Copyright And Confidentiality

牧云/罗伟
Network | IaaS | PaaS | ServiceMesh
交流 学习 沉淀 成长 分享
olaf.luo@foxmail.com
https://www.yuque.com/wei.luo

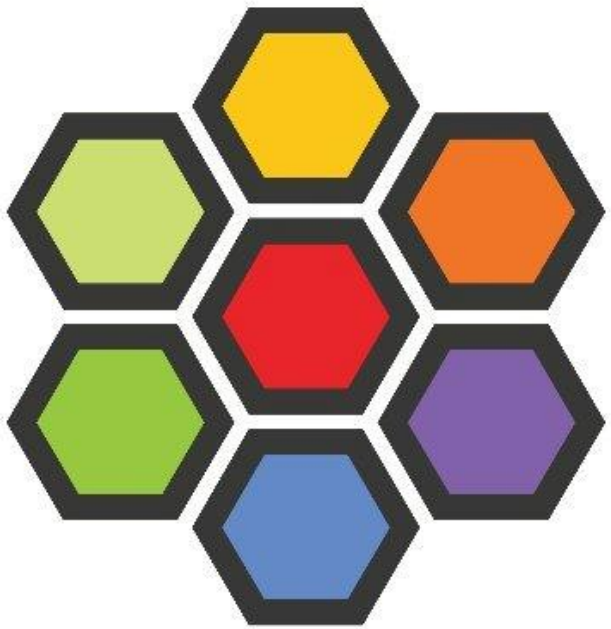
Rowan Luo

交流 学习 沉淀 成长 分享

Kubernetes CNI Cilium

Revision Date: 2021/12/21
Version：v1.2
DocID：CN00002021XLW

Kubernetes CNI - Cilium



https://cilium.io/
https://github.com/cilium

Kubernetes CNI - Cilium With eBPF

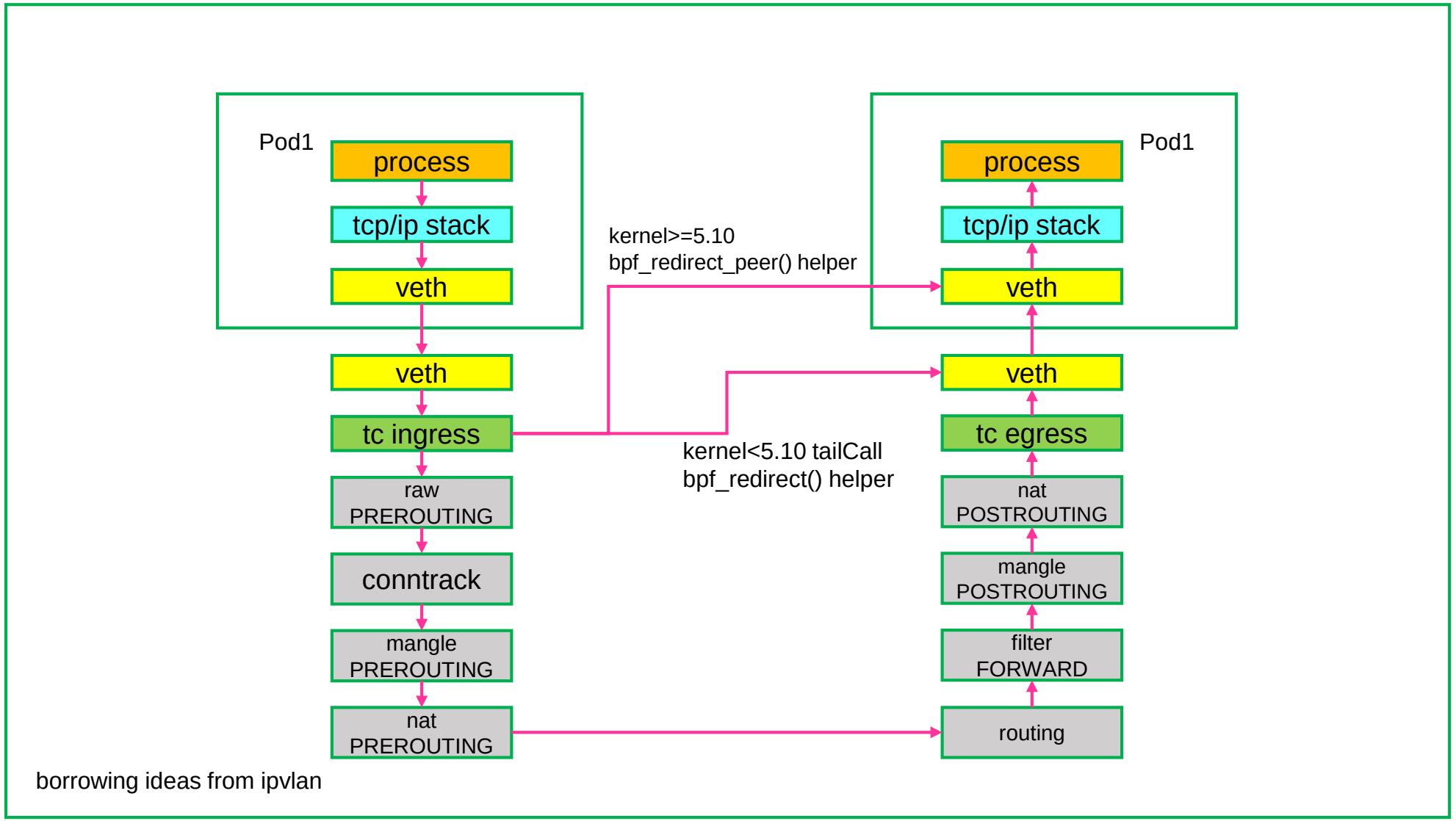
eBPF is a revolutionary technology with origins in the Linux kernel that can run sandboxed programs in an operating system kernel. It is used to safely and efficiently extend the capabilities of the kernel without requiring to change kernel source code or load kernel modules.

Historically, the operating system has always been an ideal place to implement observability, security, and networking functionality due to the kernel's privileged ability to oversee and control the entire system. At the same time, an operating system kernel is hard to evolve due to its central role and high requirement towards stability and security. The rate of innovation at the operating system level has thus traditionally been lower compared to functionality implemented outside of the operating system.

eBPF changes this formula fundamentally. By allowing to run sandboxed programs within the operating system, application developers can run eBPF programs to add additional capabilities to the operating system at runtime. The operating system then guarantees safety and execution efficiency as if natively compiled with the aid of a Just-In-Time (JIT) compiler and verification engine. This has led to a wave of eBPF-based projects covering a wide array of use cases, including next-generation networking, observability, and security functionality.

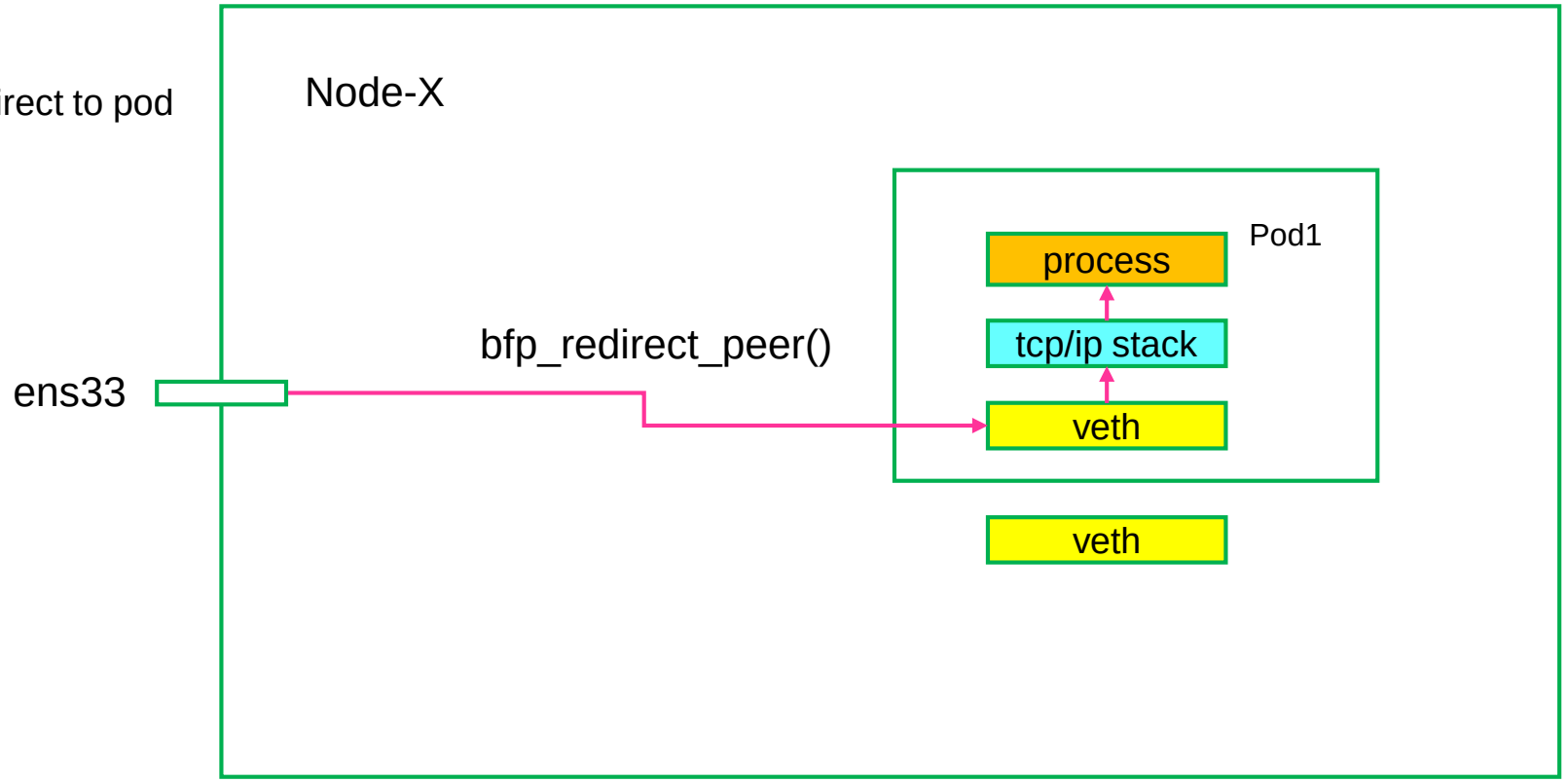
Today, eBPF is used extensively to drive a wide variety of use cases: Providing high-performance networking and load-balancing in modern data centers and cloud native environments, extracting fine-grained security observability data at low overhead, helping application developers trace applications, providing insights for performance troubleshooting, preventive application and container runtime security enforcement, and much more. The possibilities are endless, and the innovation that eBPF is unlocked has only just begun.

Kubernetes CNI - Cilium DataPath



Kubernetes CNI - Cilium DataPath - bfp_redirect_peer()

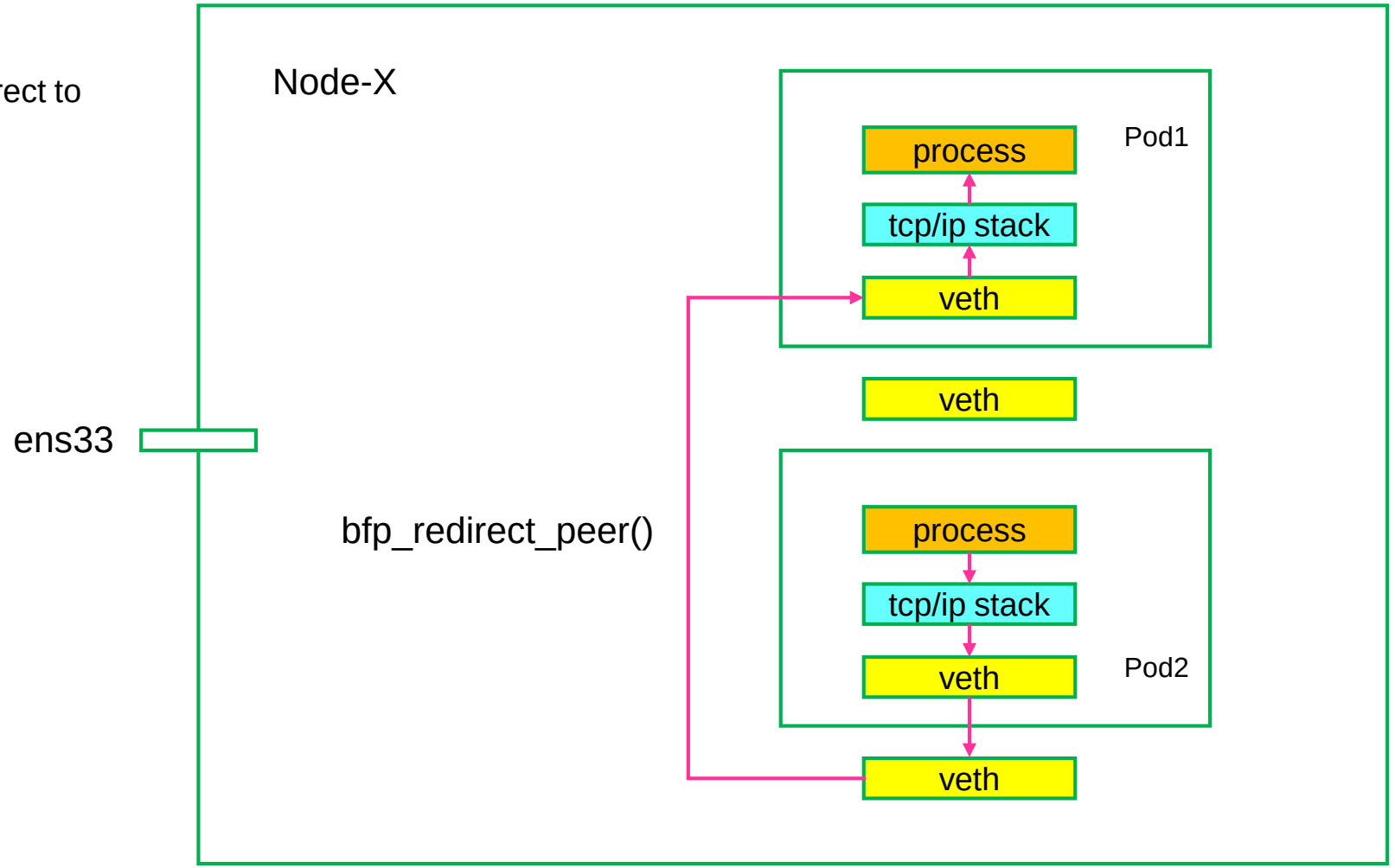
①: ROOT namespace redirect to pod namespace



The bpf_redirect_peer() enables switching network namespaces from the ingress of the NIC to the ingress of the Pod without a software interrupt rescheduling point when traversing the network namespace. The physical NIC can thus push packets up the stack into the application's socket residing in a different Pod namespace in one go. This also leads to quicker application wake-up for picking up the received data. Similarly, rescheduling points are reduced from 2 to 1 for local Pod-to-Pod communication resulting in better latency there as well.

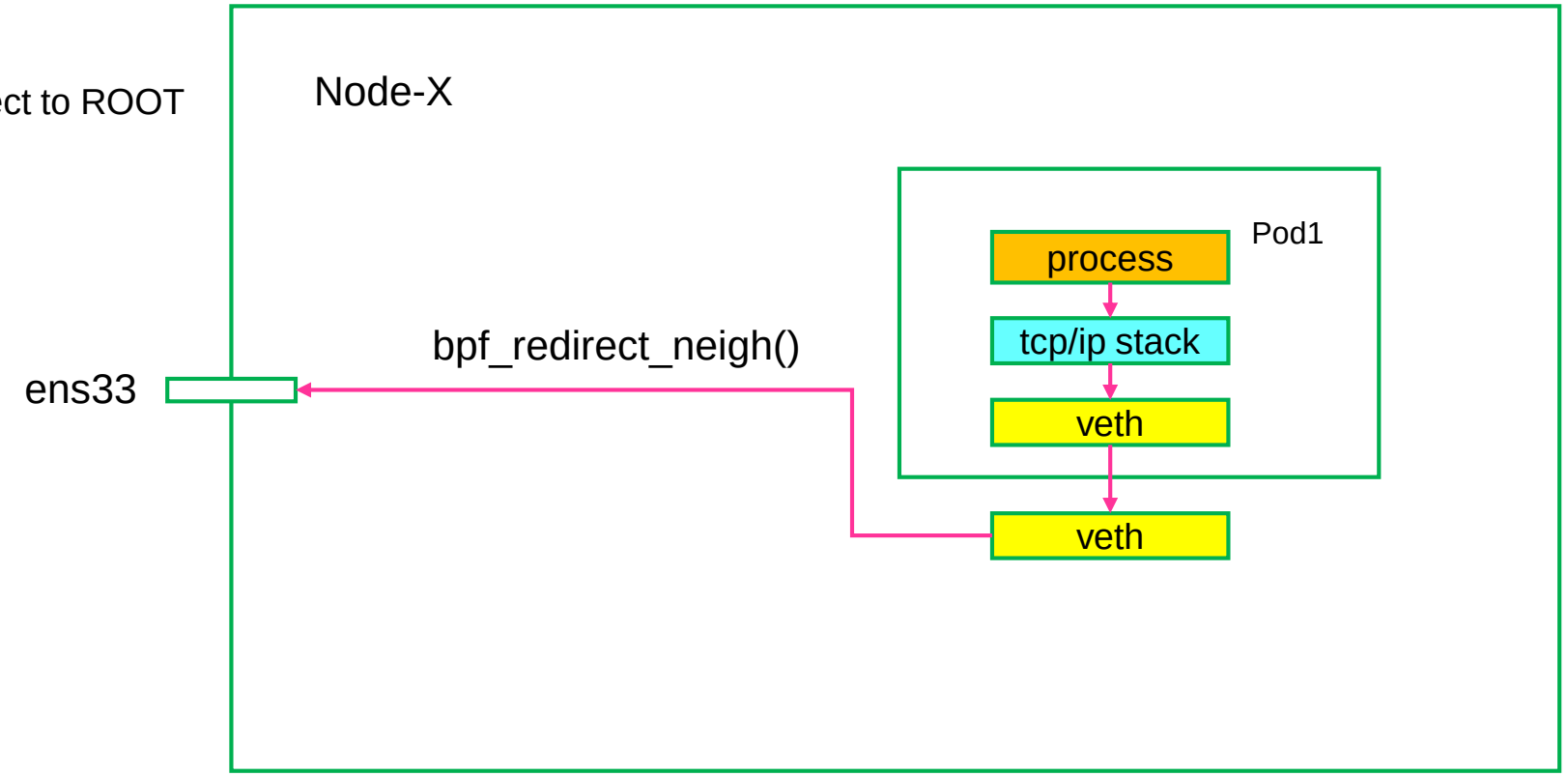
Kubernetes CNI - Cilium DataPath - bfp_redirect_peer()

②: pod namespace 1 redirect to
pod namespace 2



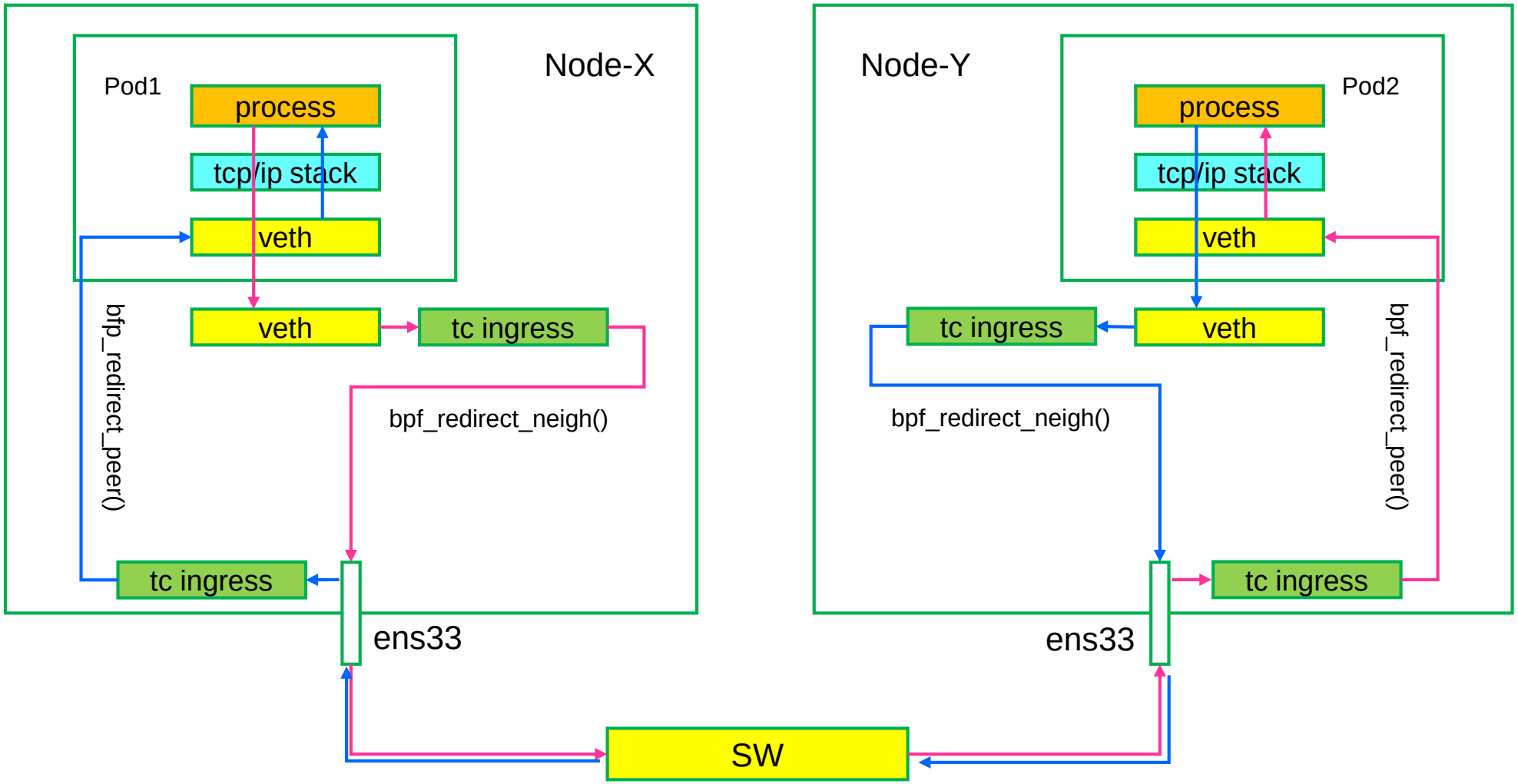
Kubernetes CNI - Cilium DataPath - bpf_redirect_neigh()

③: pod namespace redirect to ROOT namespace

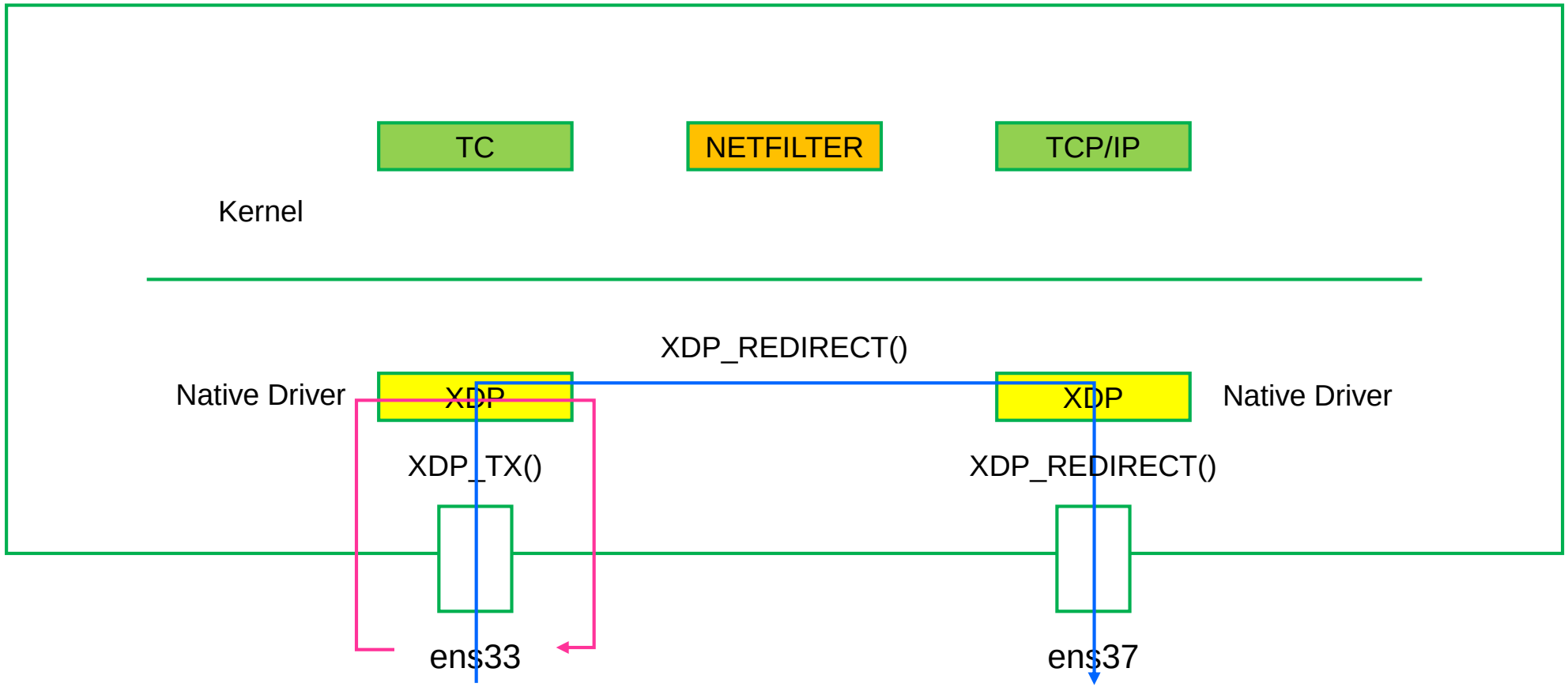


The bpf_redirect_neigh() handles a Pod's egress traffic by injecting the traffic into the Linux kernel's neighboring subsystem, allowing to find the next hop and resolving layer 2 addresses for the network packet. Performing the forwarding only in tc eBPF layer and not pushing the packet further up the networking stack also provides proper back pressure for the TCP stack and feedback for TCP's TSQ (TCP Small Queues) mechanism to reduce potential excessive queueing of TCP packets. That is, feedback is given to the TCP stack that the packet has left the node instead of inaccurately providing it too early when it would be pushed up to the host stack for routing. This is now possible because the packet's socket association can be kept intact when it is passed down into the NIC driver.

Kubernetes CNI - Cilium DataPath[Pod To Pod]



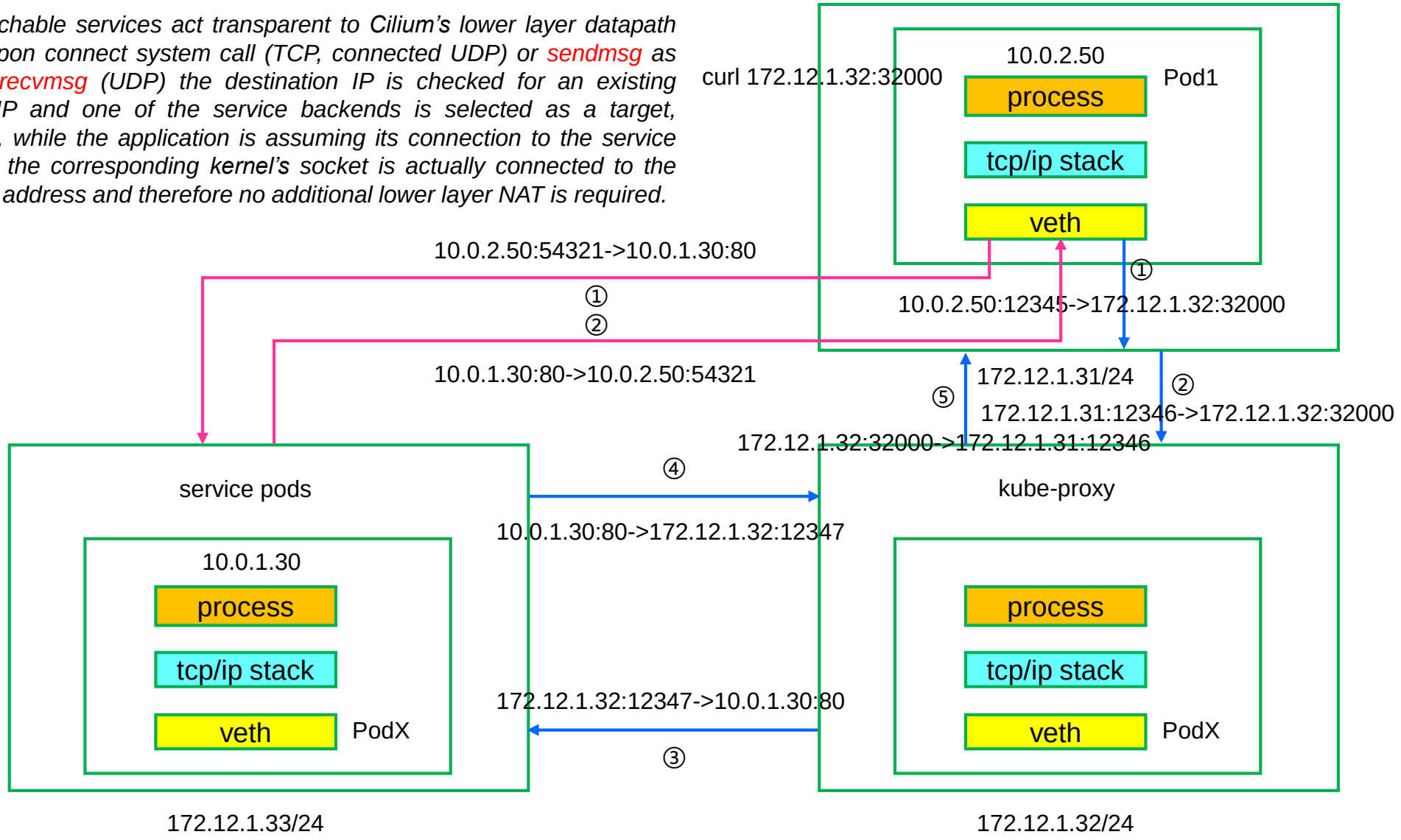
Kubernetes CNI - Cilium DataPath XDP(xdpdrv)



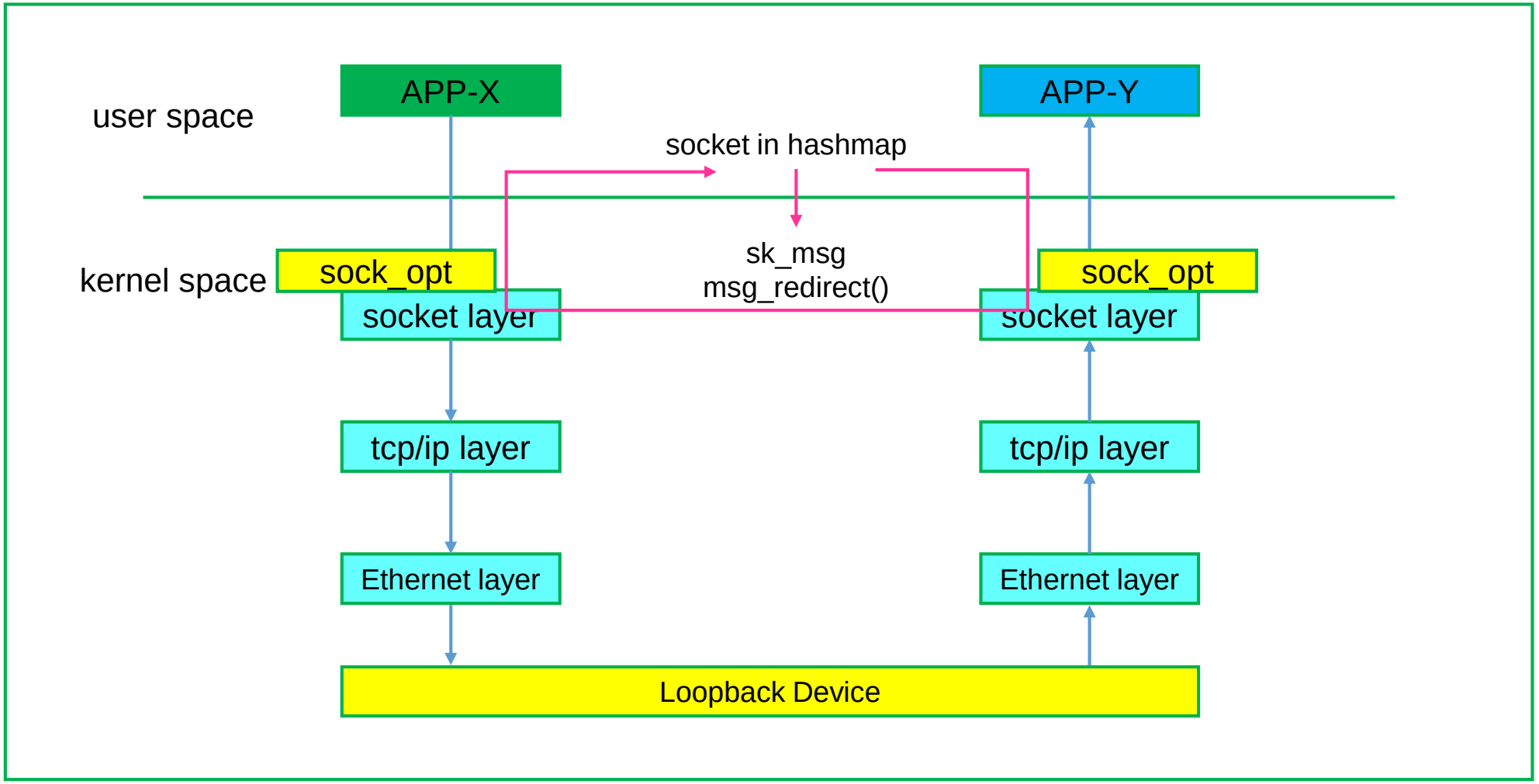
With XDP_TX the BPF program has an efficient option to transmit the network packet out of the same NIC it just arrived on again. This is typically useful when few nodes are implementing, for example, firewalling with subsequent load balancing in a cluster and thus act as a hairpinned load balancer pushing the incoming packets back into the switch after rewriting them in XDP BPF.
XDP_REDIRECT is similar to XDP_TX in that it is able to transmit the XDP packet, but through another NIC. Another option for the XDP_REDIRECT case is to redirect into a BPF cpumap, meaning, the CPUs serving XDP on the NIC's receive queues can continue to do so and push the packet for processing the upper kernel stack to a remote CPU.https://prototype-kernel.readthedocs.io/en/latest/index.html#

Kubernetes CNI - Cilium DataPath Host-Reachable Services

Host-reachable services act transparent to Cilium's lower layer datapath in that upon connect system call (TCP, connected UDP) or sendmsg as well as recvmsg (UDP) the destination IP is checked for an existing service IP and one of the service backends is selected as a target, meaning, while the application is assuming its connection to the service address, the corresponding kernel's socket is actually connected to the backend address and therefore no additional lower layer NAT is required.



Kubernetes CNI - Cilium DataPath 2+ Container in Same Pod



https://docs.cilium.io/en/latest/bpf/
https://01.org/blogs/xuyizhou/2021/accelerate-istio-dataplane-ebpf-part-1

“交流，学习，成长，提升”

Burly Luo 的赞赏码

Copyright And Confidentiality

牧云/罗伟
Network | IaaS | PaaS | ServiceMesh
交流 学习 沉淀 成长 分享
olaf.luo@foxmail.com
https://www.yuque.com/wei.luo