MakeIT 2022

University *of Ljubljana*
Faculty *of Computer and Information Science*

Intro to BPF and BPF-powered Kubernetes Service Mesh

dr. Matjaž Pančur

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

kubectl describe

Matjaž Pančur, PhD
matjaz.pancur@fri.uni-lj.si

University of Ljubljana
Faculty of Computer and Information Science

K8s/containers, DevOps, Infra, Cloud-native, Security, High Performance (Cloud) Computing

Industry collab.: PoCs, hard problems, applied research, consulting

Head of "Garaža FRI" (https://garaza.io)

- Tech Accelerator @ UL FRI

LinkedIn: https://www.linkedin.com/in/matjazpancur/

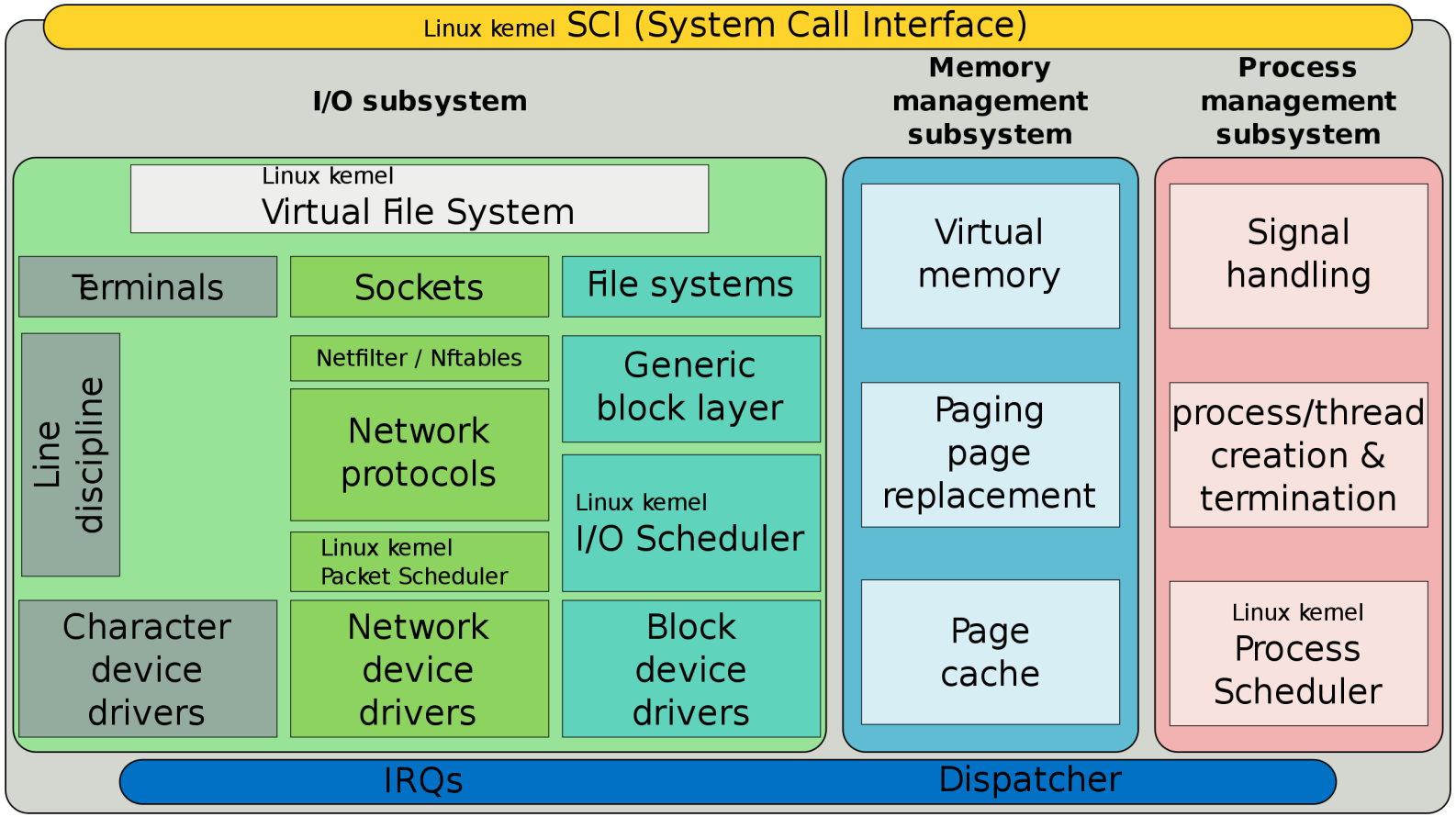
kubectl get agenda -o slide

- Standard Linux networking stack, iptables, alternatives
- BPF/XDP
- BPF-powered Kubernetes Service Mesh

Goal: introduce BPF/XDP concepts, pros and cons in a K8s context

Linux kernel and net stack

- Many parts; kernel and userspace control apps; battle-tested, stable



Kernel, standard net stack, iptables

- **iptables**: filtering, firewall, load balancing, NAT etc.
  - Sequential rules, many chains
  - Quite optimized data path in kernel
  - (Poor) performance when app (in userspace!) deals with a lot of traffic

Alternatives in standard Linux net stack

- nftables? ipset? ipvs?
- Kernel-bypass?
  - PF_RING ZC, DPDK (Data Plane Development Kit), VPP (Vector Packet Processing)
  - Poll based drivers vs. using interrupts
  - Usually L2/L3 use cases vs. socket-delivery use cases with Linux netw. stack

Standard Linux netw. stack in K8s cluster

Problems get worse with scaling:

- Connectivity, observability, security, troubleshooting

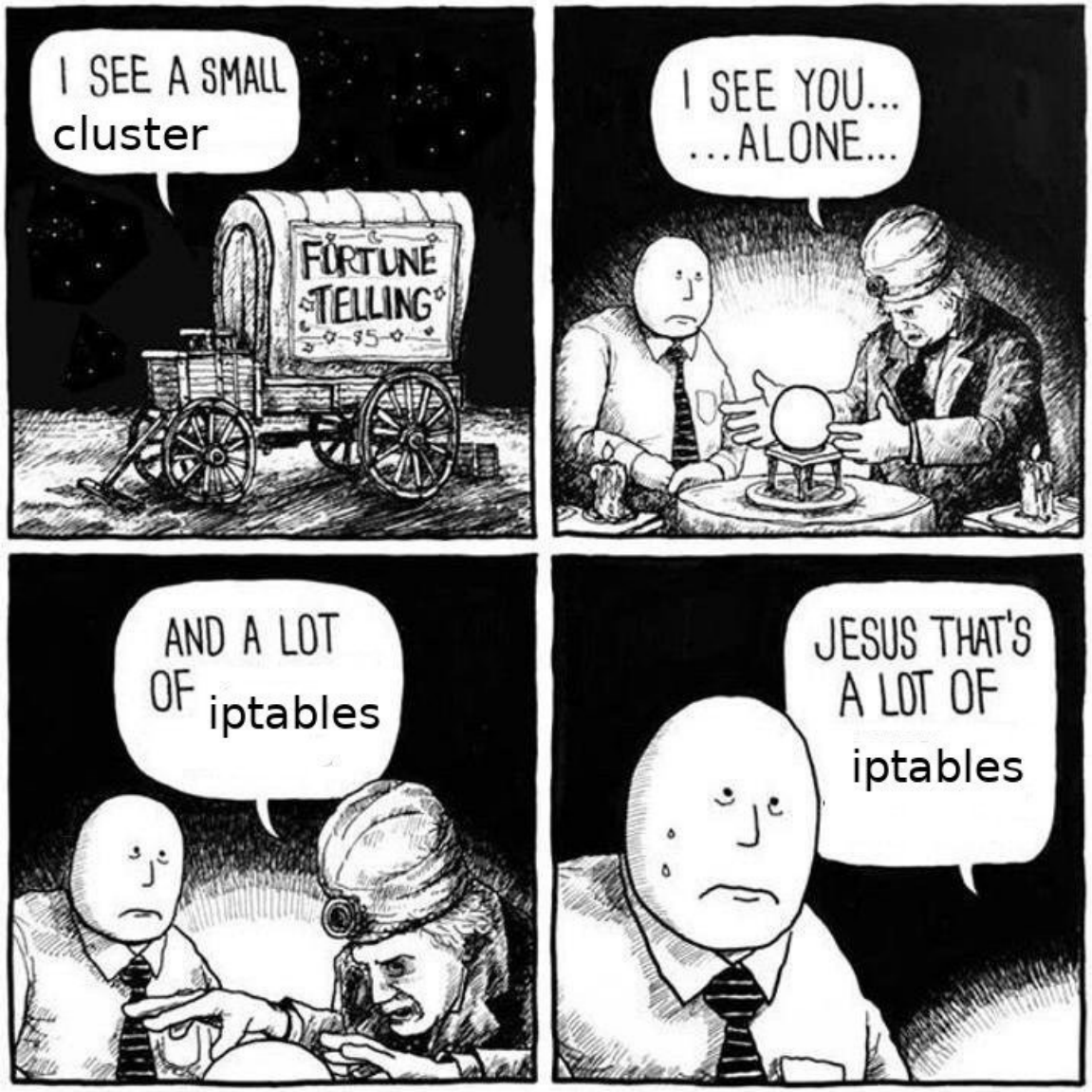
- Popular solution in K8s: use a "sidecar model" (helper containers in a Pod – e.g., Istio service mesh)

Standard Linux netw. stack in K8s cluster

- Problems with “sidecar model”
  - High CPU cost
  - Just “local” visibility for security events that span multiple workloads

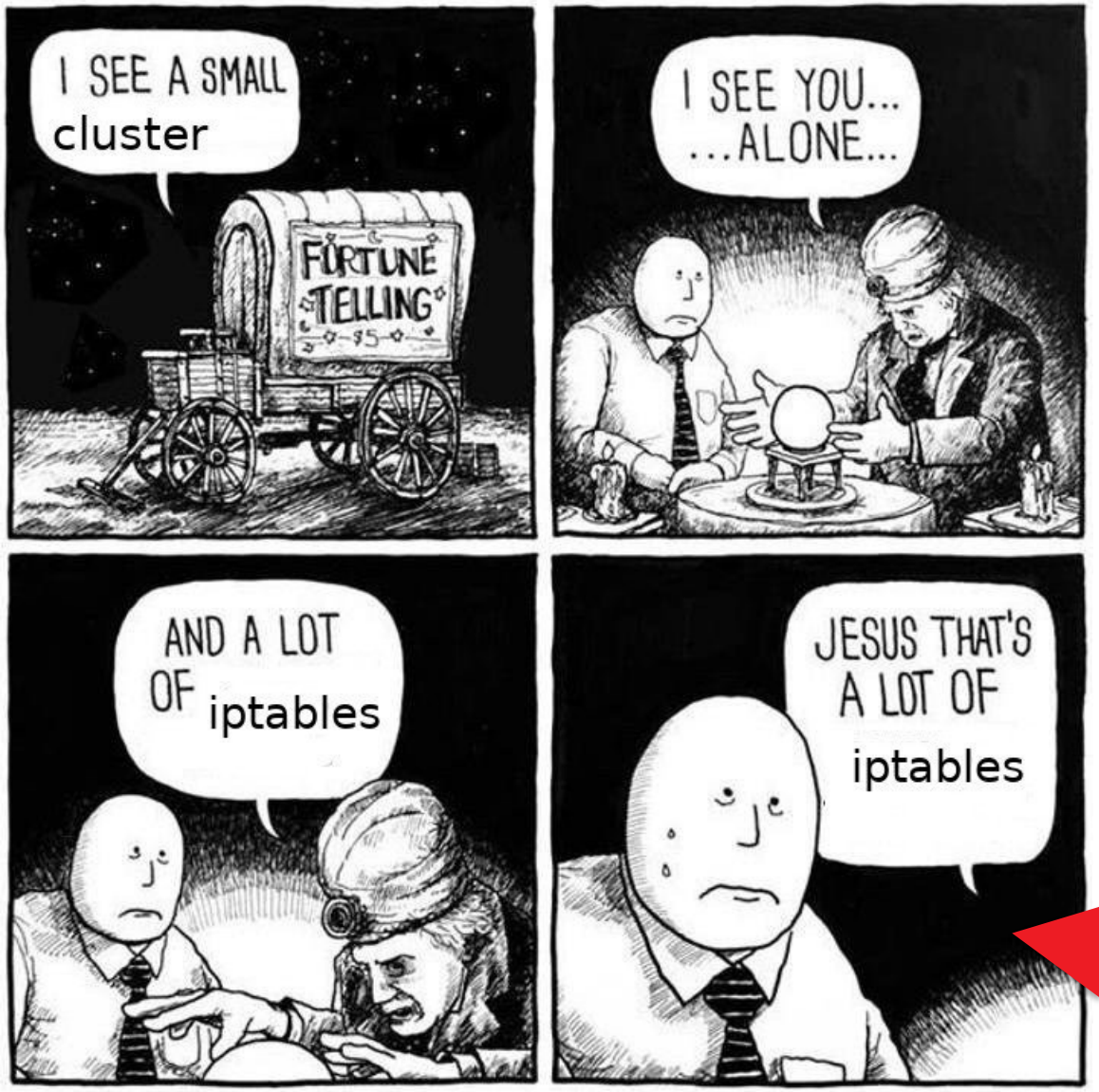
- IP/port-based mechanisms not a good fit for highly dynamic cloud-native environments e.g. Kubernetes
  - (ab)using iptables

(Ab)using iptables in K8s clusters...



Source: https://nixcraft.tumblr.com/post/175278276307/kubernetes-networking

(Ab)using iptables in K8s clusters...



...been there...

Source: https://nixcraft.tumblr.com/post/175278276307/kubernetes-networking

... lead to iptables as a superpower?

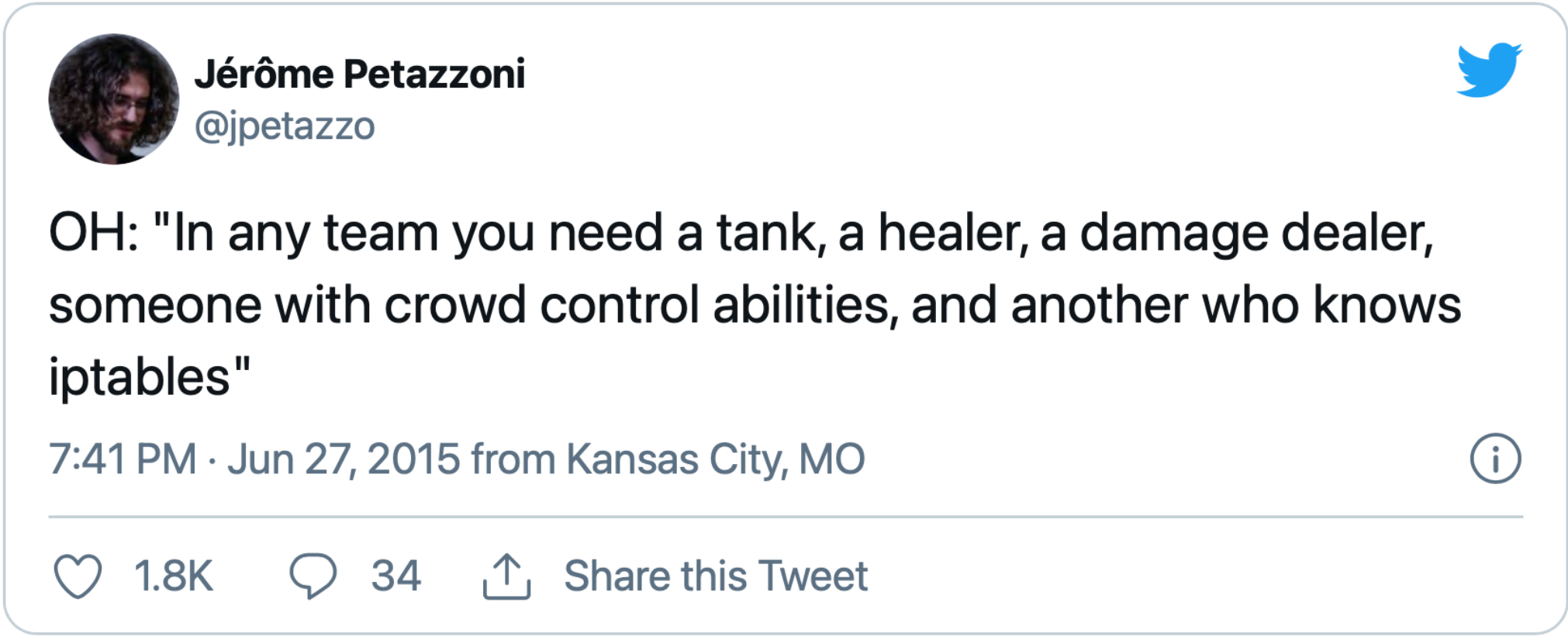
Jérôme Petazzoni
@jpetazzo

OH: "In any team you need a tank, a healer, a damage dealer, someone with crowd control abilities, and another who knows iptables"

7:41 PM · Jun 27, 2015 from Kansas City, MO

1.8K 34 Share this Tweet

… lead to iptables as a superpower?

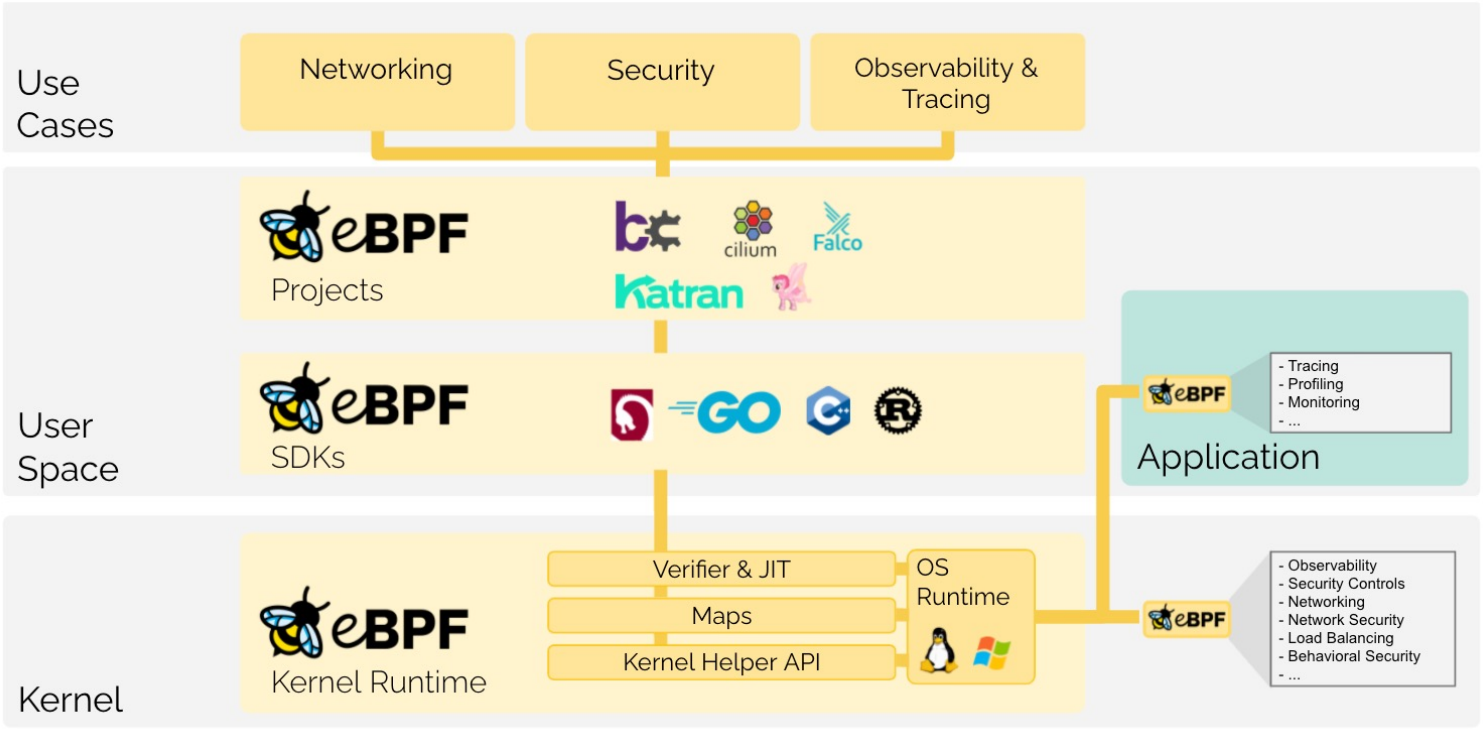


New IT superhero team: Ip Tablesman and Little Bobby Tables as his sidekick?

Don’t know Bobby Tables? See https://xkcd.com/327/

Intro to BPF

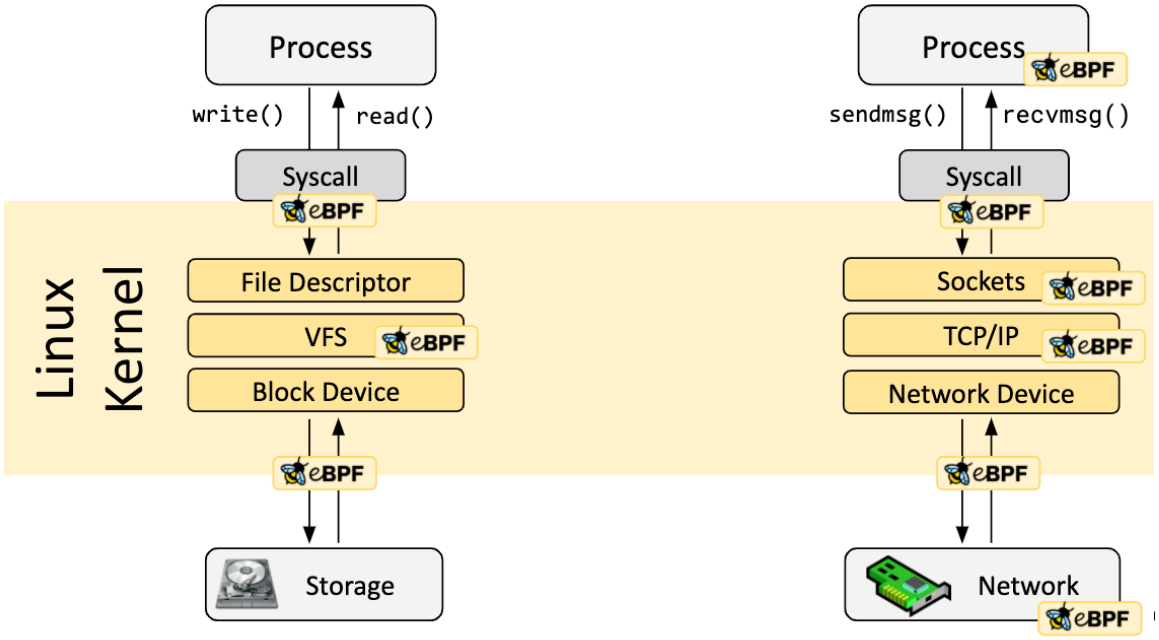
- (extended) Berkeley Packet Filter
  - classic BPF (since 1992): all *nix
  - eBPF (since 2014): Linux (+ Windows)
- Enables running custom code in Linux kernel
  - As an alternative to: add to upstream kernel or write a kernel module



Intro to BPF

- **In-kernel** execution engine
  - **Low overhead** when mapping to native code
  - **Verifiable** for safety

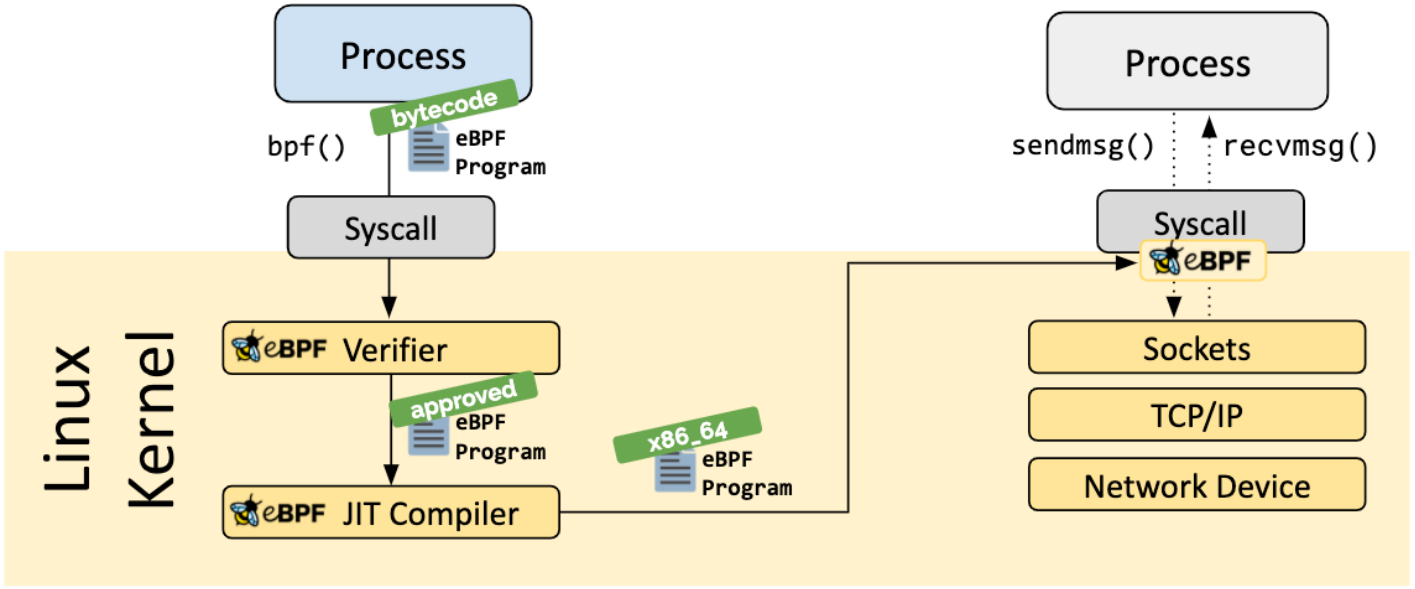
- **Event-driven**; attach *at various hook points* (kernel and userspace!)
  - syscall,
    packet arrival,
    func. entry/exit,
    kprobe/uprobe,
    etc.



https://ebpf.io/

Intro to BPF

- **Verification**: ensures that the eBPF program is safe to run



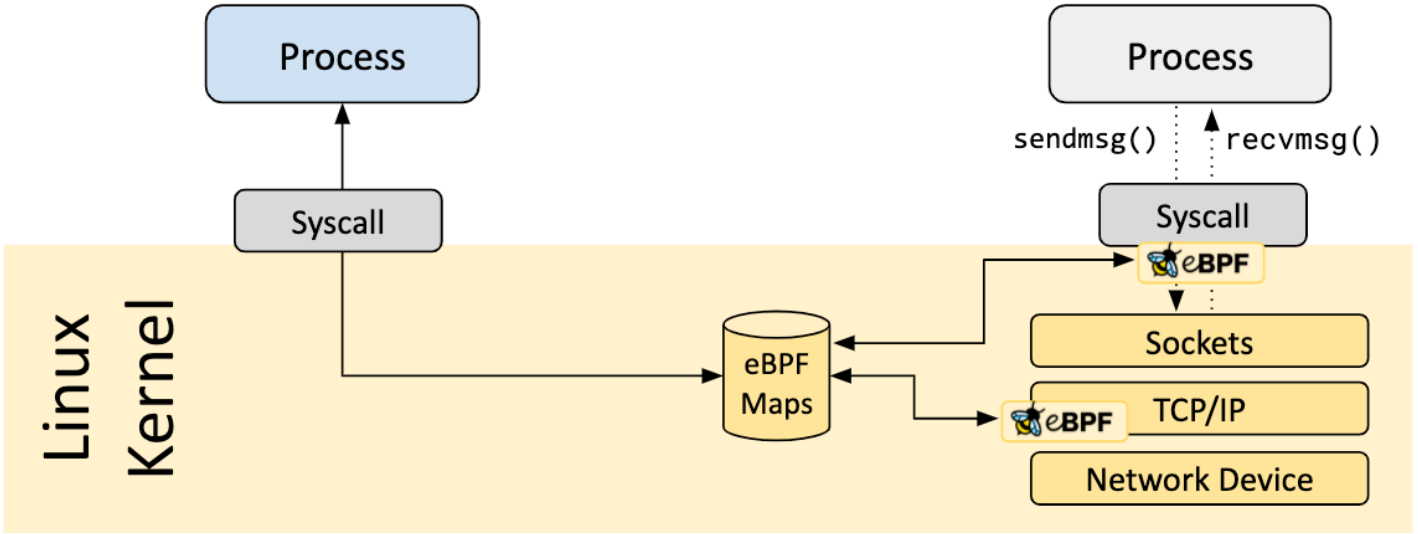
Checks that:

- Process has required capabilities (privileges)
- Program does not crash or otherwise harm the system
- always runs to completion (e.g. no unbounded loops)

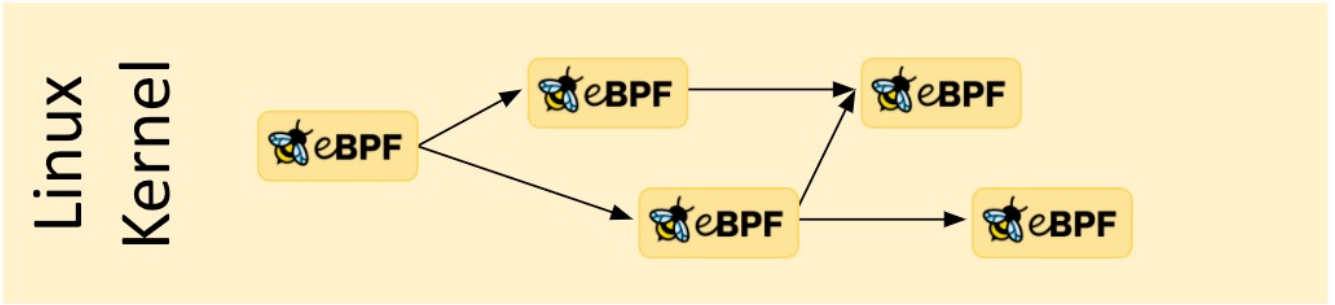
https://ebpf.io/

Intro to BPF

- BPF Maps (share info, store state)



- BPF tail and function calls (composability)



https://ebpf.io/

Intro to BPF

- Helper calls
  - Well known and stable API; differ for BPF prog. type
  - Cannot call into arbitrary kernel functions as this binds to specific kernel version

https://ebpf.io/

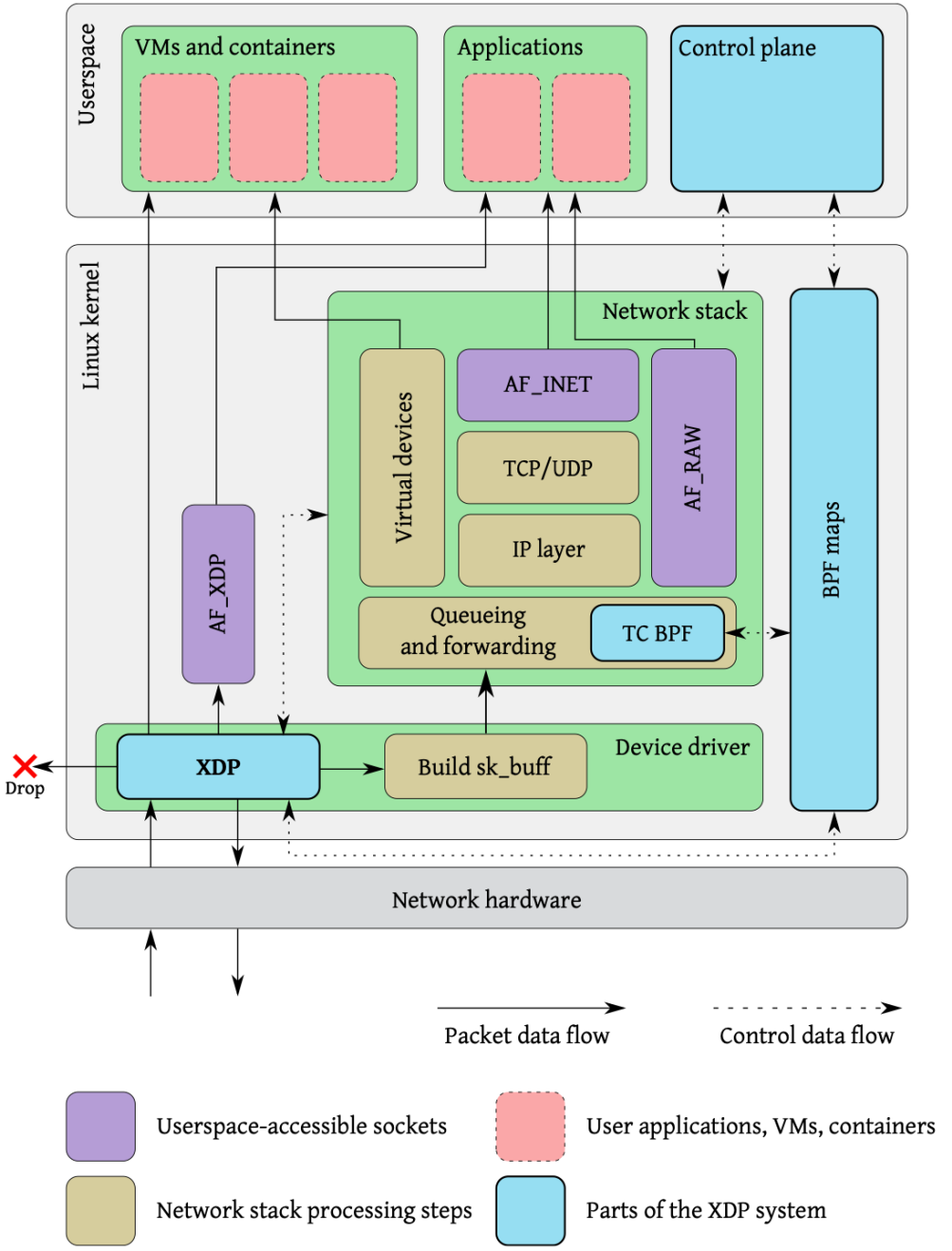
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

XDP - eXpress Data Path

- Process packets in kernel as soon as possible, bypass netw. stack

- XDP operation modes
  - native (driver!)
  - offload (NIC hw!)
  - generic

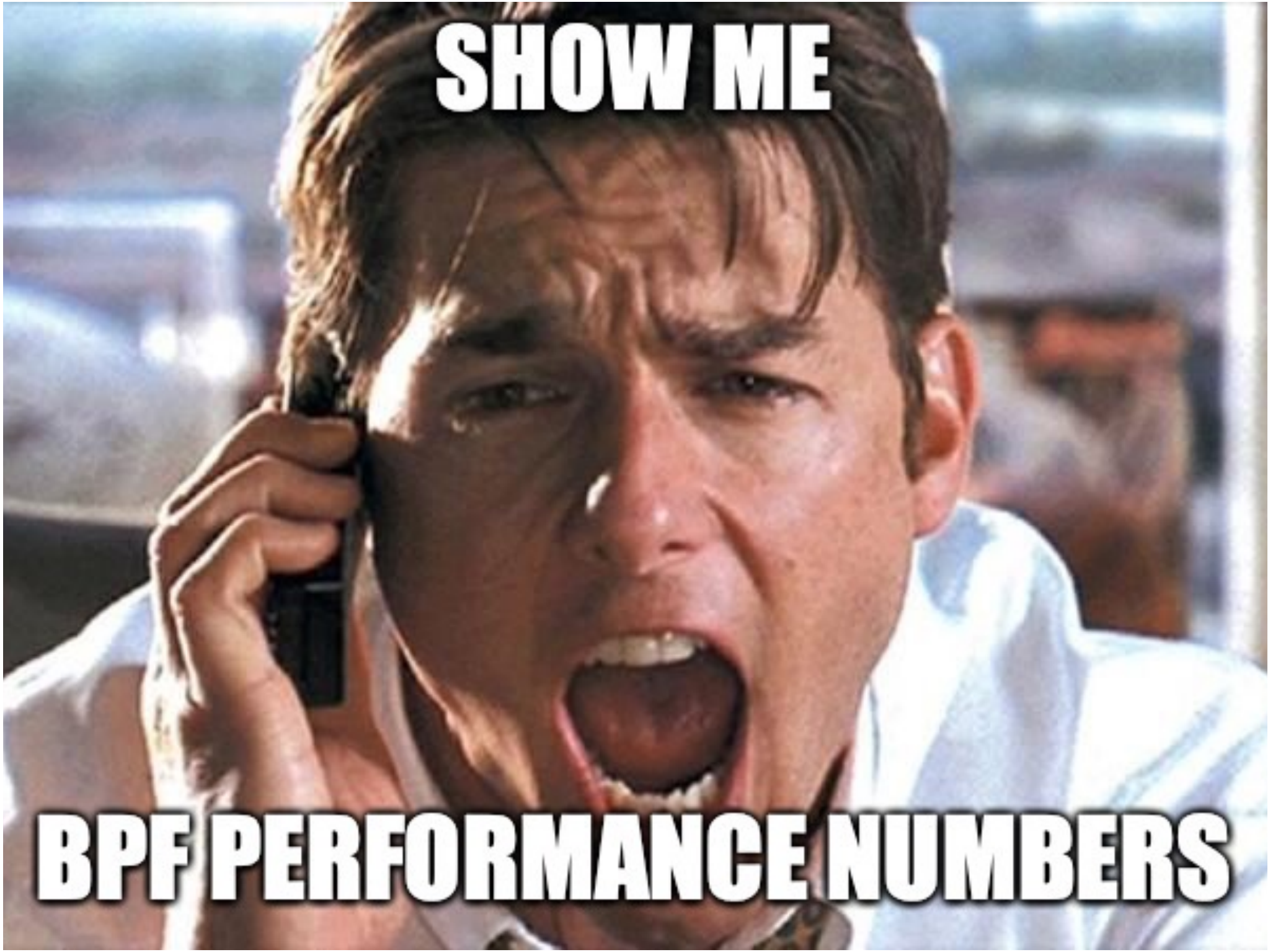
- XDP_DROP, XDP_PASS, XDP_TX, XDP_REDIRECT
- tc/BPF (traffic control) vs XDP/BPF



https://doi.org/10.1145/3281411.3281443

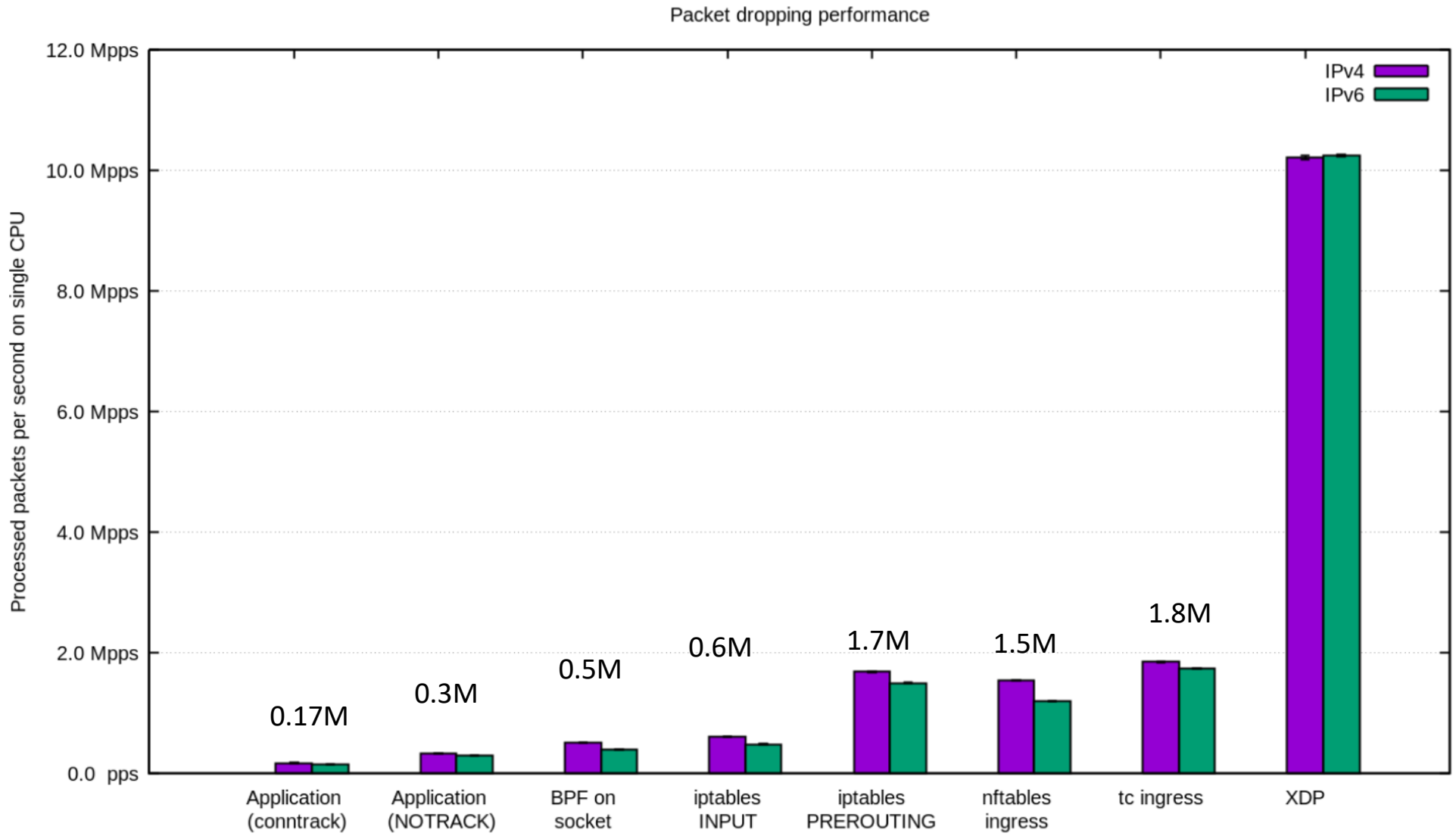
BPF cons

- Needs a fairly new kernel
  - for all available features/optimizations; Cilium min. 4.9.17, >=5.10 for max. features
  - RH backports features for some older kernel versions
- Different development toolchains and expertise
- (Probably?) Lack of in-house expertise
- Usually still slower than kernel-bypass


SHOW ME
BPF PERFORMANCE NUMBERS

Kernel bottlenecks

CF packet drop, single core, 10G nic, traffic 14Mpps small UDP



https://blog.cloudflare.com/how-to-drop-10-million-packets/

Packet drop XDP vs DPDK vs traditional

- 100G NIC

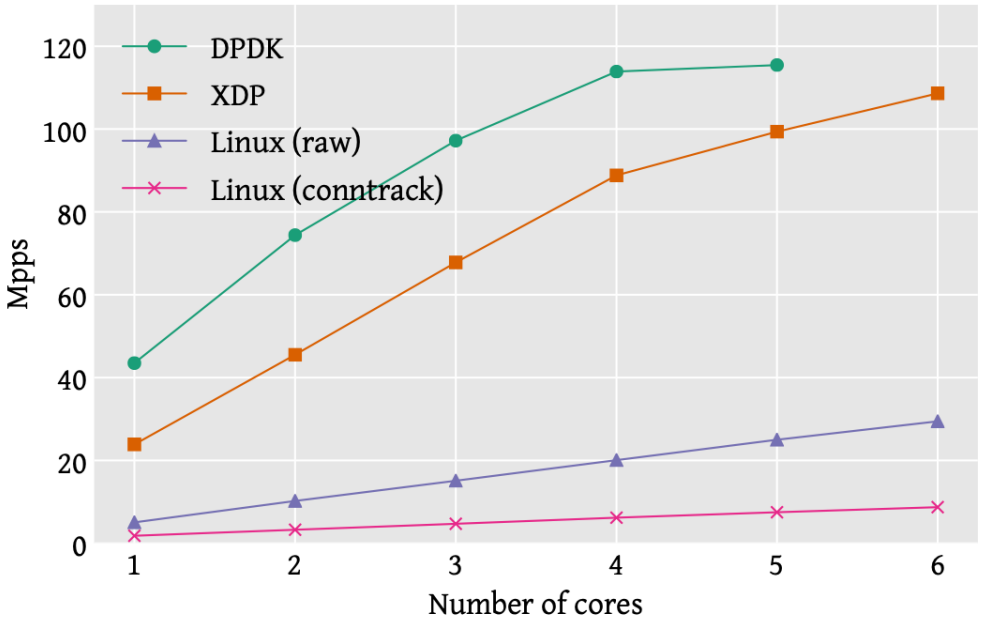


Figure 3: Packet drop performance. DPDK uses one core for control tasks, so only 5 are available for packet processing.

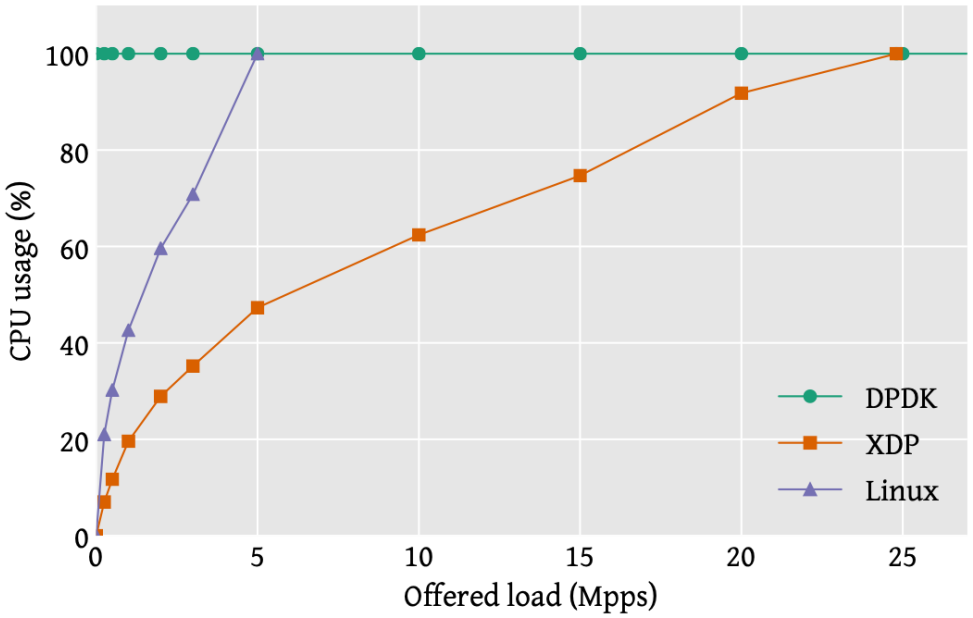
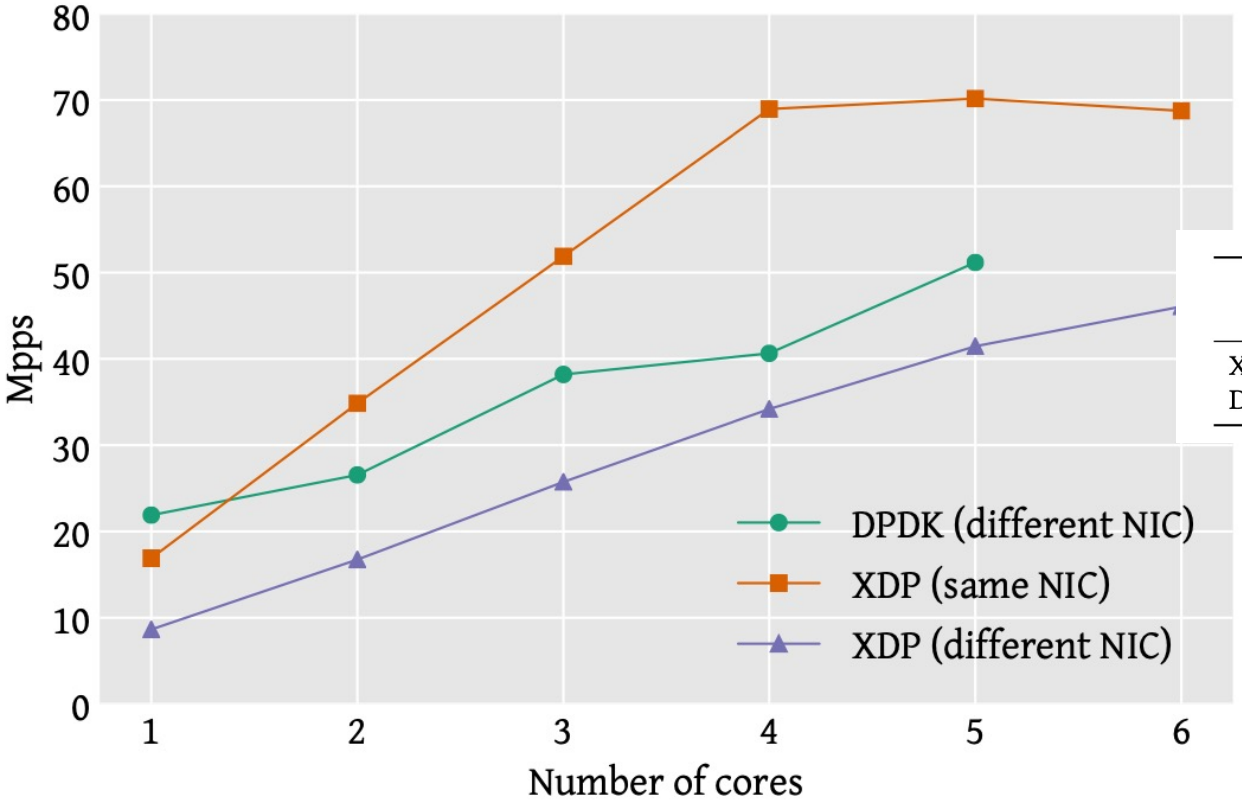


Figure 4: CPU usage in the drop scenario. Each line stops at the method's maximum processing capacity. The DPDK line continues at 100% up to the maximum performance shown in Figure 3.

https://doi.org/10.1145/3281411.3281443

Packet forwarding and latency



	Average		Maximum		$< 10\mu s$	
	100 pps	1 Mpps	100 pps	1 Mpps	100 pps	1 Mpps
XDP	$82\mu s$	$7\mu s$	$272\mu s$	$202\mu s$	0%	98.1%
DPDK	$2\mu s$	$3\mu s$	$161\mu s$	$189\mu s$	99.5%	99.0%

Figure 5: Packet forwarding throughput. Sending and receiving on the same interface takes up more bandwidth on the same PCI port, which means we hit the PCI bus limit at 70 Mpps.

https://doi.org/10.1145/3281411.3281443

TCP peformance during DDoS

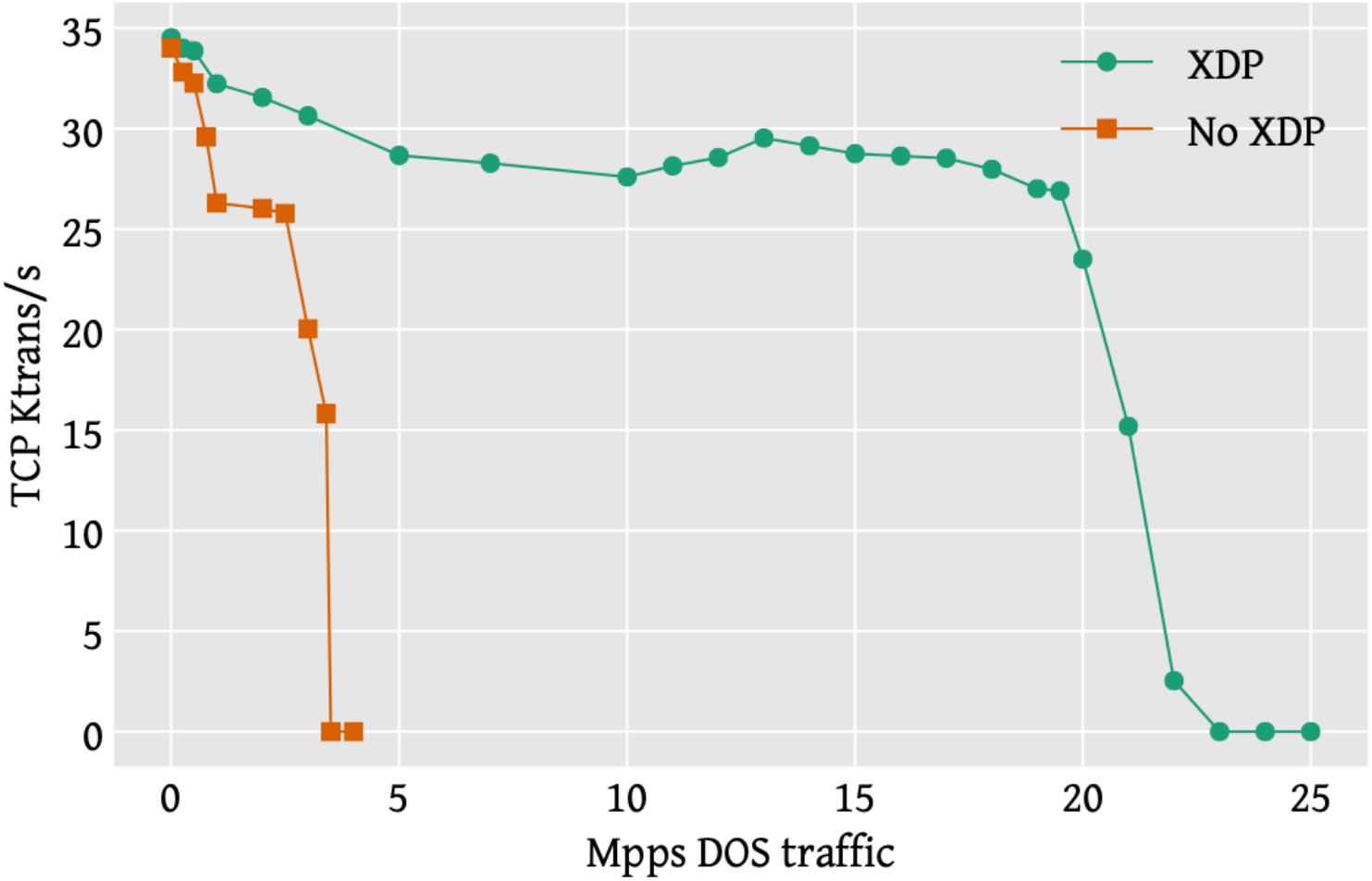


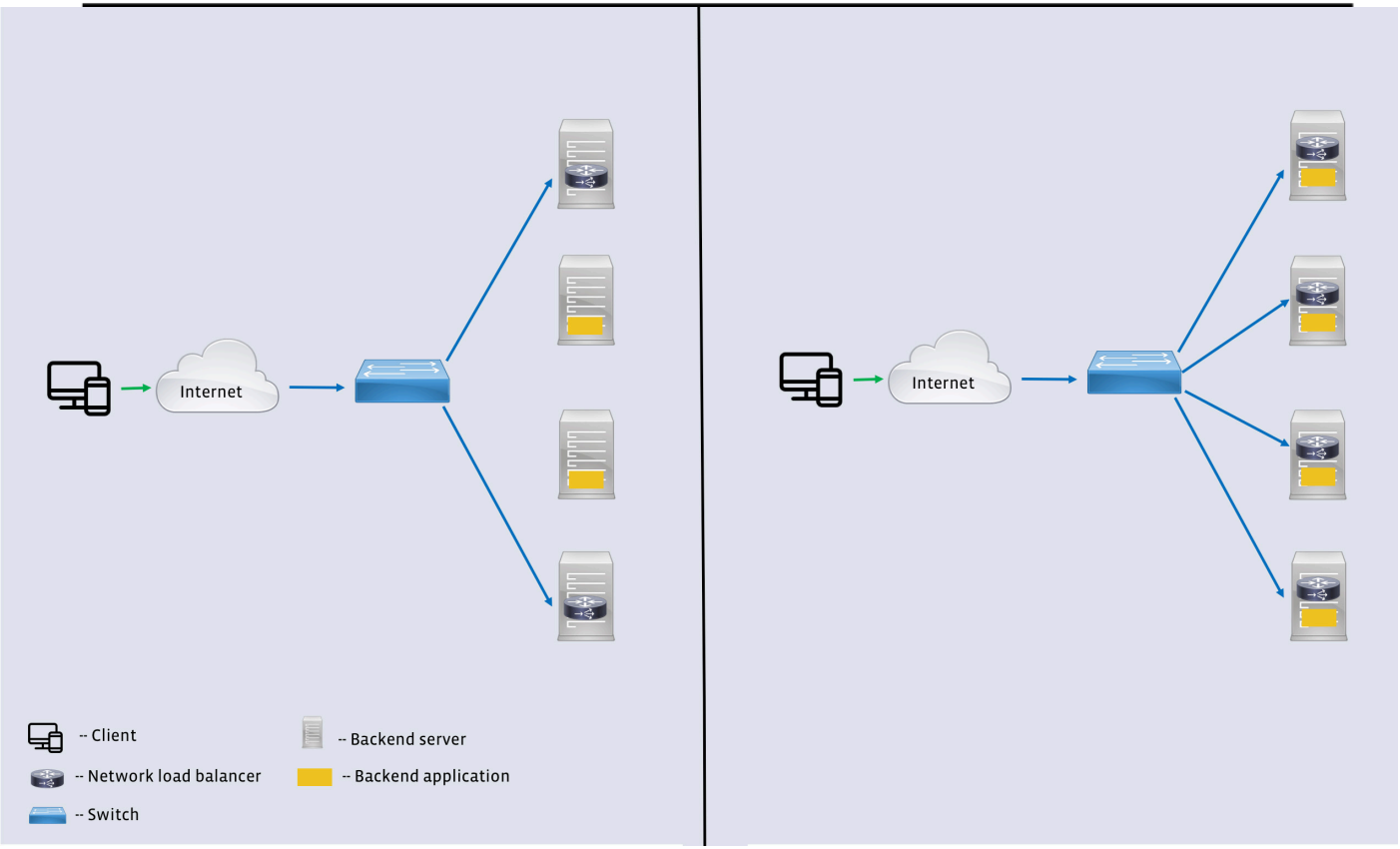
Figure 7: DDoS performance. Number of TCP transactions per second as the level of attack traffic directed at the server increases.

https://doi.org/10.1145/3281411.3281443

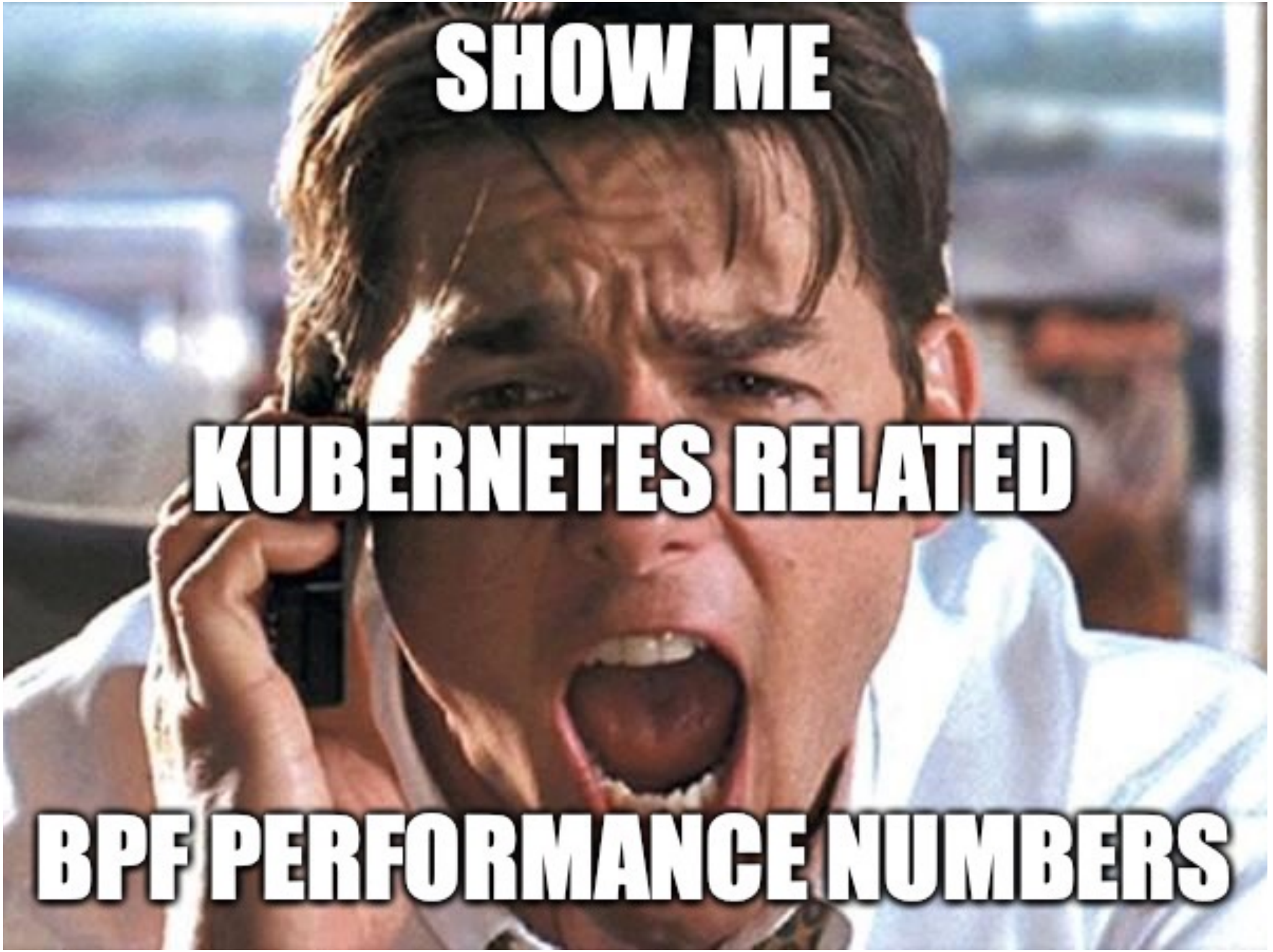
Facebook's Load Balancer Katran

Table 2: Load balancer performance (Mpps).

CPU Cores	1	2	3	4	5	6
XDP (Katran)	5.2	10.1	14.6	19.5	23.4	29.3
Linux (IPVS)	1.2	2.4	3.7	4.8	6.0	7.3



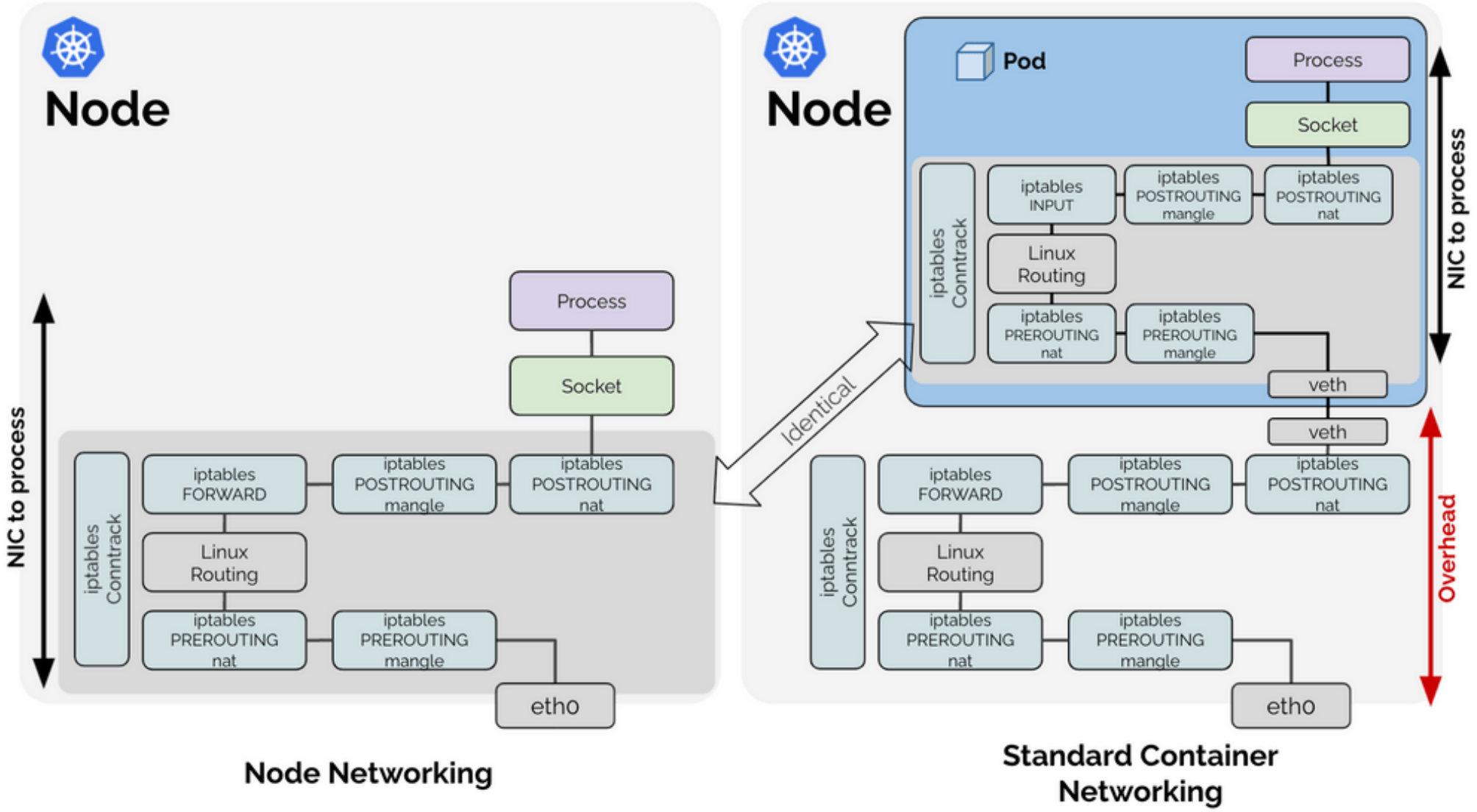
https://doi.org/10.1145/3281411.3281443, https://github.com/facebookincubator/katran


SHOW ME
KUBERNETES RELATED
BPF PERFORMANCE NUMBERS

Benefits of BPF in a K8s net plugin

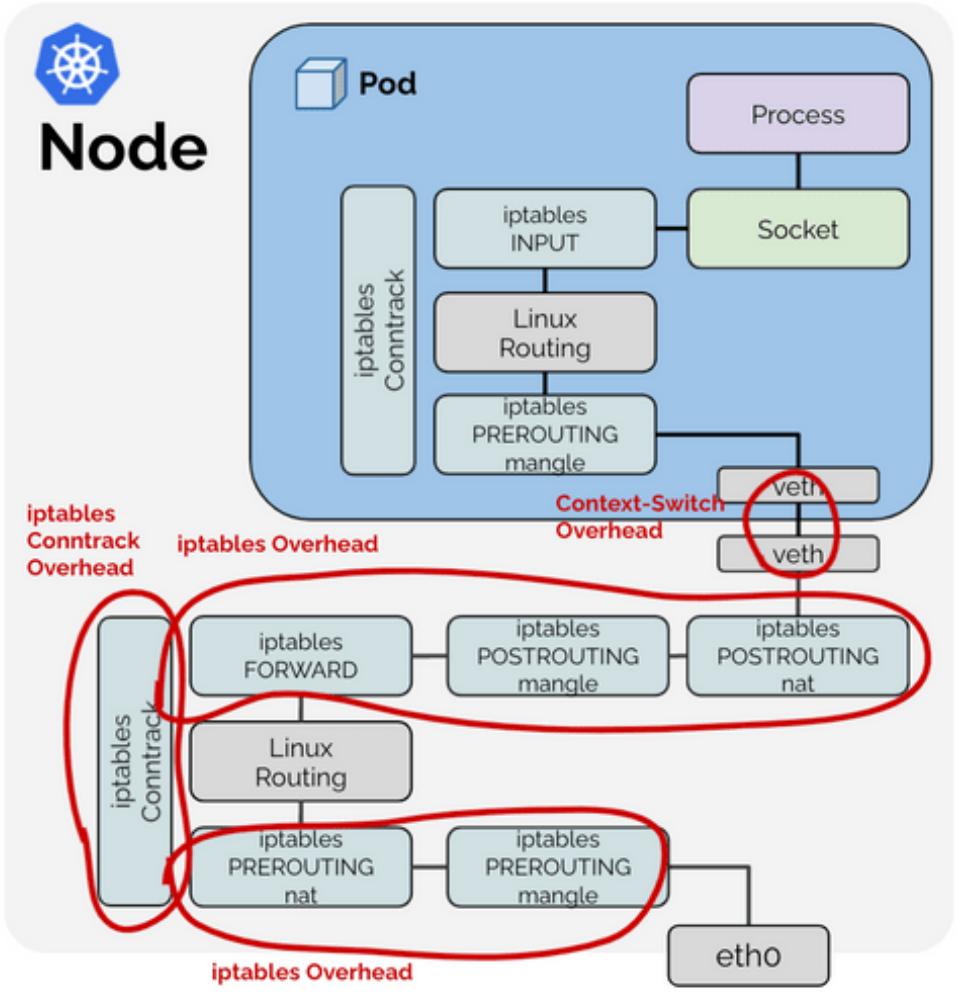
- **Cilium** - first k8s native BPF netw. plugin, most features, optimizations
- Calico – often as a default plugin; standard Linux kernel optimizations (ipset, ipvs mode, etc.)
- Why Calico switched to BPF, if they have solid "position"?
  - Higher throughput, less CPU/GBit
  - It has native support for Kubernetes services (enables **kube-proxyless** K8s)

Does Container Networking add an overhead?

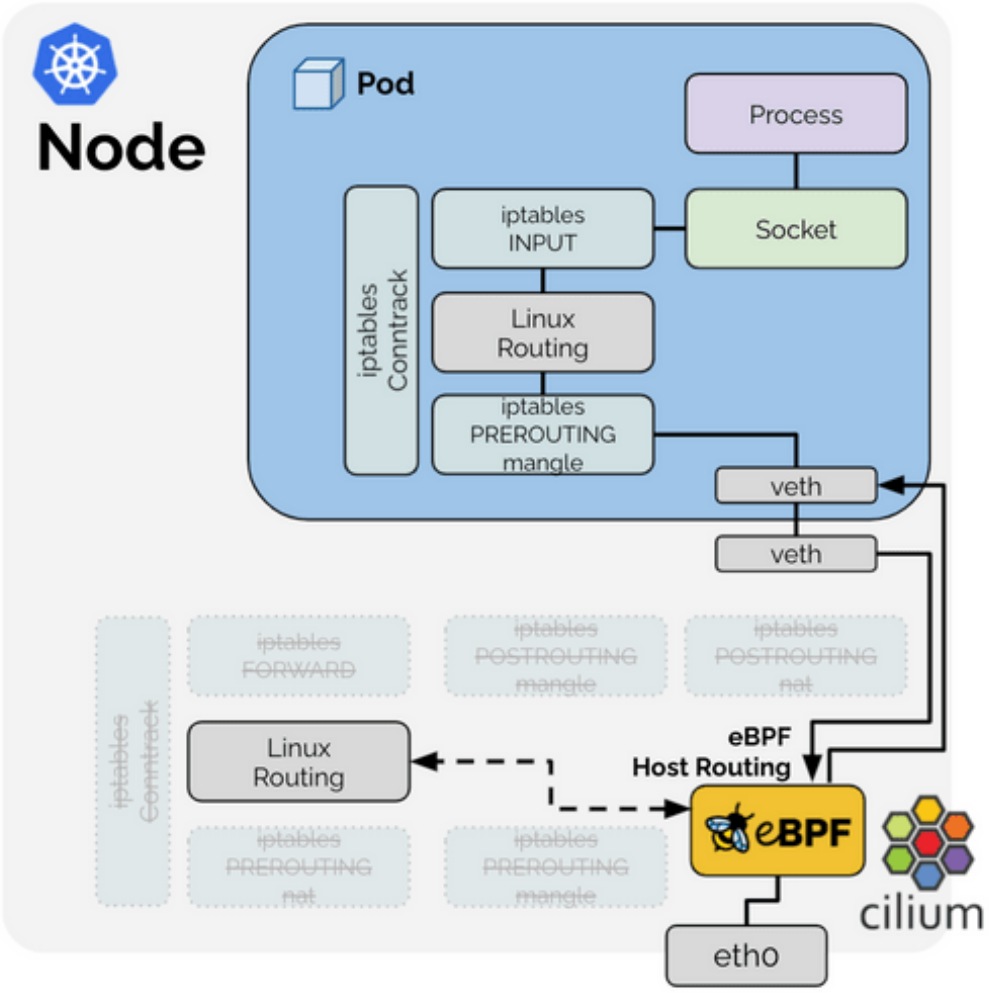


https://cilium.io/blog/2021/05/11/cni-benchmark#overhead

eBPF Host-Routing



Standard Container Networking

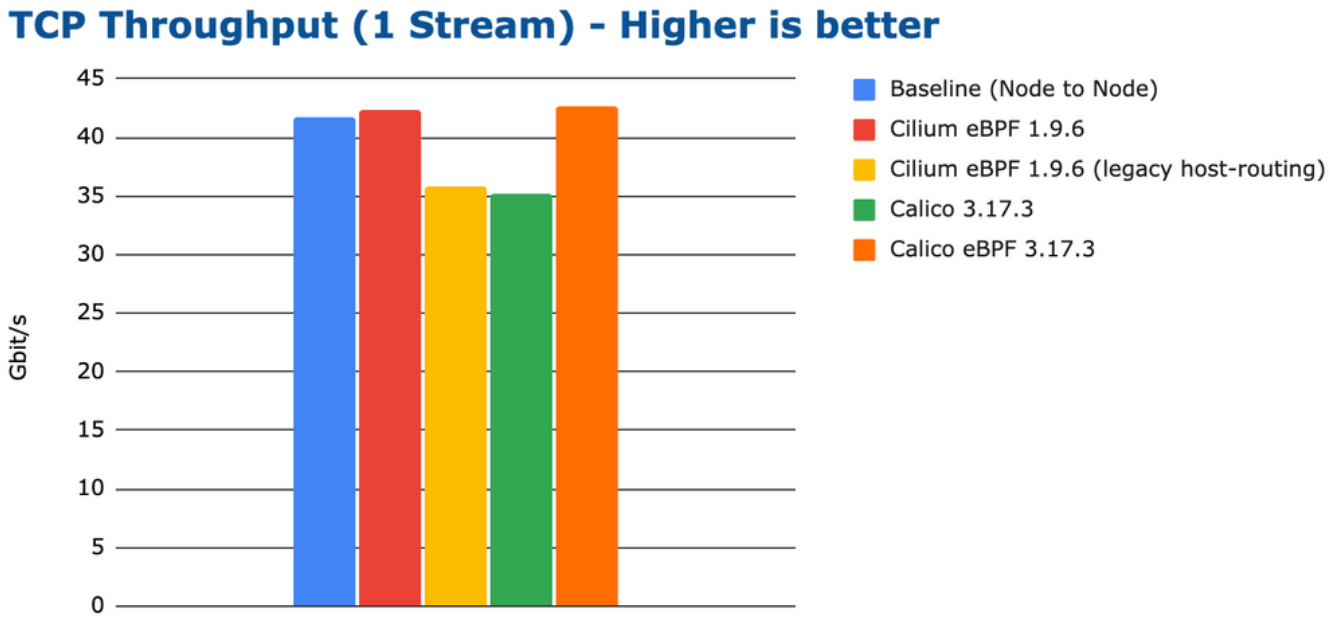


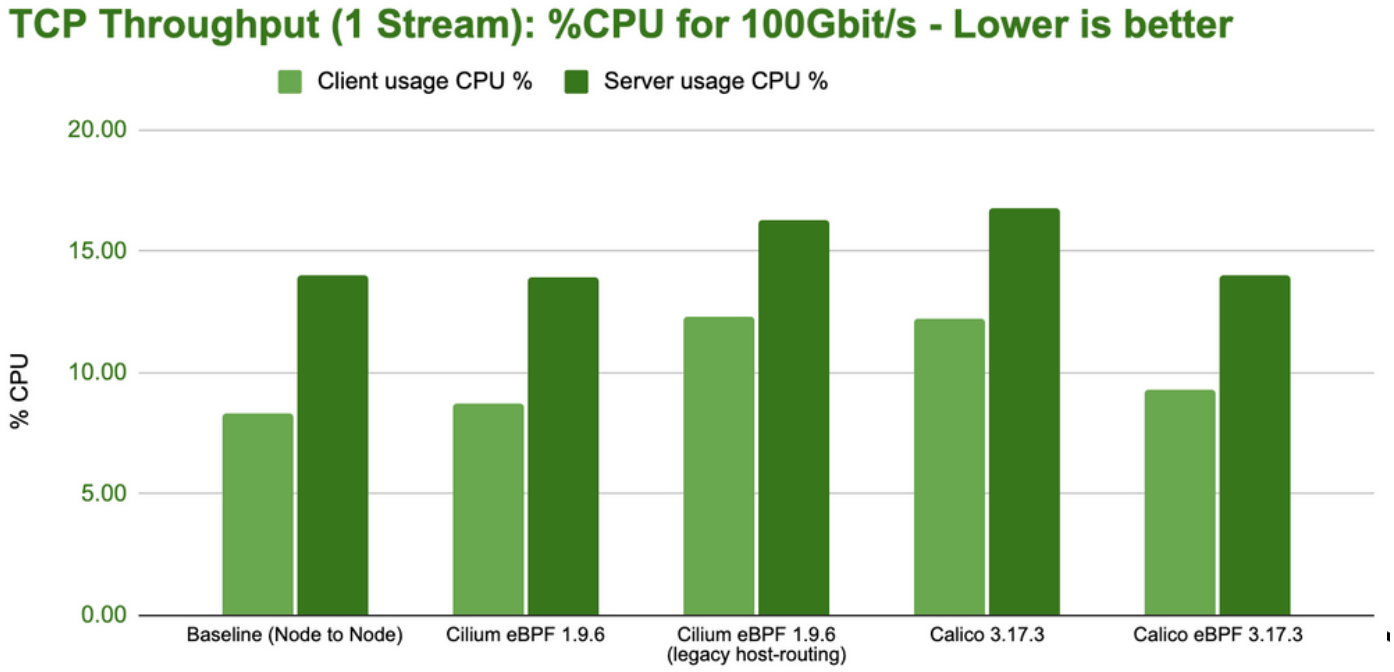
Cilium eBPF Container Networking

https://cilium.io/blog/2021/05/11/cni-benchmark#overhead

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Throughput 100G NIC (1 stream)

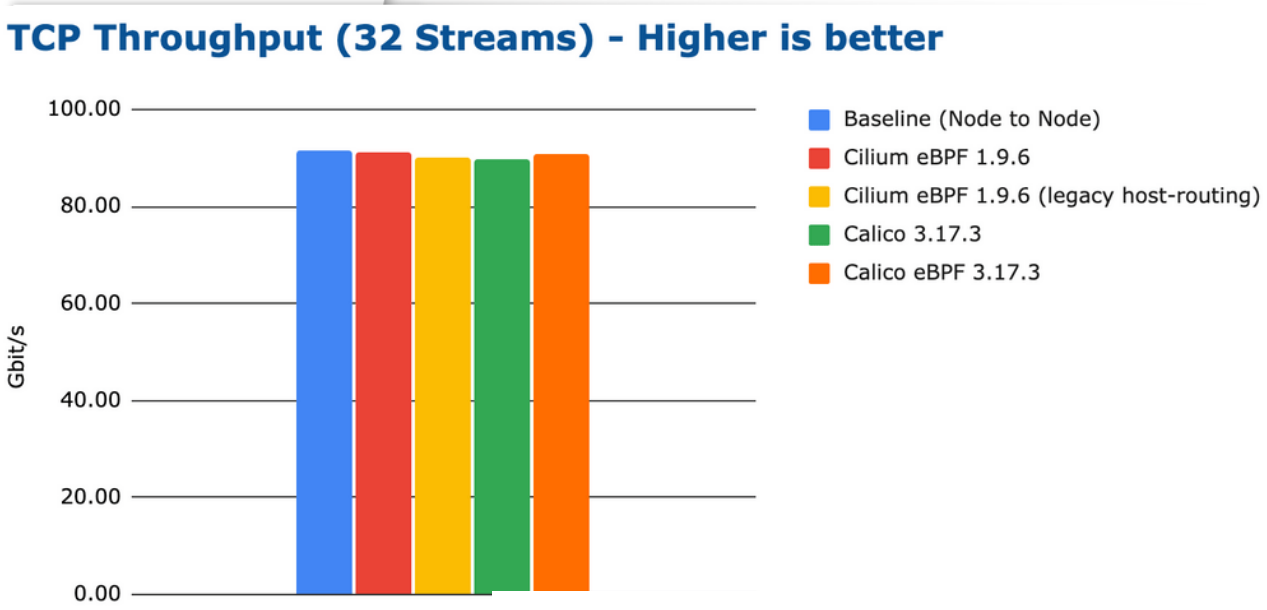


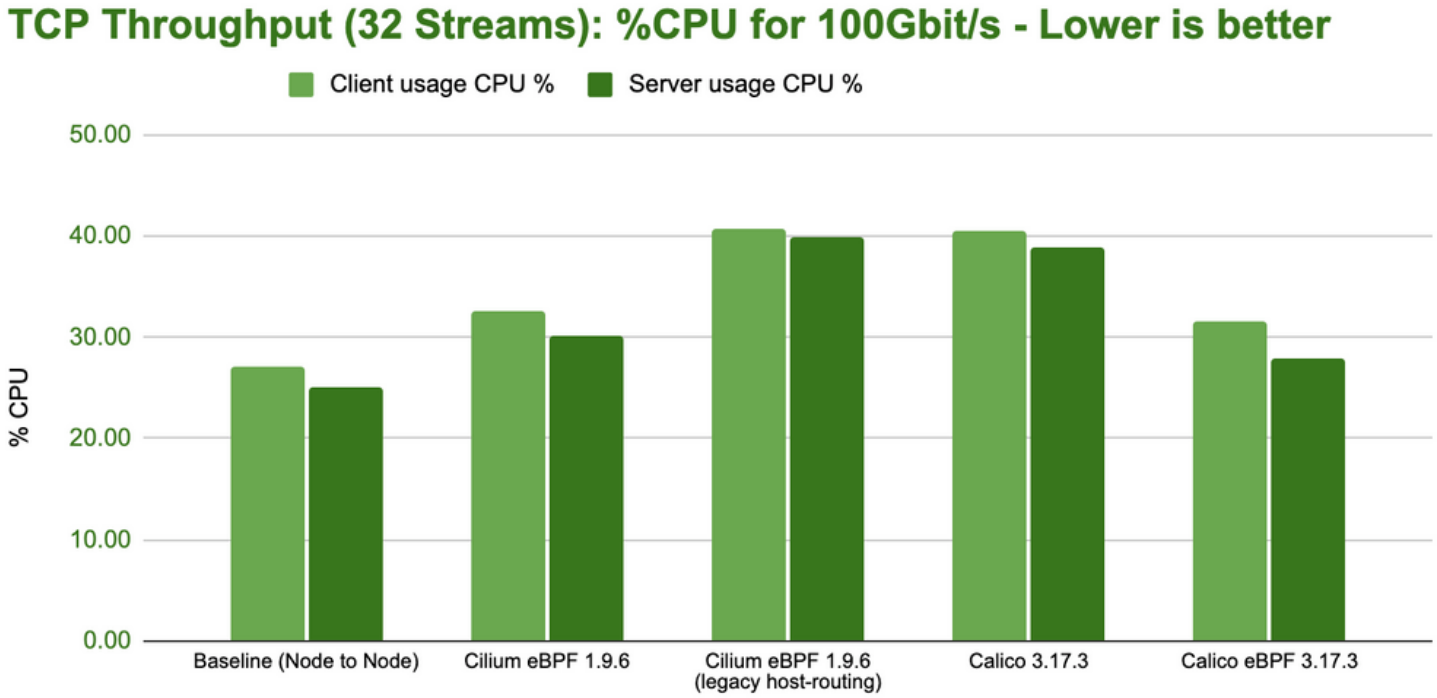


https://cilium.io/blog/2021/05/11/cni-benchl

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Throughput 100G NIC (32 streams on 32 core cpu)



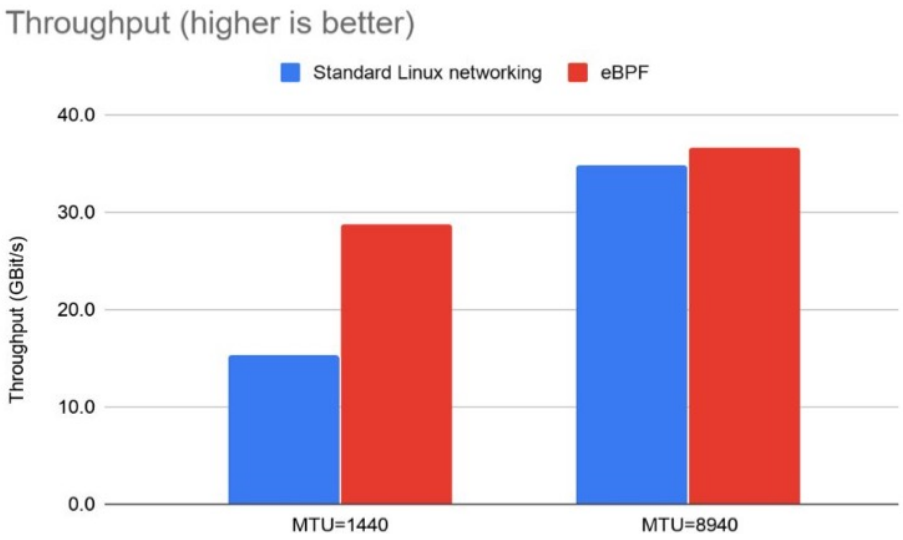


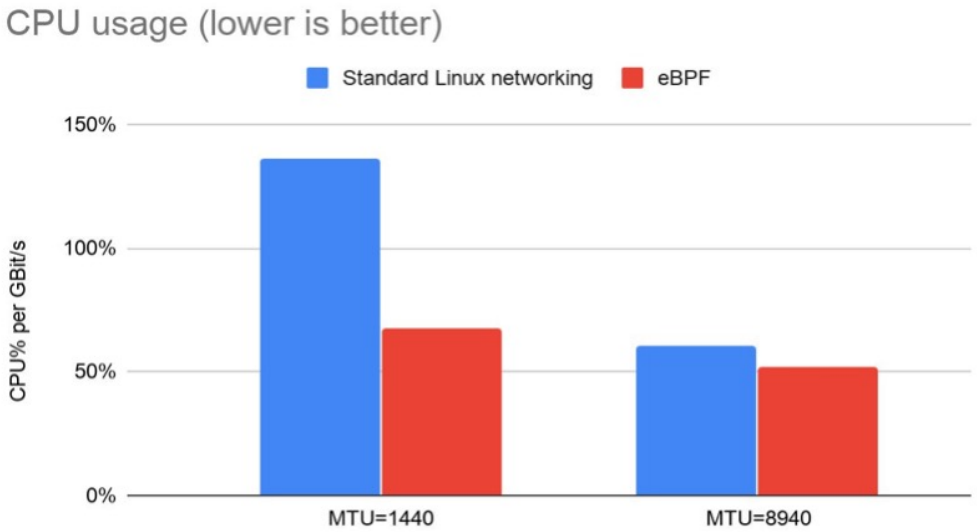
https://cilium.io/blog/2021/05/11/cni-b

Calico BPF dataplane

- Pod to pod throughput and CPU

40 Gbps network, running qperf in single pod





https://www.cncf.io/wp-content/uploads/2020/08/Calico-eBPF-Dataplane-CNCF-Webinar-Slides.pdf

Benefits of BPF in a K8s net plugin

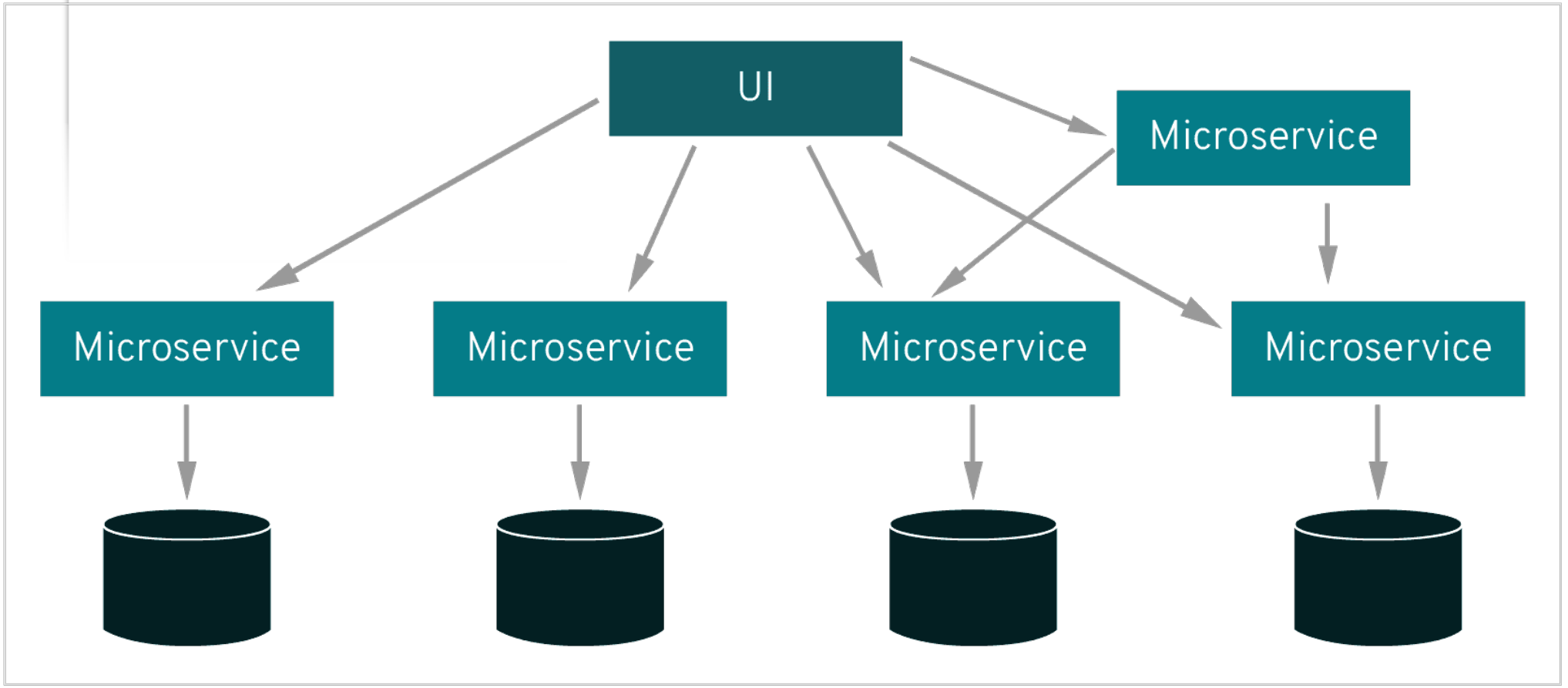
- Kube-proxyless Calico with BPF:
  - Smaller first packet latency to services
  - Preserves external client source IP addresses all the way to the pod
  - Supports DSR (Direct Server Return)
  - Uses less CPU than kube-proxy to keep the dataplane in sync

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko

Service mesh

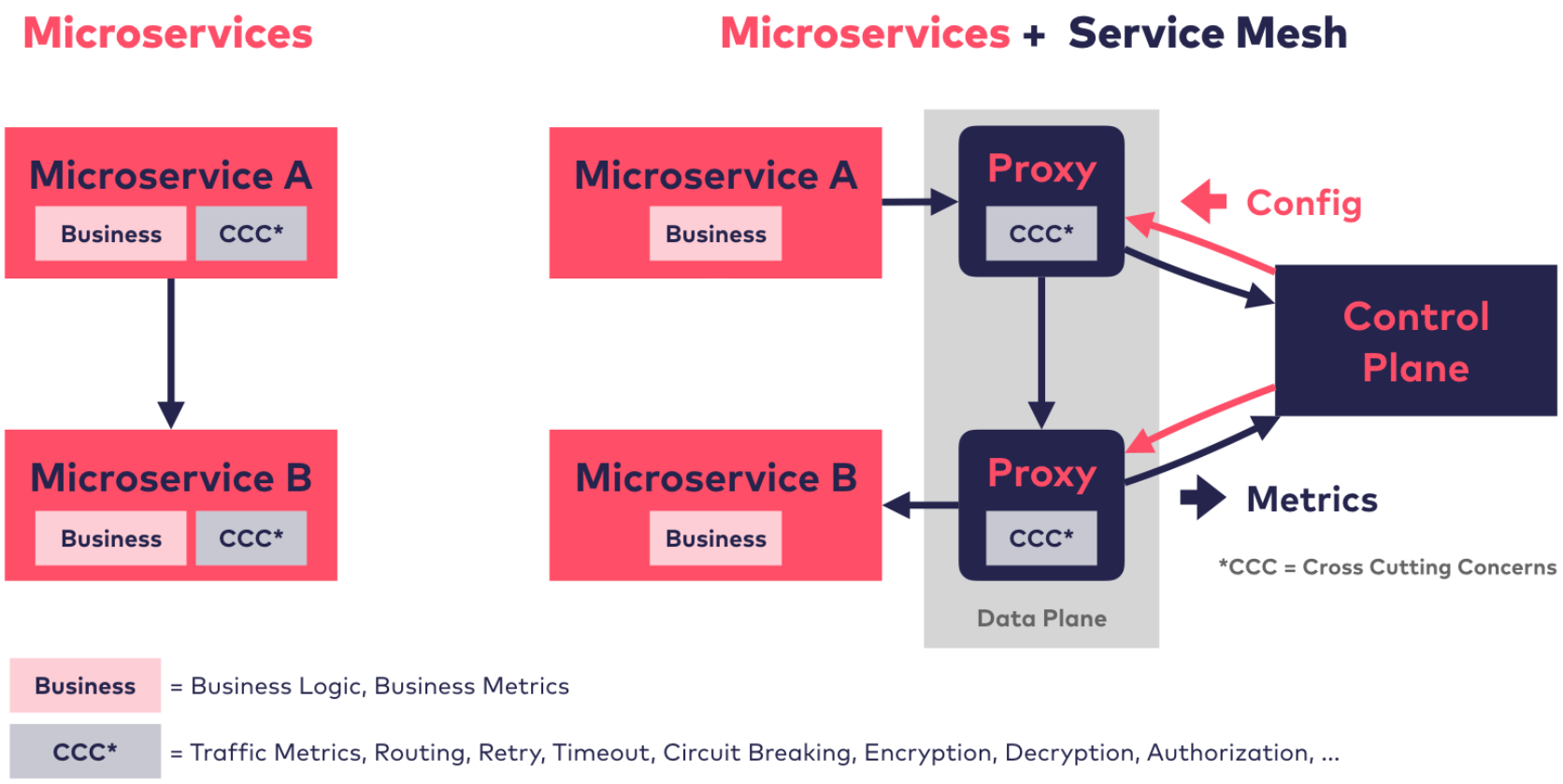
Traditional vs Cilium’s BPF implementation

Microservice architecture



https://www.redhat.com/en/topics/microservices/what-is-a-service-mesh

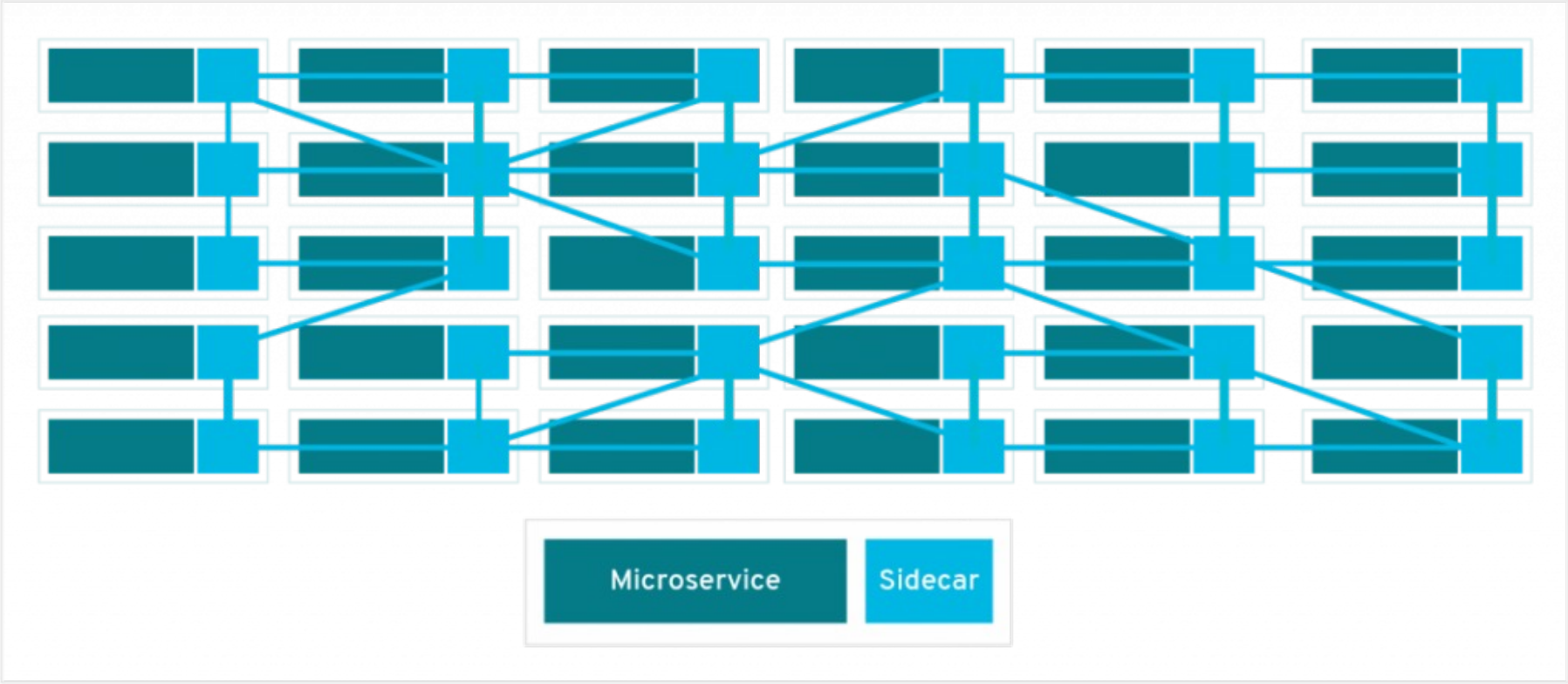
What is a service mesh



https://servicemesh.es

Service mesh

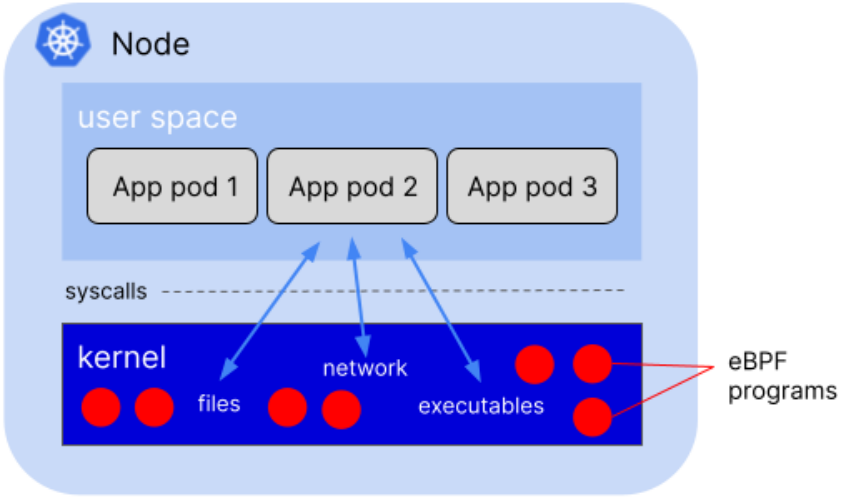
- Sidecar problem



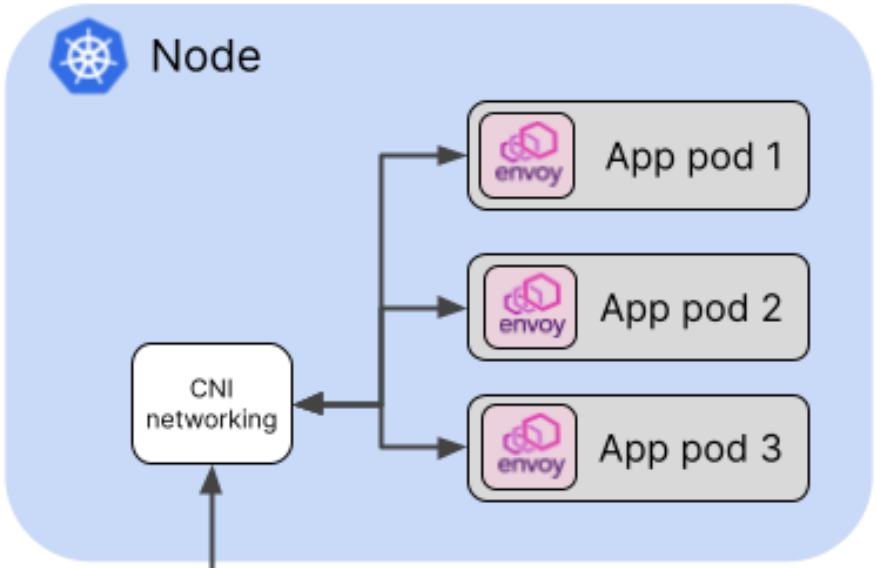
redhat.com/architect/why-when-service-mesh

K8s "sidecarless" service mesh w/ BPF

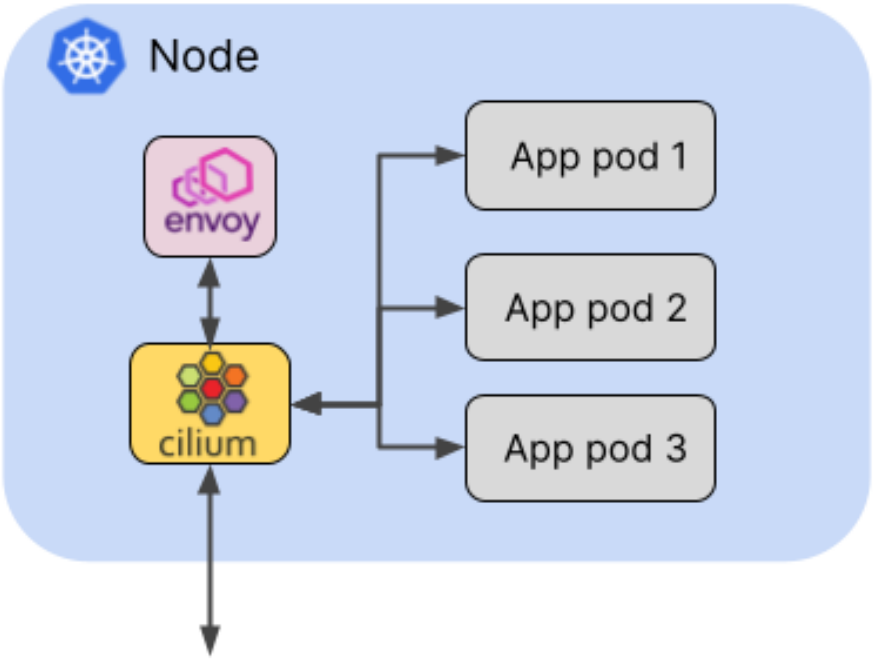
- Sidecar model needs labeling to inject a proxy (e.g. Istio)
- BPF: CRD is used cluster wide (can even apply for pre-existing pods)



Sidecar Proxy Model



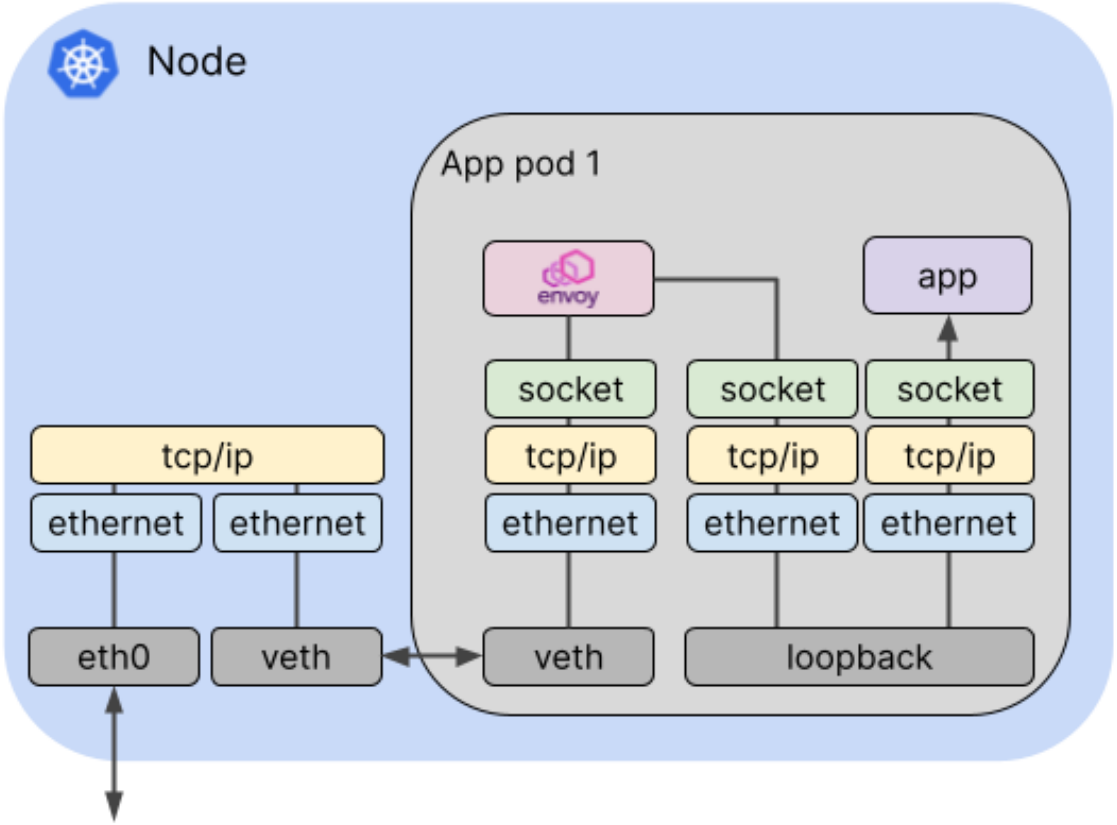
Sidecarless Proxy Model



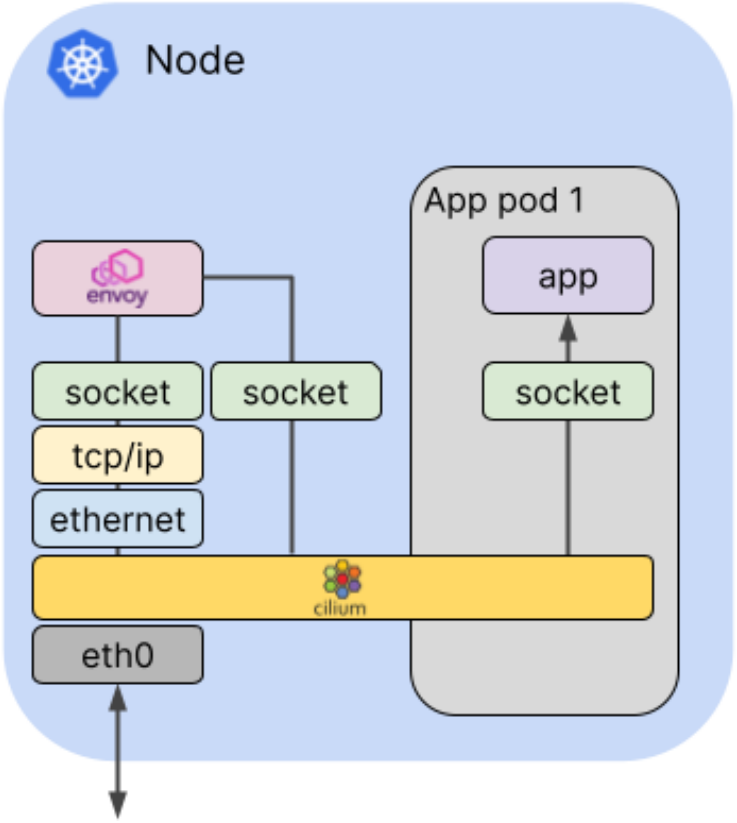
https://thenewstack.io/how-ebpf-streamlines-the-service-mesh/

"sidecarless" service mesh w/ BPF

Service mesh with traditional networking



Sidecarless model, eBPF acceleration



https://thenewstack.io/how-ebpf-streamlines-the-service-mesh/

Kernel versions for advanced features

Cilium Feature	Minimum Kernel Version
IPv4 fragment handling	>= 4.10
Restrictions on unique prefix lengths for CIDR policy rules	>= 4.11
IPsec Transparent Encryption in tunneling mode	>= 4.19
WireGuard Transparent Encryption	>= 5.6
Host-Reachable Services	>= 4.19.57, >= 5.1.16, >= 5.2
Kubernetes Without kube-proxy	>= 4.19.57, >= 5.1.16, >= 5.2
Bandwidth Manager (beta)	>= 5.1
Local Redirect Policy (beta)	>= 4.19.57, >= 5.1.16, >= 5.2
Full support for Session Affinity	>= 5.7
BPF-based proxy redirection	>= 5.7
BPF-based host routing	>= 5.10
Socket-level LB bypass in pod netns	>= 5.7
Egress Gateway (beta)	>= 5.2

https://docs.cilium.io/en/stable/operations/system_requirements/

TL;DL

- Take a look at BPF/XDP; test your use cases
- Performance is not the only advantage! (observability, security + security runtime enforcement, auditing, etc.)

- Check your K8s network plugin
  - hint: **Cilium;** Calico w/BPF; Multus (enables more than one network plugin)
  - Hosted vs on-premise
- BPF-accelerated Cilium Service mesh (beta!)

- K8s on the edge, ARM support (Cilium >=1.10)
- BPF powered malware?


I, FOR ONE, WELCOME OUR NEW
BPF
OVERLORDS
imgflip.com

Questions?

References

- D. Borkmann: BPF as a Fundamentally Better Dataplane, eBPF Summit 2020
- projectcalico.org
- cilium.io
- bpf.io
- T. Høiland-Jørgensen et al: The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel, CoNEXT 2018, ACM https://doi.org/10.1145/3281411.3281443
- Many others (attributed directly on slides)