BPF as a revolutionary technology for the container landscape

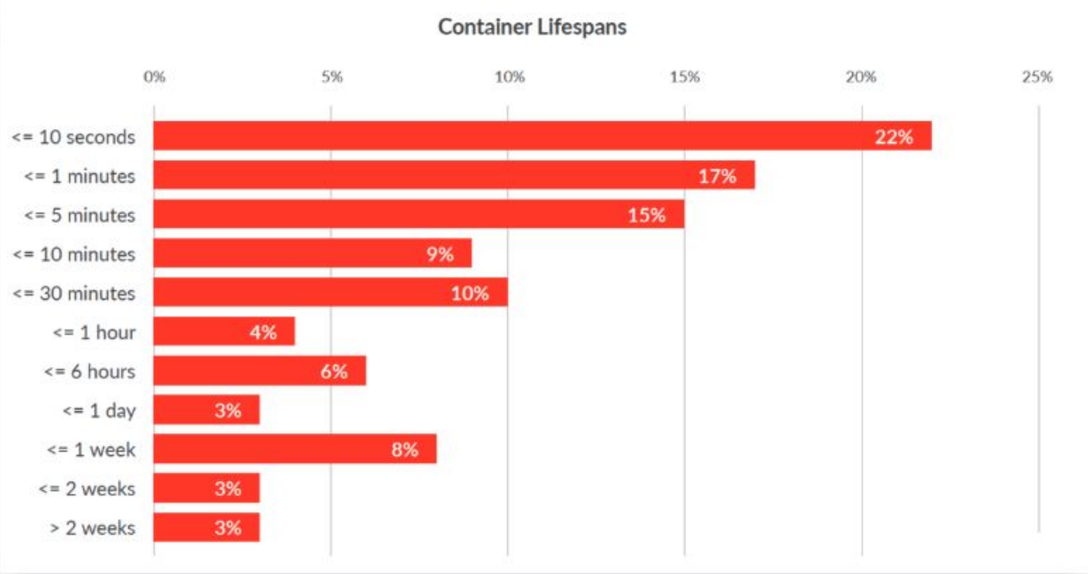
Daniel Borkmann, Cilium.io
FOSDEM'20

eBPF cilium

Landscape: continuously decreasing lifetime

The short life of containers

Comparing container lifespans year over year, we found that the number of containers that are alive for 10 seconds or less has doubled to 22%. In fact, the number of containers that live for 5 minutes or less grew by 2X as well.

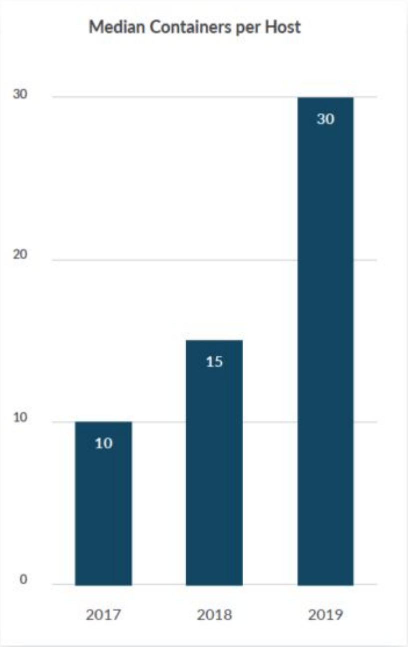


Source: sysdig '19 container usage report

Landscape: continuously increasing density

Containers-per-host density increases 100%

Over the past year, the median number of containers per host doubled to 30, compared to 15 in 2018 and 10 in 2017.

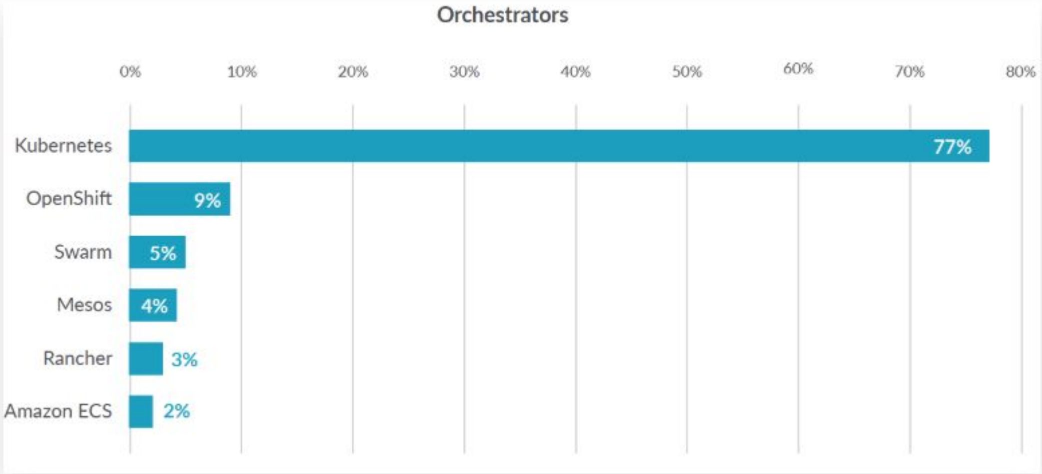


Source: sysdig '19 container usage report

Landscape: Kubernetes as main orchestrator

Container orchestration: Kubernetes dominates

It's no surprise that as the de facto container orchestration tool, Kubernetes takes a whopping 77% share of orchestrators in-use. That number expands to 89% when you add in Red Hat OpenShift and Rancher – both built with Kubernetes. Here's the current breakdown:



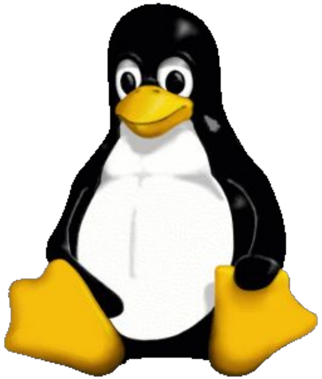
Source: sysdig '19 container usage report

Landscape: Linux kernel as common denominator

Must provide building blocks for ...

- Isolation (namespaces)
- Resource management (cgroups)
- Network connectivity
- Security policies
- […]

… AND must withstand ever increasing scalability needs and high churn frequencies ...



Landscape: Linux kernel as common denominator

… while coping with subsystems and user interfaces originally designed long ago and subject to the “never break user space” paradigm.

Few examples in networking: tc, iptables/netfilter

Both designed for extensibility in general, but within inflexible overall framework for today’s needs.

Processing pipeline becomes part of the API contract.

Complex rules then significantly slow down fast-path.

Source: reddit.com/r/ArchitecturePorn/

Landscape: Linux kernel as common denominator

Given the need to support wide range of kernels, system software often stuck in such framework.

Policy logic then gets deeply baked into codebase, significant effort to rewrite.

Random pick, libnetwork:

```
[....]
        args = []string{
                "!", "-i", bridgeName,
                "-o", bridgeName,
                "-p", proto,
                "-d", destAddr,
                "--dport", strconv.Itoa(destPort),
                "-j", "ACCEPT",
        }
        if err := ProgramRule(Filter, c.Name, action, args); err != nil {
                return err
        }
[...]
```

Source: xkcd.com/1421/

Landscape: Linux kernel as common denominator

… but also Kubernetes itself relies a lot on iptables/netfilter for its Service implementation.

Issues in face of container scalability needs:

- Low and unpredictable packet latency
- Slow update time
- Reliability issues
- Inflexibility

https://github.com/kubernetes/community/blob/master/sig-scalability/blogs/k8s-services-scalability-issues.md (Jan 2020)

Performance

```
# perf top -a -e cycles:k

PerfTop:   16326 irqs/sec  (all, 4 CPUs)
-------------------------------------------------------------------------------
     8.79%  [kernel]            [k] native_sched_clock
     4.99%  [ip_tables]         [k] ipt_do_table
     3.09%  [e1000e]            [k] e1000_irq_enable
     2.51%  [nf_conntrack]      [k] __nf_conntrack_find_get
     2.03%  [kernel]            [k] fib_table_lookup
     1.98%  [kernel]            [k] sched_clock_cpu
     1.75%  [nf_conntrack]      [k] tcp_packet
     1.65%  [nf_conntrack]      [k] nf_conntrack_tuple_taken
     [...]
```

Reliability


DNS intermittent delays of 5s #56903

Closed mikksoone opened this issue on Dec 6, 2017 · 230 comments

mikksoone commented on Dec 6, 2017 • edited

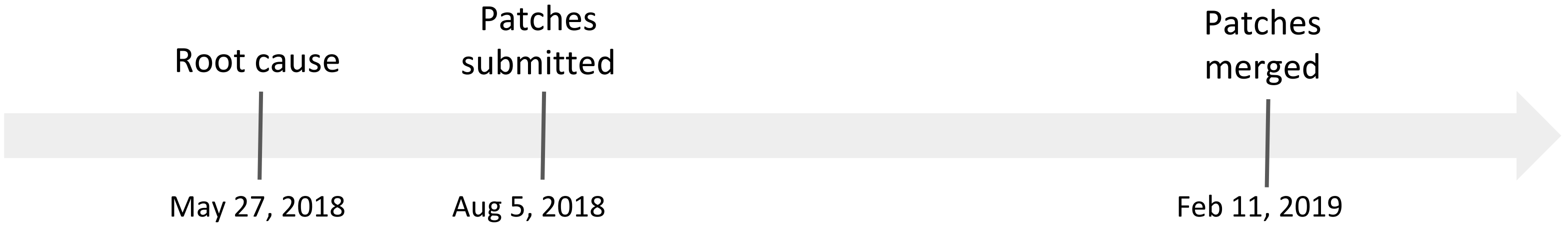
Is this a BUG REPORT or FEATURE REQUEST?:
/kind bug

What happened:
DNS lookup is sometimes taking 5 seconds.

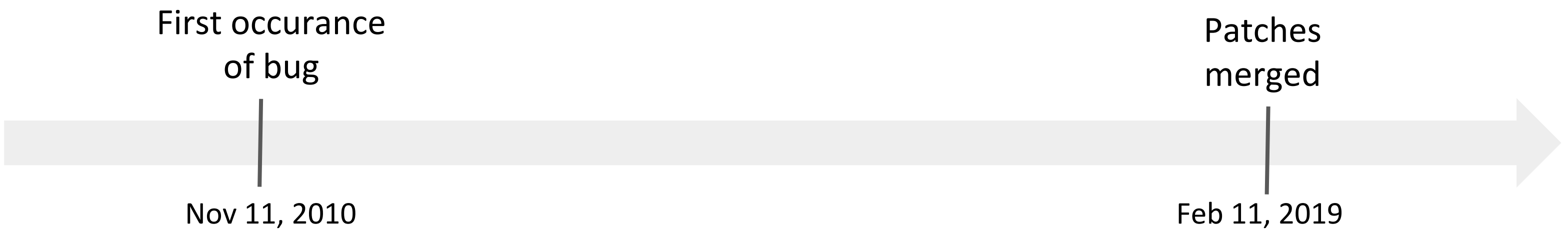
What you expected to happen:
No delays in DNS.

Assignees
No one assigned

Labels
area/dns
kind/bug
sig/network



Reliability



Compatibility issues along the way

kube-proxy currently incompatible with `iptables >= 1.8` #71305

Open **drags** opened this issue on Nov 21, 2018 · 75 comments · May be fixed by #82966 or #84420

Ensure iptables tooling does not use the nftables backend

In Linux, nftables is available as a modern replacement for the kernel's iptables subsystem. The `iptables` tooling can act as a compatibility layer, behaving like iptables but actually configuring nftables. This nftables backend is not compatible with the current kubeadm packages: it causes duplicated firewall rules and breaks `kube-proxy`.

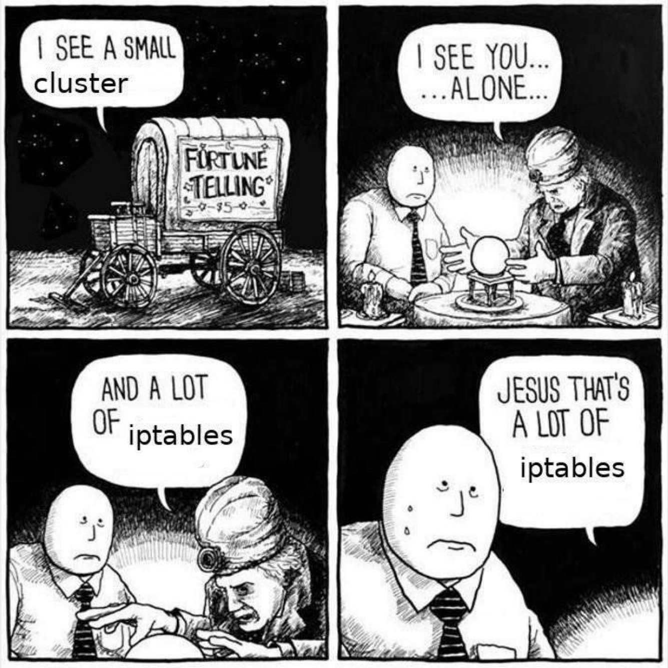
If your system's `iptables` tooling uses the nftables backend, you will need to switch the `iptables` tooling to 'legacy' mode to avoid these problems. This is the case on at least Debian 10 (Buster), Ubuntu 19.04, Fedora 29 and newer releases of these distributions by default. RHEL 8 does not support switching to legacy mode, and is therefore incompatible with current kubeadm packages.

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/ (Jan 2020)

Debuggability

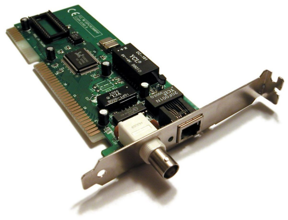
```
# iptables-save -c

*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
[1:10] -A FORWARD -i eth0 -s 172.17.0.0/16 -j DROP
```

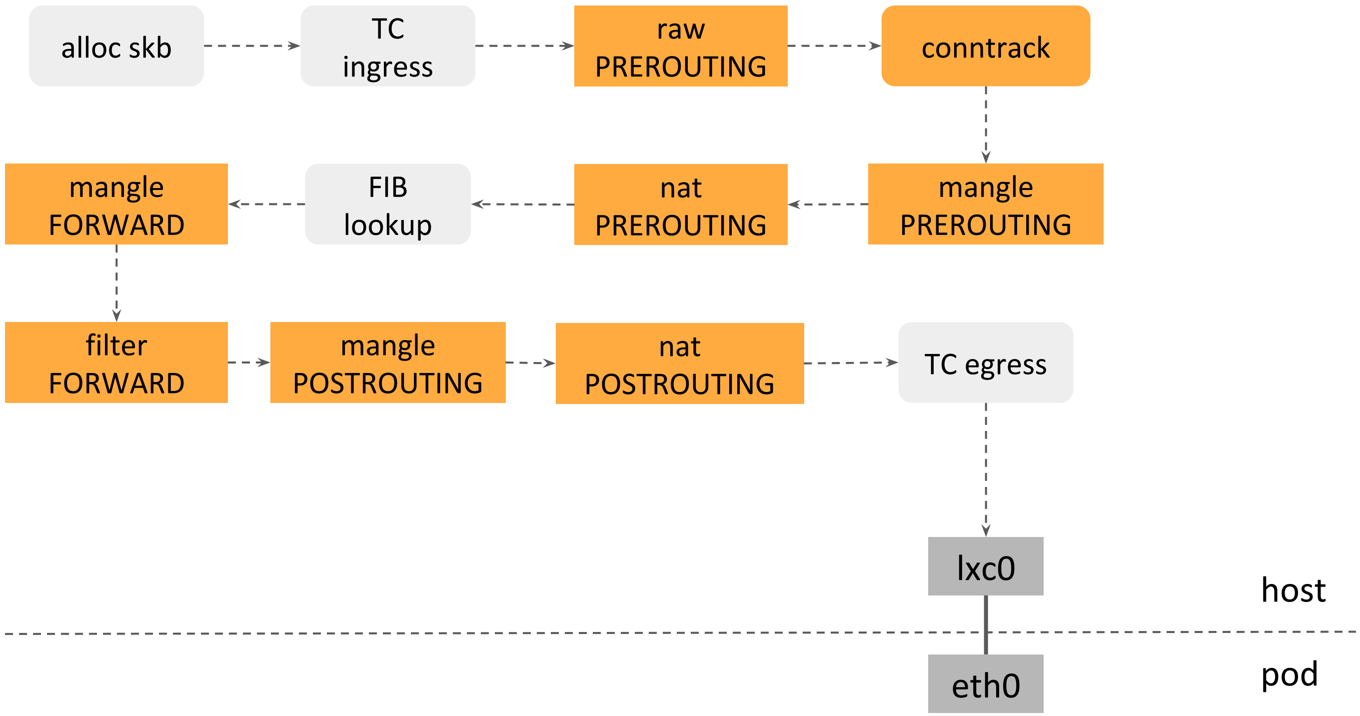
Debuggability



Source: reddit.com/r/networkingmemes/comments/8u7jyz/container_networking/

Packet flow





Source: commons.wikimedia.org/w/index.php?curid=122201

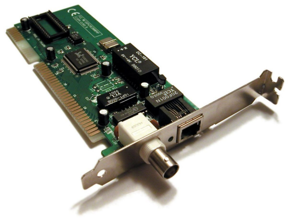
ClusterIP with iptables

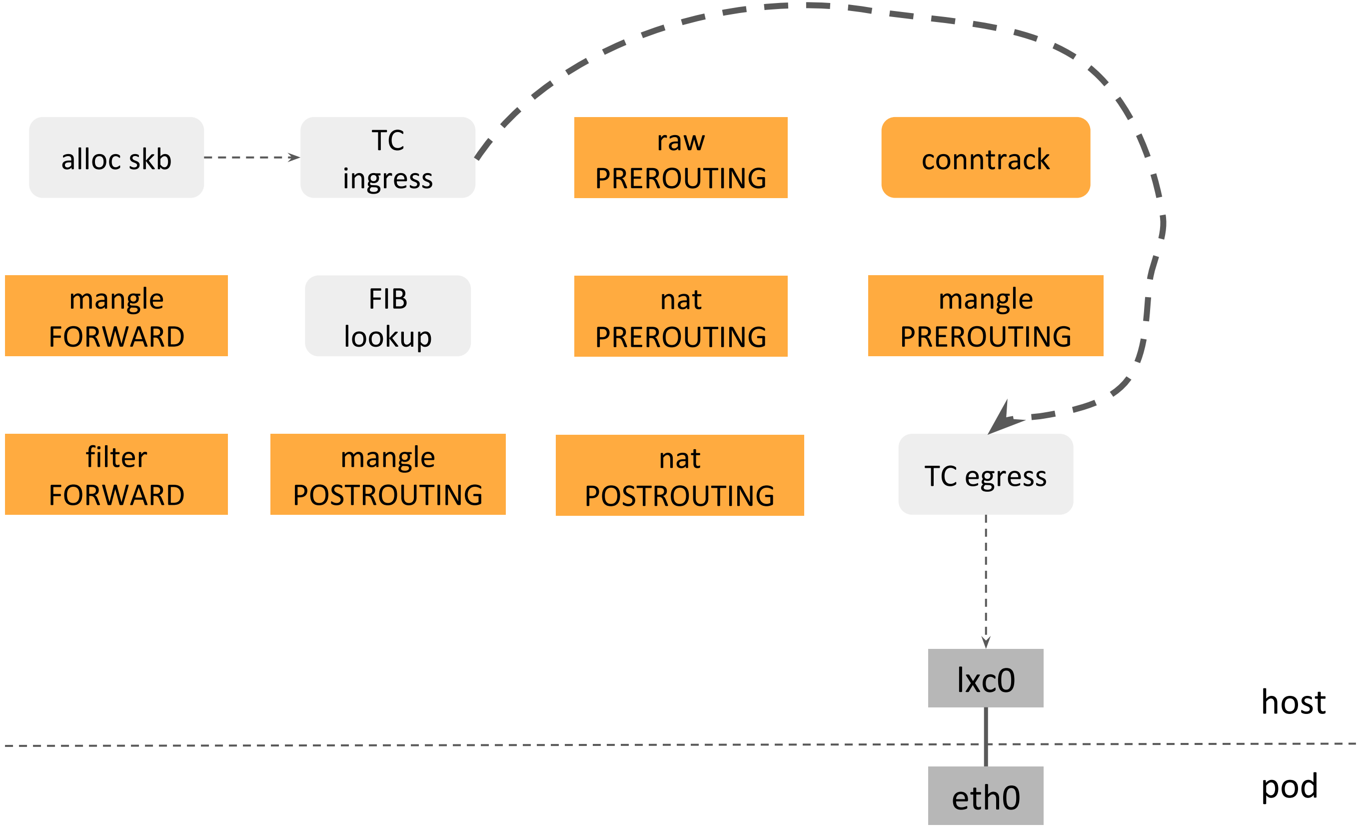
```
$ kubectl get svc nginx
NAME   TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)
nginx  ClusterIP   3.3.3.3      <none>        80/TCP

$ kubectl get endpoints nginx
NAME    ENDPOINTS
nginx   1.1.1.1:80, 1.1.2.2:80
```

```
-t nat -A PREROUTING -m conntrack --ctstate NEW -j KUBE-SERVICES

-A KUBE-SERVICES ! -s 1.1.0.0/16 -d 3.3.3.3/32 -p tcp -m tcp --dport 80 -j KUBE-MARK-MASQ
-A KUBE-SERVICES -d 3.3.3.3/32 -p tcp -m tcp --dport 80 -j KUBE-SVC-NGINX

-A KUBE-SVC-NGINX -m statistic --mode random --probability 0.50 -j KUBE-SEP-NGINX1
-A KUBE-SVC-NGINX -j KUBE-SEP-NGINX2

-A KUBE-SEP-NGINX1 -s 1.1.1.1/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-NGINX1 -p tcp -m tcp -j DNAT --to-destination 1.1.1.1:80
-A KUBE-SEP-NGINX2 -s 1.1.2.2/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-NGINX2 -p tcp -m tcp -j DNAT --to-destination 1.1.2.2:80
```

nat
PREROUTING

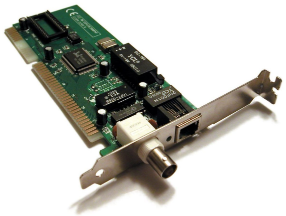
Packet flow

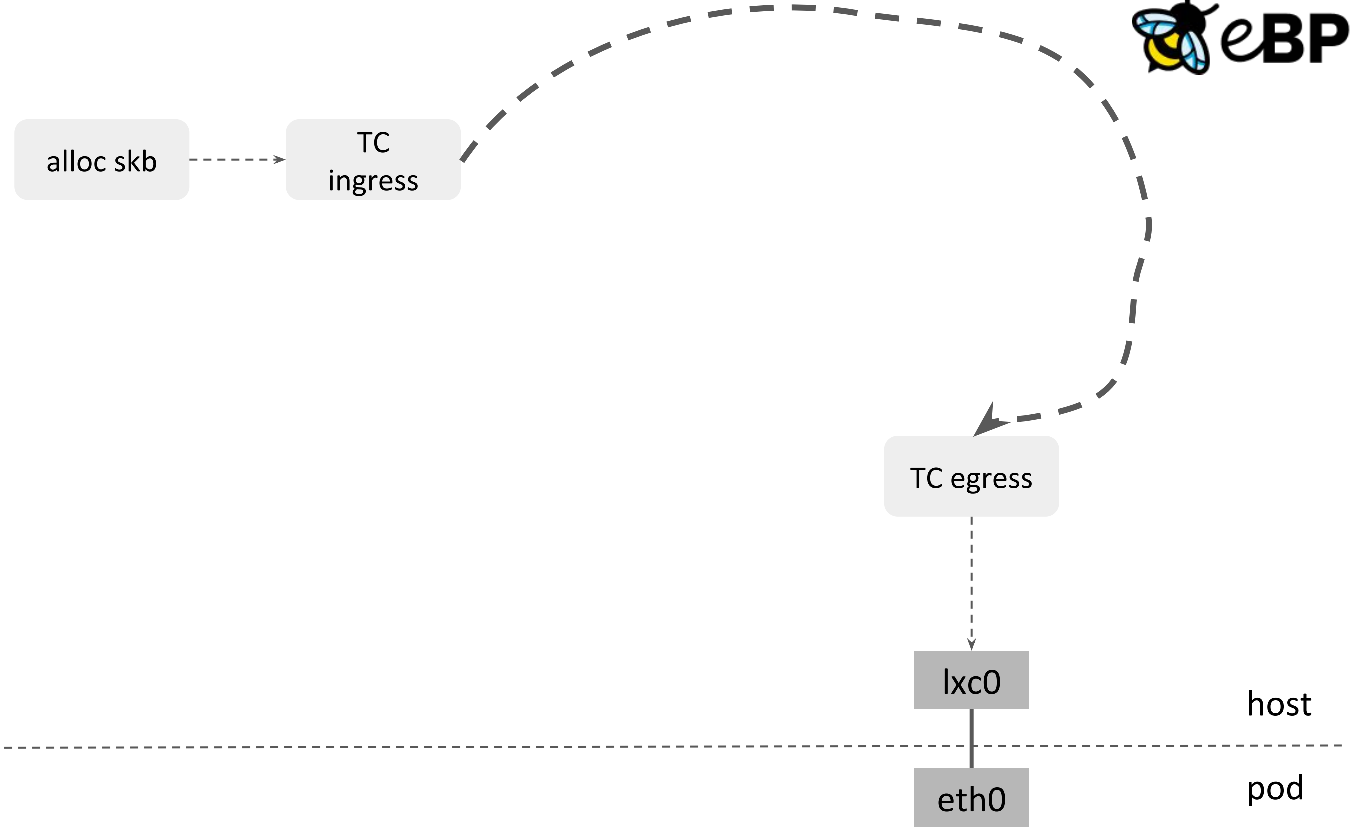




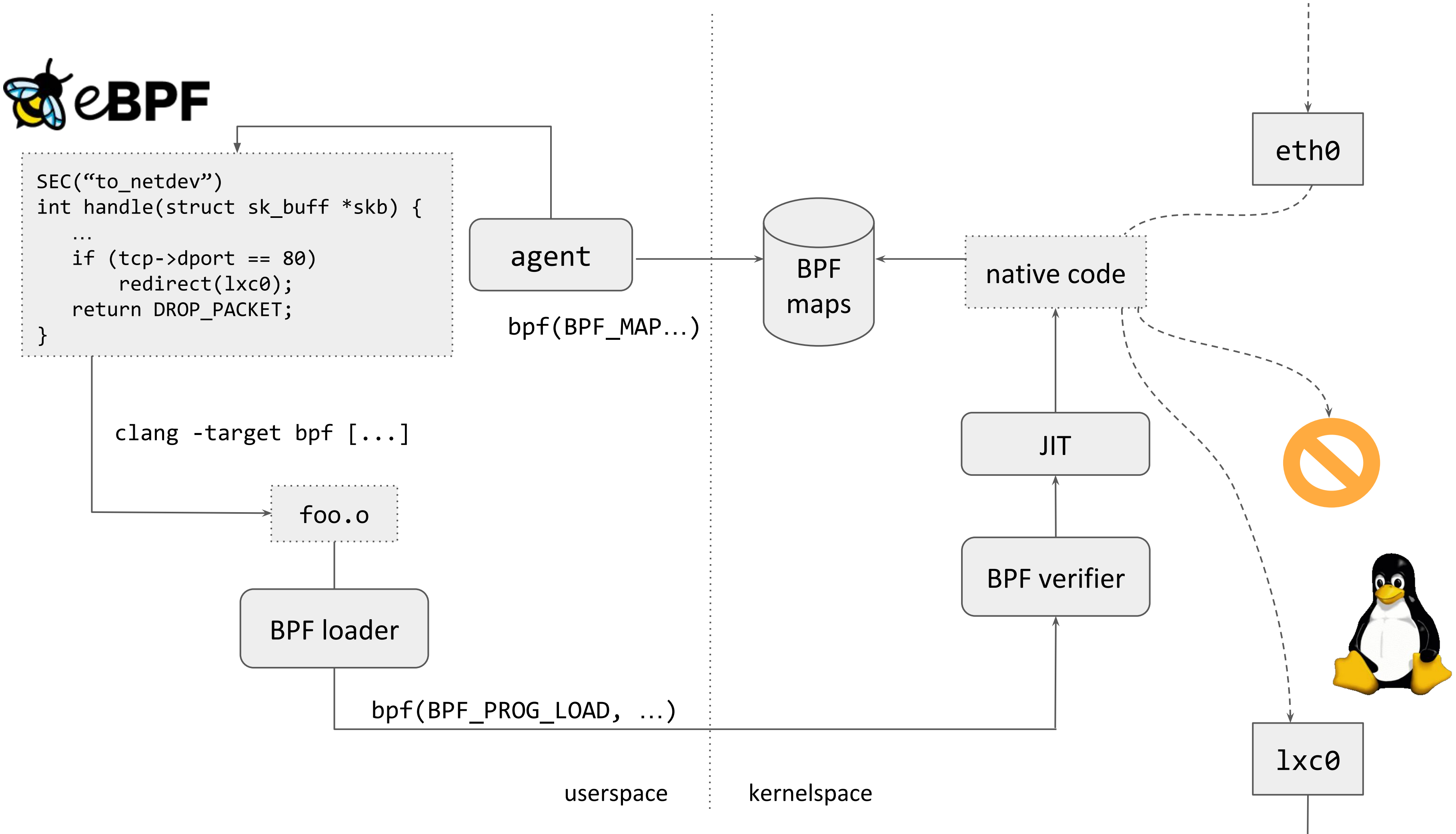
Source: commons.wikimedia.org/w/index.php?curid=122201

Packet flow





Source: commons.wikimedia.org/w/index.php?curid=122201


eBPF
SEC("to_netdev")
int handle(struct sk_buff *skb) {
...
if (tcp->dport == 80)
redirect(lxc0);
return DROP_PACKET;
}
agent
bpf(BPF_MAP...)
BPF maps
native code
eth0
clang -target bpf [...]
foo.o
JIT
BPF verifier
BPF loader
bpf(BPF_PROG_LOAD, ...)
lxc0
userspace
kernelspace

BPF as a radical shift towards full programmability

Freedom to let user tinker with the kernel through BPF programs, but with safety-belt on.

Main use-cases in networking, tracing and security subsystems, e.g. in networking, allows to fully define the forwarding pipeline.

Stable API guarantees as with syscalls. Native speed as with kernel modules. Atomic program updates on live kernel without service disruption. Designed for performance and solving production use-cases.

eBPF

287 contributors (Jan 2016 to Jan 2020):

- 466 Daniel Borkmann (Cilium; maintainer)
- 290 Andrii Nakryiko (Facebook)
- 279 Alexei Starovoitov (Facebook; maintainer)
- 217 Jakub Kicinski (Facebook, formerly Netronome)
- 173 Yonghong Song (Facebook)
- 168 Martin KaFai Lau (Facebook)
- 159 Stanislav Fomichev (Google)
- 148 Quentin Monnet (Cilium, formerly Netronome)
- 148 John Fastabend (Cilium)
- 118 Jesper Dangaard Brouer (Red Hat)
- [...]

Large-scale users:

TheRustyTwit
@rusty_twit

Replying to @LaF0rge

Well, iptables perf used to be "mostly good enough". Replacing it has taken so long because it requires a radically different approach; nice to see it finally happening!

12:46 AM · Apr 18, 2018 · Twitter for Android

BPF in Kubernetes networking and security: enter Cilium

- Datapath implemented in BPF
- Networking
  - Cilium-CNI or chaining on top of most other CNIs
- Kubernetes Services implementation
- Network Policies
  - Identity-based, DNS aware, API aware
- Multicluster, Encryption
- Native Envoy and Istio Integration
  - Transparent Envoy injection (per-node or sidecar)
  - Accelerated proxy redirection, Transparent SSL visibility
- All Open Source at github.com/cilium/cilium

cilium

Path towards replacing kube-proxy with BPF in Cilium

```
$ kubectl -n kube-system delete ds kube-proxy
```

kube-proxy

1. ClusterIP

- In-cluster access via virtual IP

2. NodePort

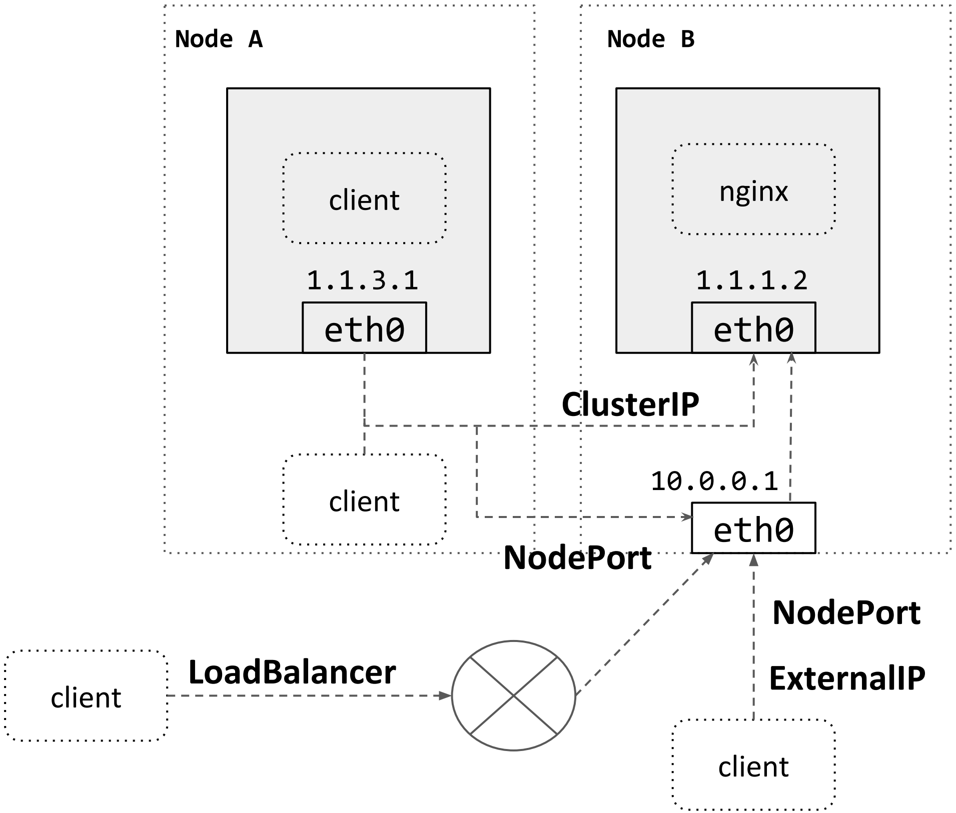
- Access from outside / inside via node IP + port

3. ExternalIP

- Access from outside via external IP

4. LoadBalancer

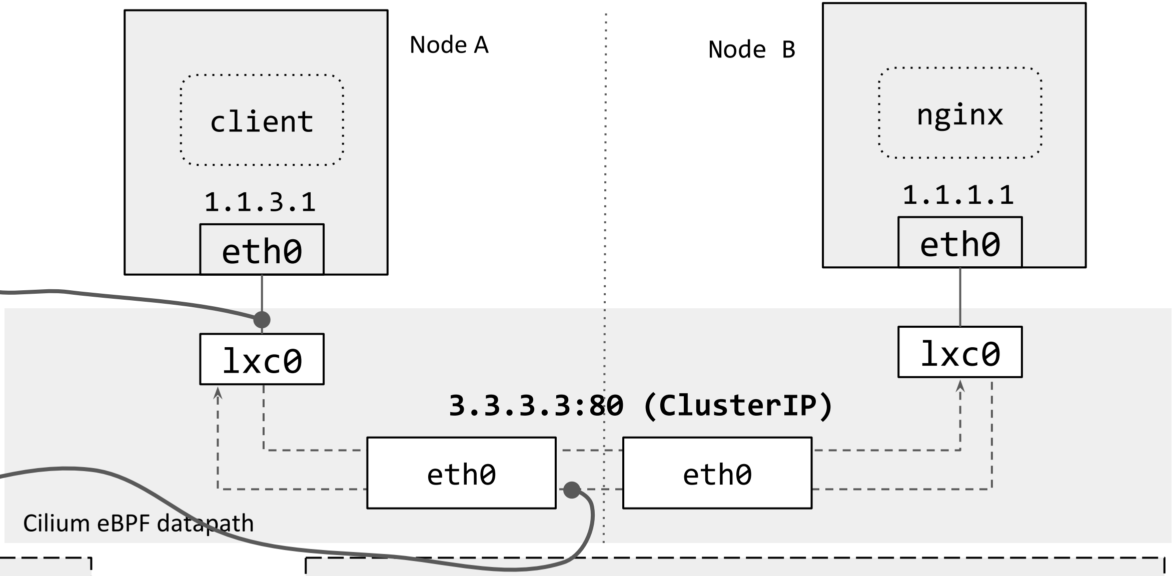
- Access from outside via external LB



ClusterIP (pod to pod) in Cilium

1. Lookup dst in SVC map
2. If found:
   a. Select EP
   b. DNAT
   c. Create SVC CT
   d. Create Egress CT

1. Lookup Egress CT
2. If found:
   a. Rev-DNAT xlation
   b. Redirect to lxc0



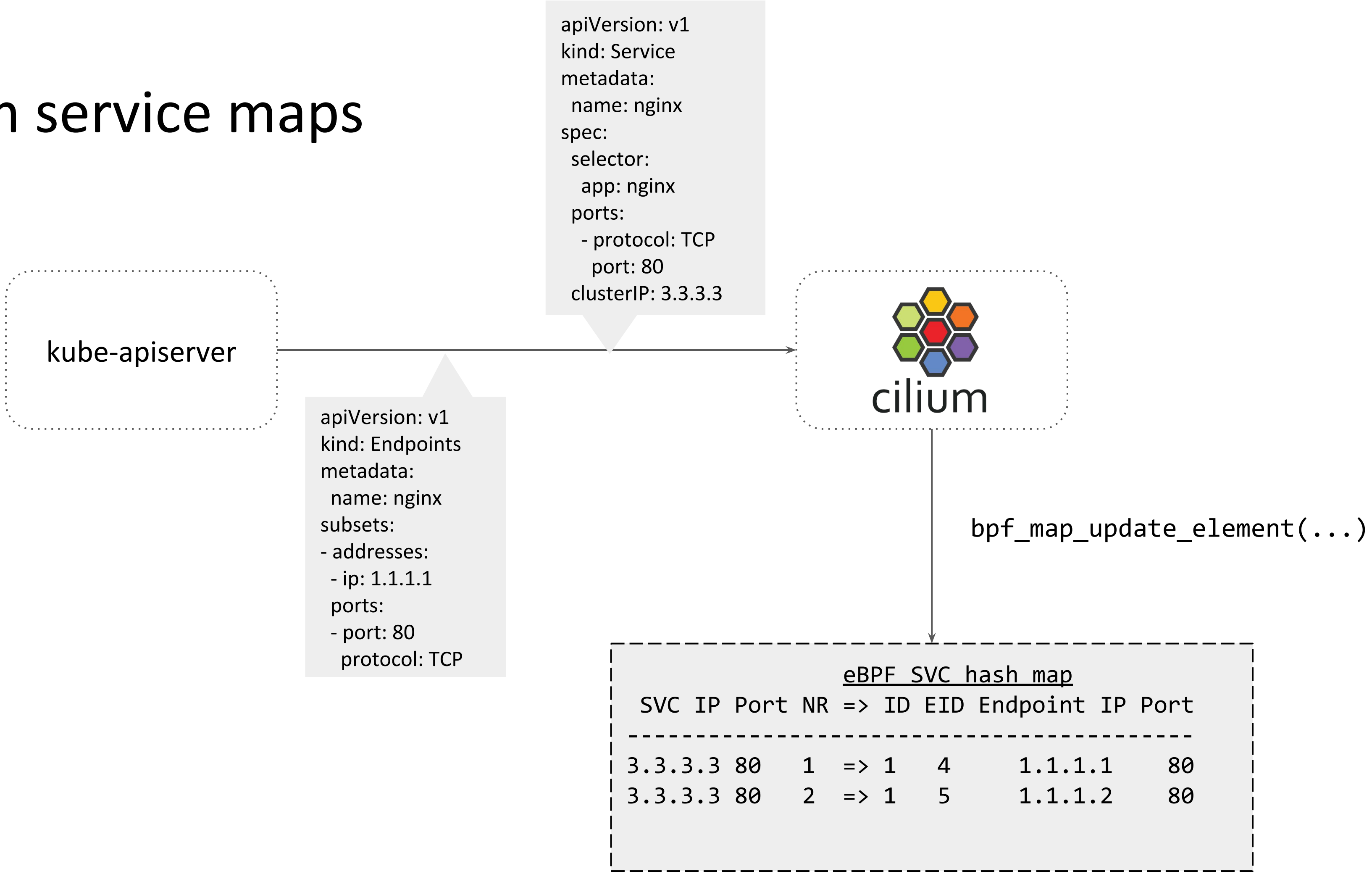
eBPF SVC hash map

SVC IP	Port	NR	=>	ID	EID	Endpoint IP	Port
3.3.3.3	**80**	**1**	**=>**	**1**	**4**	**1.1.1.1**	**80**
3.3.3.3	80	2	=>	1	5	1.1.1.2	80

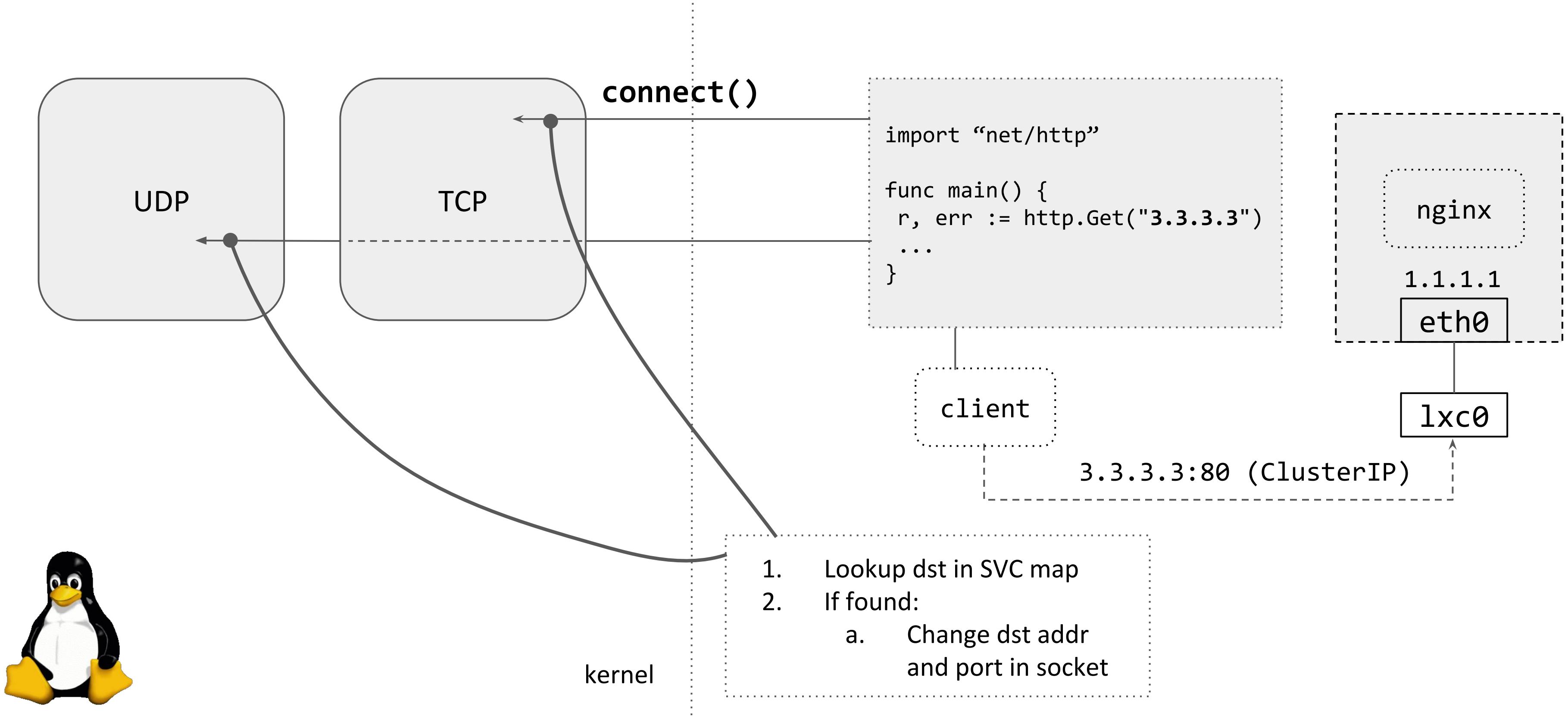
eBPF conntrack LRU map

srcIP	sPort	dstIP	dPort	Type	=>	EID|SVCID
1.1.3.1	**4321**	**3.3.3.3**	**80**	**SVC**	**=>**	**4**
1.1.3.1	**4321**	**1.1.1.1**	**80**	**Egress**	**=>**	**1**

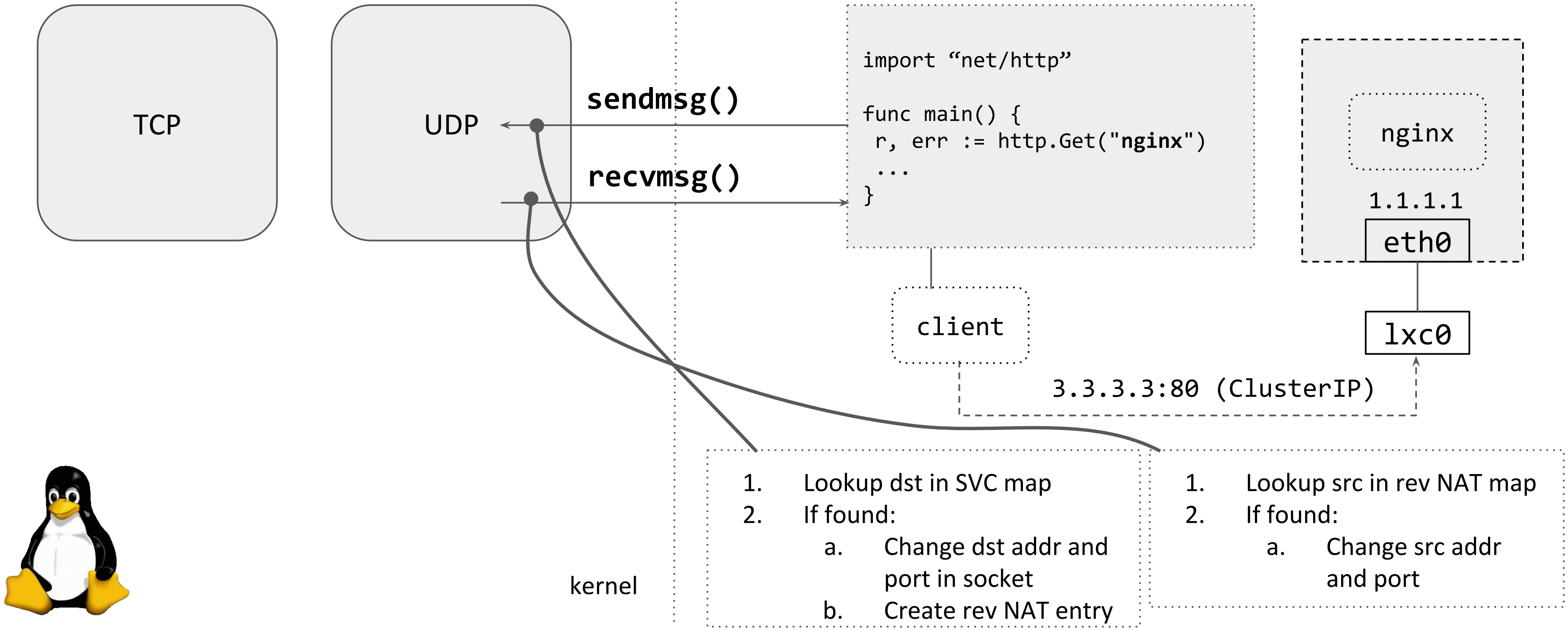
Cilium service maps



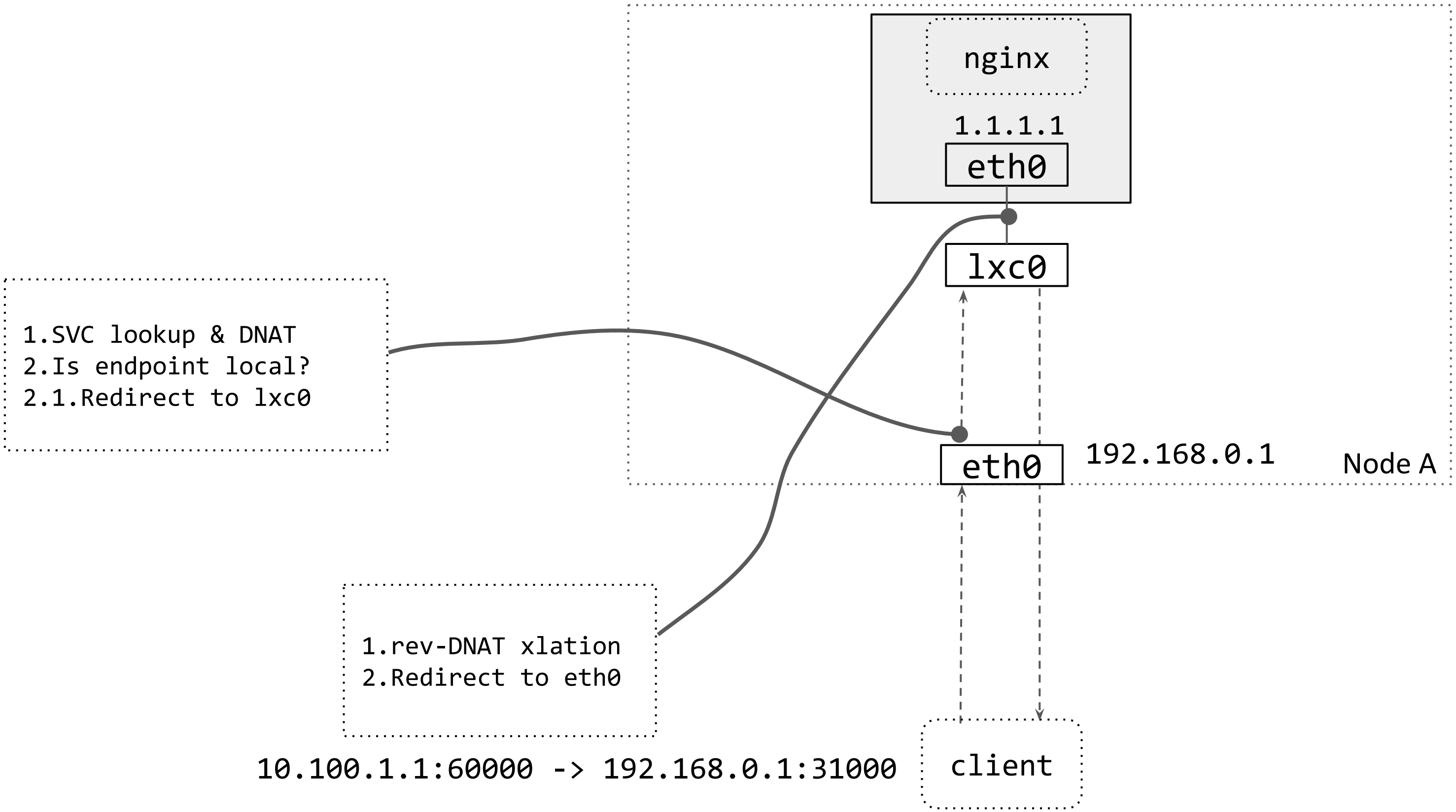
ClusterIP (host or pod to pod) in Cilium



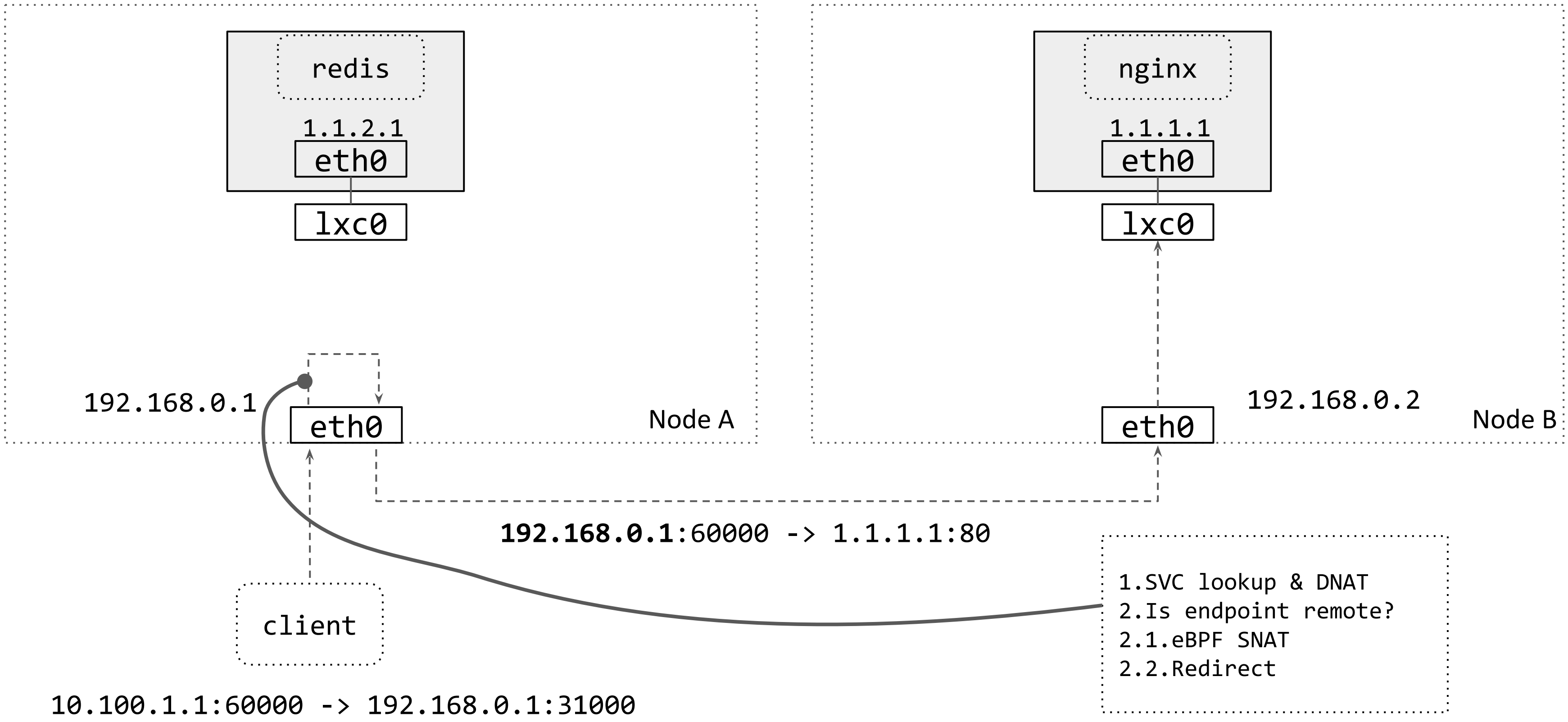
ClusterIP (host or pod to pod) in Cilium



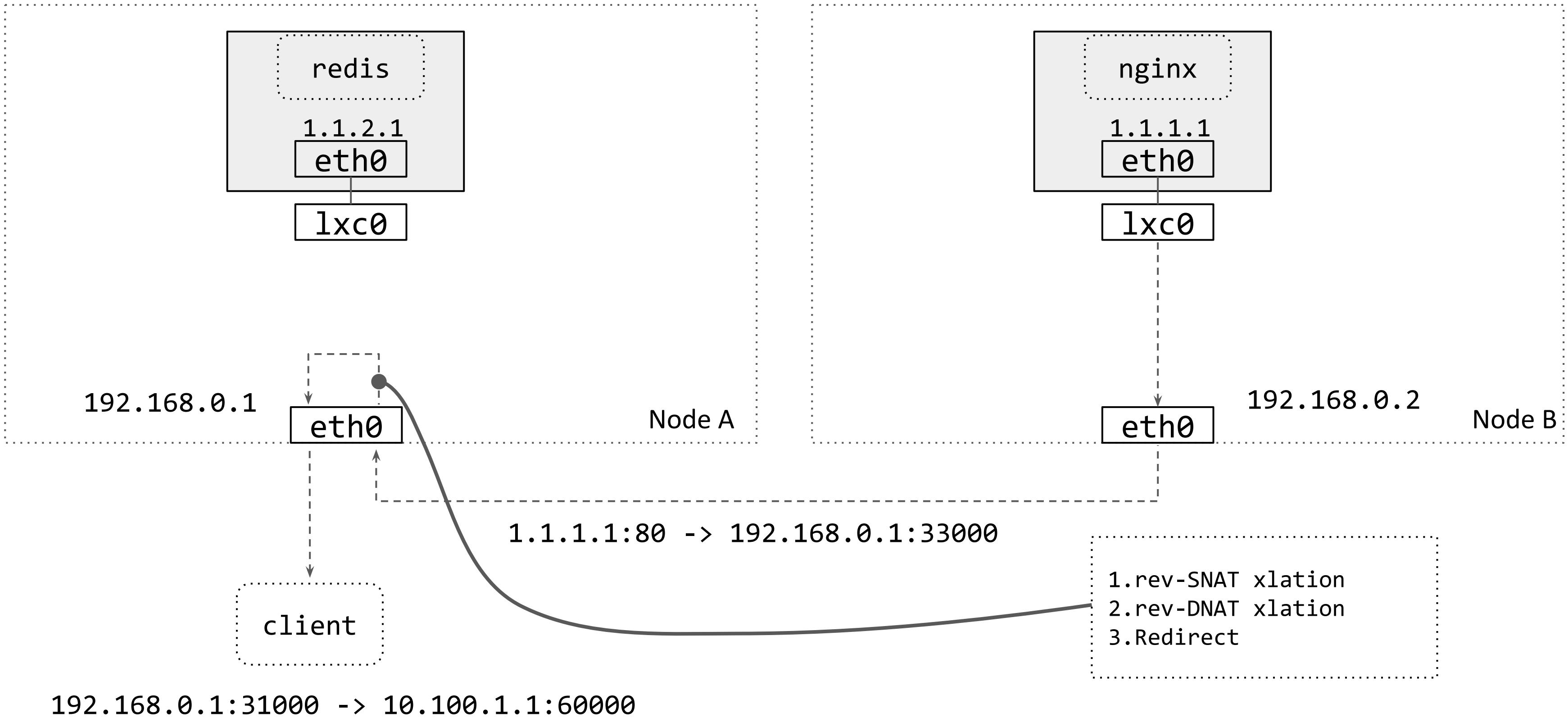
NodePort with service endpoint on local node in Cilium



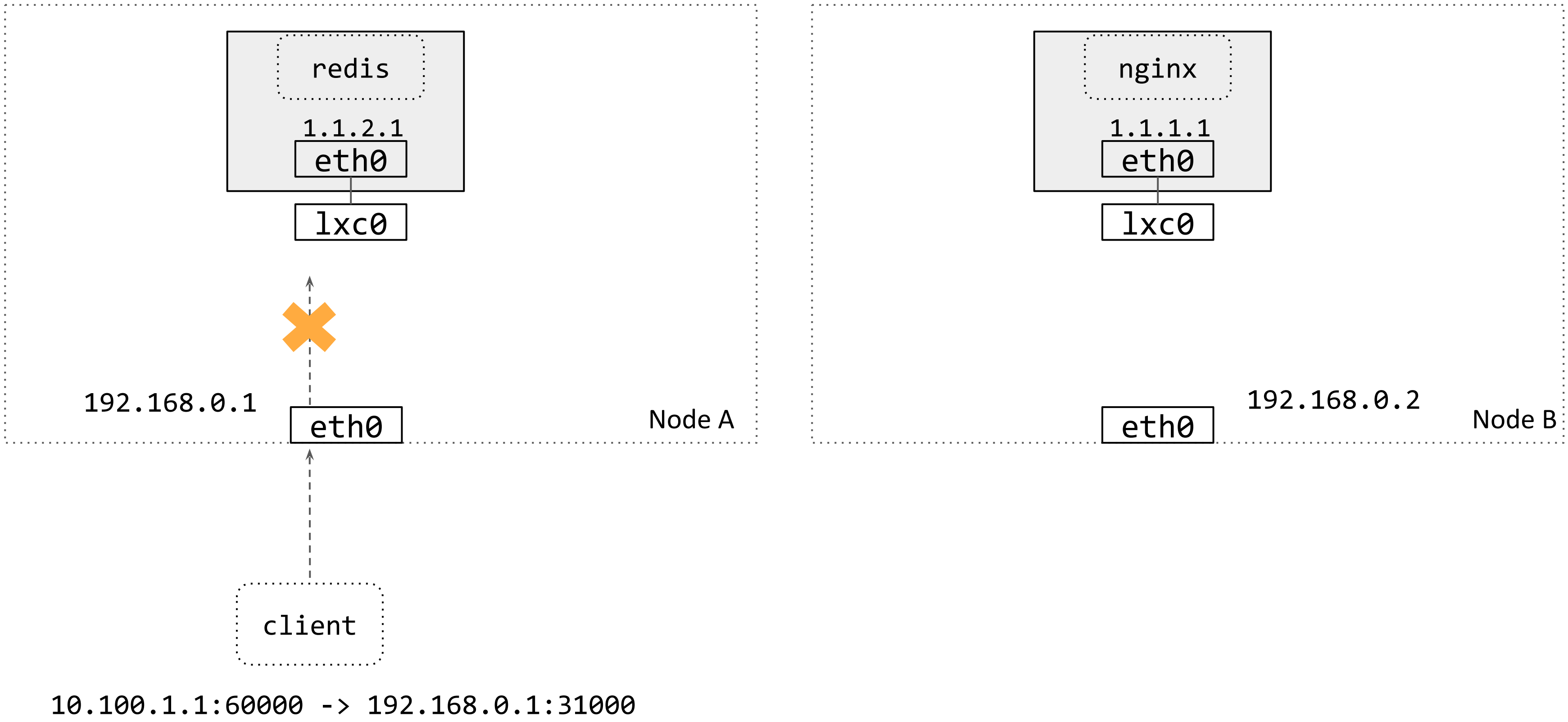
NodePort with service endpoint on remote node in Cilium



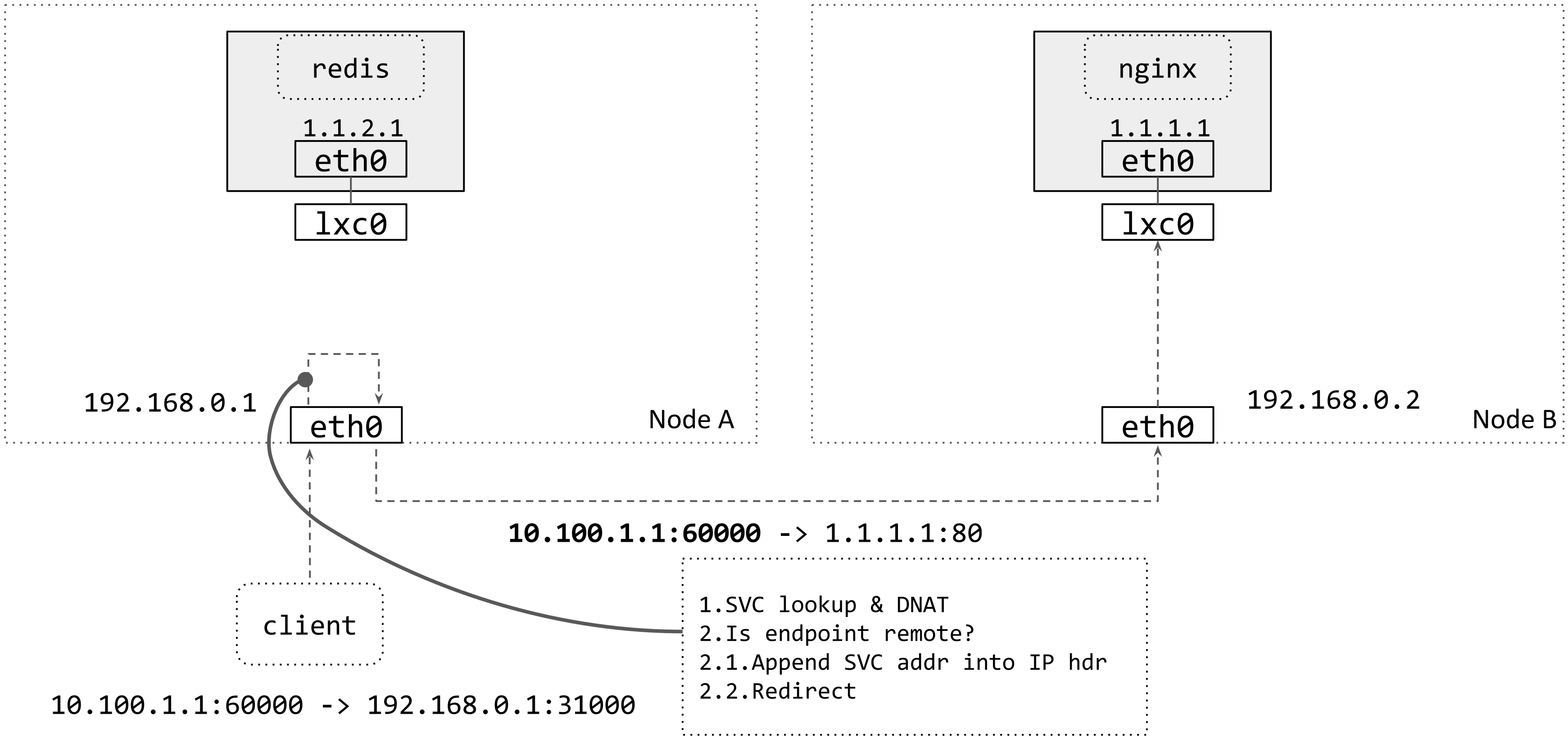
NodePort with service endpoint on remote node in Cilium



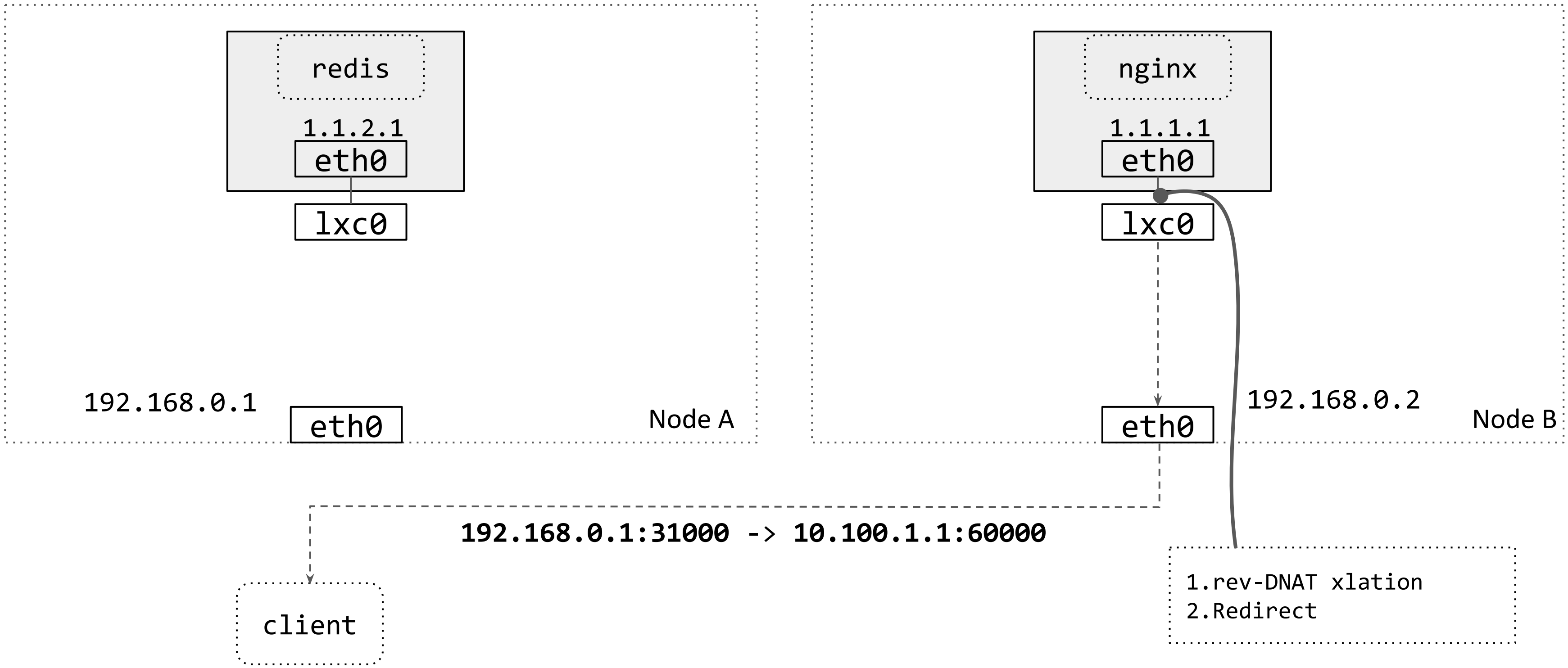
NodePort externalTrafficPolicy=Local



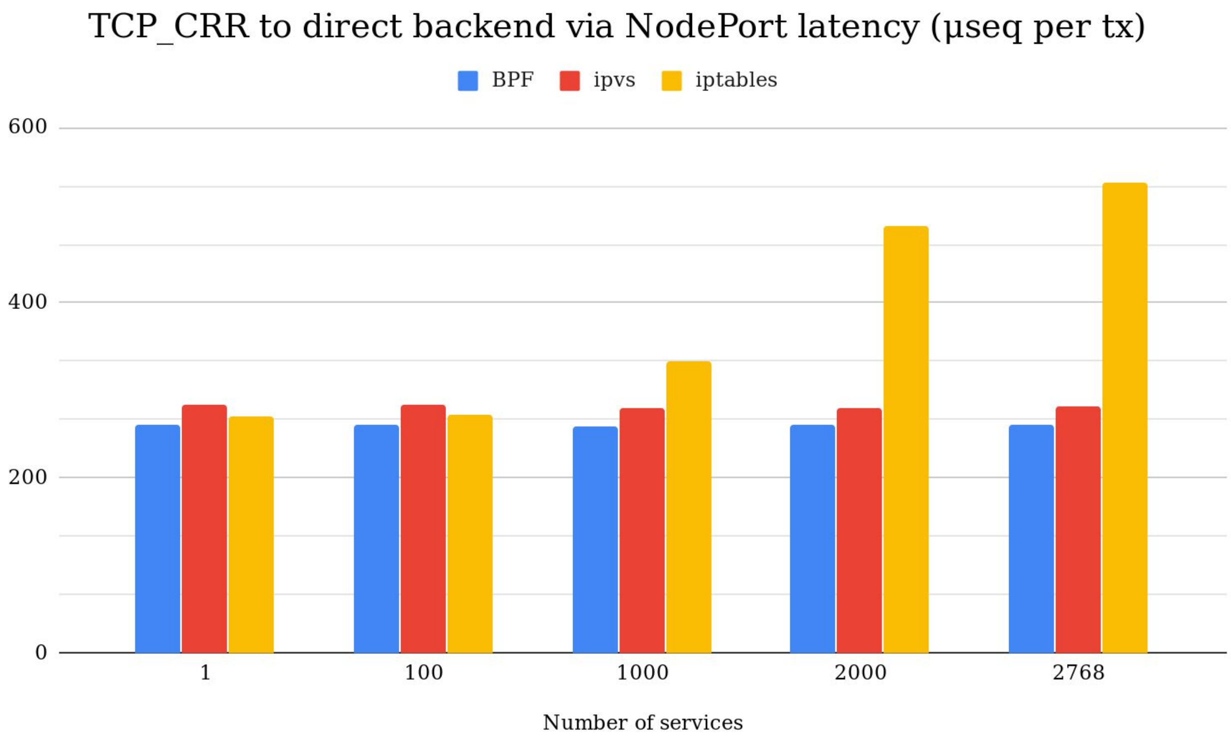
NodePort (DSR) in Cilium



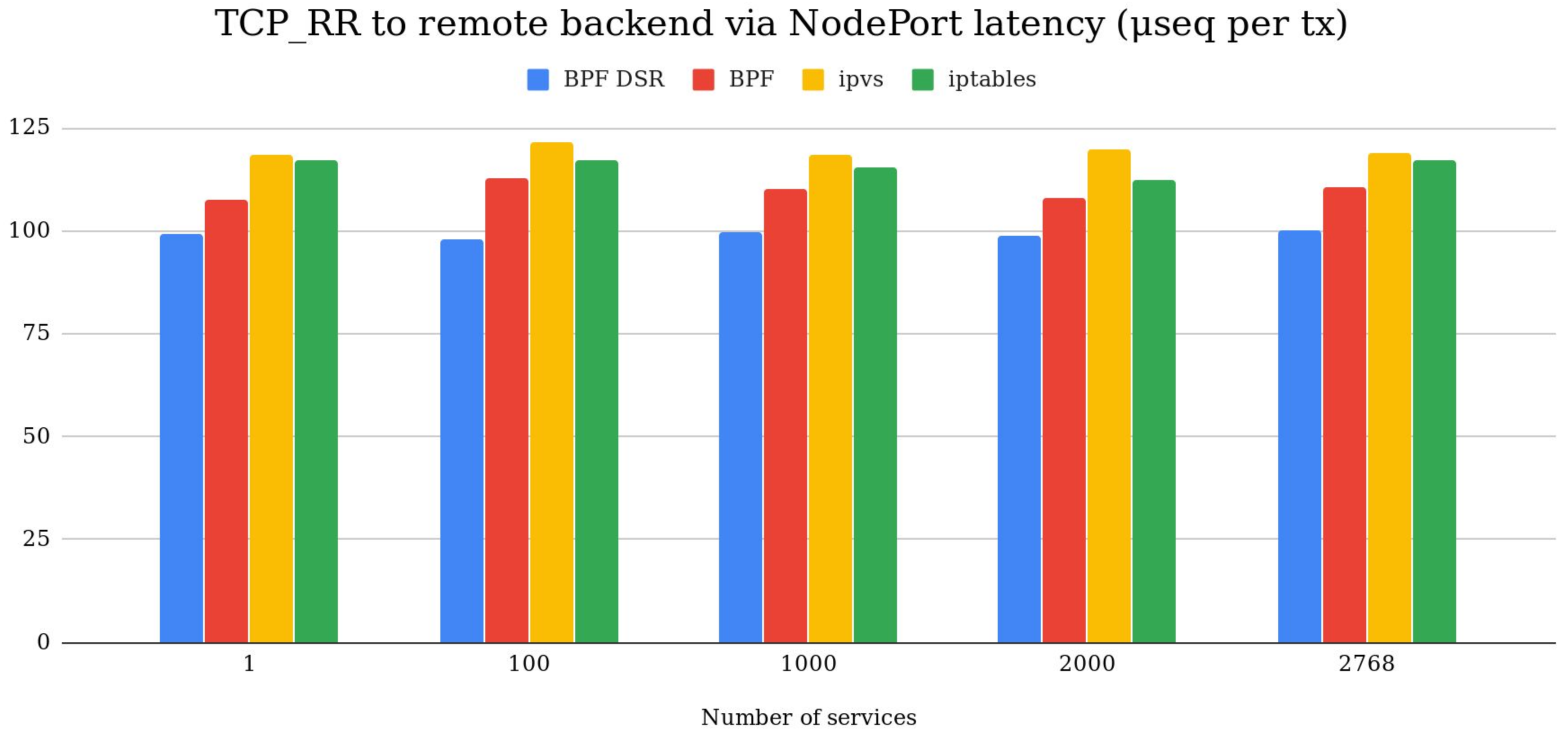
NodePort (DSR) in Cilium



Performance (lower is better)

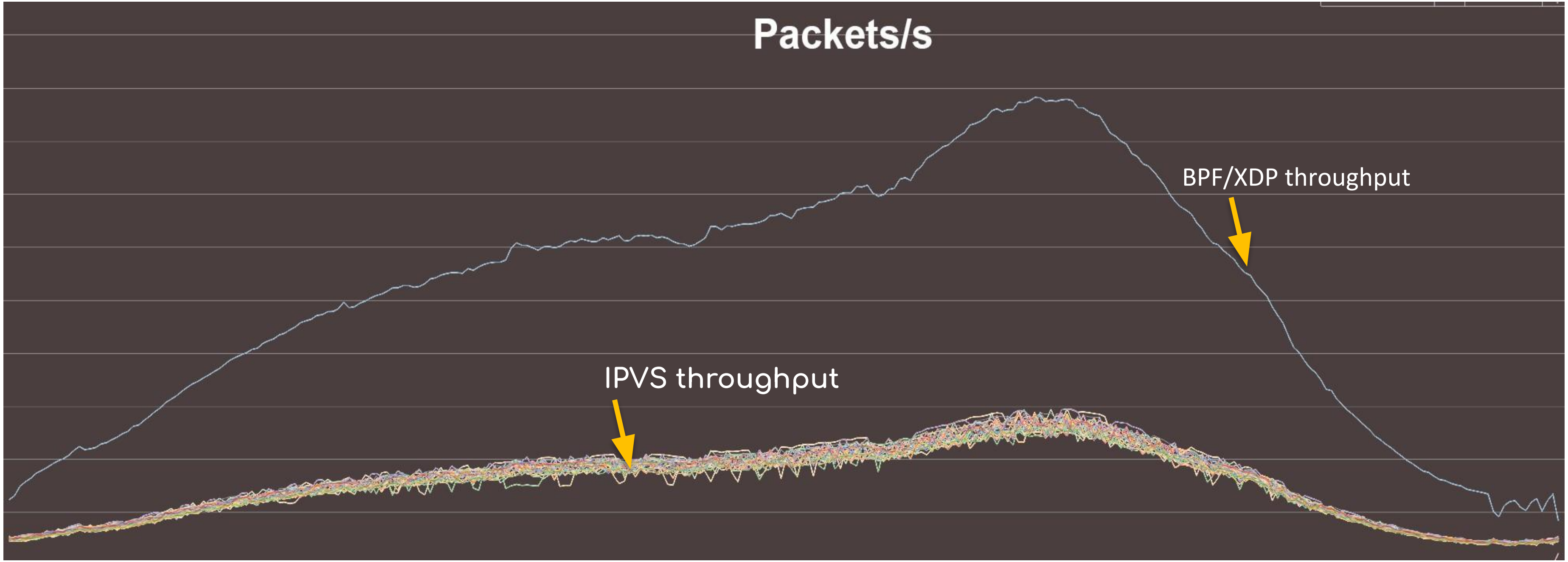


Performance (lower is better)



WIP for Cilium: XDP for hop to remote node (DSR, SNAT)

Native XDP finally supported by all 3 major cloud providers. 🎉



Source: https://www.netdevconf.org/2.1/slides/apr6/zhou-netdev-xdp-2017.pdf

tl;dr

Performance

- Better performance and latency over kube-proxy (ipvs and iptables)
- Fast service updates

Reliability

- Less LOC in datapath
- No need to wait for a new kernel release to fix a bug

Visibility

- Better tooling for introspection and troubleshooting

Compatibility

- No more exec iptables

Customization

- Ability to change datapath behaviour on the fly
- Fully integrated with rest of Cilium BPF datapath features

Want to liberate yourself from kube-proxy?

Howto: https://cilium.link/kubeproxy-free
Code: https://github.com/cilium/cilium
Slack: https://cilium.io/slack

eBPF

cilium