Mesos Networking

Christos Kozyrakis, Spike Curtis

Kapil Arya, Connor Doyle, Niklas Nielsen, Tarak Parekh, Alex Pollitt

The State of Mesos Networking

Containers share the slave's IP address

Containers can use any port on the slave

Service discovery using per-slave proxies

- localhost:8888 on any slave redirects to a specific service

This was OK Initially

For clusters where

- a single framework manages all services
- there are only a few, long-running services
- there is a single version of each service

But it's Problematic Now

For clusters where

- services are launched by tens of frameworks
- there are thousands of services with high churn
- multiple version of each service

  prod/test/dev, US/AMEA/Asia, …

Problem #1: Port Conflicts

If two apps want to use same port on a slave, one fails to start

Alternative: port isolator enforces non-overlapping port ranges

→ service discovery problem for the app that does not get standard port

Alternative: bridged networking

→ service discovery problem for the app behind the bridge

Problem #2: Service Discovery

How do multiple frameworks manage proxy settings?

How do clients know which version of a service is at each port?

Do we update the proxies in 10K slaves every time a service starts?

Problem #3: No Isolation

How do we stop a test app from connecting with a prod app?

How we isolate different users, services, or divisions?

How do we stop DoS attacks within the cluster?

This makes no sense…

Mesos Networking Redux

Per-container IP addresses

- Routable within and, if needed, outside the cluster
- No port conflicts

DNS-based service discovery

- Discovery using hostnames (A & SRV records, HTTP interface)

Network isolation

- Based on coarse-grain or fine-grain security policies

Implementation

One feature set, many pluggable implementations

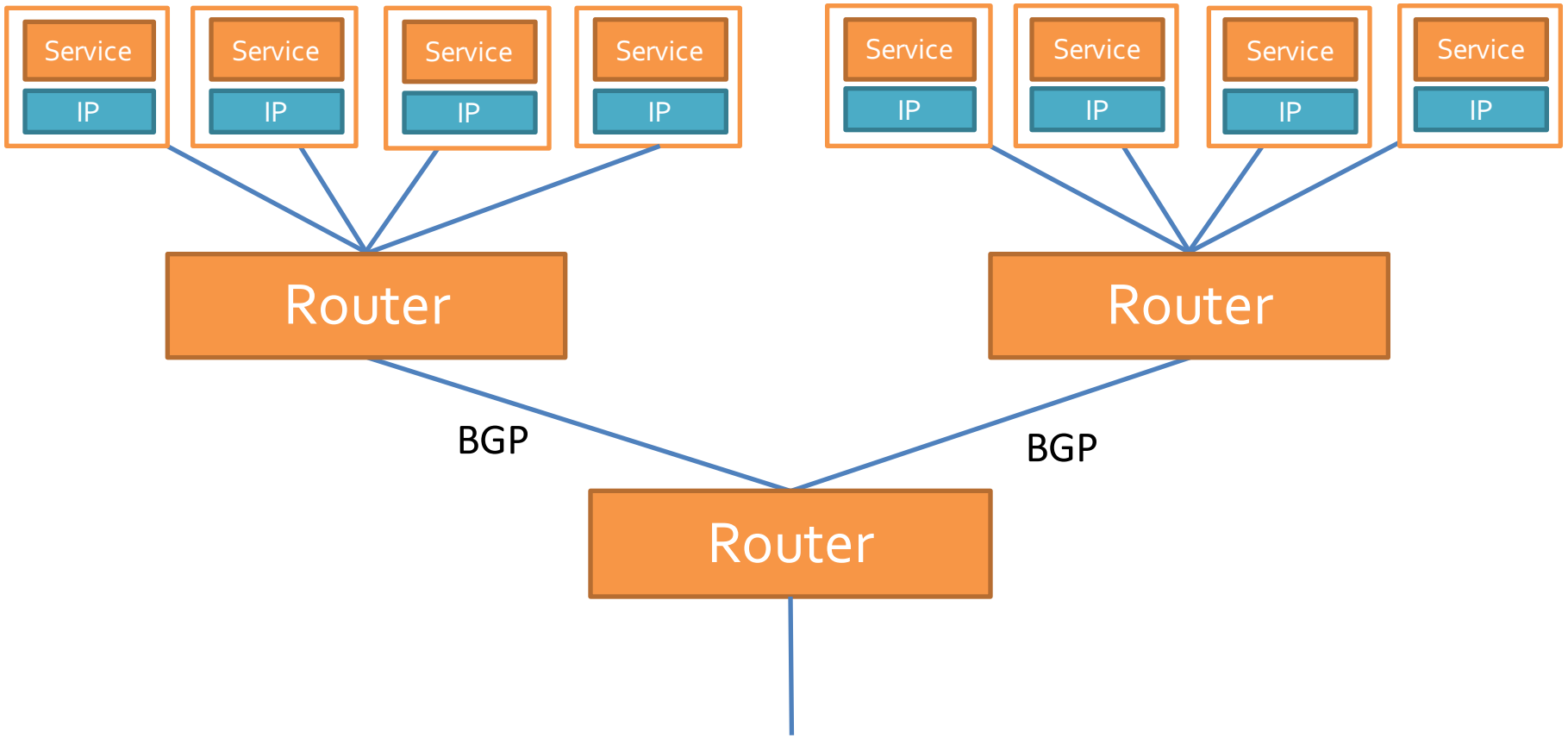
- Different network virtualization technologies (L2 or L3)
- Different IP address management schemes
- Different DNS servers

First implementation based on Project Calico

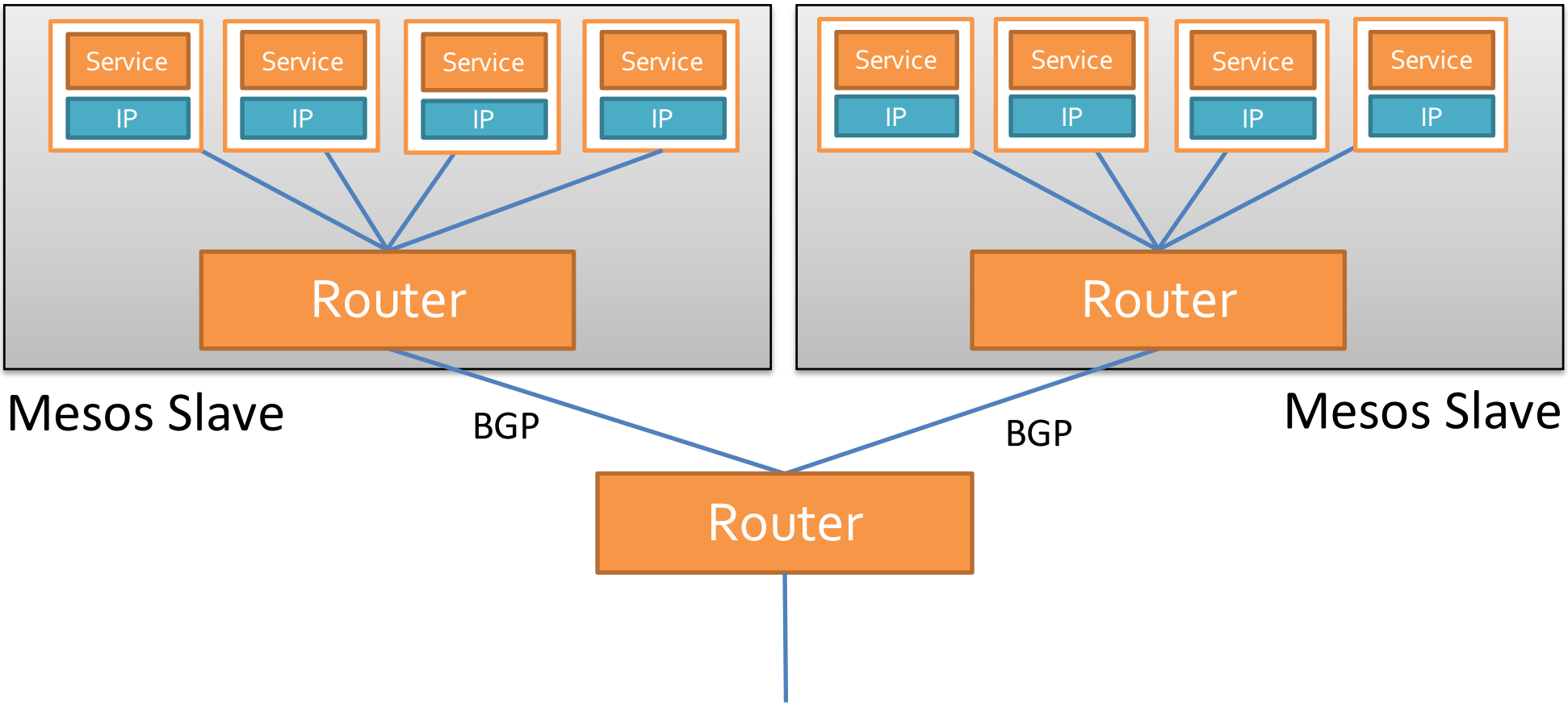
- L3-based network virtualization & isolation
- Simple, scalable, open-source


PROJECT
CALICO

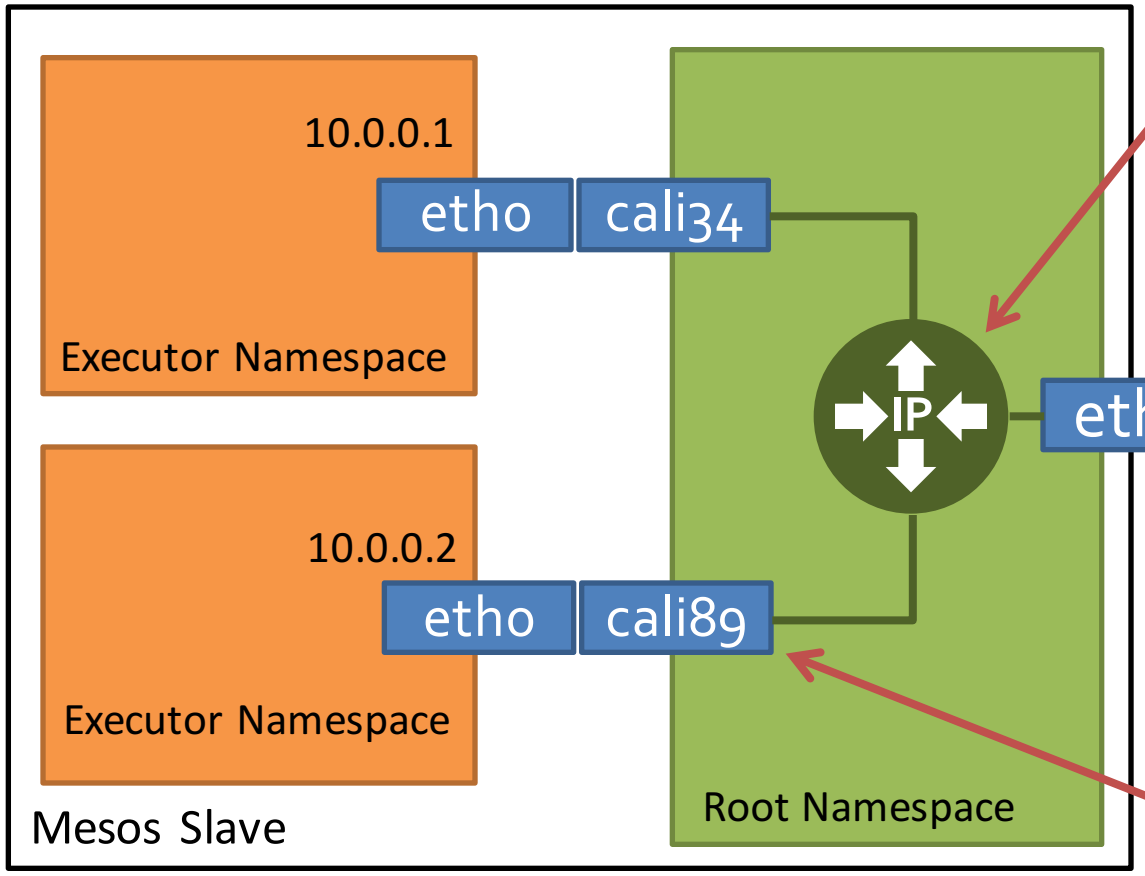
Build the DC network like the Internet



Build the DC network like the Internet



Calico Data Plane



Linux Kernel Routing

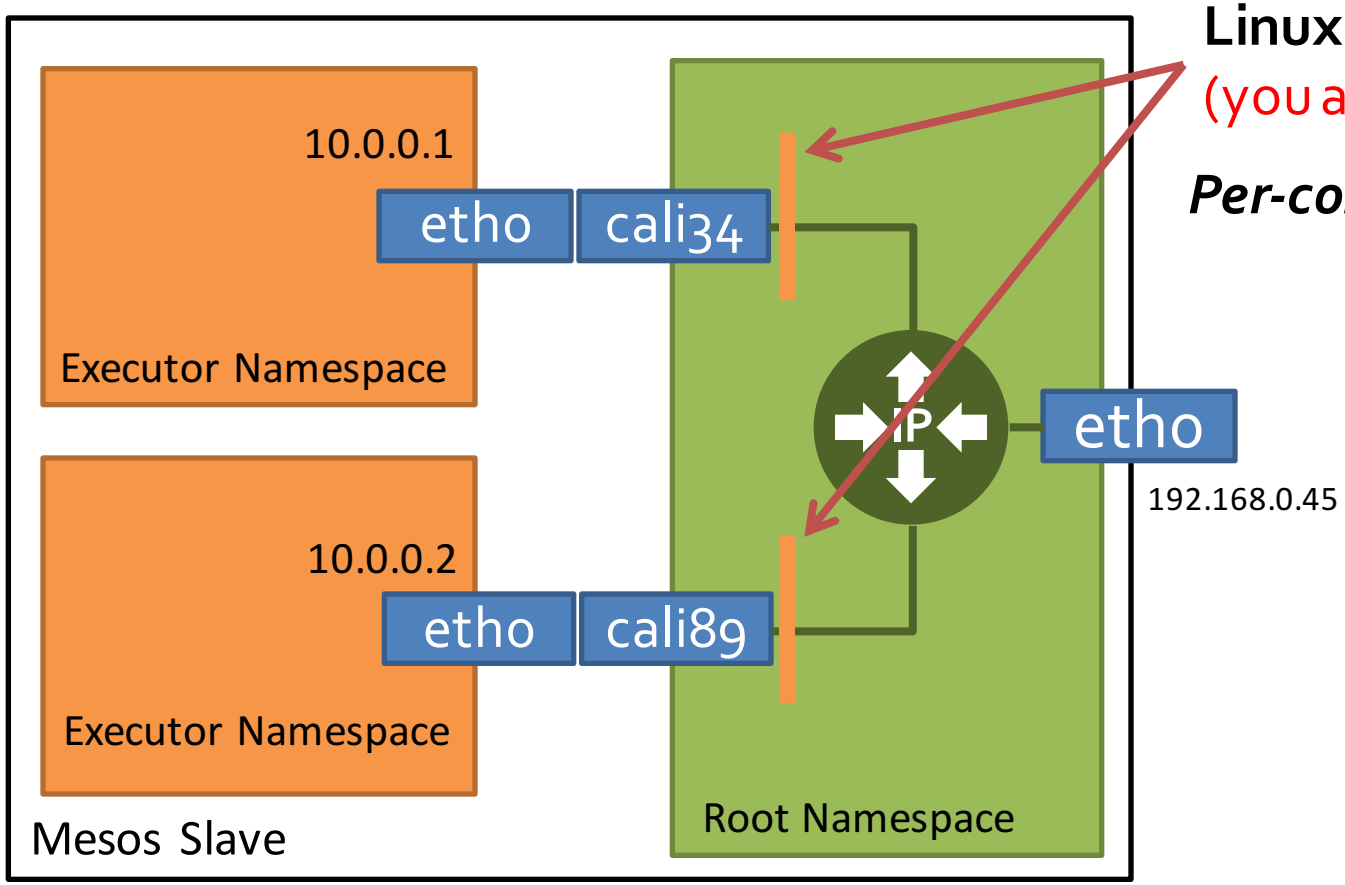
(you already have this!)

```
default via 192.168.0.1 dev eth0
192.168.0.0/24 dev eth0  src 10.0.2.15
10.0.0.1/32 dev cali34 scope global
10.0.0.2/32 dev cali89 scope global
10.0.1.40/32 via 192.168.0.29 dev eth0
10.0.2.53/32 via 192.168.0.131 dev eth0
```

Containers on other slaves

veth pair (kernel version 2.6.24+)

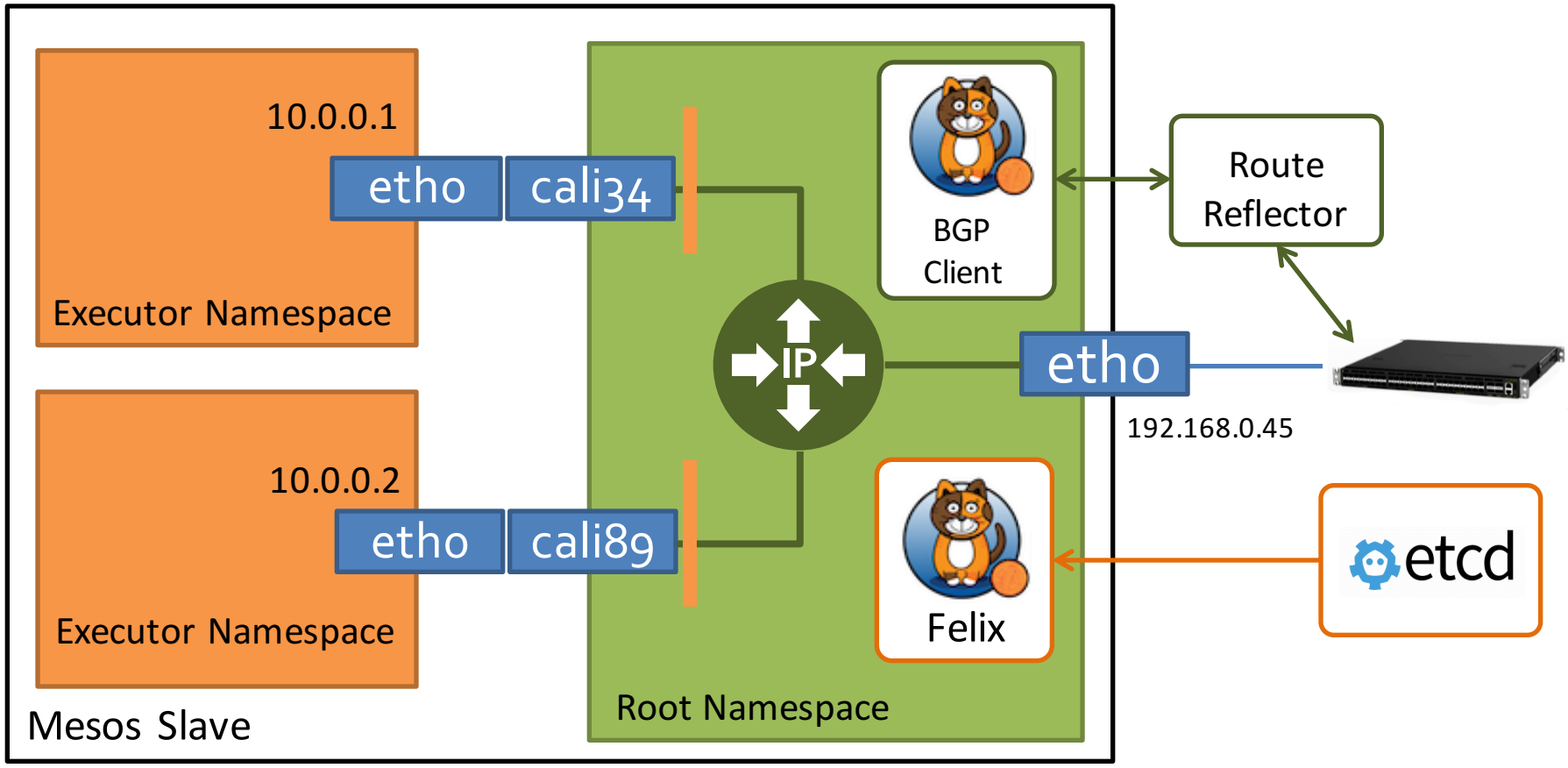
Calico Data Plane



Linux Kernel Filtering (iptables)
(you already have this!)

Per-container distributed firewall

Calico Control Plane



Mesos – Calico Integration

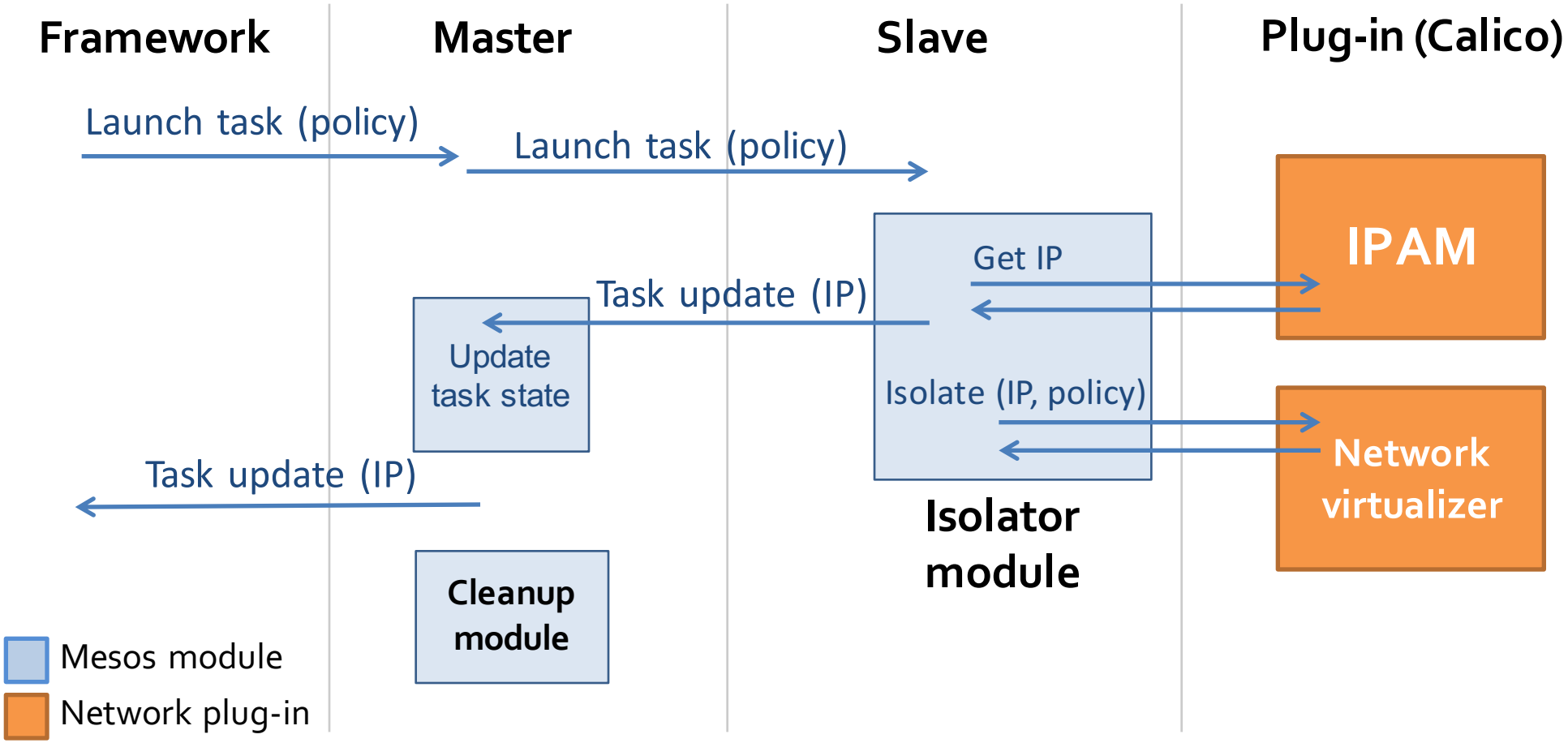
Networking isolator

Calico IP address management – IPAM (plug-in)

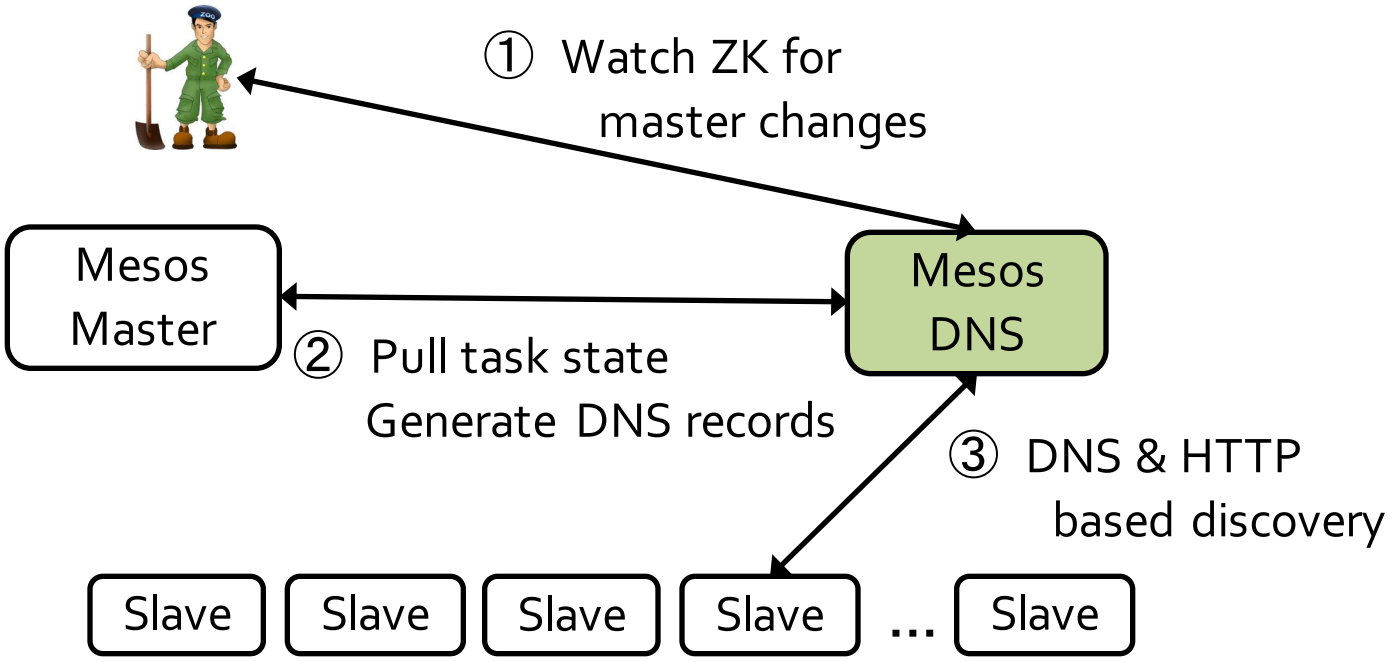
Calico network virtualizer (plug-in)

Master cleanup module

Networking Workflow



Mesos-DNS



nginx_prod.marathon.mesos → 10.13.17.95

_nginx_prod._tcp.marathon.mesos →10.13.17.95:8181

Networking Demo

Mesos cluster with 2 slaves

Launching 4 probe tasks

- Each probe listens to port 9000
- Each probe tries to reach all other probes

We want all 4 to launch successfully (no port conflicts)

We want to isolate them into two groups of 2 probes

Networking Demo

Roadmap

Code release (open source)

Integration with Mesosphere DCOS

Interfaces for coarse-grain and fine-grain isolation policies

Other plug-in implementations

Flexible task naming in Mesos-DNS

Network QoS

Summary

Mesos networking features

- Per-container IP addresses
- DNS-based service discovery
- Network isolation

1st implementation using Project Calico

Try it and contribute!

References

https://mesosphere.com/

http://www.projectcalico.org/

https://github.com/mesosphere/net-modules

https://github.com/mesosphere/mesos-dns