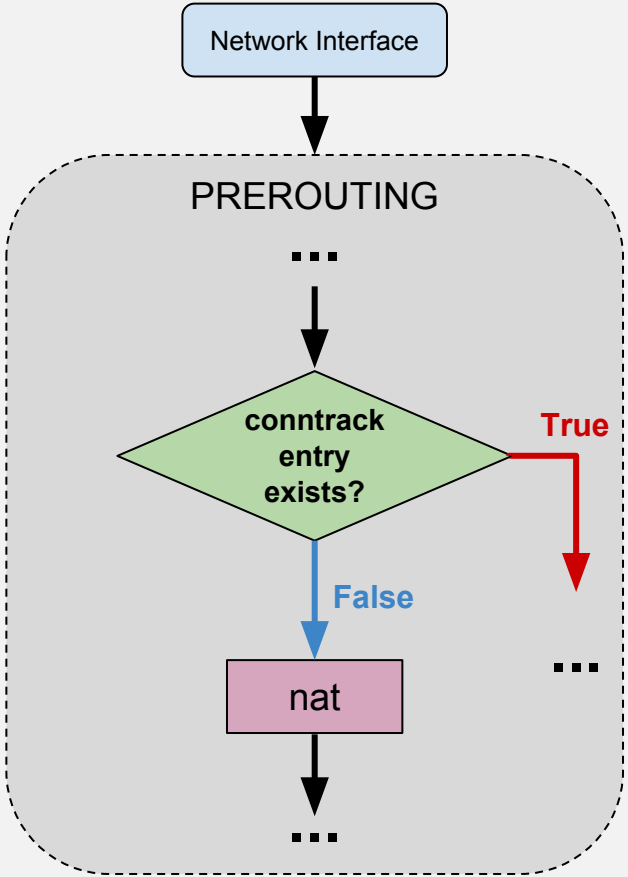
Conntrack in a Nutshell

- Linux kernel connection-tracking
- Remembers address translations
- Based on the 5-tuple
- Reversed on the return path

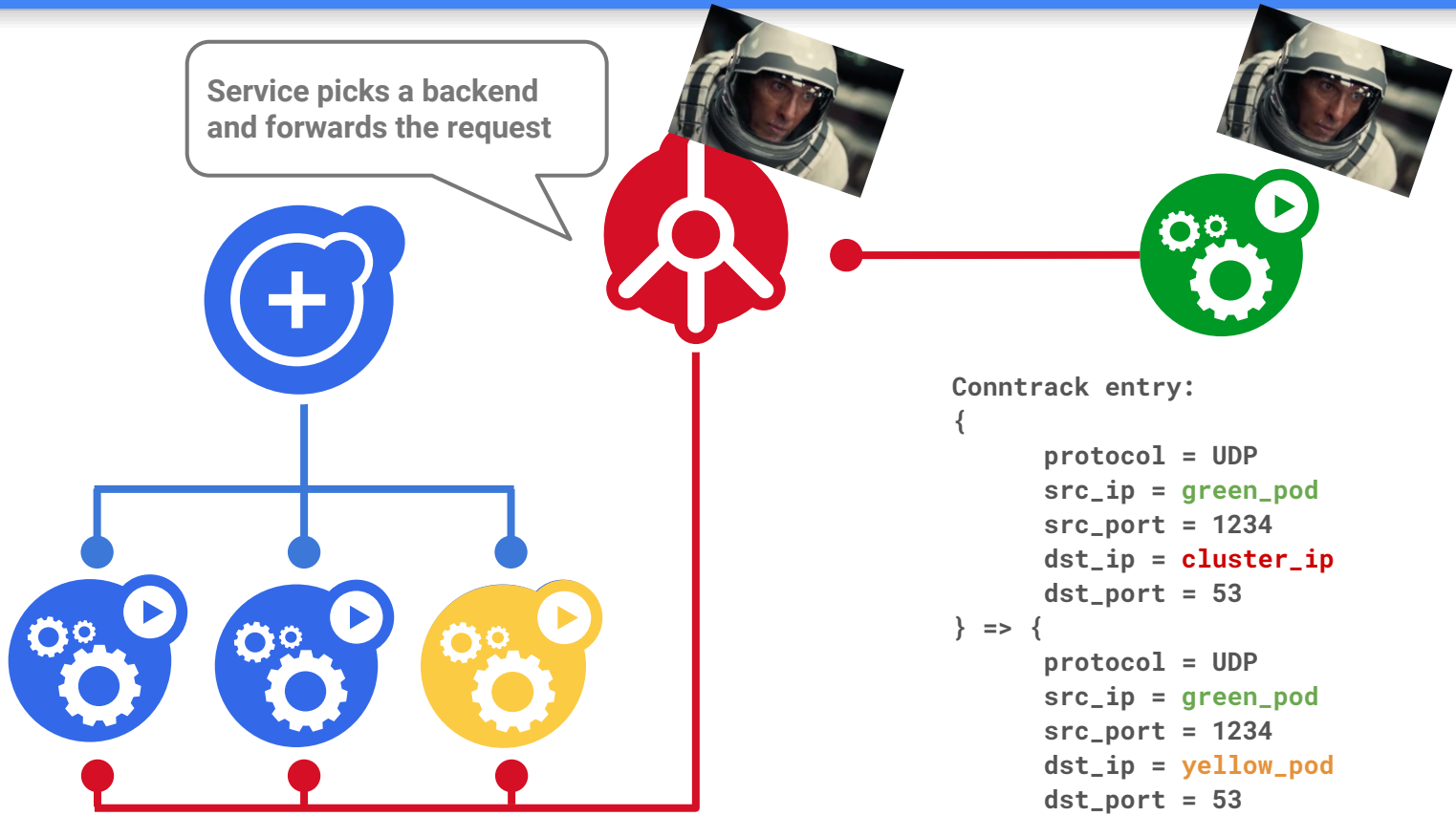
```
{
    protocol = TCP
    src_ip = pod1
    src_port = 1234
    dst_ip = svc1
    dst_port = 80
} => {
    protocol = TCP
    src_ip = pod1
    src_port = 1234
    dst_ip = pod99
    dst_port = 80
}
```

Netfilter in a Nutshell

- Linux packet filtering framework
- Provides “**hooks**” to intercept and manipulate network packets
- Capable of packet filtering, network address translation, and port translation
- iptables, ebtables, conntrack table and etc...

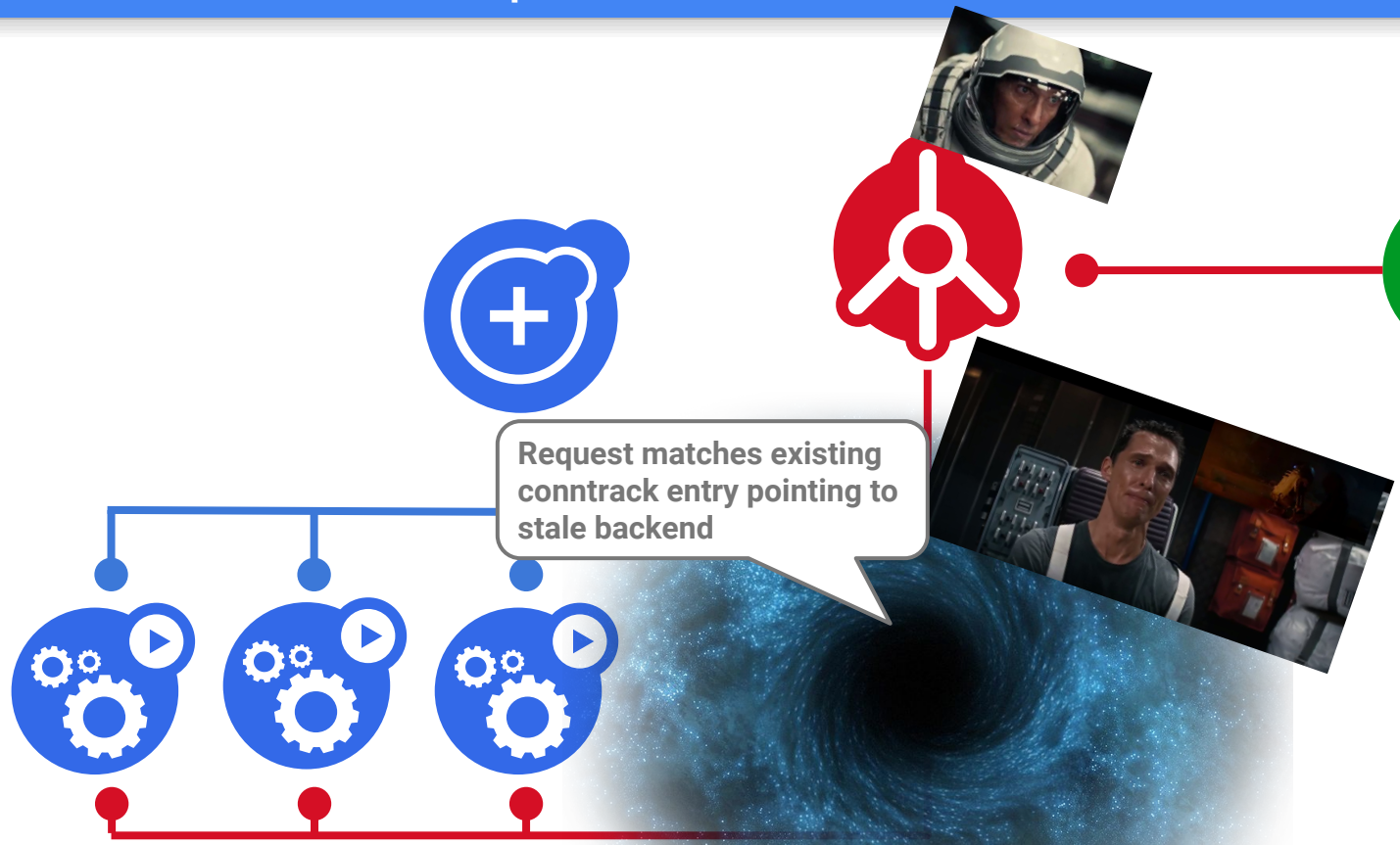


Blackhole - Explained




```
Conntrack entry:
{
        protocol = UDP
        src_ip = green_pod
        src_port = 1234
        dst_ip = cluster_ip
        dst_port = 53
} => {
        protocol = UDP
        src_ip = green_pod
        src_port = 1234
        dst_ip = yellow_pod
        dst_port = 53
}
```

Blackhole - Explained




```
Conntrack entry:
{
        protocol = UDP
        src_ip = green_pod
        src_port = 1234
        dst_ip = cluster_ip
        dst_port = 53
} => {
        protocol = UDP
        src_ip = green_pod
        src_port = 1234
        dst_ip = yellow_pod
        dst_port = 53
}
```

Blackhole - Lesson Learned

Conntrack

+

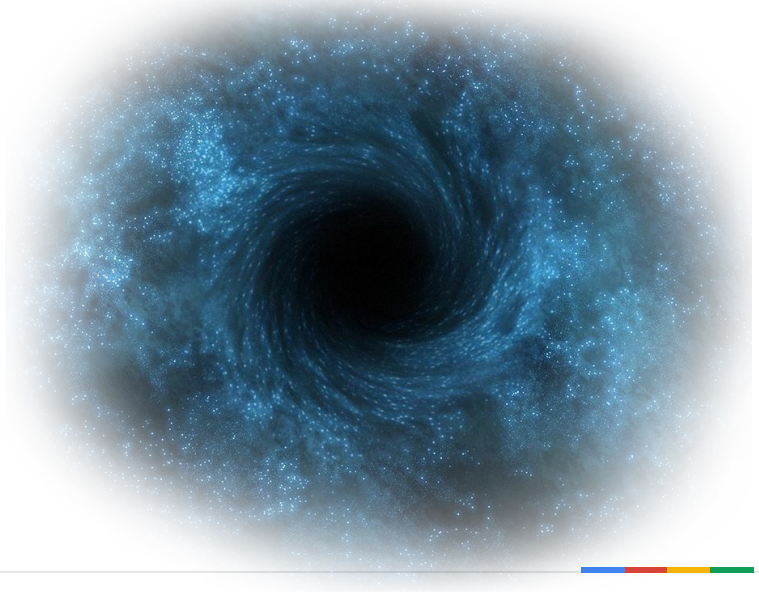
NAT

+

UDP

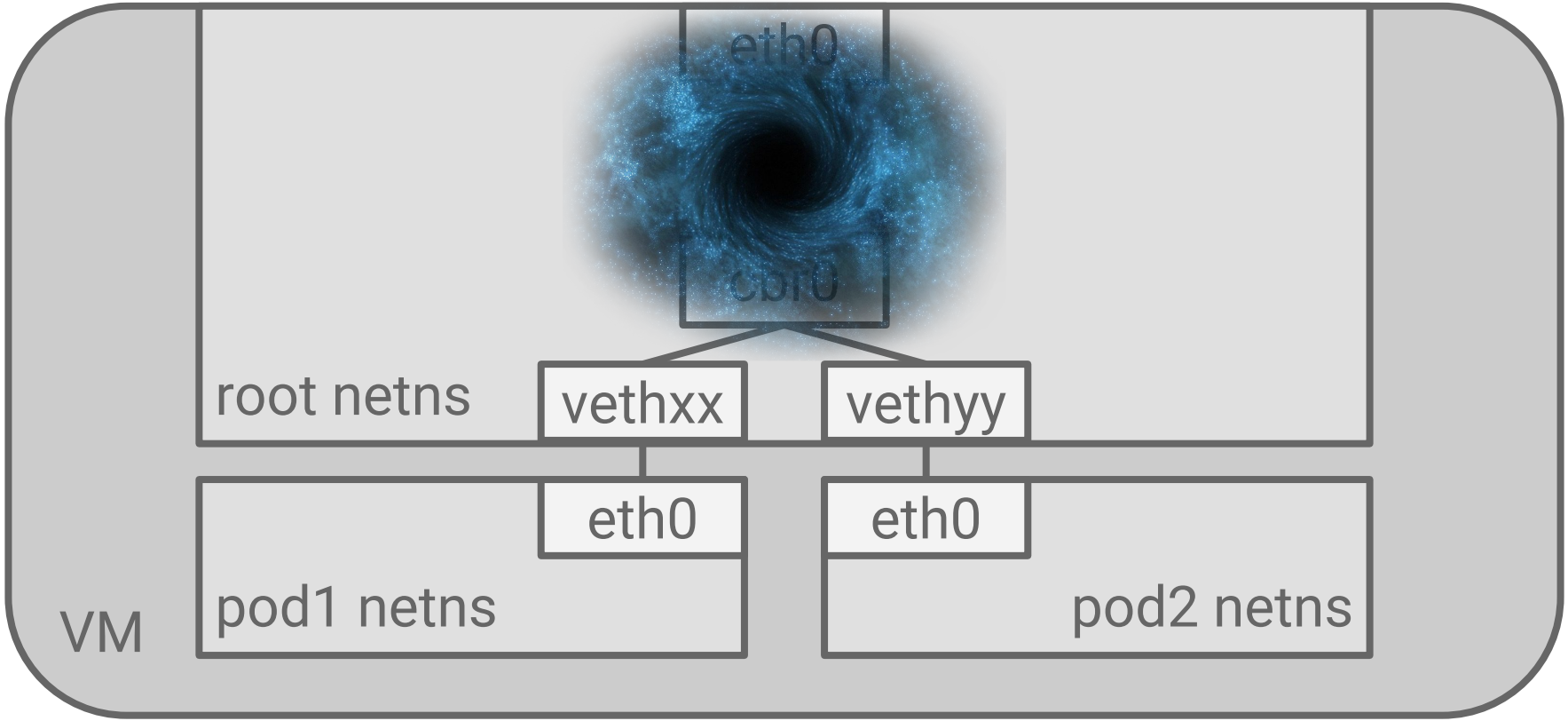
+

Ephemeral nature of pods =



Case Study: Yet Another Blackhole

Blackhole #2 - Set Up



Blackhole #2 - Explained

- Memory Pressure
- Systemd Networkd got OOM killed
- Systemd Networkd bug - On restart, reset:

net.ipv4.conf.eth0.forwarding = 0

Lesson Learned

- Dig deeper
- OS/Kernel config matters

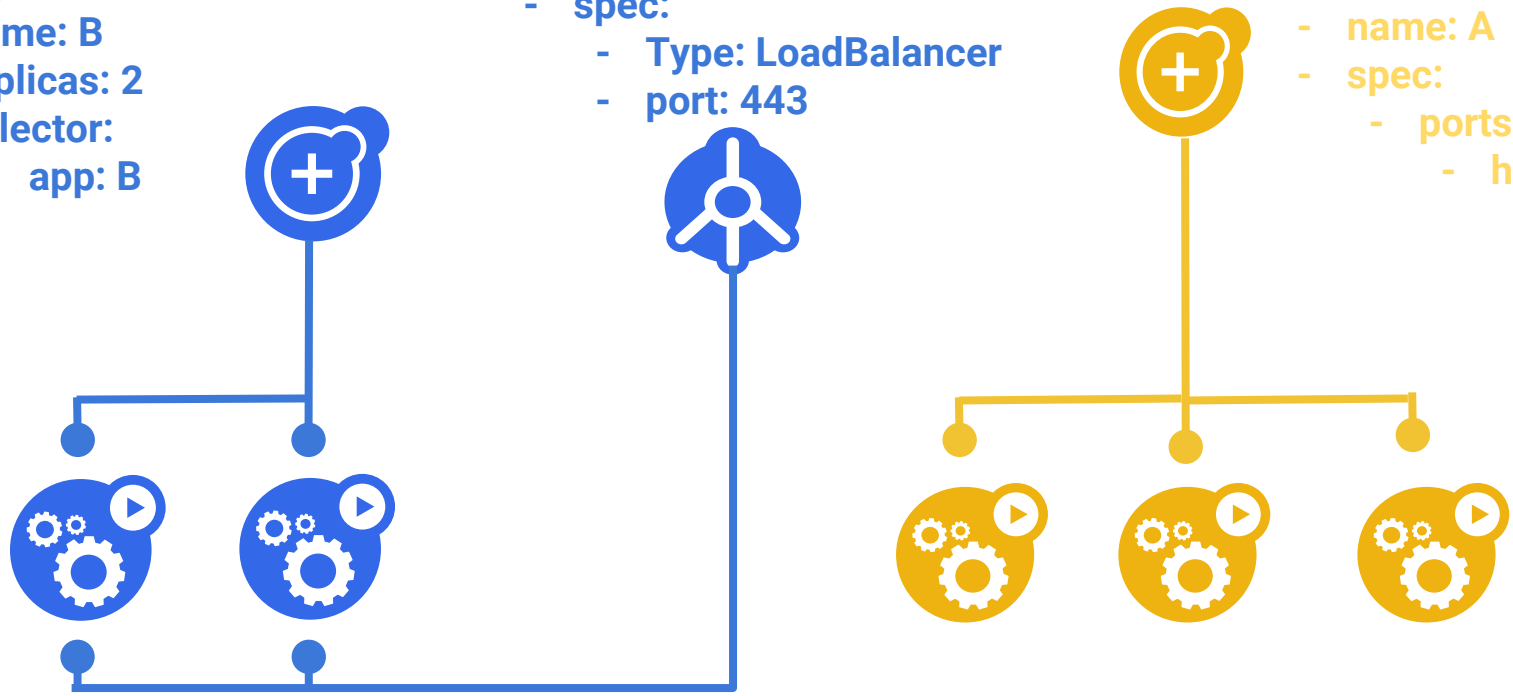
Case Study: Wormhole

Wormhole - Set Up

Deployment
- name: B
- replicas: 2
- selector:
  - app: B

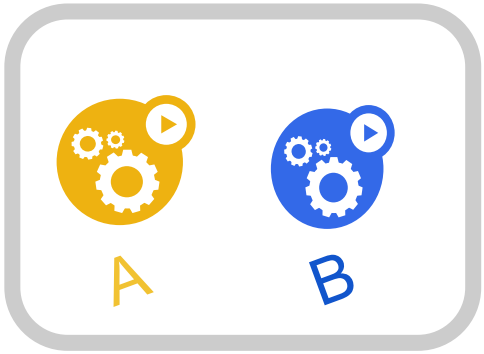
Service
- app: B
- spec:
  - Type: LoadBalancer
  - port: 443

DaemonSet
- name: A
- spec:
  - ports:
    - hostPort: 443

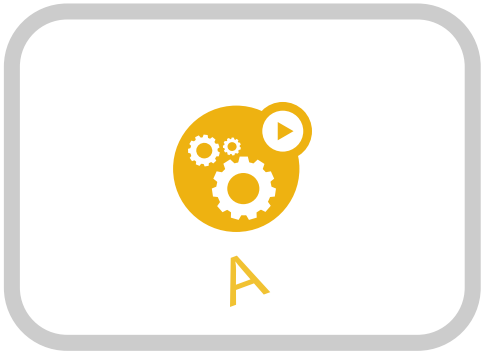


Wormhole - What Happened

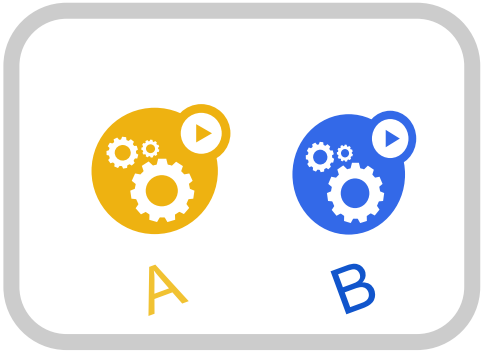
35.194.18.174



10.128.0.2



10.128.0.3



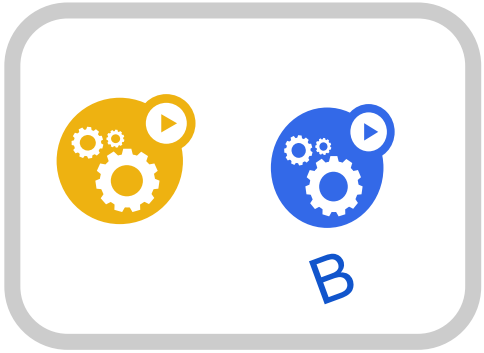
10.128.0.4

Wormhole - What Happened

LB

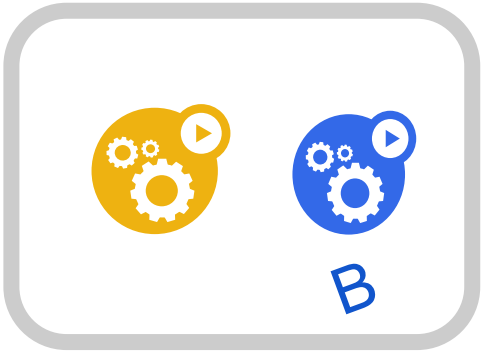
35.194.18.174

Client hits LB VIP at
443 to talk to service B



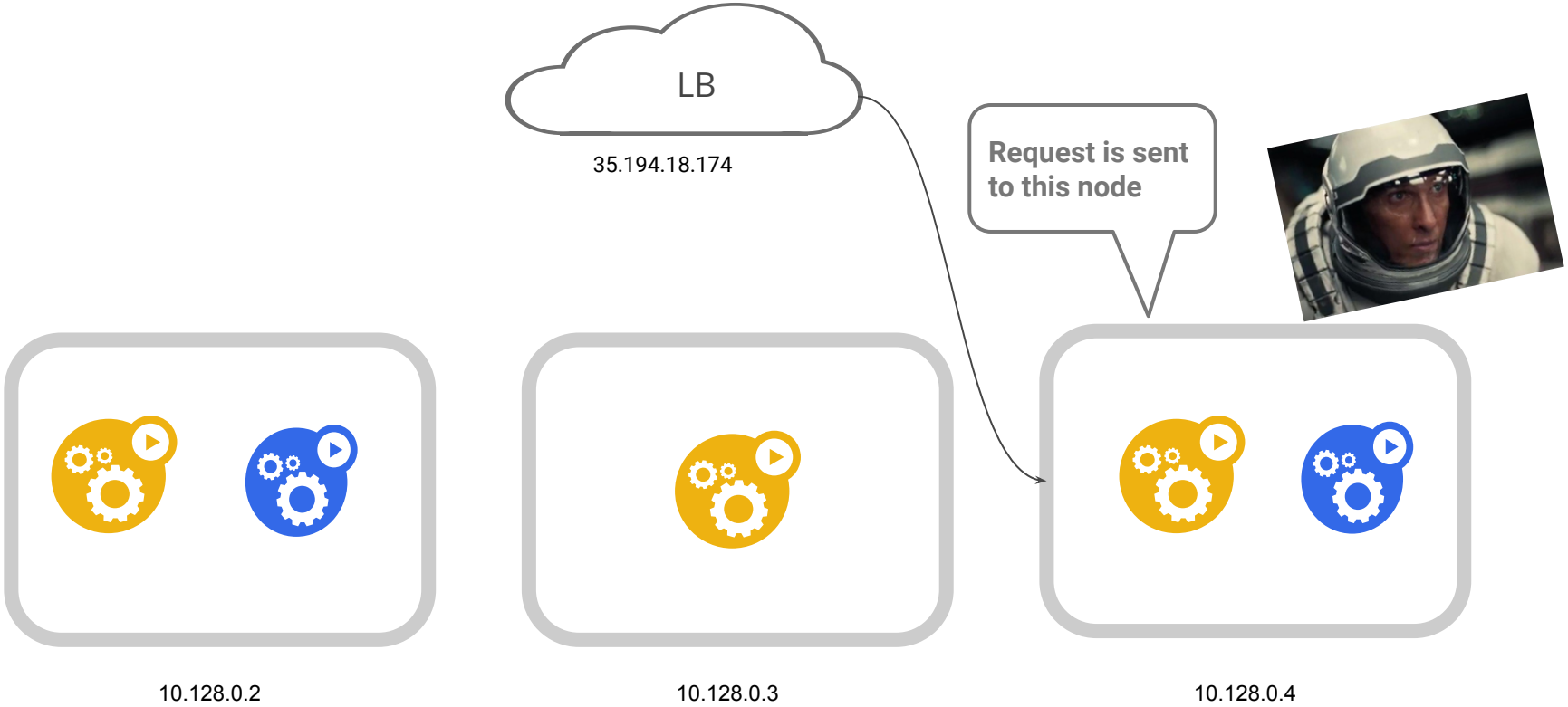
10.128.0.2

10.128.0.3

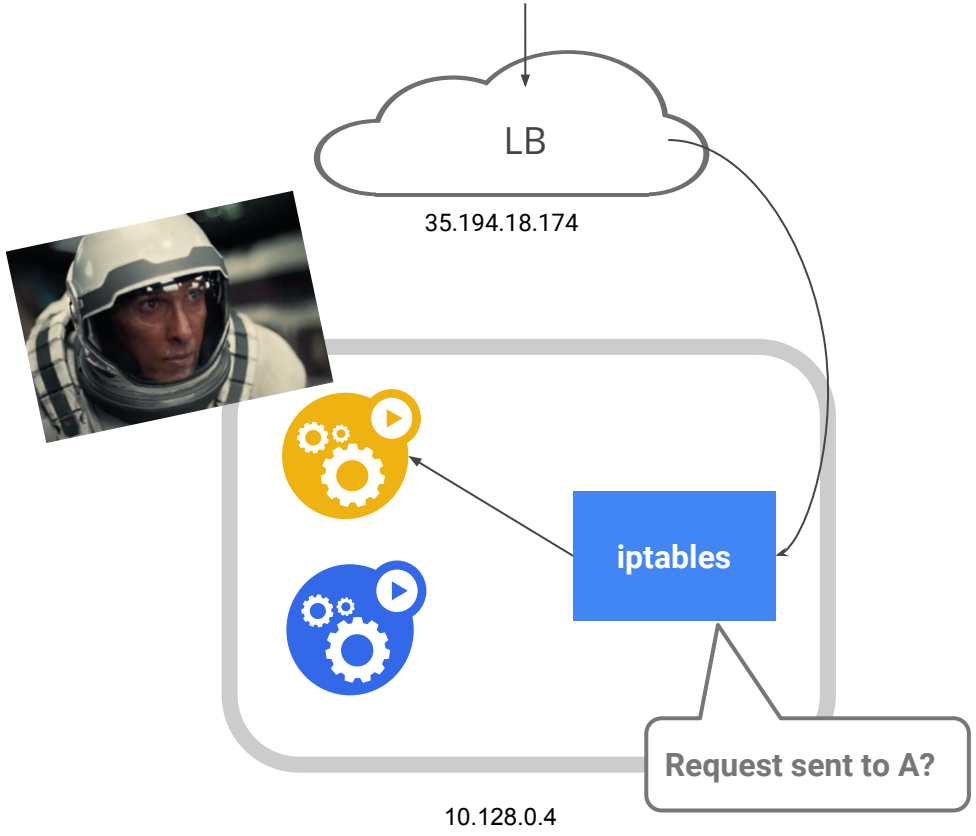


10.128.0.4

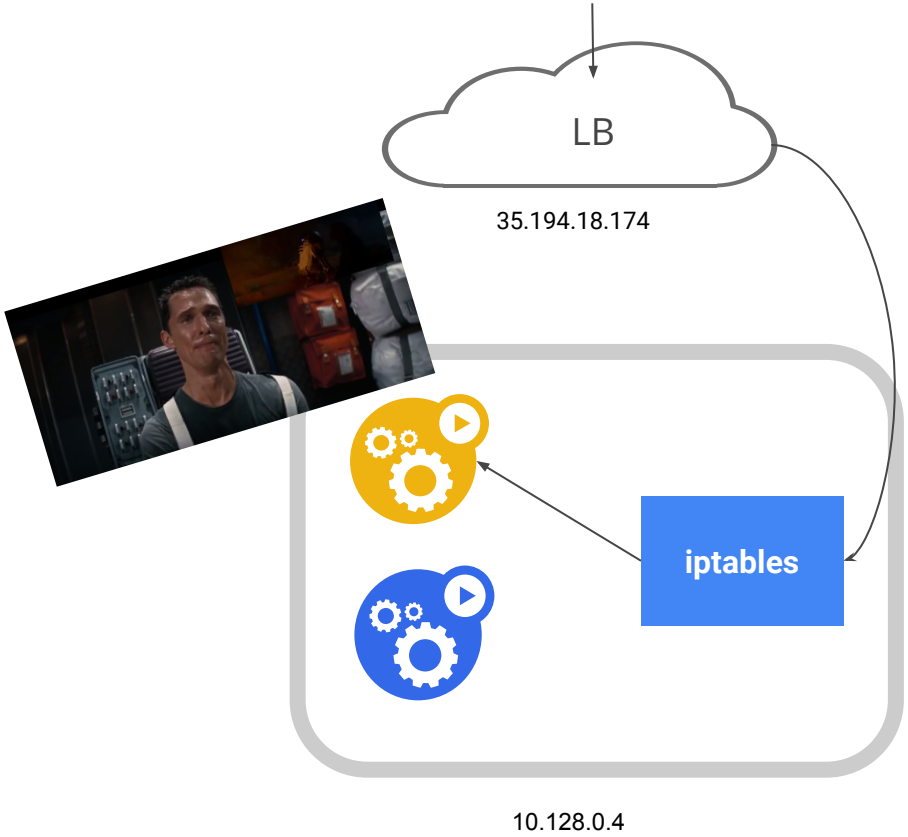
Wormhole - What Happened



Wormhole - What Happened

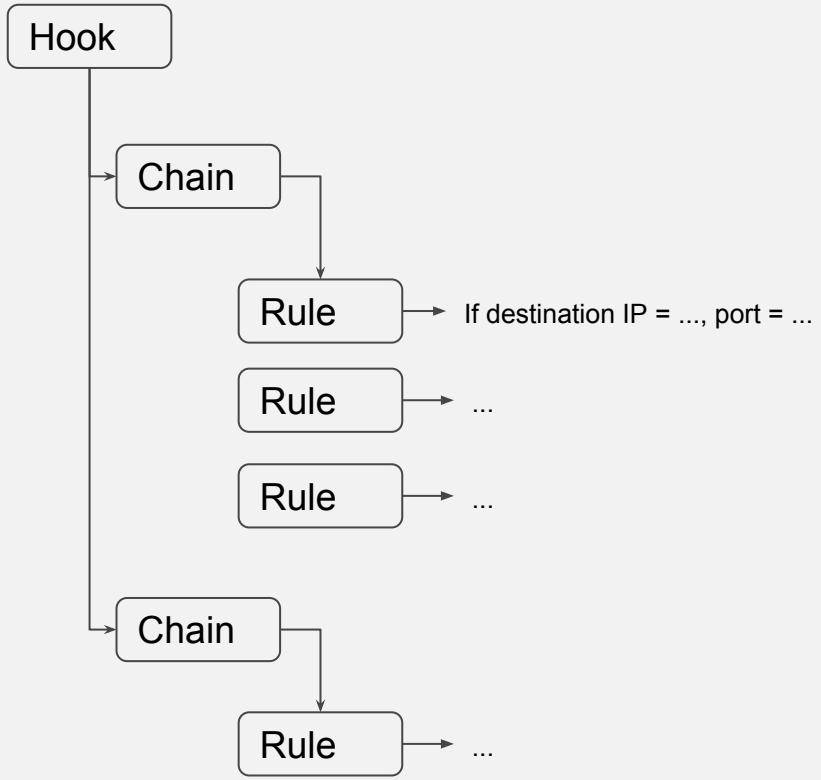


Wormhole - What Happened

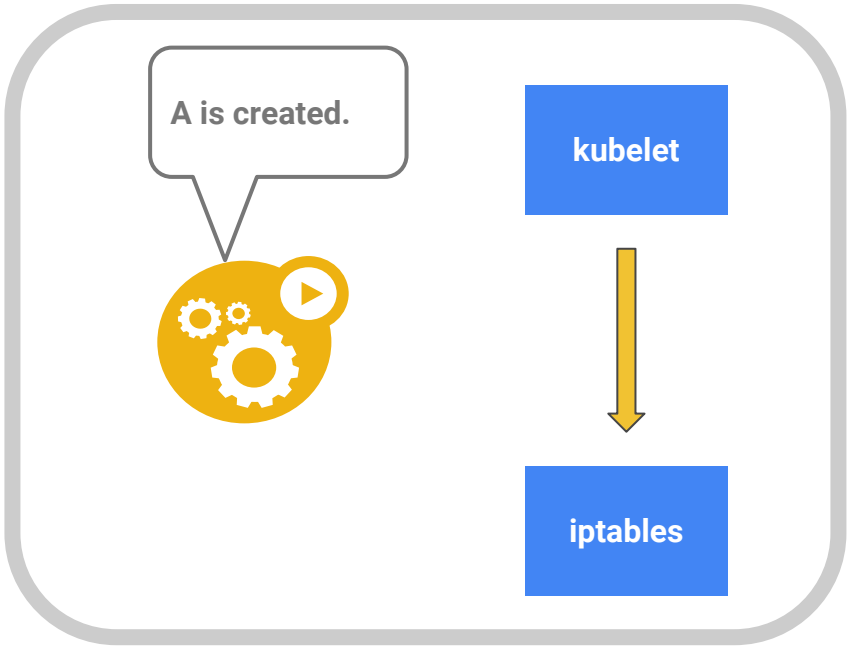


Iptables in a Nutshell

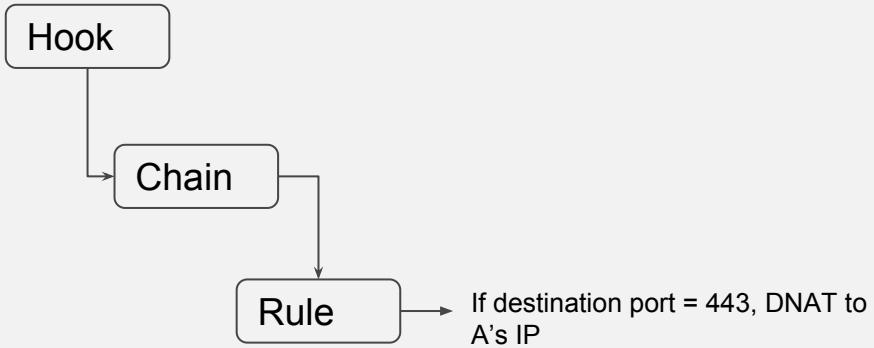
- Implements service routing + “load balancing” in k8s.
- Configured by both kube-proxy & kubelet.
- Implemented using Netfilter hooks



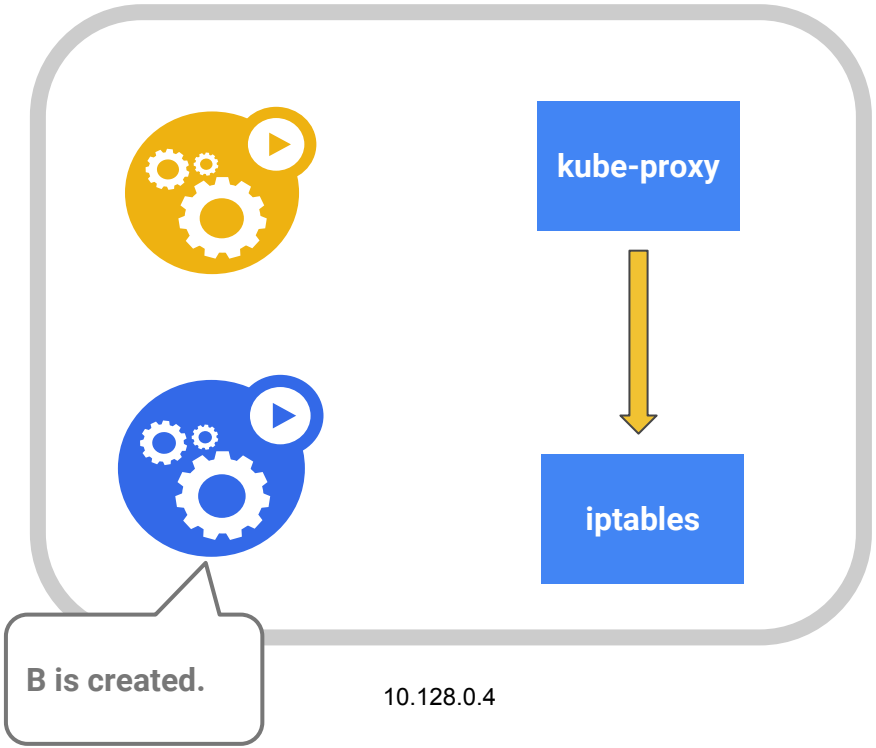
Wormhole - Why

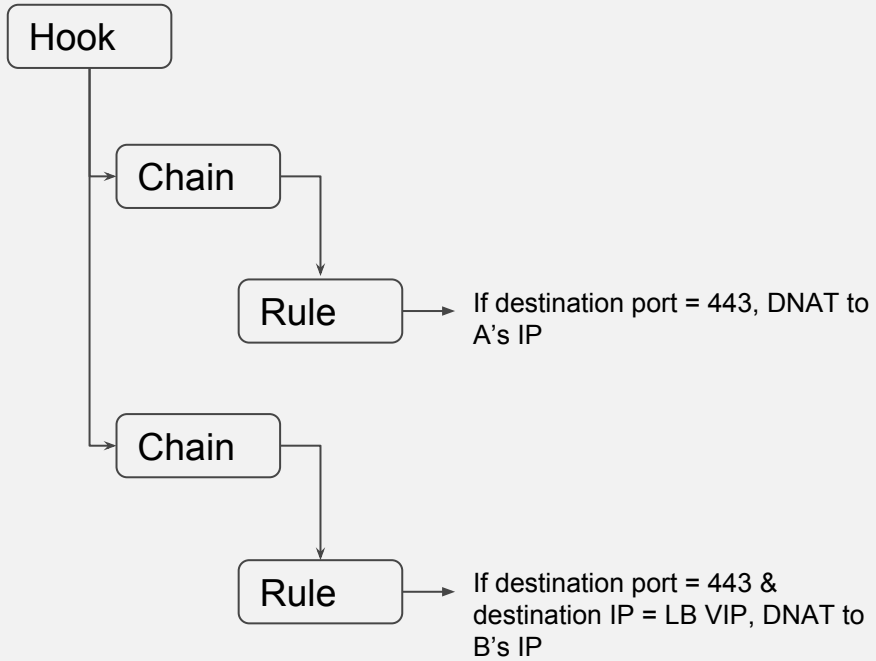


10.128.0.4

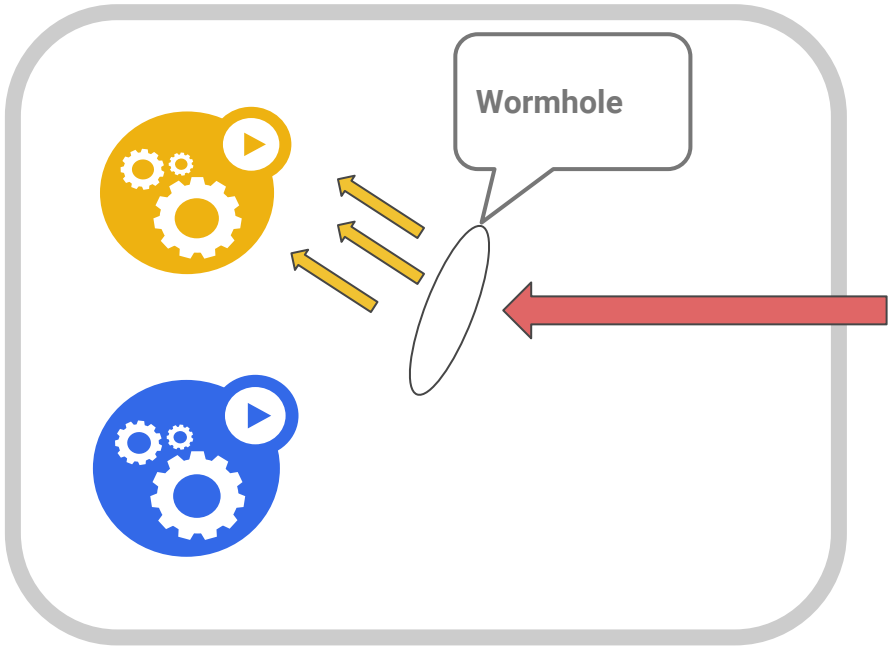


Wormhole - Why

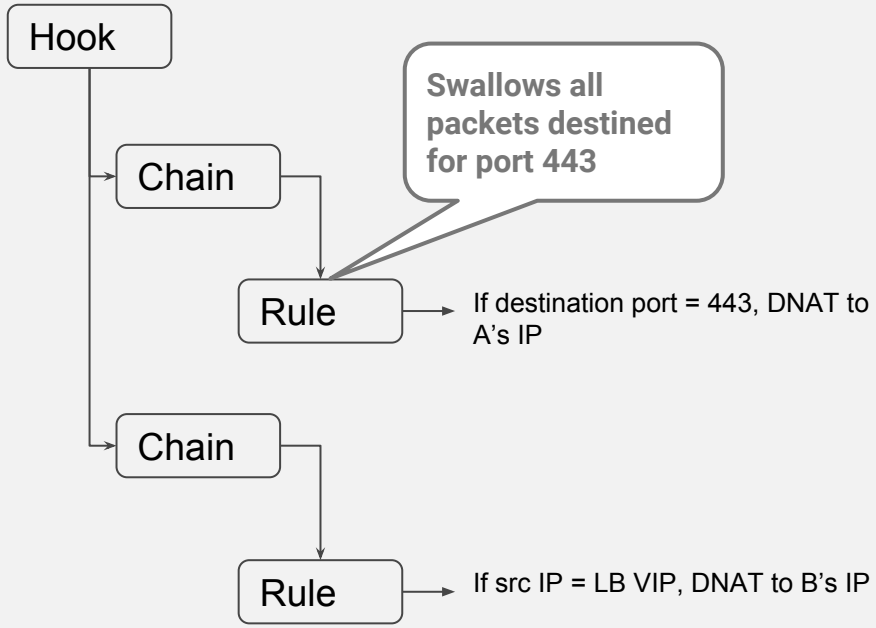




Wormhole - Why



10.128.0.4



Wormhole - Lesson Learned

- Iptables is tricky
- Rules should be as explicit as possible (i.e narrow)
- Rules should be precedence agnostic

Troubleshooting Best Practices

What's in my iptables?

```
$ iptables-save

  -A OUTPUT ... "kubernetes service portals" -j KUBE-SERVICES

  -A KUBE-SERVICES -d 10.0.16.10/32 -p udp ... "kube-system/kube-dns:dns cluster IP" -m
  udp --dport 53 -j KUBE-SVC-TCOU7JCQXEZGVUNU

  -A KUBE-SVC-TCOU7JCQXEZGVUNU ... "kube-system/kube-dns:dns" -m statistic --mode random
  --probability 0.50000000000 -j KUBE-SEP-RWNL743MFJNVLAU2
  -A KUBE-SVC-TCOU7JCQXEZGVUNU ... "kube-system/kube-dns:dns" -j KUBE-SEP-NCG4O2FBJHD7SOS3

  -A KUBE-SEP-RWNL743MFJNVLAU2 -p udp -m comment --comment "kube-system/kube-dns:dns" -m
  udp -j DNAT --to-destination 10.8.3.4:53
  -A KUBE-SEP-NCG4O2FBJHD7SOS3 -p udp -m comment --comment "kube-system/kube-dns:dns" -m
  udp -j DNAT --to-destination 10.8.3.6:53
```

What's in my iptables?

```
$ iptables-save

  -A OUTPUT ... "kubernetes service portals" -j KUBE-SERVICES

  -A KUBE-SERVICES -d 10.0.16.10/32 -p udp ... "kube-system/kube-dns:dns cluster IP" -m
  udp --dport 53 -j KUBE-SVC-TCOU7JCQXEZGVUNU
```

10.0.16.10

What's in my iptables?

```
$ iptables-save

  -A OUTPUT ... "kubernetes service portals" -j KUBE-SERVICES

  -A KUBE-SERVICES -d 10.0.16.10/32 -p udp ... "kube-system/kube-dns:dns cluster IP" -m
  udp --dport 53 -j KUBE-SVC-TCOU7JCQXEZGVUNU

-A KUBE-SVC-TCOU7JCQXEZGVUNU ... "kube-system/kube-dns:dns" -m statistic --mode random
  --probability 0.50000000000 -j KUBE-SEP-RWNL743MFJNVLAU2
  -A KUBE-SVC-TCOU7JCQXEZGVUNU ... "kube-system/kube-dns:dns" -j KUBE-SEP-NCG4O2FBJHD7SOS3
```

What's in my iptables?

```
$ iptables-save

-A KUBE-SVC-TCOU7JCQXEZGVUNU ... "kube-system/kube-dns:dns" -m statistic --mode random
--probability 0.50000000000 -j KUBE-SEP-RWNL743MFJNVLAU2
-A KUBE-SVC-TCOU7JCQXEZGVUNU ... "kube-system/kube-dns:dns" -j KUBE-SEP-NCG4O2FBJHD7SOS3

-A KUBE-SEP-RWNL743MFJNVLAU2 -p udp -m comment --comment "kube-system/kube-dns:dns" -m
udp -j DNAT --to-destination 10.8.3.4:53
-A KUBE-SEP-NCG4O2FBJHD7SOS3 -p udp -m comment --comment "kube-system/kube-dns:dns" -m
udp -j DNAT --to-destination 10.8.3.6:53
```

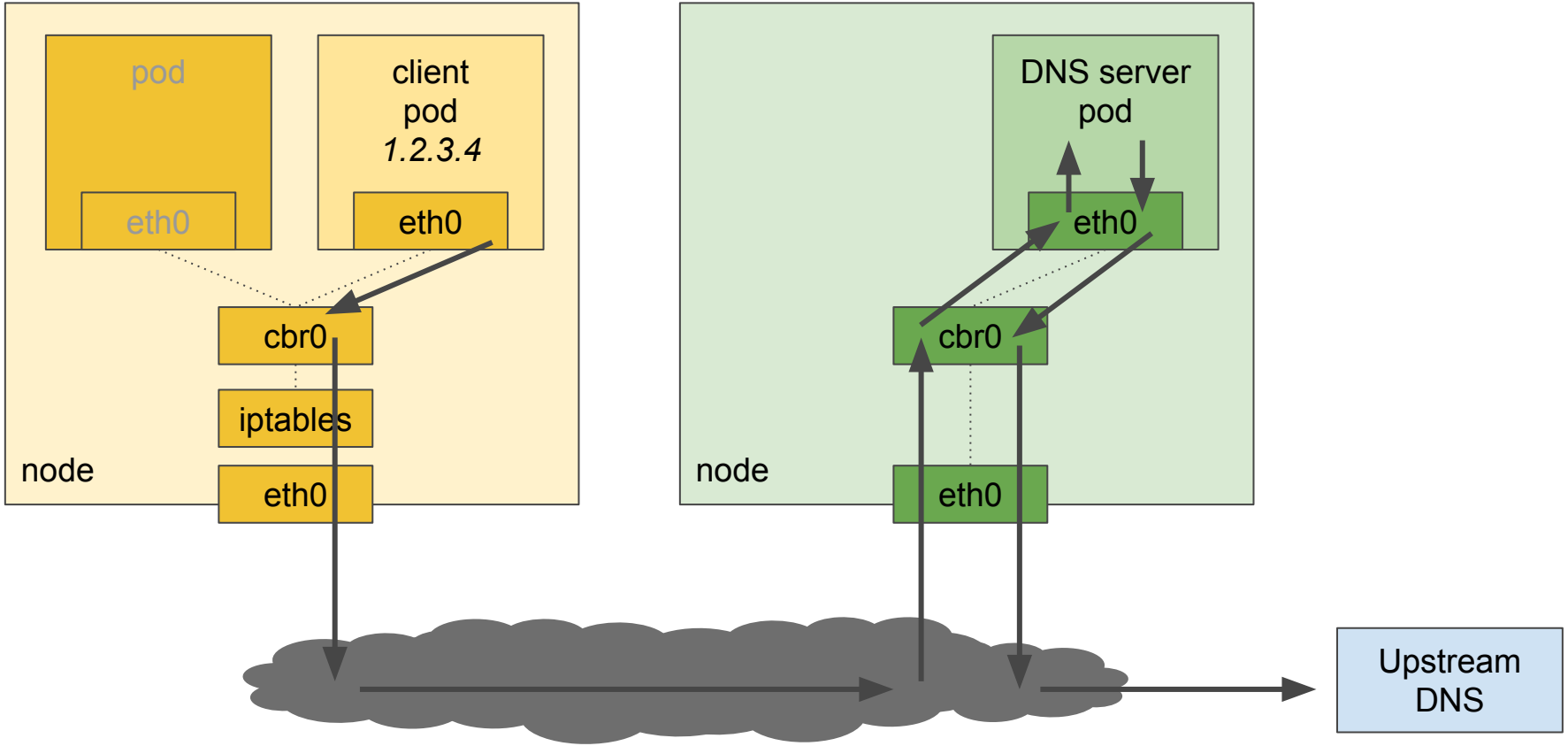
What's in my conntrack?

```
$ conntrack -L


ipv4     2 tcp      6 77 TIME_WAIT src=10.84.0.1 dst=10.84.0.3 sport=32804 dport=8080 src=10.84.0.3
  dst=10.84.0.1 sport=8080 dport=32804 [ASSURED] mark=0 zone=0 use=2

.
.
.
.
.
.
.
.

ipv4     2 tcp      6 87 TIME_WAIT src=35.191.255.128 dst=10.128.0.4 sport=49798 dport=31024 src=10.84.0.7
  dst=10.84.0.1 sport=8080 dport=49798 [ASSURED] mark=0 zone=0 use=2
```

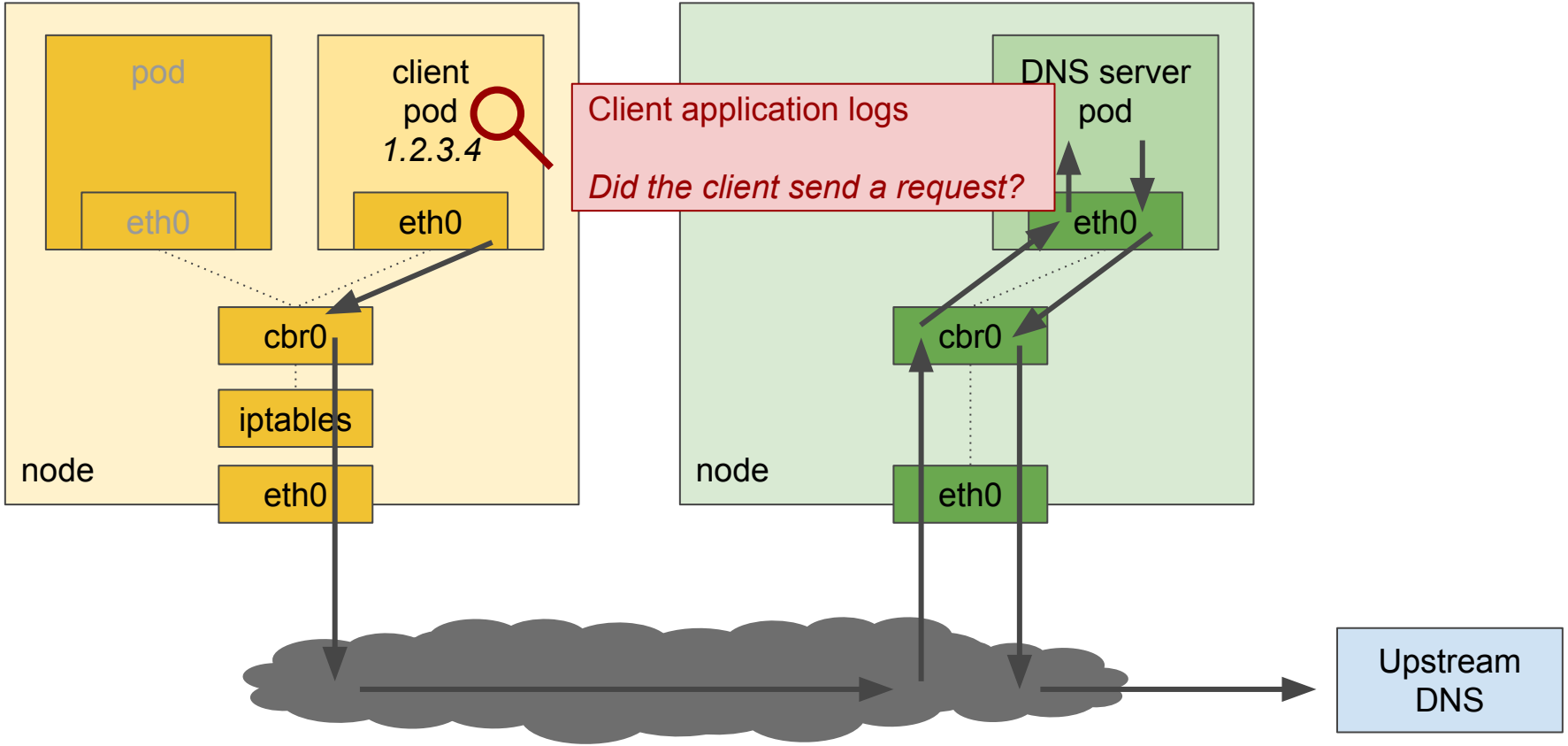
Data Path Inspection

- User Space
- Kernel
- Network Fabric

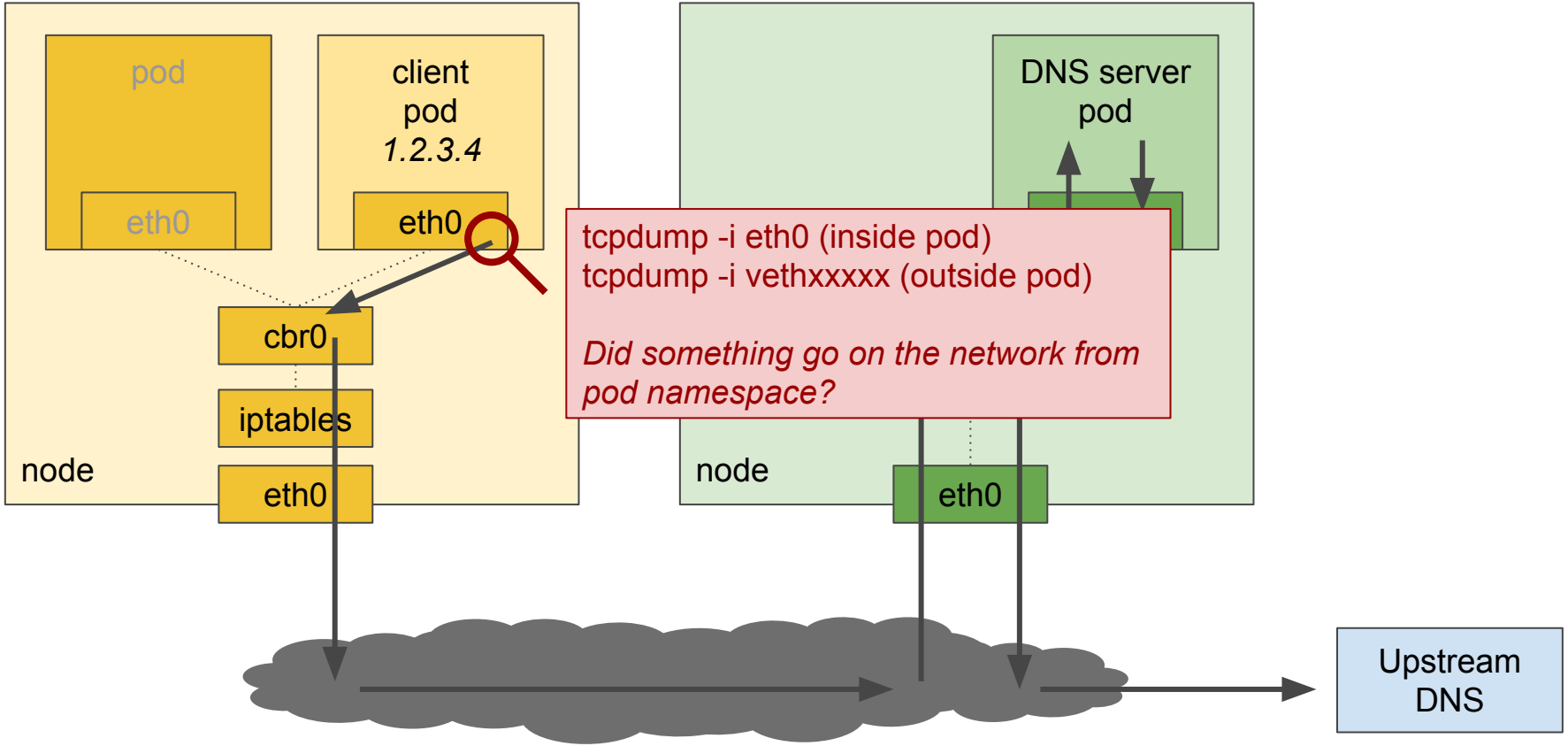
Example: Customer complains that DNS queries are timing out



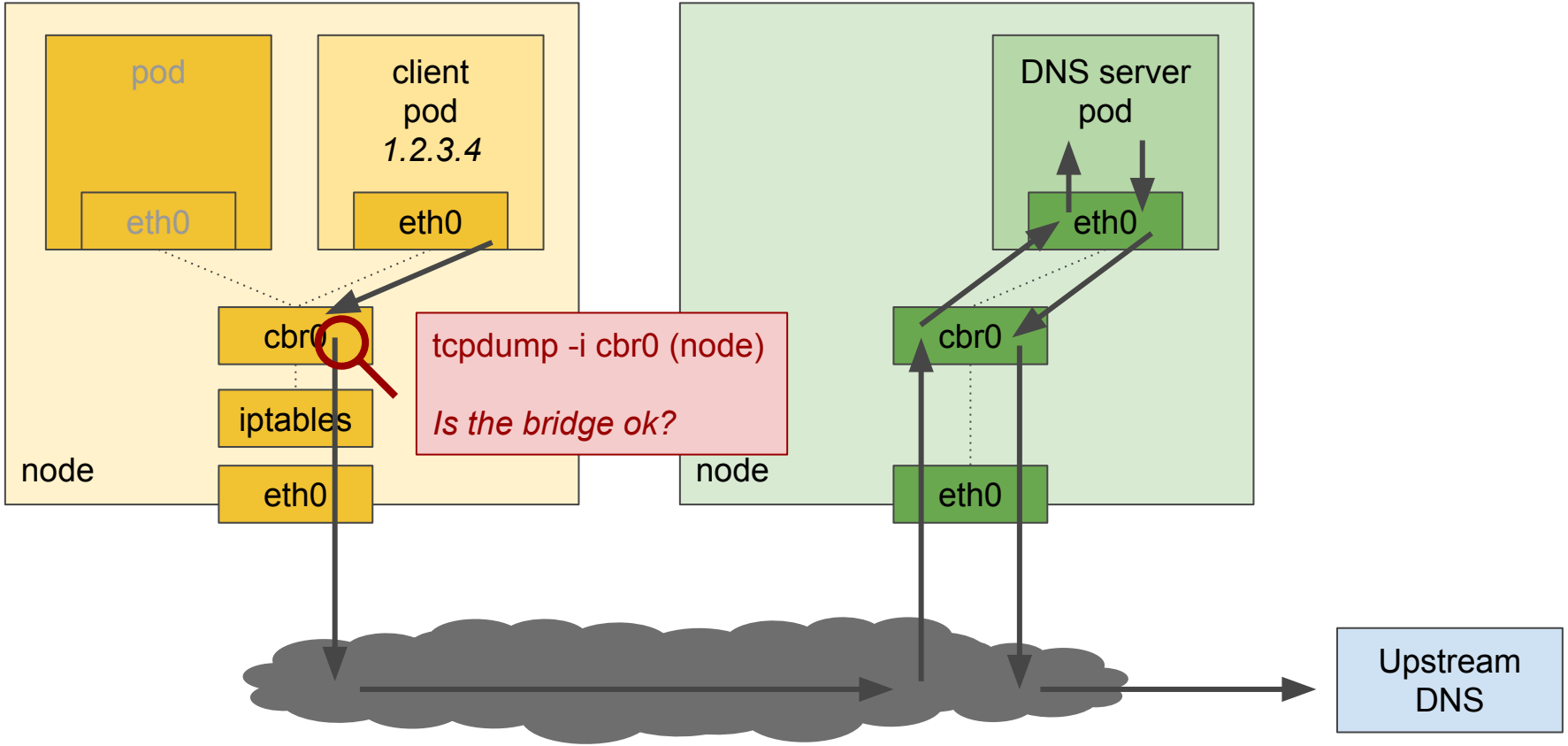
Example: Customer complains that DNS queries are timing out



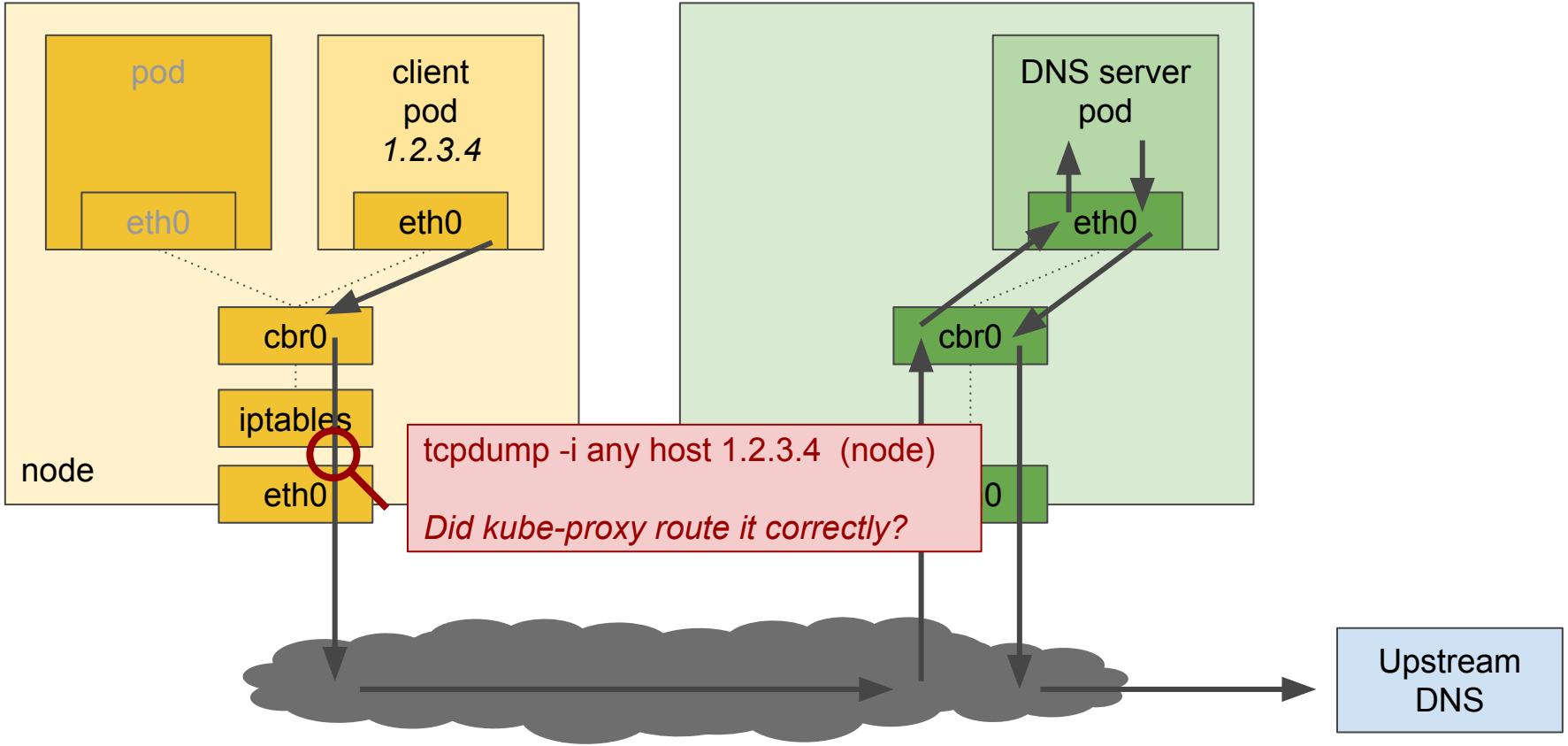
Example: Customer complains that DNS queries are timing out



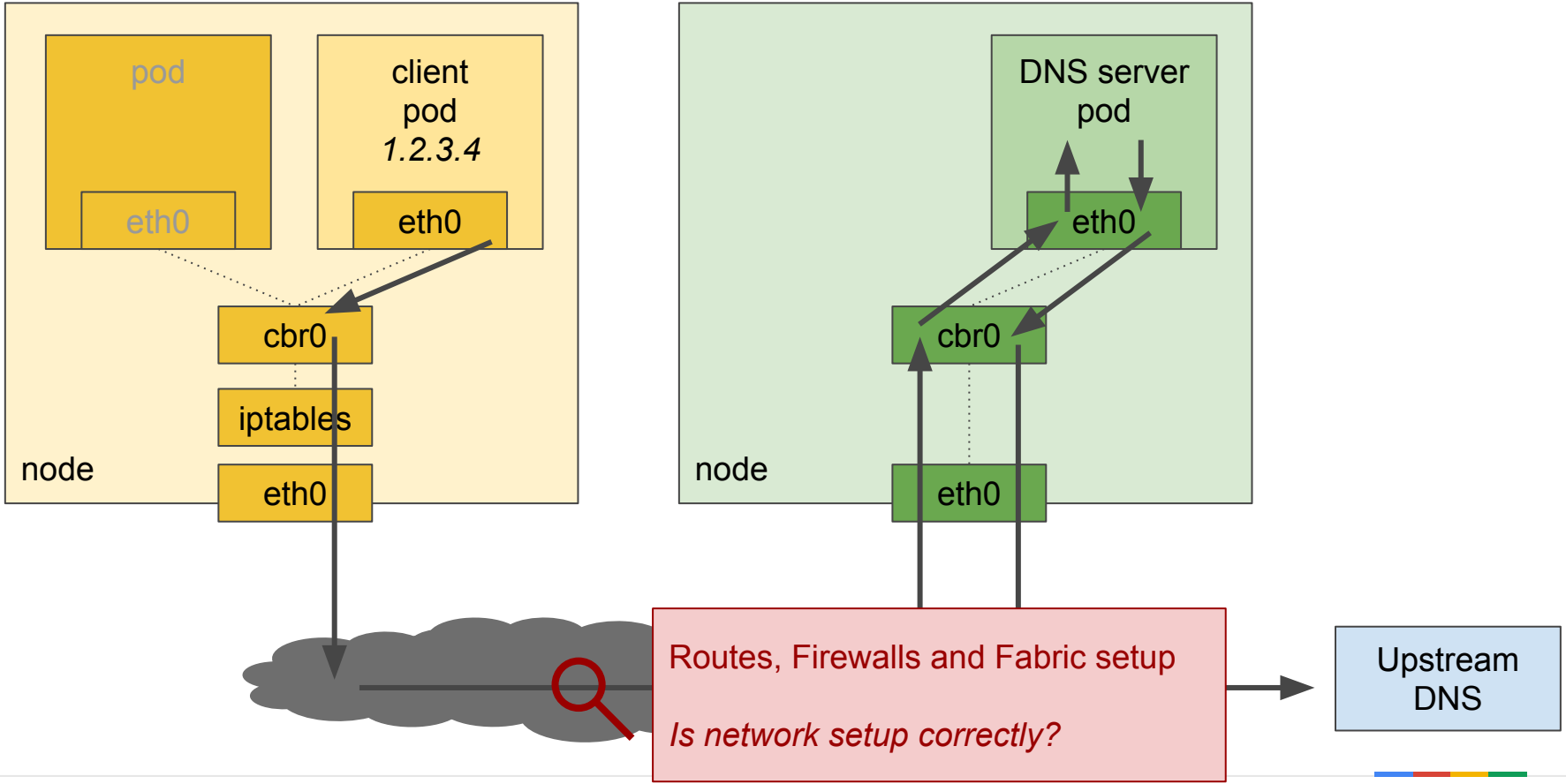
Example: Customer complains that DNS queries are timing out



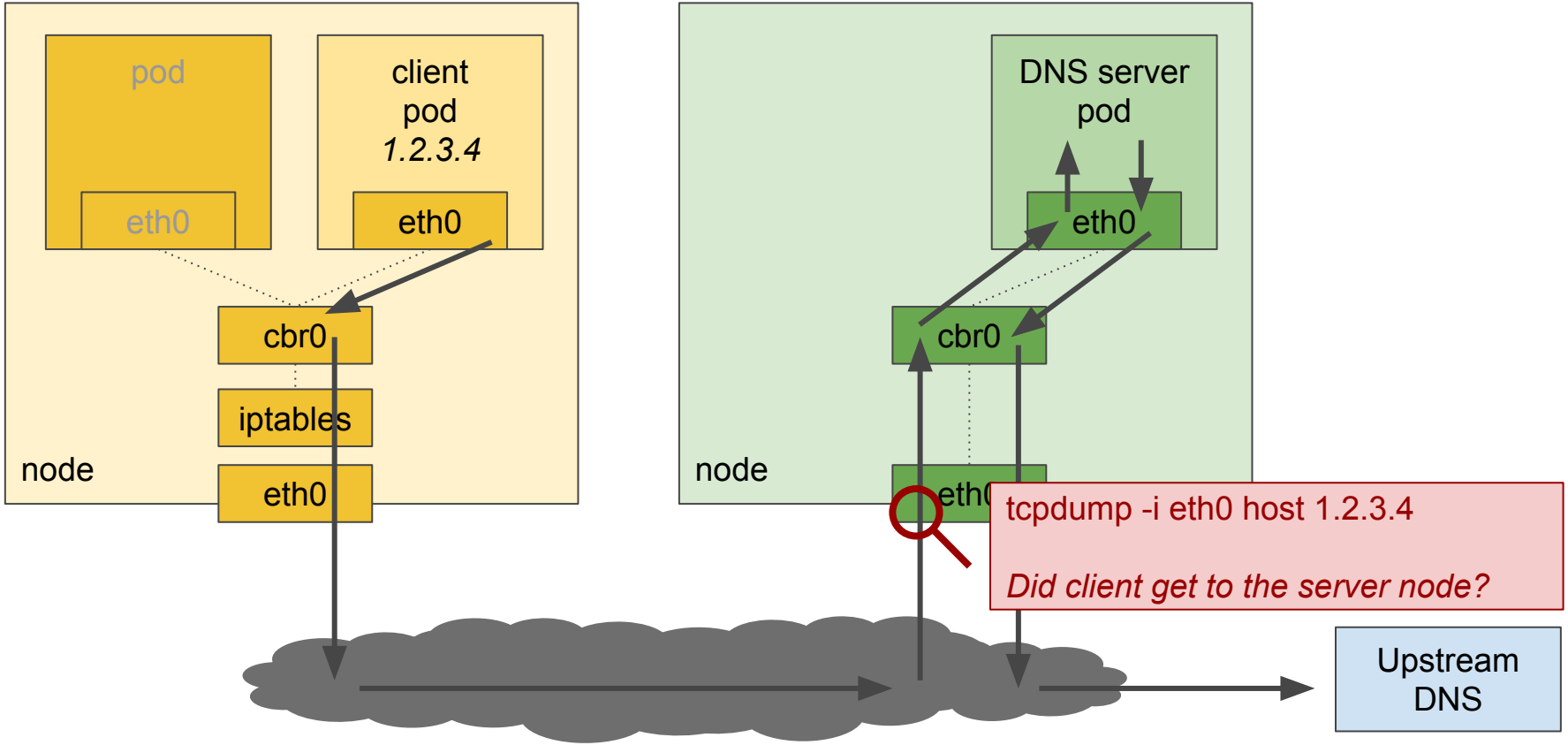
Example: Customer complains that DNS queries are timing out



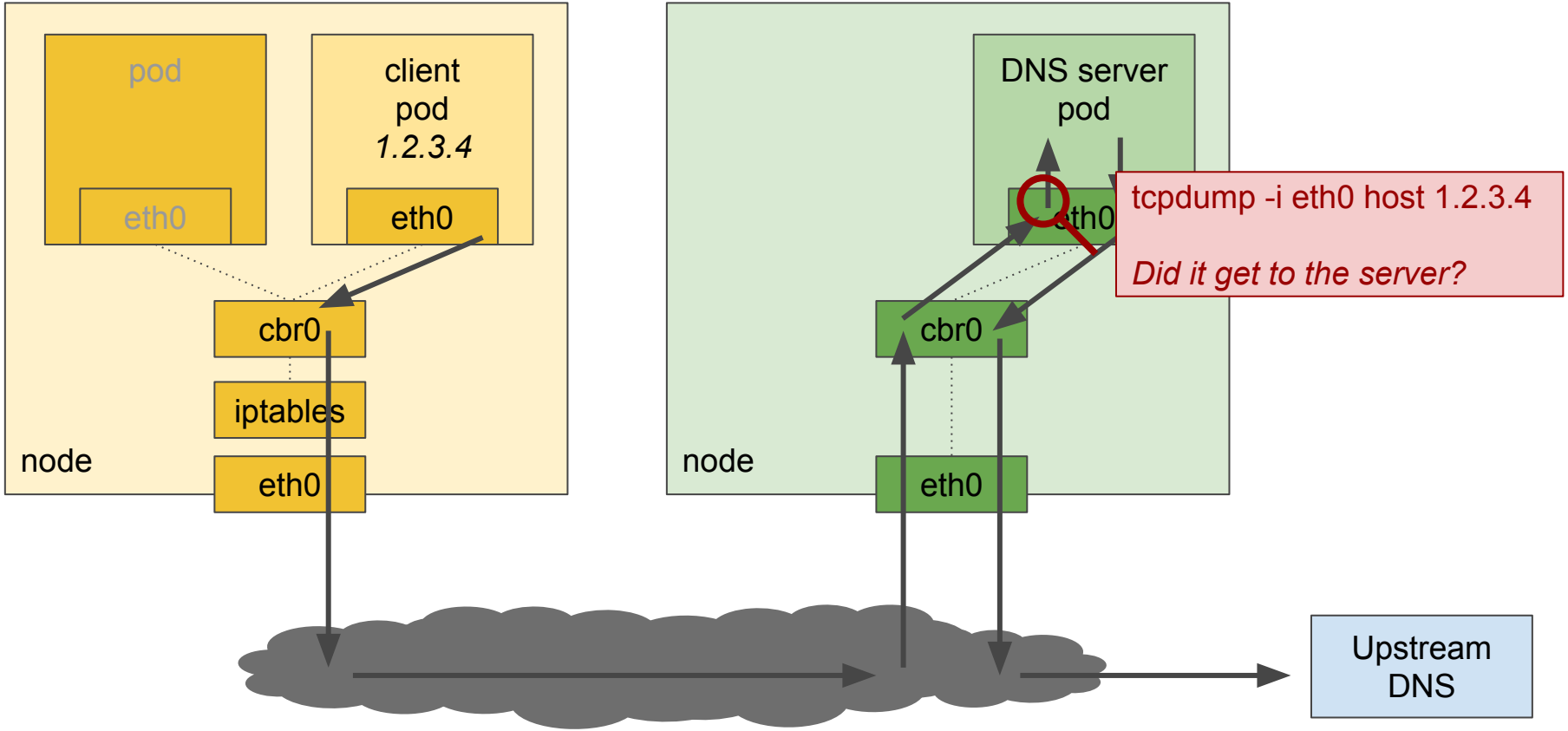
Example: Customer complains that DNS queries are timing out



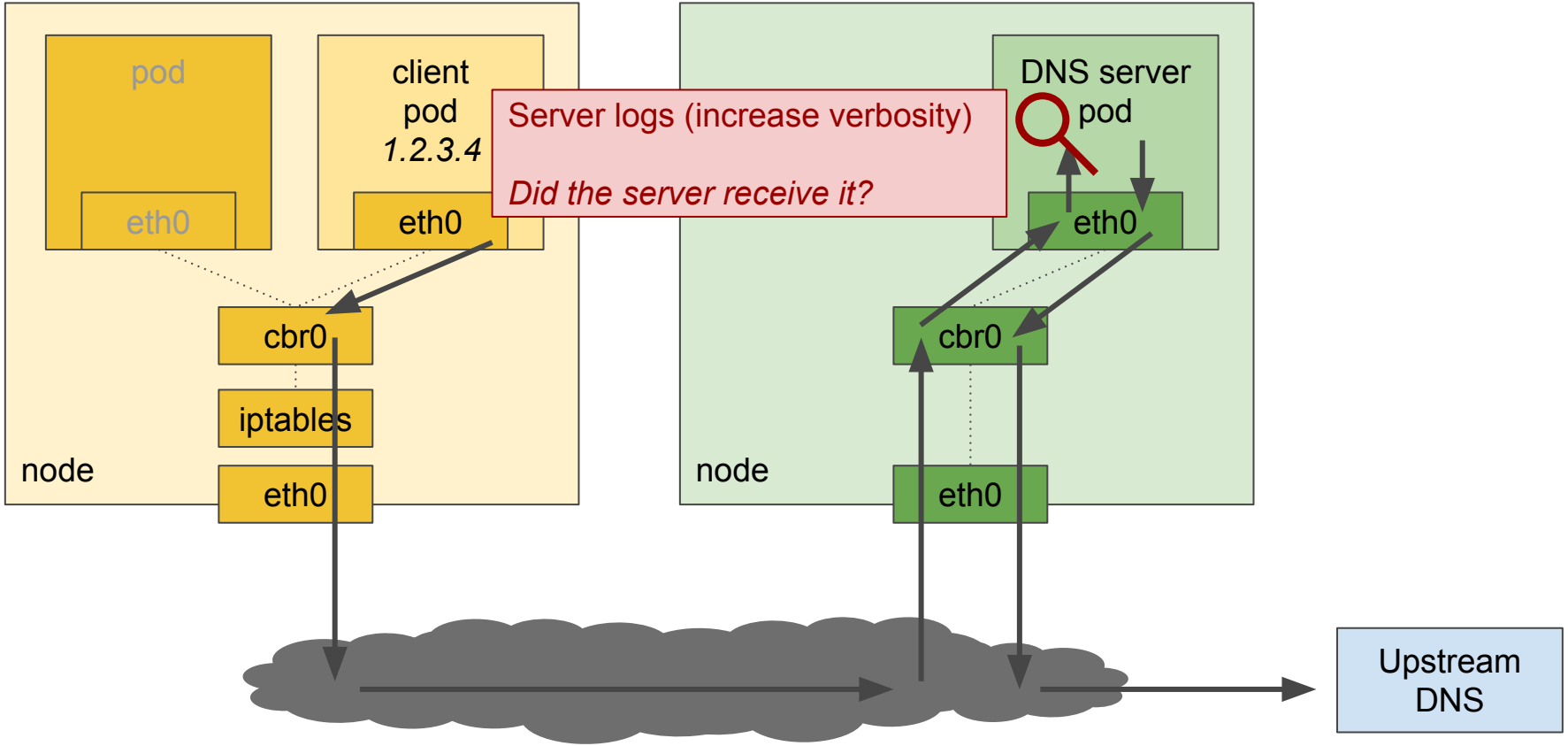
Example: Customer complains that DNS queries are timing out



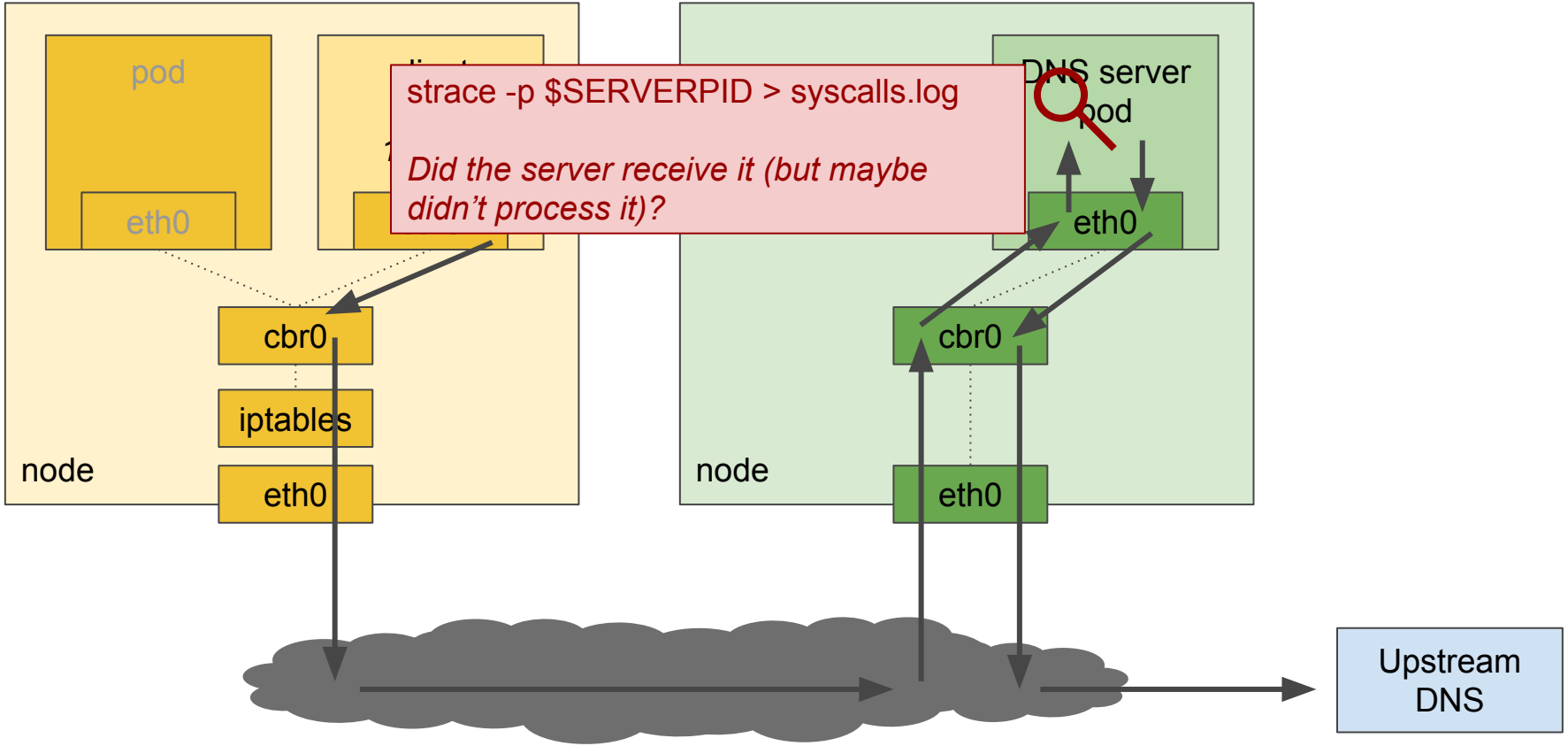
Example: Customer complains that DNS queries are timing out



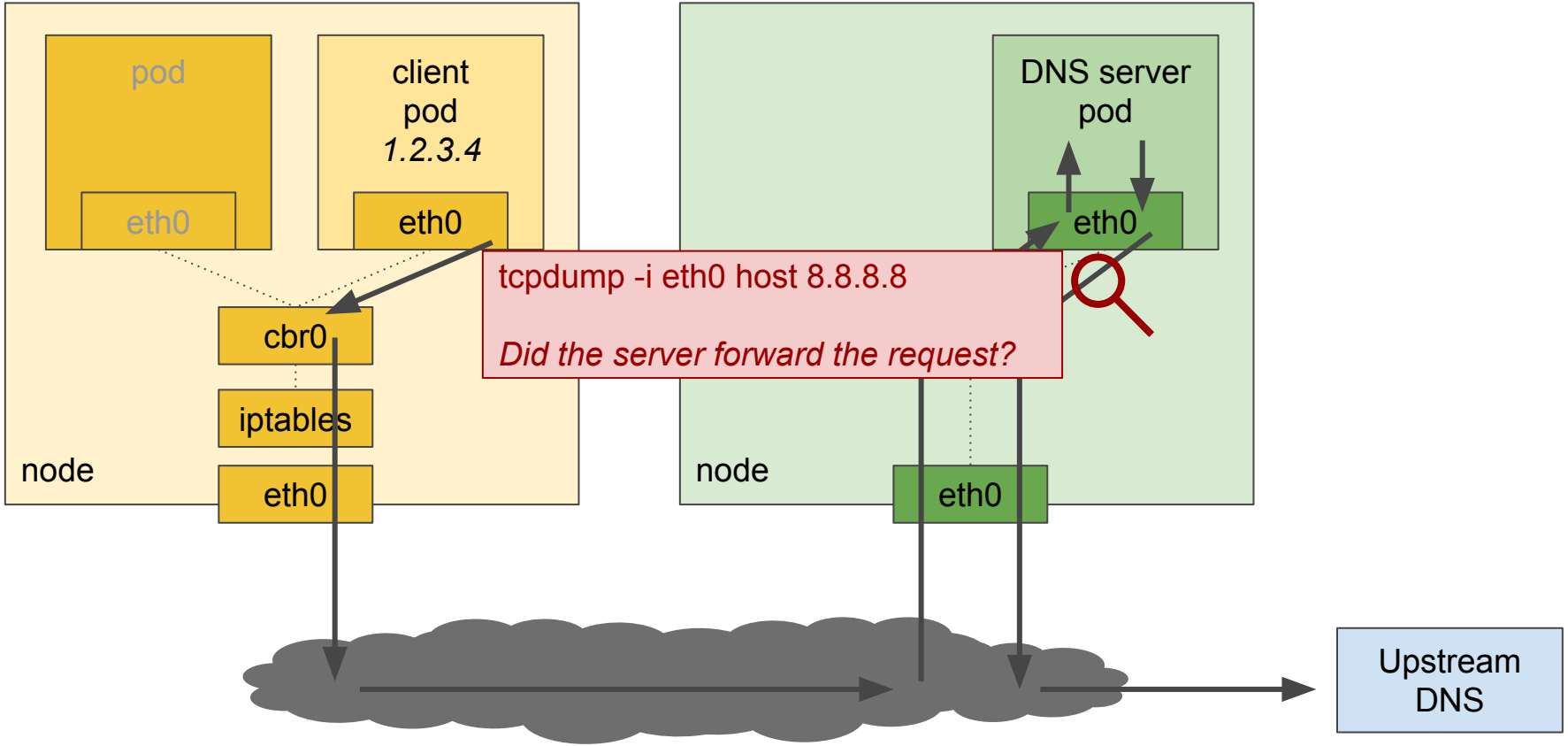
Example: Customer complains that DNS queries are timing out



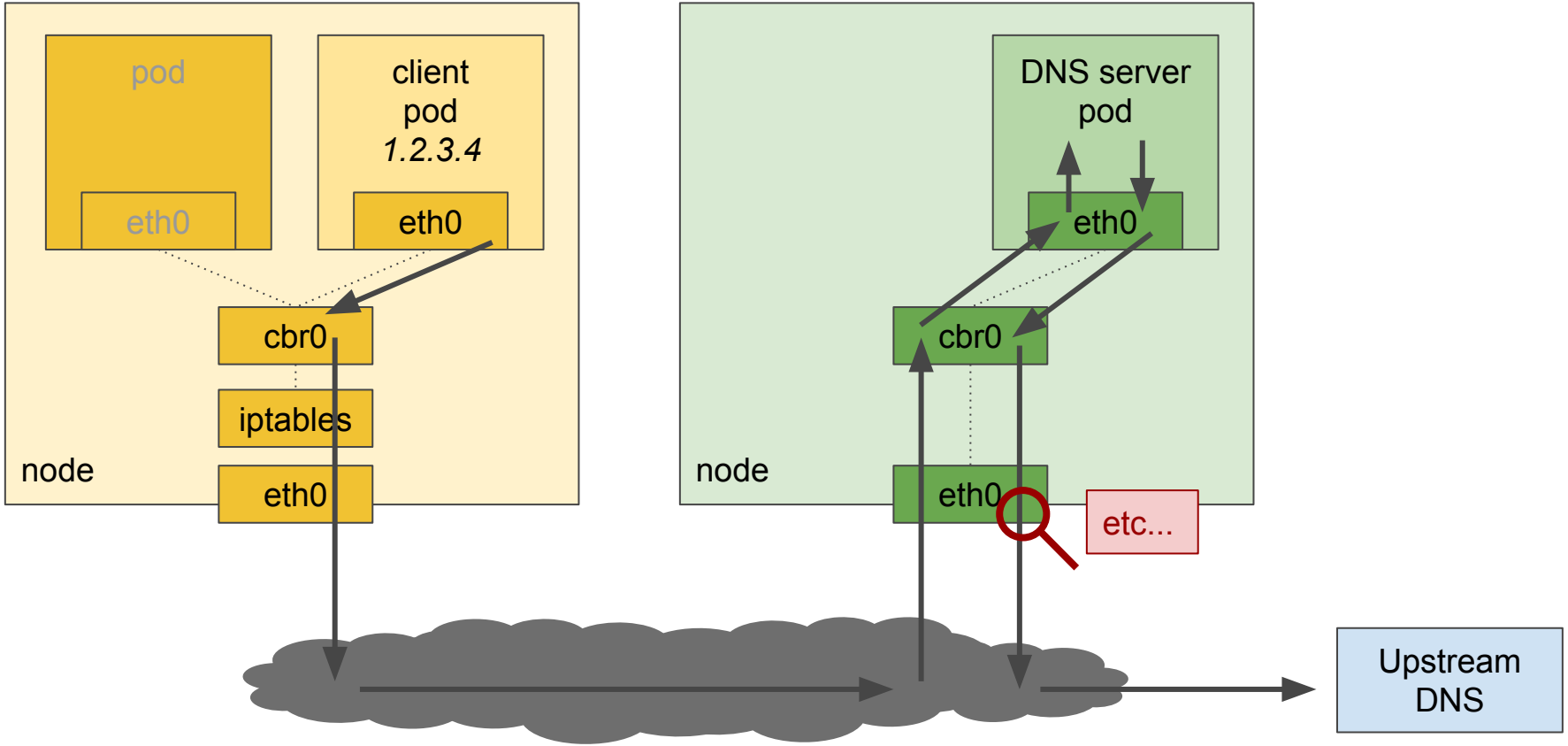
Example: Customer complains that DNS queries are timing out



Example: Customer complains that DNS queries are timing out



Example: Customer complains that DNS queries are timing out



Conclusion

Networking is hard

BUT

K8s Networking is not “**Magic**”

Thank You!

Backup Slides

TOC

- Blackhole:
  - UDP + Conntrack+ IPtables natting == bad idea in linux
    - Kernel sysctl dependency (networkd)
- Wormhole:
  - Loadbalancer vs. hostport
    - Iptables conflict
- Actionable:
  - Iptables
    - Pain point: no iptables history
  - Conntrack
  - Tcpdump (kube-dns troubleshooting)
    - Pain point: only gets end result
    - kube-dns troubleshooting)
- MISC(may not):
  - When SNAT happens:
    - Hairpin
    - Double hop (LB, Nodeport)
    - MASQ external