
KubeCon
CloudNativeCon
North America 2023

Turning up Performance to 11: Cilium, NetKit Devices, and Going Big with TCP

Daniel Borkmann (Isovalent)

KubeCon North America 2023

Experiment for this talk: What would it take to turn the network performance knob to 11?

... and why is it relevant in the first place?

Scale:

- Migrating more workloads to K8s envs
- Connecting multiple clusters in a mesh

Sustainability:

- Better utilization of the existing infrastructure
- Reduction of off/on-prem costs

Performance:

- Better RPC workload latencies
- Escalating bulk data demands from AI/ML

Home > Data Center

Cisco lays groundwork for 800G networks as AI, 5G and video traffic demands grow

Cisco brings an 800Gbps line card and better packet management to Silicon One-based 8000 Series routers as AI, 5G and video traffic demands grow.

By Michael Cooney

Senior Editor, Network World | APR 5, 2023 9:39 AM PDT

Home > Data Center

Cisco lays groundwork for 800G networks as AI, 5G and video traffic demands grow

Cisco brings an 800Gbps line card and better packet management to Silicon One-based 8000 Series routers as AI, 5G and video traffic demands grow.

By Michael Cooney
Senior Editor, Network World | APR 5, 2023

sues Jobs

VentureBeat

Security | Data Infrastructure | Automation | Enterprise

AI and ML: The new frontier for data center innovation and optimization

Home > Data Center

Cisco lays groundwork for 800G networks as AI, 5G and video traffic demands grow

Cisco brings an 800Gbps line card and better packet management to Silicon One-based 8000 Series routers as AI, 5G and video traffic demands grow.

By Michael Cooney
Senior Editor, Network World | APR 5, 2023

sues Jobs

VentureBeat

Security | Data Infrastructure | Automation | Enterprise

AI and ML: The new frontier for data center innovation and optimization

The Register

Hyperscale datacenter capacity set to triple because of AI demand

And it's going to suck... up more power too

Dan Robinson

Wed 18 Oct 2023 // 16:45 UTC

Total capacity of hyperscale datacenters is set to grow almost threefold over the next six years on the back of AI demand, substantially increasing the amount of power required by those facilities.

Home > Data Center

Cisco lays groundwork for 800G networks as AI, 5G and video traffic demands grow

Cisco brings an 800Gbps line card and better packet management to Silicon
8000 Series routers as AI, 5G and video traffic demands grow.

By Michael Cooney
Senior Editor, Network World | APR 5, 2023

sues Jobs

VentureBeat

Security | Data Infrastructure | Aut

AI and ML: The new fron
data center innovation a
optimization

TECHNOLOGIES > ANALOG

Ethernet Switch Silicon Doubles Bandwidth to 51.2 Tb/s

Aug. 19, 2022

Broadcom's Tomahawk 5 switch ASIC packs 512 high-performance 100G PAM4 SerDes transceivers.

James Morra

Related To: Electronic Design

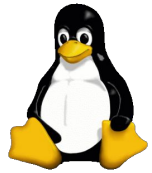
Hyperscale datacenter capacity set to triple because of AI demand

And it's going to suck... up more power too

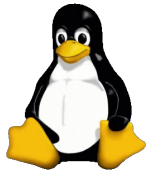
Dan Robinson

Wed 18 Oct 2023 // 16:45 UTC

Total capacity of hyperscale datacenters is set to grow almost threefold over the next six years on the back of AI demand, substantially increasing the amount of power required by those facilities.

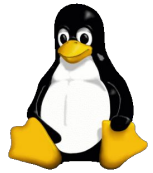
How does a network platform for K8s look like which would address future demands and how much can we benefit from it today?*



How does a network platform for K8s look like which would address future demands and how much can we benefit from it today?*

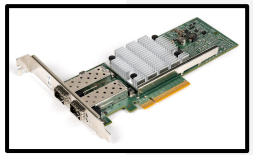
* without rewriting existing applications

Standard Datapath Architecture:

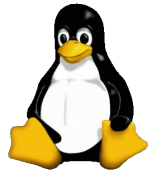


Host

- kubelet
- kube-proxy

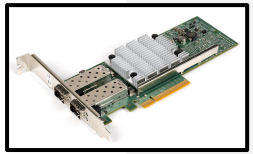


Standard Datapath Architecture:



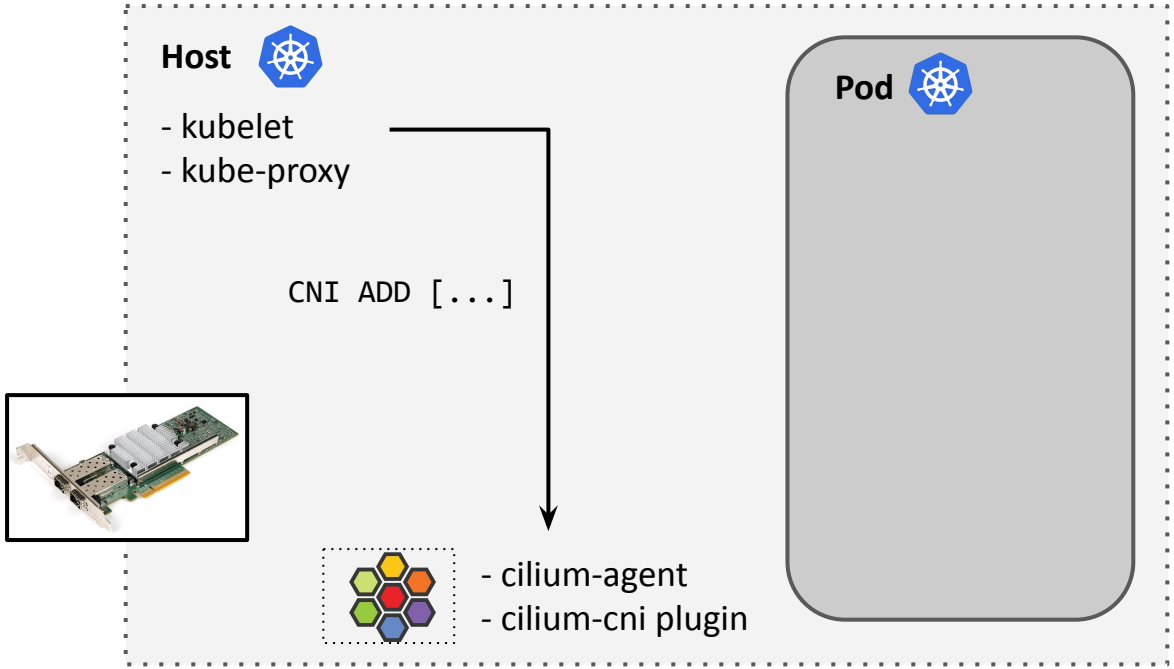
Host

- kubelet
- kube-proxy

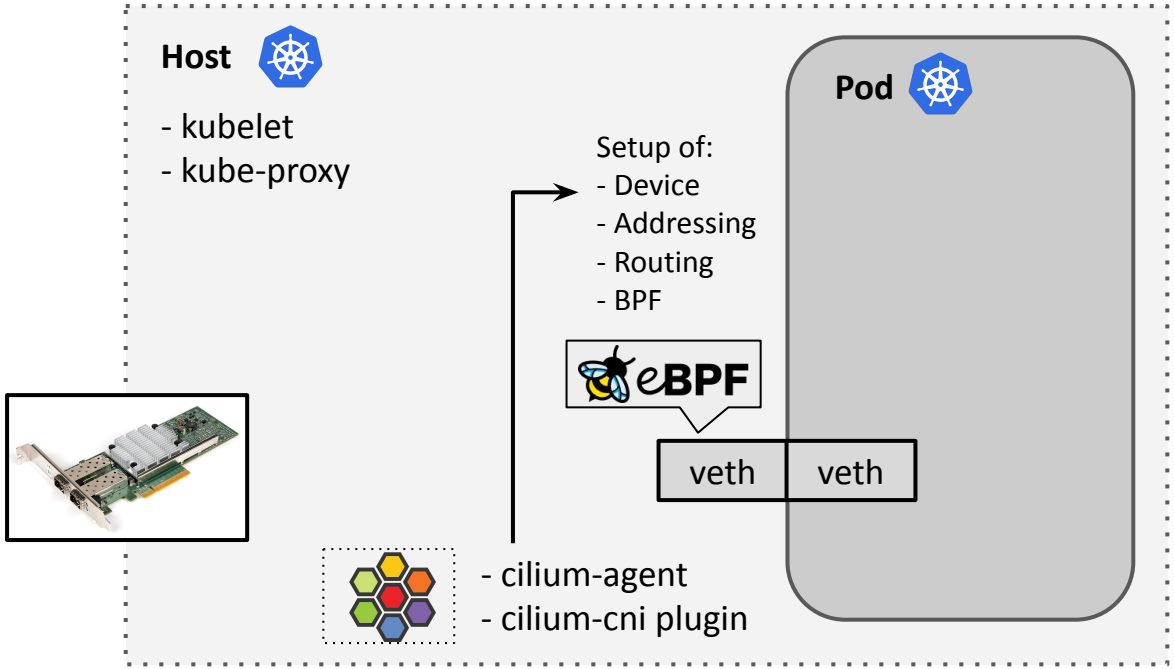


- cilium-agent
- cilium-cni plugin

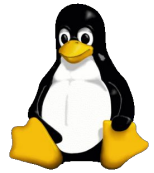
Standard Datapath Architecture:

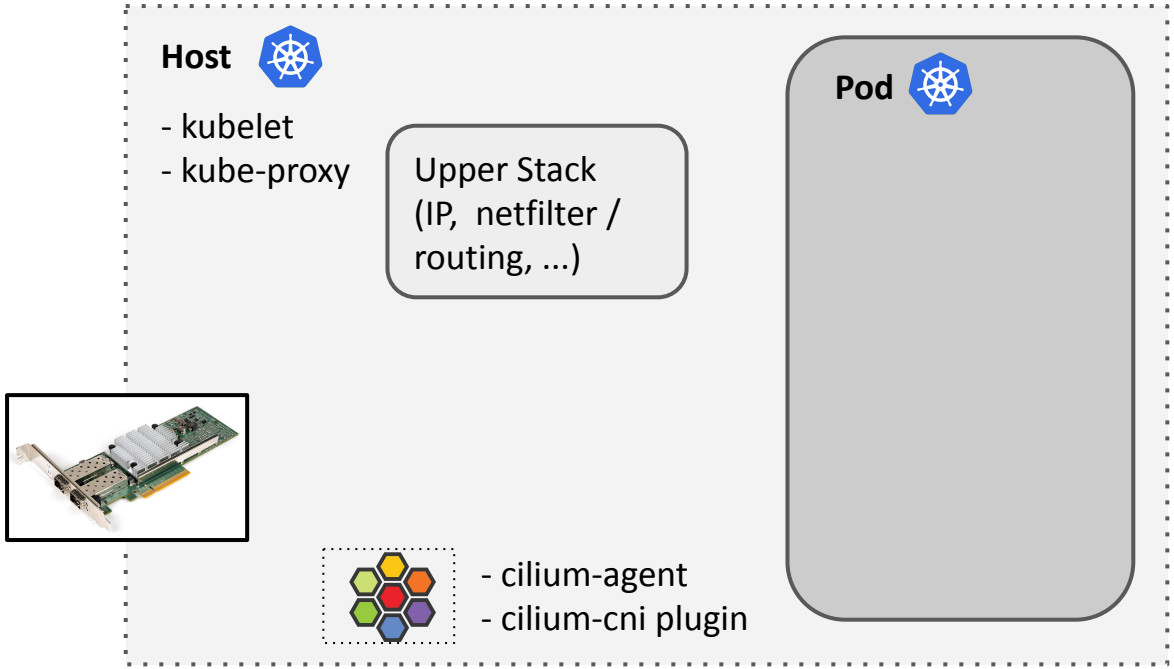


Standard Datapath Architecture:

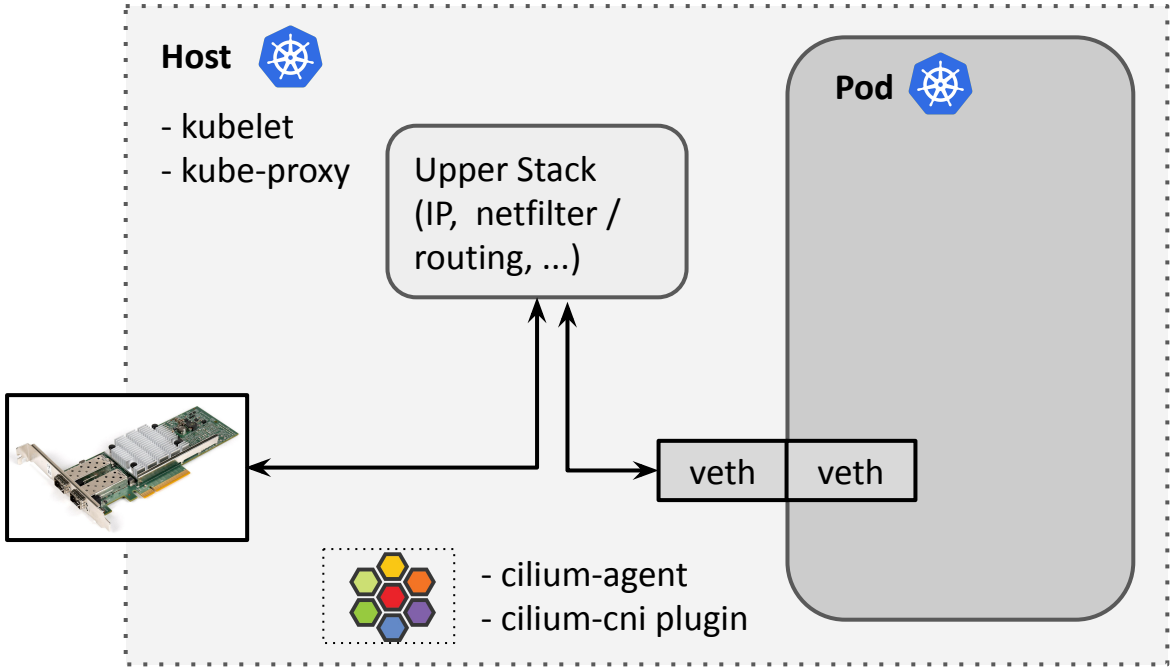


Standard Datapath Architecture:

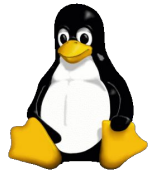




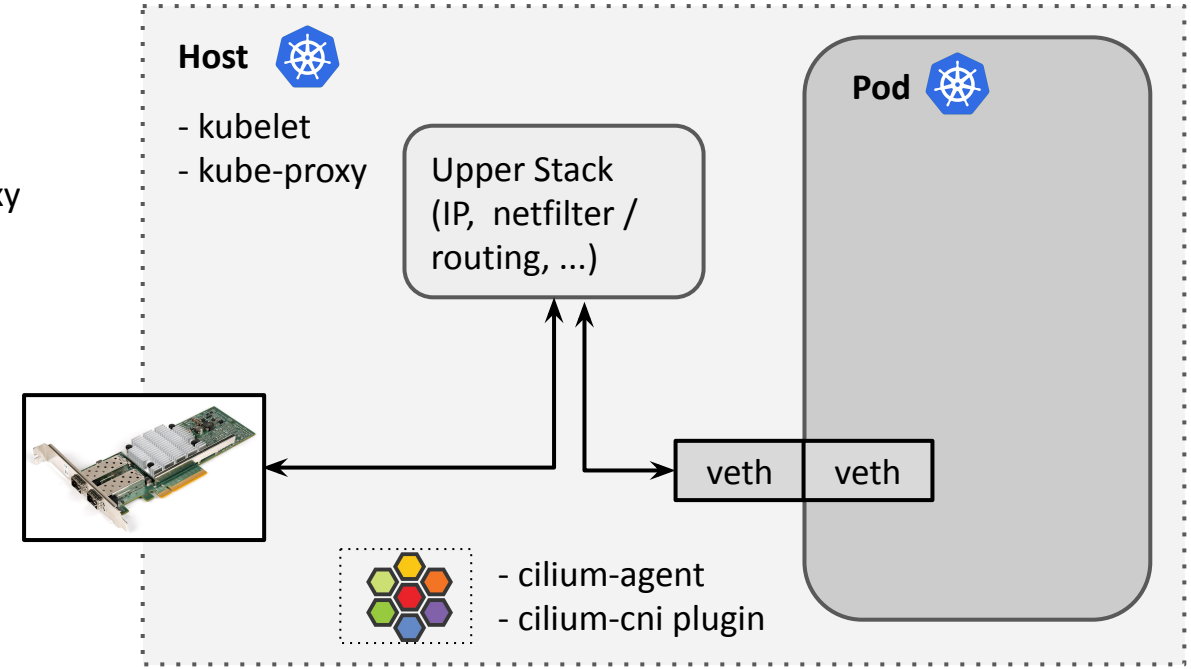
Standard Datapath Architecture:



Standard Datapath Architecture:

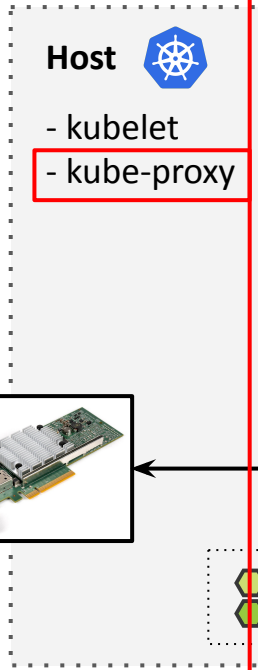


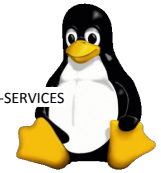
Problems:
- kube-proxy scalability
- Routing via upper stack
- Potential reasons:
  - Cannot replace kube-proxy
  - Custom netfilter rules
  - Just “went with defaults”

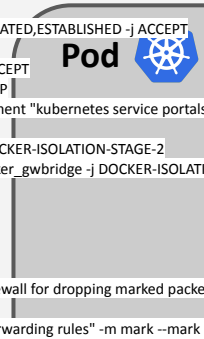


Standard Datapath Architec

Problems:
- kube-proxy scalability
- Routing via upper stack
- Potential reasons:
  - Cannot replace kube-proxy
  - Custom netfilter rules
  - Just “went with defaults”

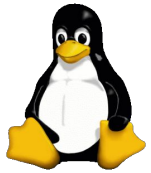




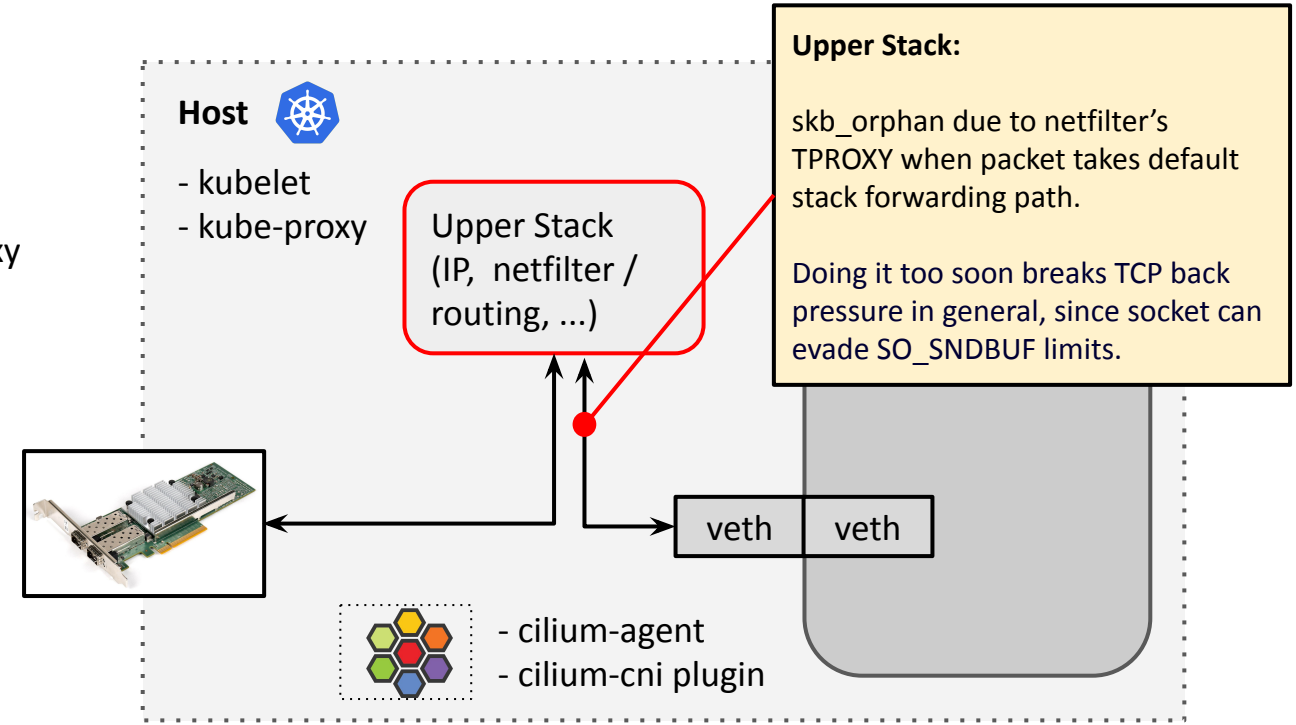



```
:DOCKER-ISOLATION-STAGE-2 - [0:0]
:DOCKER-USER - [0:0]
:KUBE-EXTERNAL-SERVICES - [0:0]
:KUBE-FIREWALL - [0:0]
:KUBE-FORWARD - [0:0]
:KUBE-SERVICES - [0:0]
-A INPUT -m conntrack --ctstate NEW -m comment --comment "kubernetes service portals" -j KUBE-SERVICES
-A INPUT -m conntrack --ctstate NEW -m comment --comment "kubernetes externally-visible service portals" -j KUBE-EXTERNAL-SERVICES
-A INPUT -j KUBE-FIREWALL
-A FORWARD -m comment --comment "kubernetes forwarding rules" -j KUBE-FORWARD
-A FORWARD -m conntrack --ctstate NEW -m comment --comment "kubernetes service portals" -j KUBE-SERVICES
-A FORWARD -j DOCKER-USER
-A FORWARD -j DOCKER-ISOLATION-STAGE-1
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A FORWARD -o docker_gwbridge -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o docker_gwbridge -j DOCKER
-A FORWARD -i docker_gwbridge ! -o docker_gwbridge -j ACCEPT
-A FORWARD -i docker_gwbridge -o docker_gwbridge -j DROP
-A OUTPUT -m conntrack --ctstate NEW -m comment --comment "kubernetes service portals" -j KUBE-SERVICES
-A OUTPUT -j KUBE-FIREWALL
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -i docker_gwbridge ! -o docker_gwbridge -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -j RETURN
-A DOCKER-ISOLATION-STAGE-2 -o docker0 -j DROP
-A DOCKER-ISOLATION-STAGE-2 -o docker_gwbridge -j DROP
-A DOCKER-ISOLATION-STAGE-2 -j RETURN
-A DOCKER-USER -j RETURN
-A KUBE-FIREWALL -m comment --comment "kubernetes firewall for dropping marked packets" -m mark --mark 0x8000/0x8000 -j DROP
-A KUBE-FORWARD -m conntrack --ctstate INVALID -j DROP
-A KUBE-FORWARD -m comment --comment "kubernetes forwarding rules" -m mark --mark 0x4000/0x4000 -j ACCEPT
-A KUBE-FORWARD -s 10.217.0.0/16 -m comment --comment "kubernetes forwarding conntrack pod source rule" -m conntrack --ctstate RELATED,ESTABLISHED
-A KUBE-FORWARD -d 10.217.0.0/16 -m comment --comment "kubernetes forwarding conntrack pod destination rule" -m conntrack --ctstate RELATED,ESTABLI
-A KUBE-SERVICES -d 10.99.38.155/32 -p tcp -m comment --comment "default/nginx-59: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port-
-A KUBE-SERVICES -d 10.96.61.252/32 -p tcp -m comment --comment "default/nginx-64: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port-
-A KUBE-SERVICES -d 10.104.166.10/32 -p tcp -m comment --comment "default/nginx-67: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port
-A KUBE-SERVICES -d 10.98.85.41/32 -p tcp -m comment --comment "default/nginx-9: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port-un
-A KUBE-SERVICES -d 10.97.138.144/32 -p tcp -m comment --comment "default/nginx-17: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port
-A KUBE-SERVICES -d 10.106.49.80/32 -p tcp -m comment --comment "default/nginx-37: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port-
-A KUBE-SERVICES -d 10.104.164.205/32 -p tcp -m comment --comment "default/nginx-5: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port
-A KUBE-SERVICES -d 10.104.25.150/32 -p tcp -m comment --comment "default/nginx-19: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port
-A KUBE-SERVICES -d 10.106.234.213/32 -p tcp -m comment --comment "default/nginx-88: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-po
-A KUBE-SERVICES -d 10.109.209.136/32 -p tcp -m comment --comment "default/nginx-33: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-po
-A KUBE-SERVICES -d 10.106.196.105/32 -p tcp -m comment --comment "default/nginx-49: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-po
-A KUBE-SERVICES -d 10.111.101.6/32 -p tcp -m comment --comment "default/nginx-53: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port-
-A KUBE-SERVICES -d 10.110.226.230/32 -p tcp -m comment --comment "default/nginx-79: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-po
-A KUBE-SERVICES -d 10.98.99.136/32 -p tcp -m comment --comment "default/nginx-6: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port-u
-A KUBE-SERVICES -d 10.99.75.233/32 -p tcp -m comment --comment "default/nginx-7: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port-u
-A KUBE-SERVICES -d 10.108.41.202/32 -p tcp -m comment --comment "default/nginx-14: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port
-A KUBE-SERVICES -d 10.97.36.249/32 -p tcp -m comment --comment "default/nginx-99: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port-
-A KUBE-SERVICES -d 10.98.213.37/32 -p tcp -m comment --comment "default/nginx-77: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port-
-A KUBE-SERVICES -d 10.107.229.31/32 -p tcp -m comment --comment "default/nginx-92: has no endpoints" -m tcp --dport 80 -j REJECT --reject-with icmp-port
```

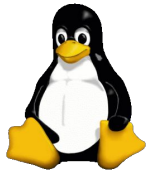
Standard Datapath Architecture:



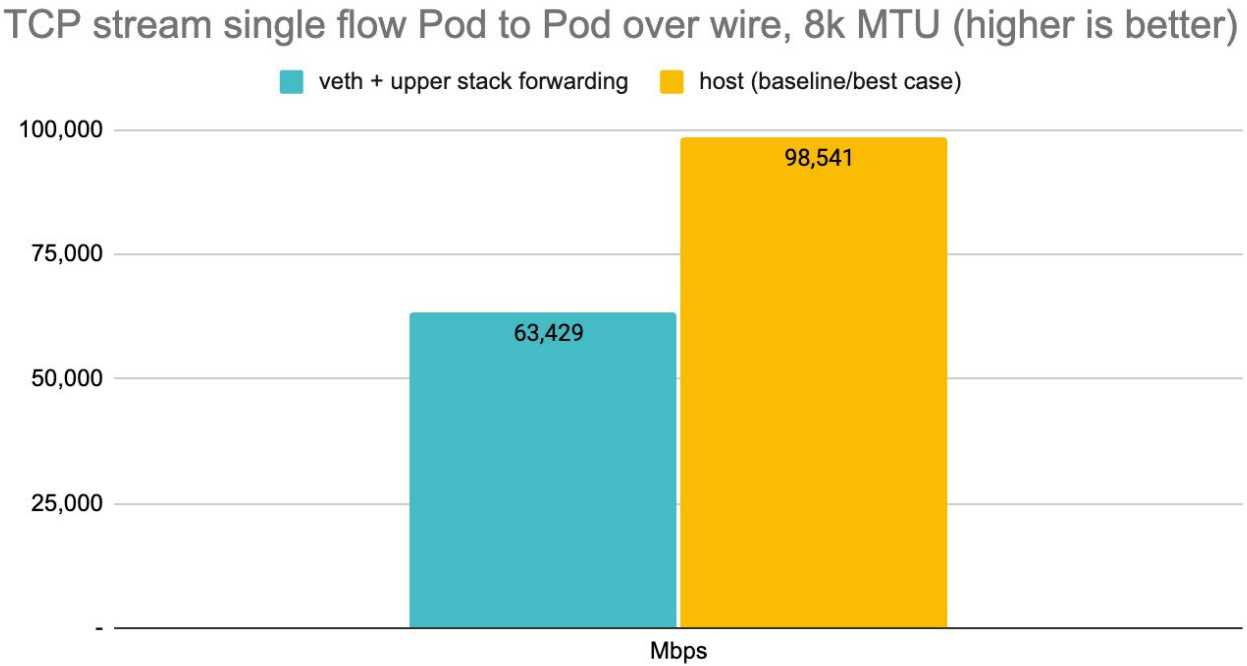
Problems:
- kube-proxy scalability
- Routing via upper stack
- Potential reasons:
  - Cannot replace kube-proxy
  - Custom netfilter rules
  - Just “went with defaults”



Standard Datapath Architecture:



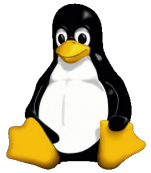
Problems:
- kube-proxy scalability
- Routing via upper stack
- Potential reasons:
  - Cannot replace kube-proxy
  - Custom netfilter rules
  - Just “went with defaults”



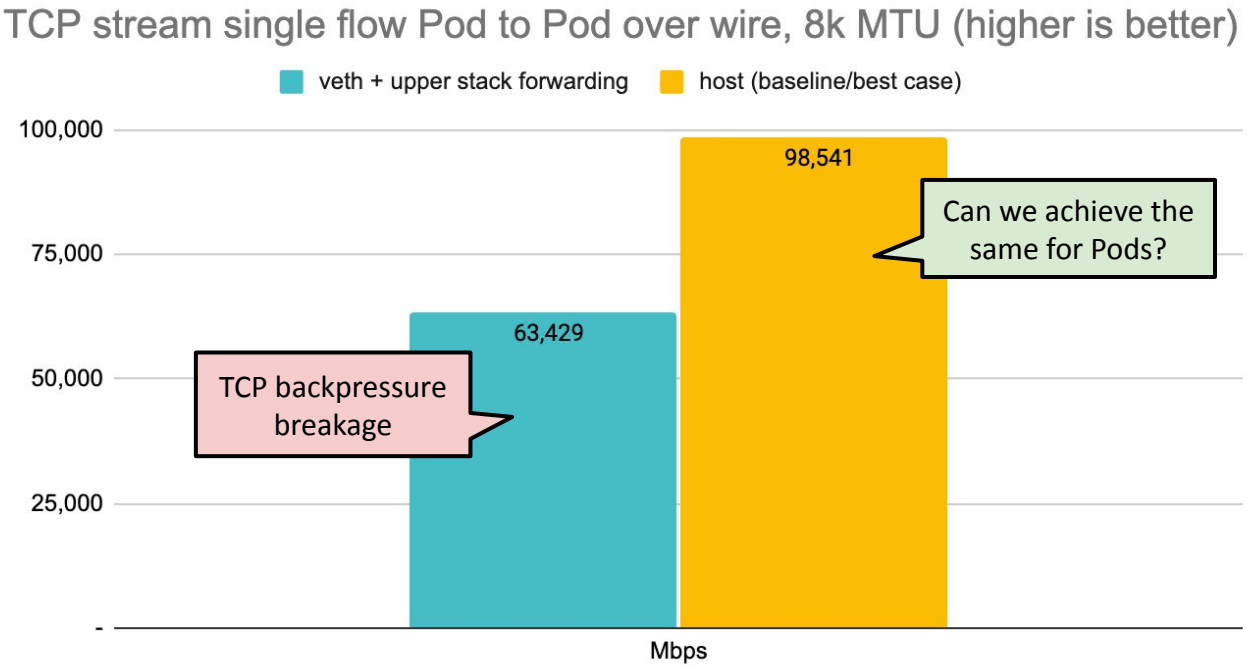
* 8264 MTU for data page alignment in GRO

Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 8264 MTU
Receiver: taskset -a -c <core> tcp_mmap -s (non-zerocopy mode), Sender: taskset -a -c <core> tcp_mmap -H <dst host>

Standard Datapath Architecture:



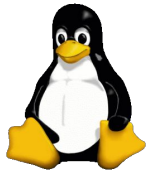
Problems:
- kube-proxy scalability
- Routing via upper stack
- Potential reasons:
  - Cannot replace kube-proxy
  - Custom netfilter rules
  - Just “went with defaults”



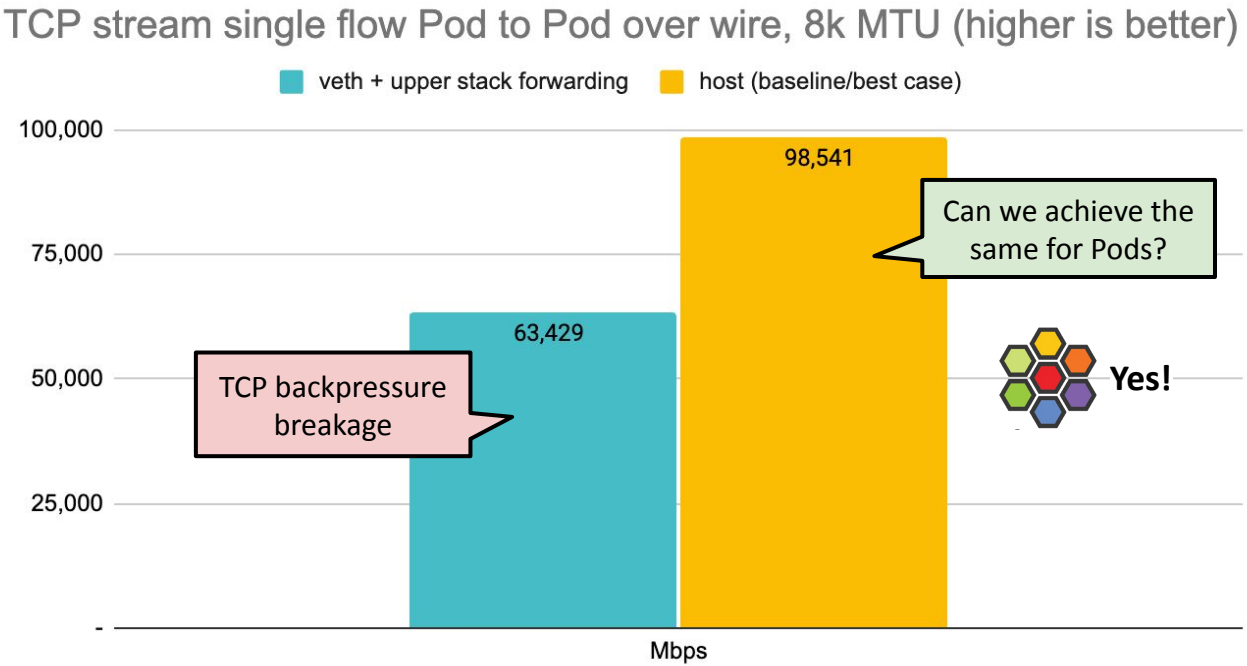
* 8264 MTU for data page alignment in GRO

Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 8264 MTU
Receiver: taskset -a -c <core> tcp_mmap -s (non-zerocopy mode), Sender: taskset -a -c <core> tcp_mmap -H <dst host>

Standard Datapath Architecture:



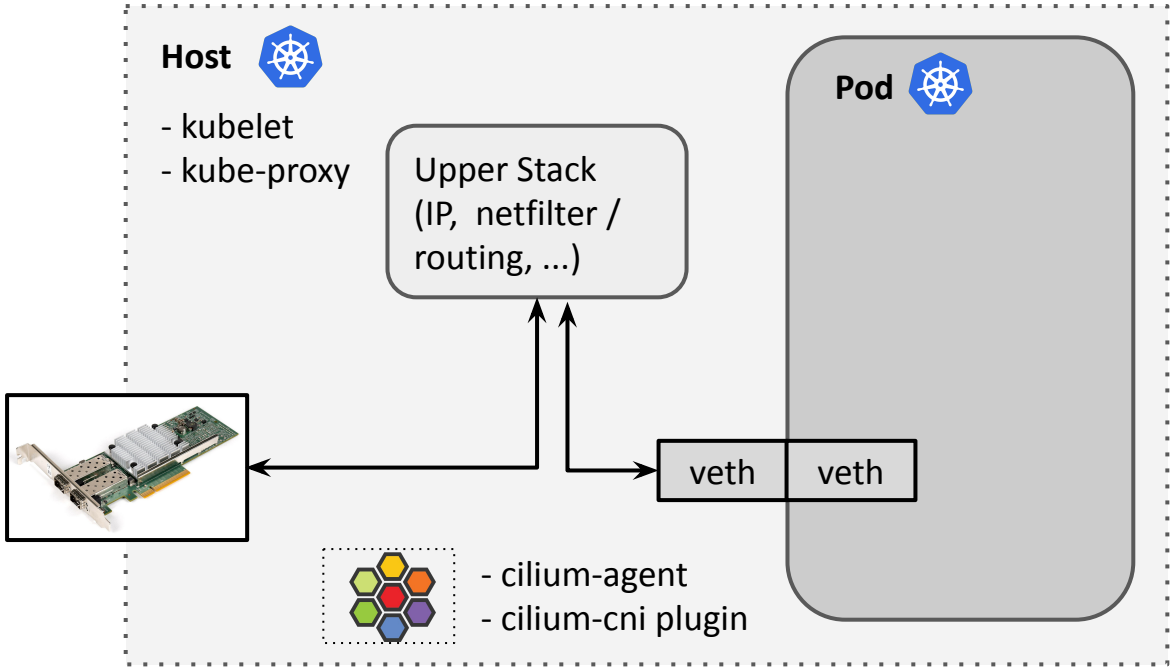
Problems:
- kube-proxy scalability
- Routing via upper stack
- Potential reasons:
  - Cannot replace kube-proxy
  - Custom netfilter rules
  - Just “went with defaults”



* 8264 MTU for data page alignment in GRO

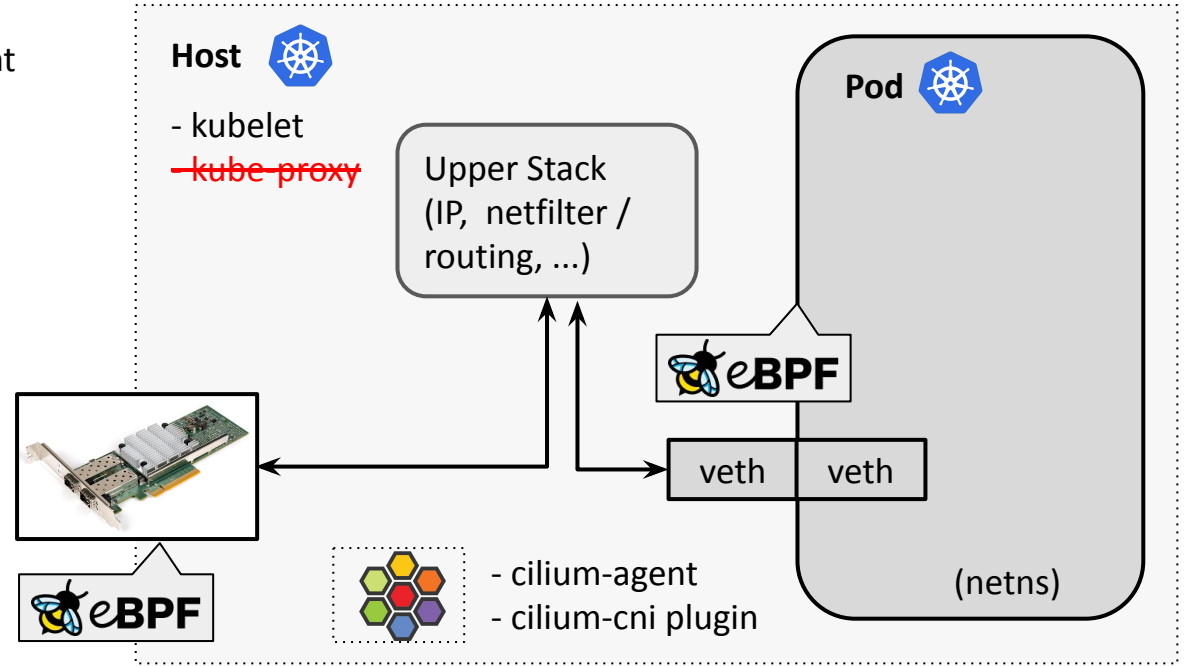
Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 8264 MTU
Receiver: taskset -a -c <core> tcp_mmap -s (non-zerocopy mode), Sender: taskset -a -c <core> tcp_mmap -H <dst host>

Standard Datapath Architecture:



Cilium Datapath Architecture (journey 2019 - today):

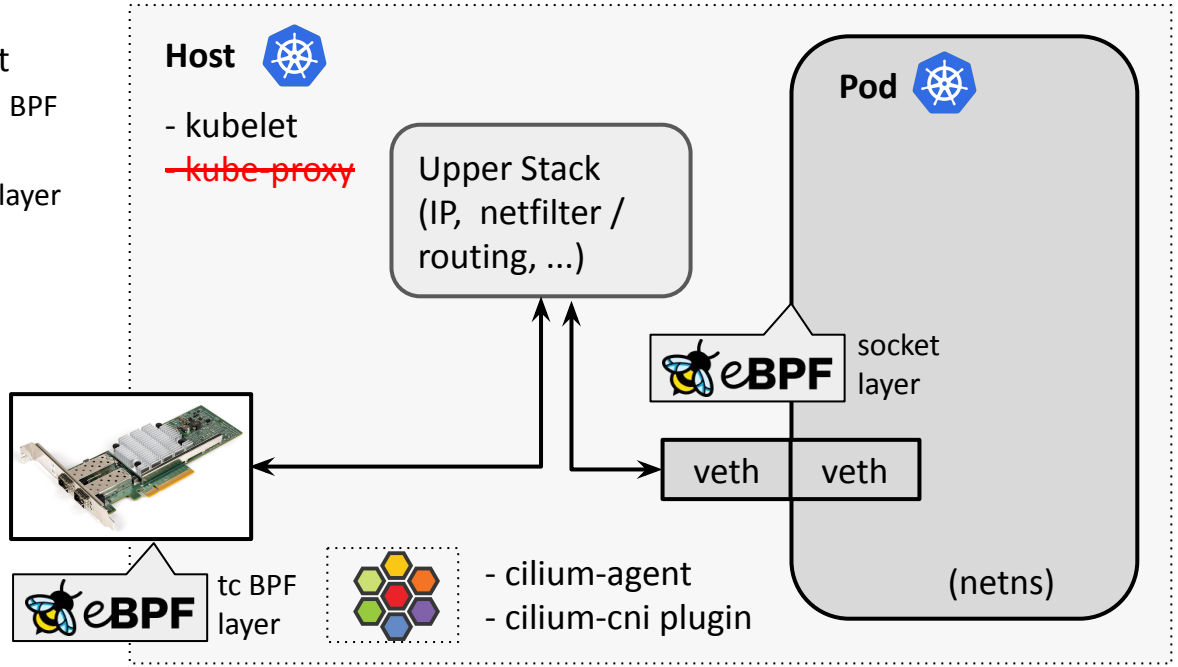
Building Blocks:
- BPF kube-proxy replacement



Cilium Datapath Architecture (journey 2019 - today):

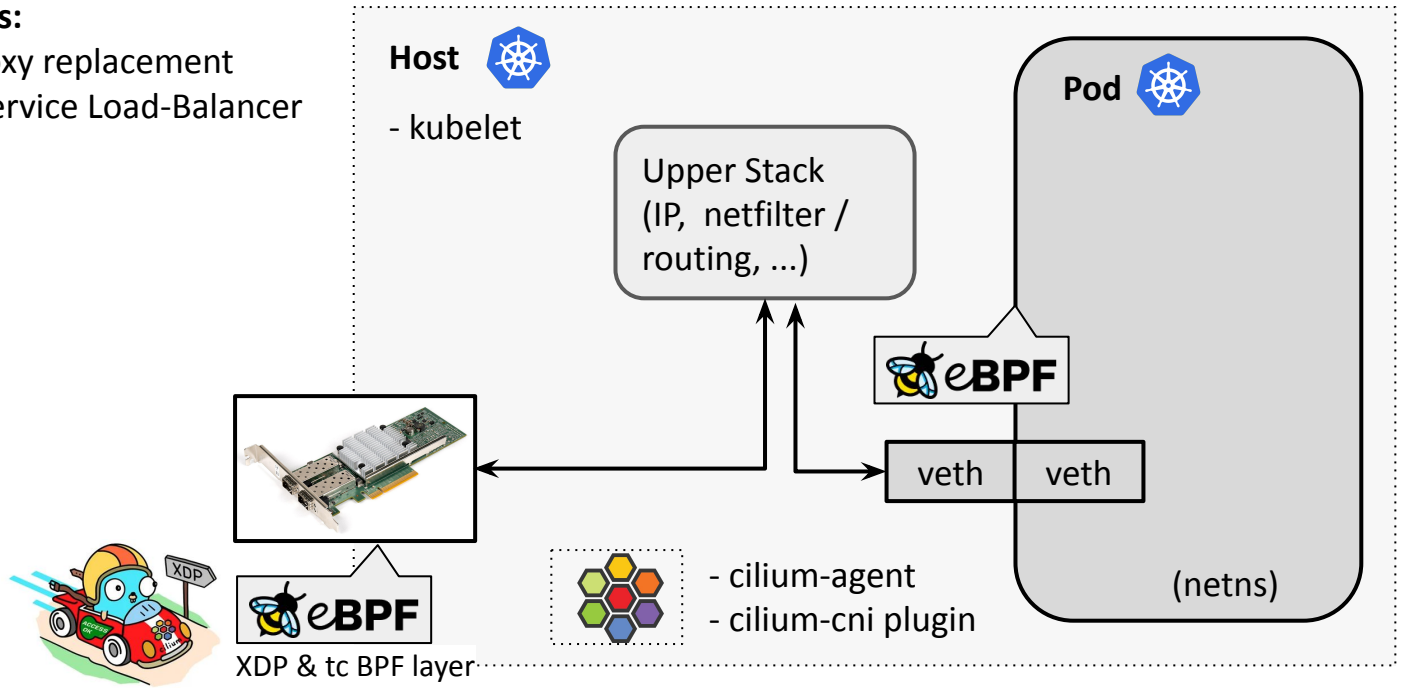
Building Blocks:

- BPF kube-proxy replacement
  - ↪ Covers all K8s service types via BPF
  - ↪ N/S: per packet NAT in tc BPF
  - ↪ E/W: per connect(2) at socket layer
  - ↪ Maglev & HostPort support



Cilium Datapath Architecture (journey 2019 - today):

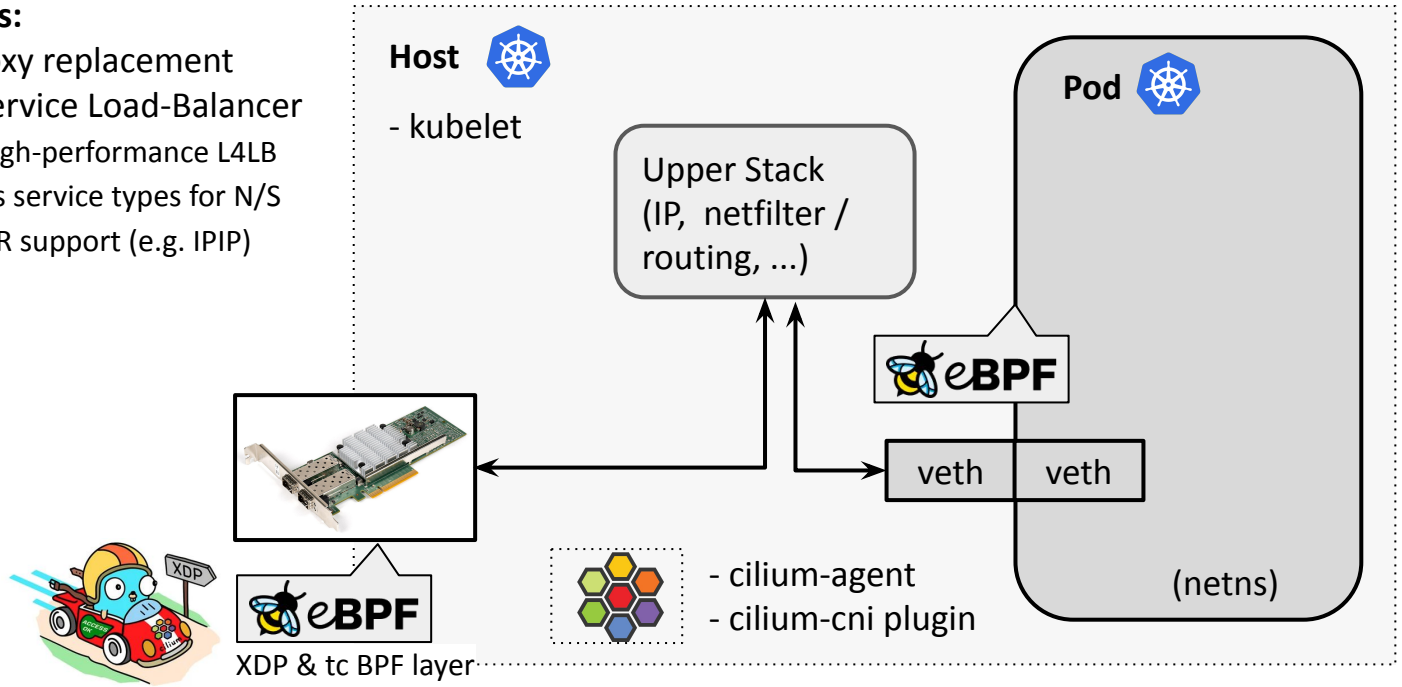
Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer



Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:

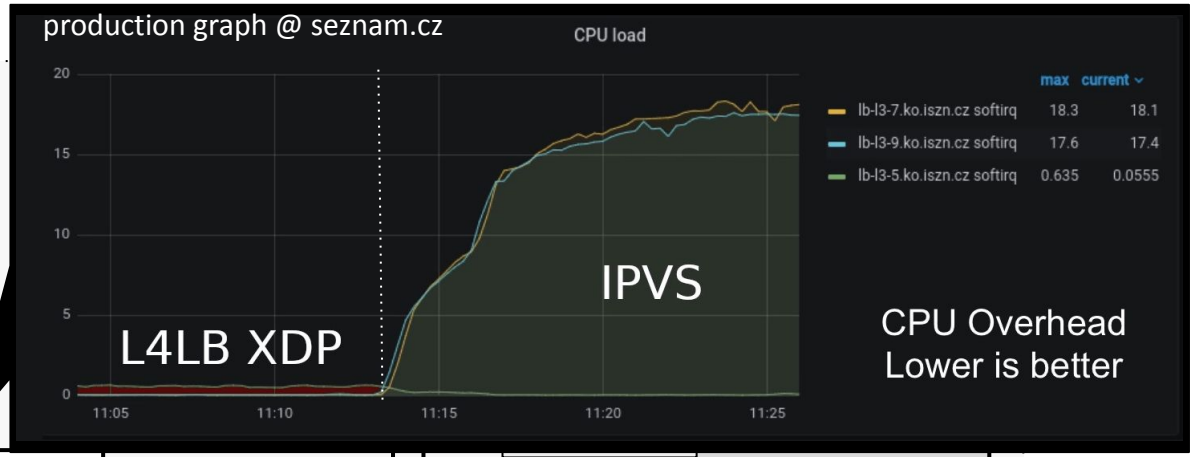
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
  - ↪ Co-located high-performance L4LB
  - ↪ Covers all K8s service types for N/S
  - ↪ Maglev & DSR support (e.g. IPIP)

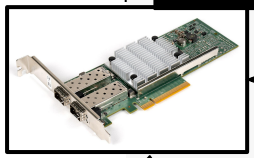


Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:

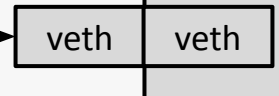
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
  - ↪ Co-located high-performance L4LB
  - ↪ Covers all K8s service types for N/S
  - ↪ Maglev & DSR support (e.g. IPIP)





XDP & tc BPF layer

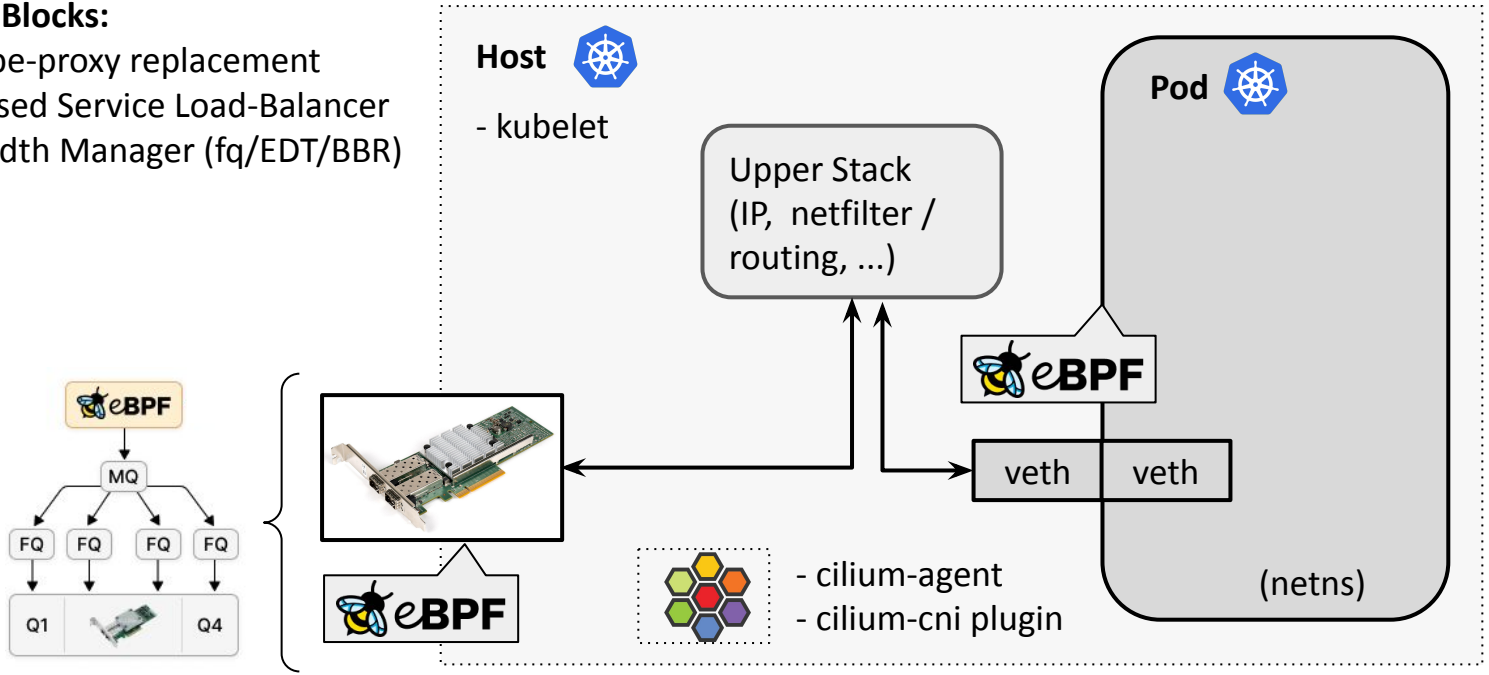
- cilium-agent
- cilium-cni plugin



(netns)

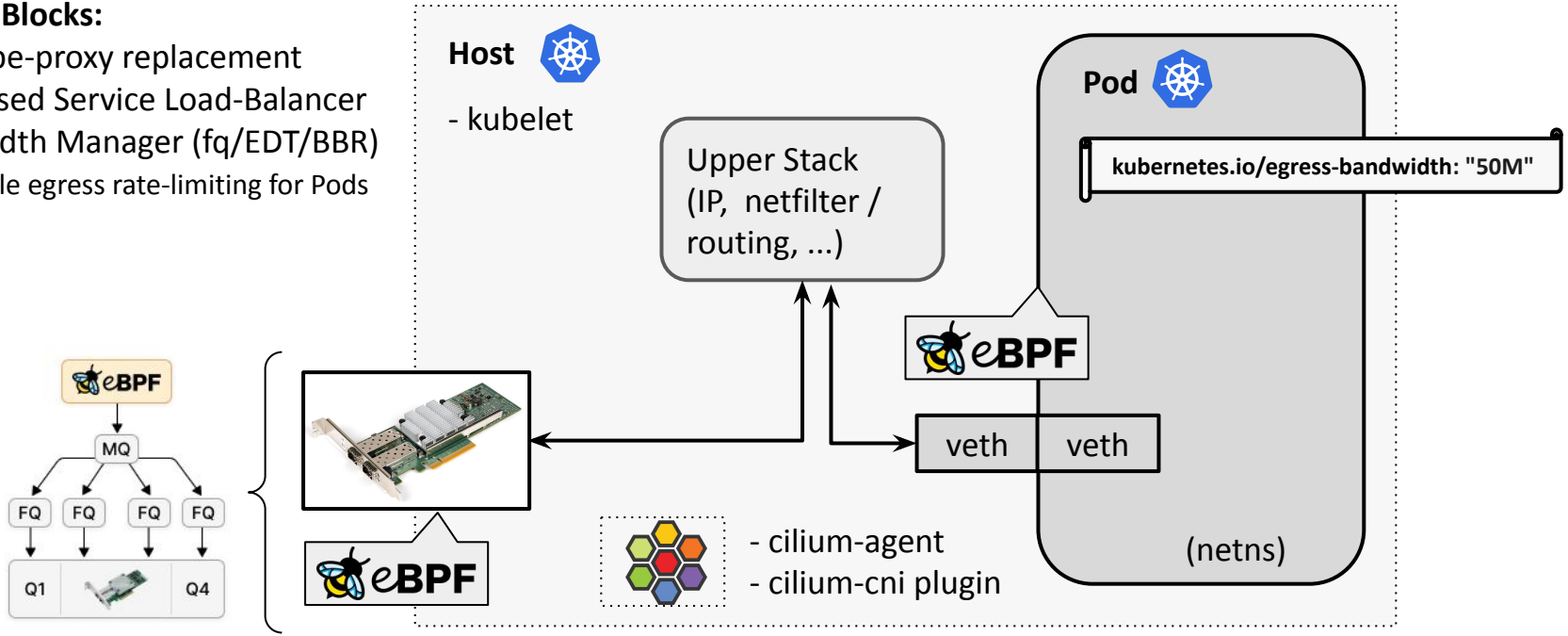
Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)



Cilium Datapath Architecture (journey 2019 - today):

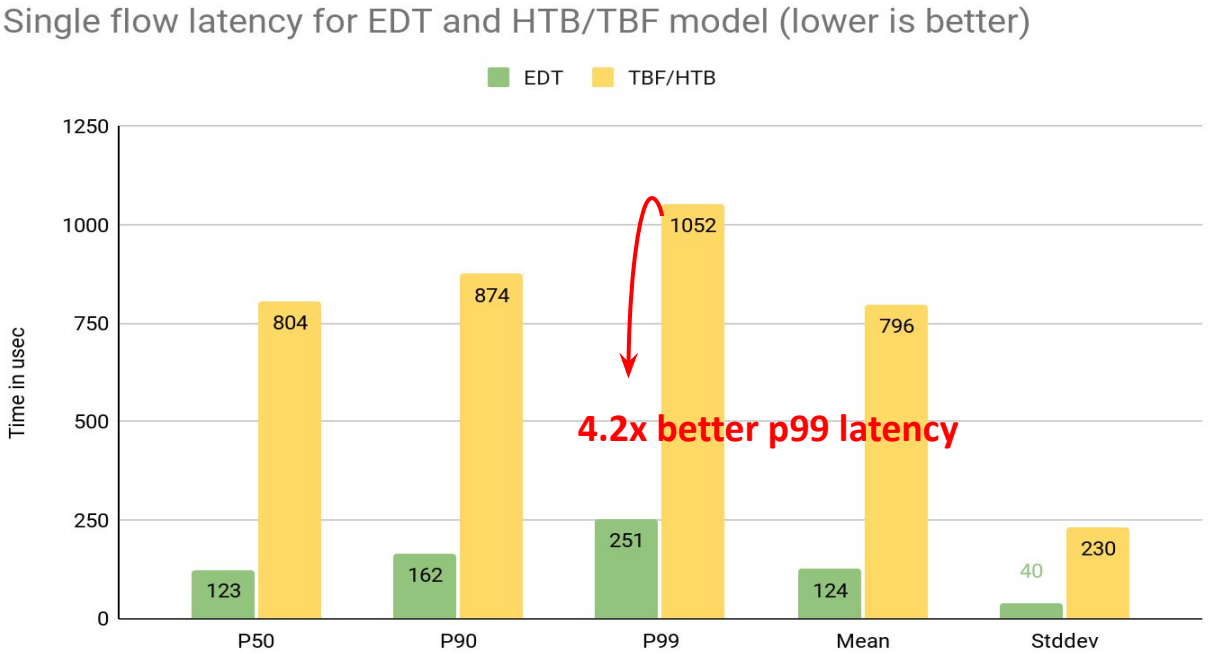
Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
  ↪ Scalable egress rate-limiting for Pods



Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:

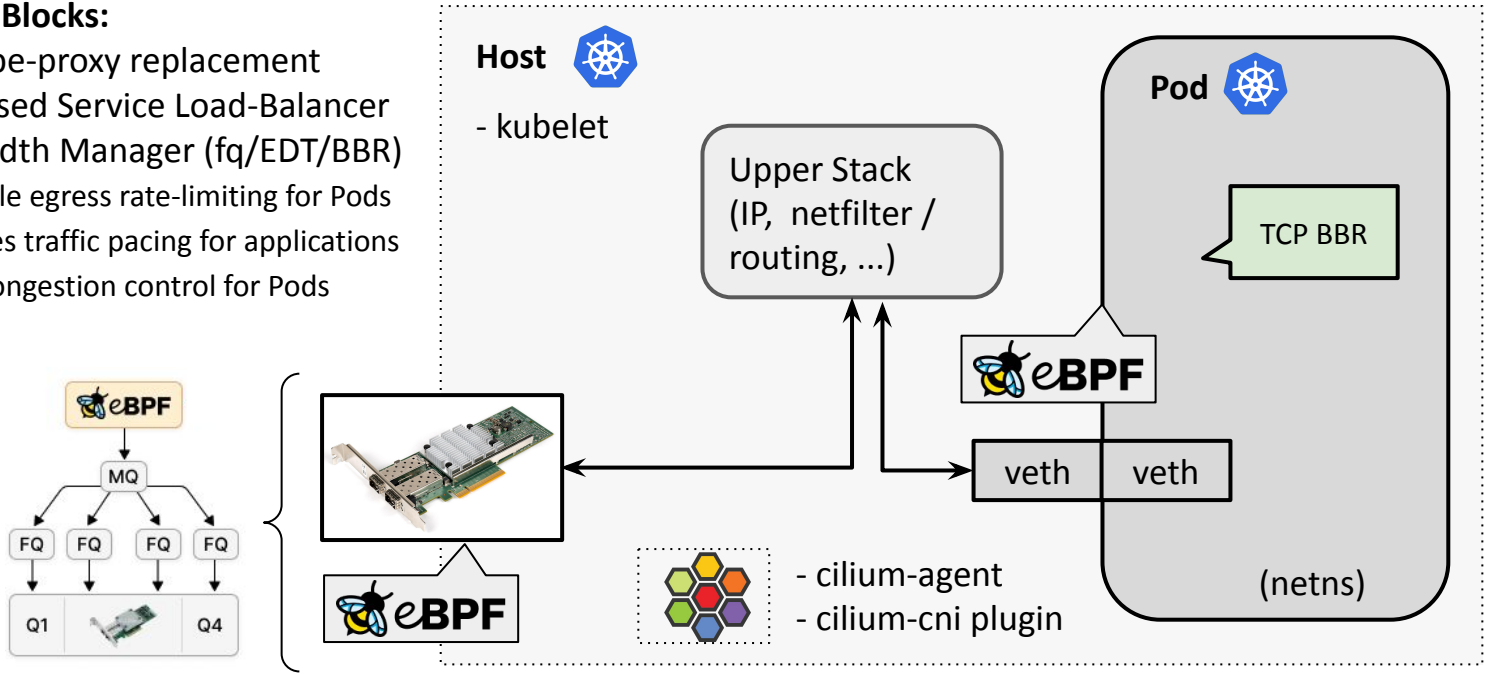
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
  - ↪ Scalable egress rate-limiting for Pods
    - Earliest departure time (EDT) via BPF
    - fq also in production at Google/Meta
    - Ready for ToS priority bands too



Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:

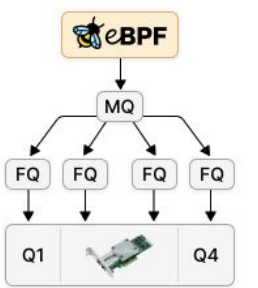
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
  - ↪ Scalable egress rate-limiting for Pods
  - ↪ Enables traffic pacing for applications
  - ↪ BBR congestion control for Pods

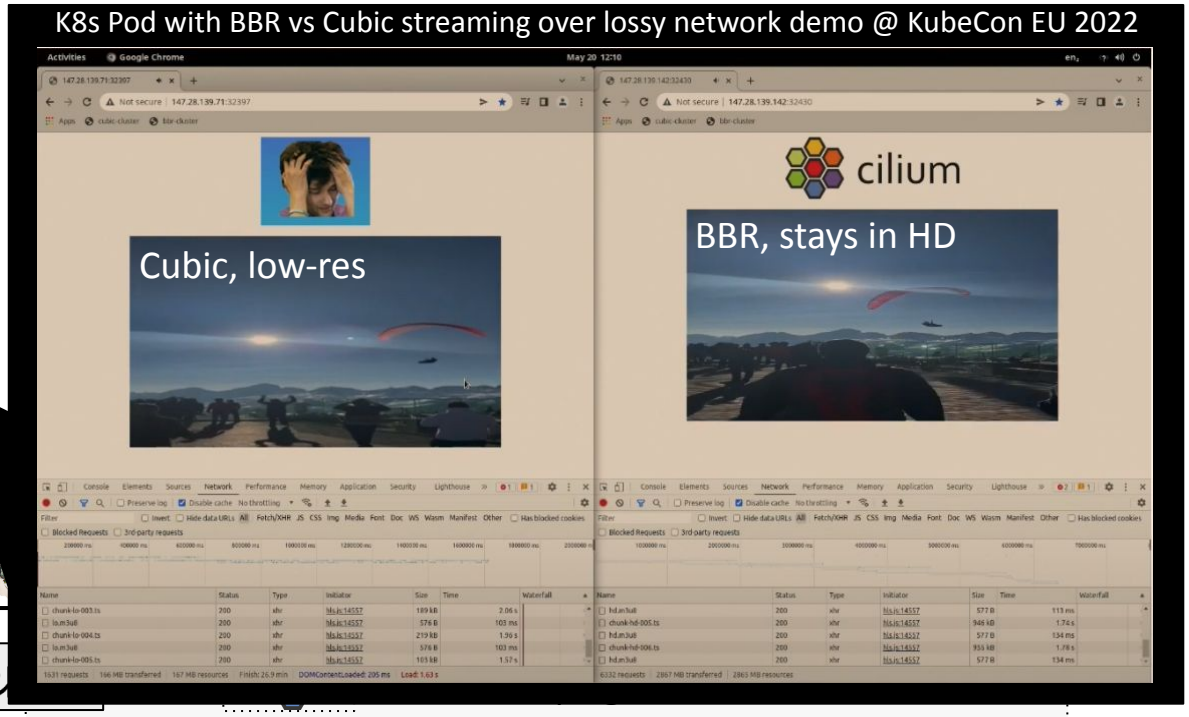


Kernel 5.18: preserving delivery timestamps across netns (Martin Lau, Daniel Borkmann)

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
  - ↪ Scalable egress rate-limiting for Pods
  - ↪ Enables traffic pacing for applications
  - ↪ BBR congestion control for Pods

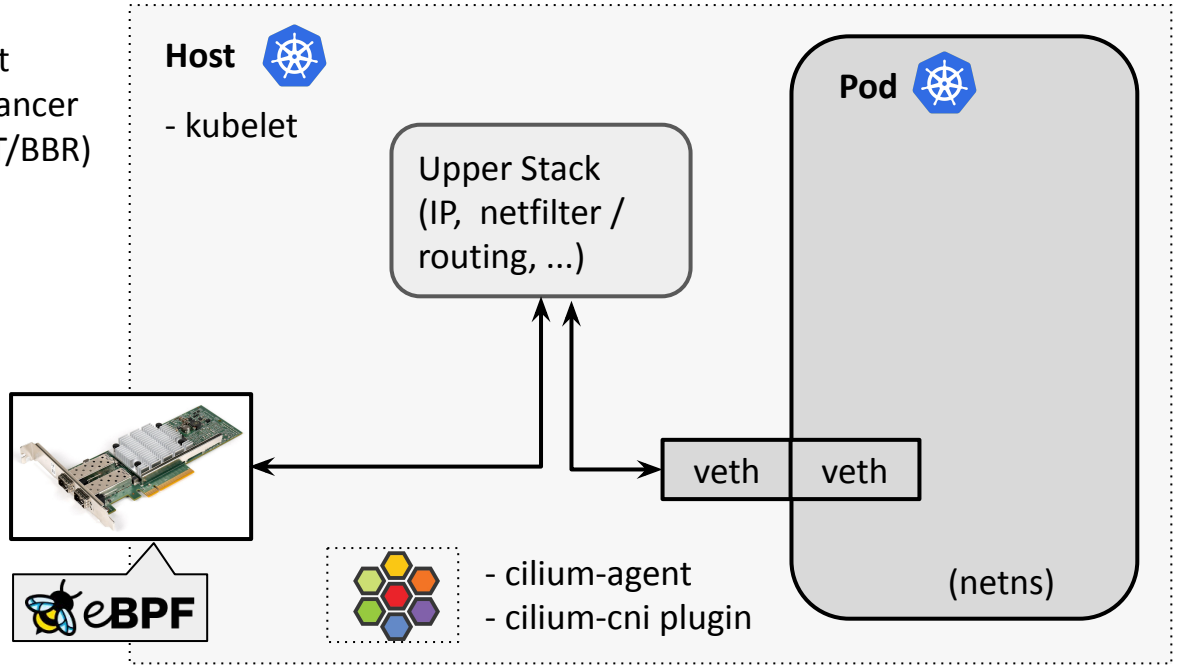




Kernel 5.18: preserving delivery timestamps across netns (Martin Lau, Daniel Borkmann)

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing

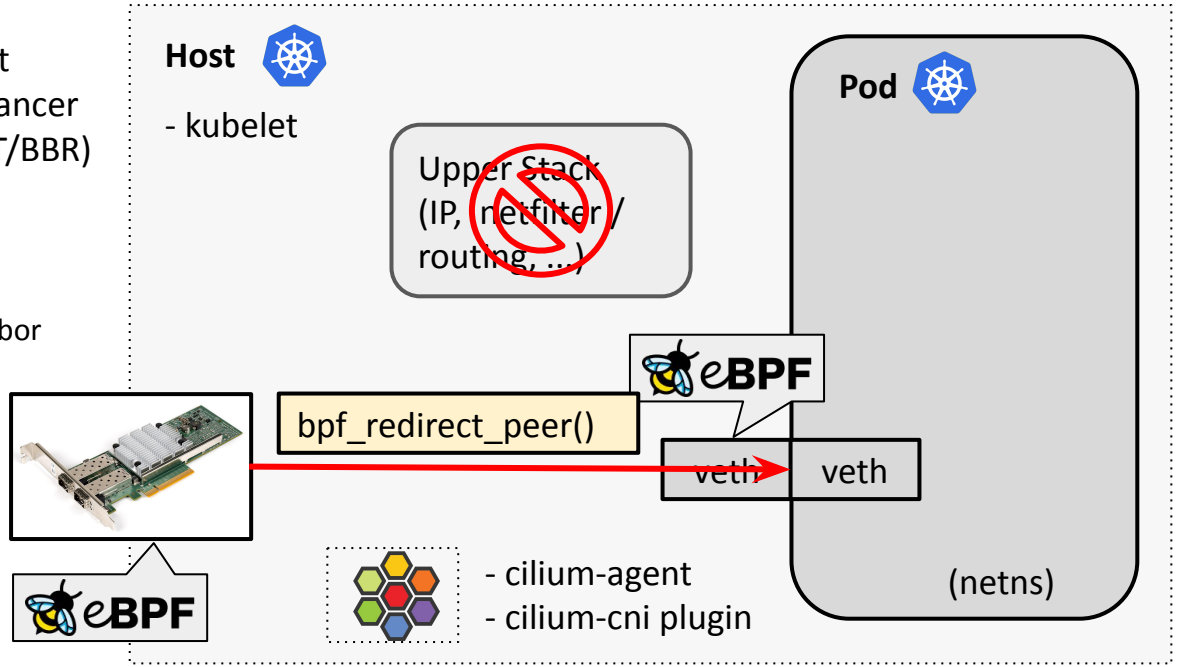


Kernel 5.10: bpf_redirect_peer and bpf_redirect_neigh helpers (Daniel Borkmann)

Cilium Datapath Architecture (journey 2019 - today):

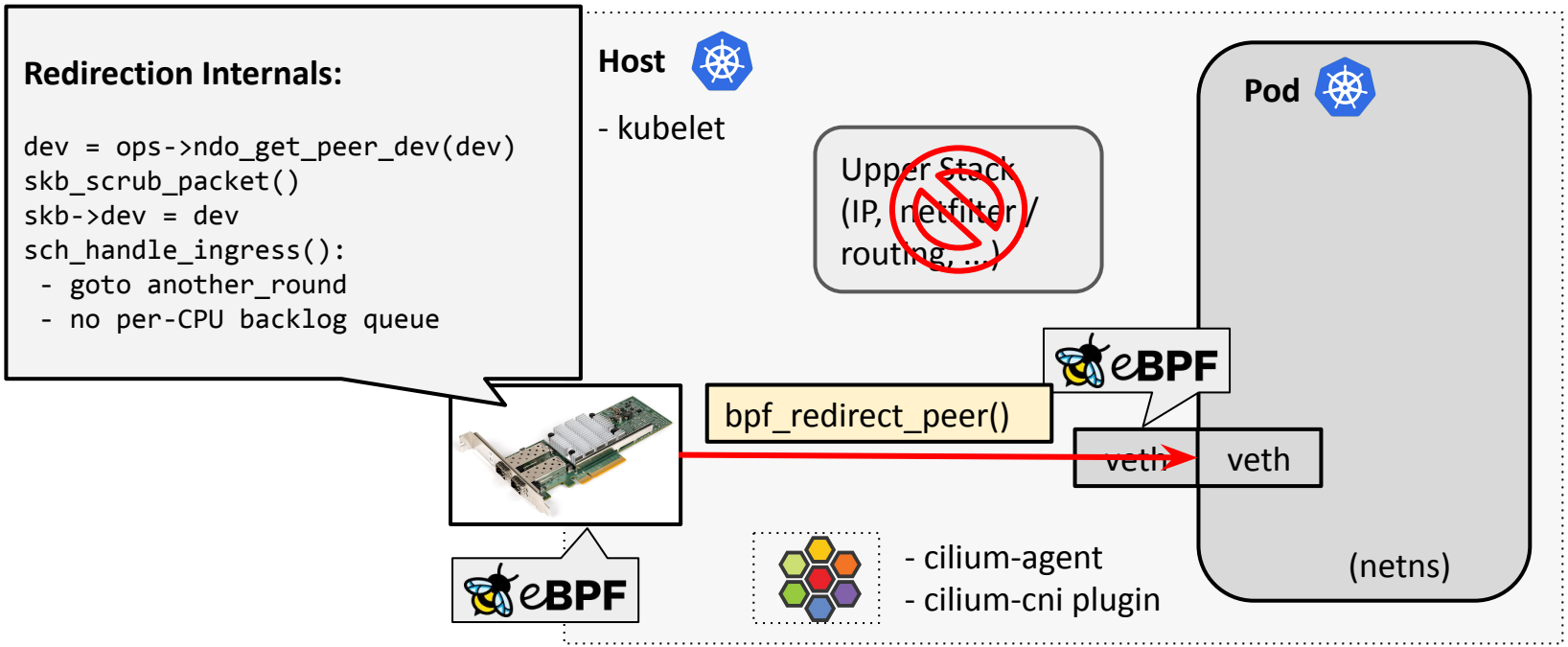
Building Blocks:

- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
  - ↪ Routing only via tc BPF layer
  - ↪ Fast netns switch on ingress
  - ↪ Helper for fib + dynamic neighbor resolution on egress



Kernel 5.10: bpf_redirect_peer and bpf_redirect_neigh helpers (Daniel Borkmann)

Cilium Datapath Architecture (journey 2019 - today):

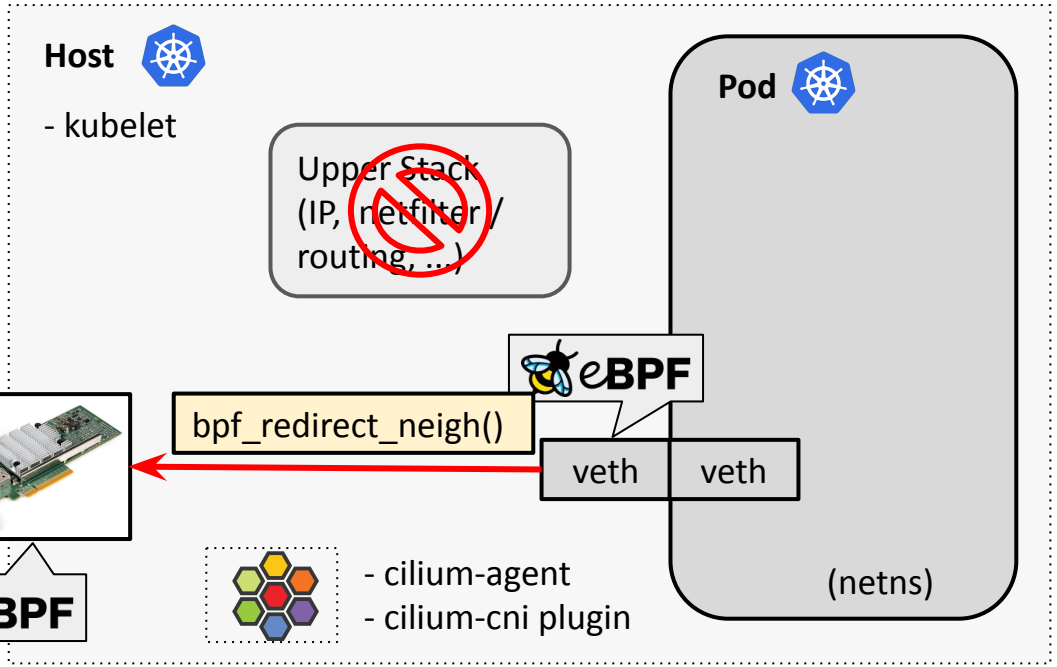


Kernel 5.10: bpf_redirect_peer and bpf_redirect_neigh helpers (Daniel Borkmann)

Cilium Datapath Architecture (journey 2019 - today):

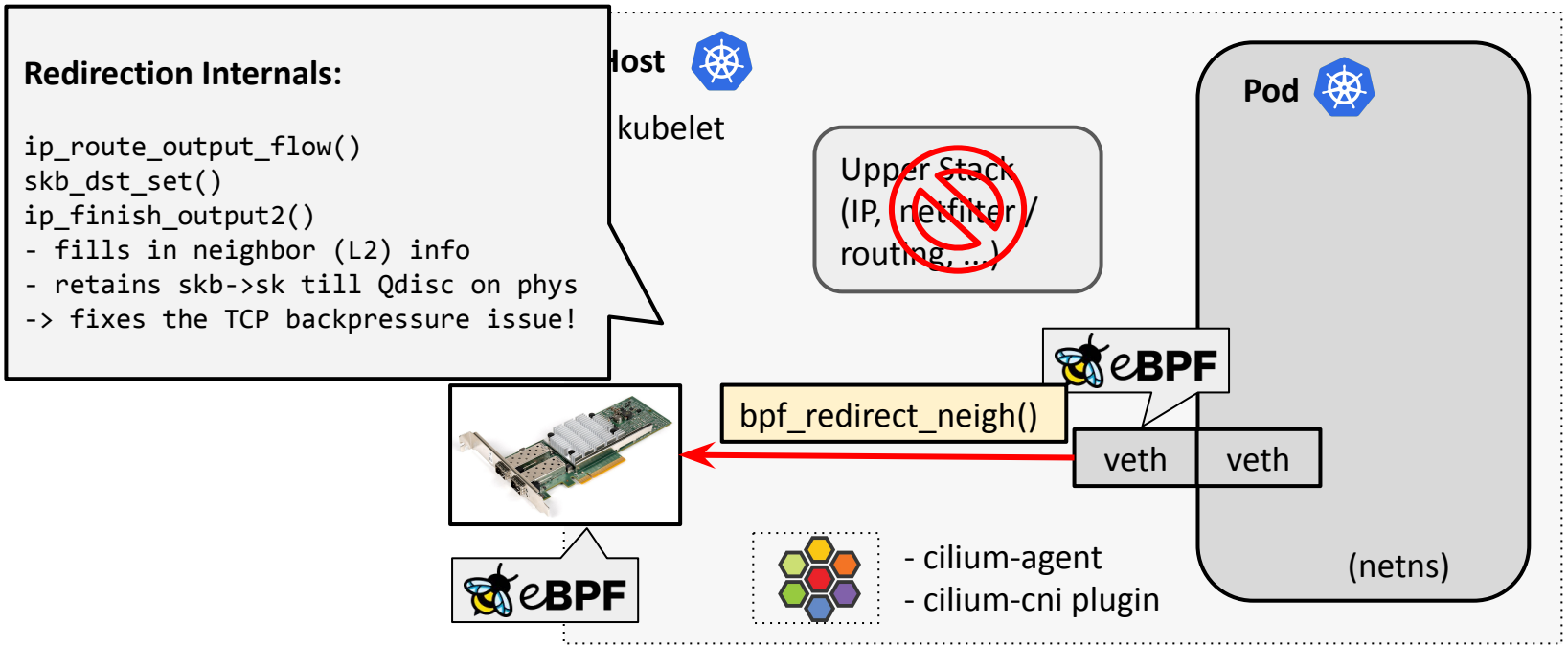
Building Blocks:

- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
  - ↪ Routing only via tc BPF layer
  - ↪ Fast netns switch on ingress
  - ↪ Helper for fib + dynamic neighbor resolution on egress



Kernel 5.10: bpf_redirect_peer and bpf_redirect_neigh helpers (Daniel Borkmann)

Cilium Datapath Architecture (journey 2019 - today):

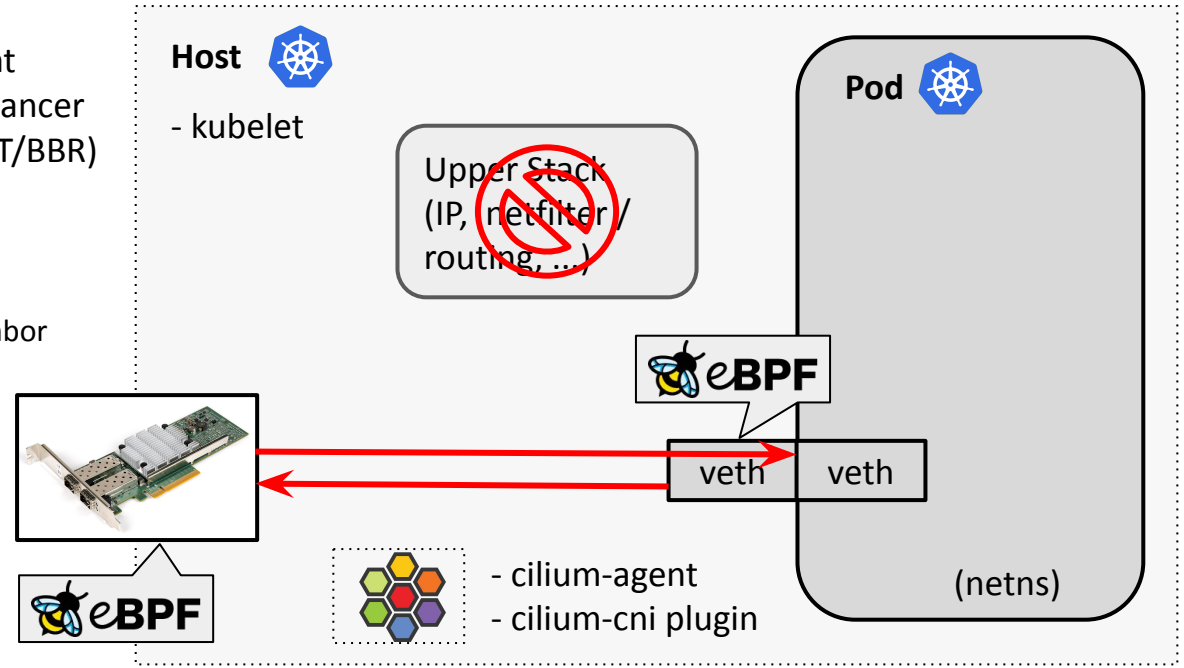


Kernel 5.10: bpf_redirect_peer and bpf_redirect_neigh helpers (Daniel Borkmann)

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:

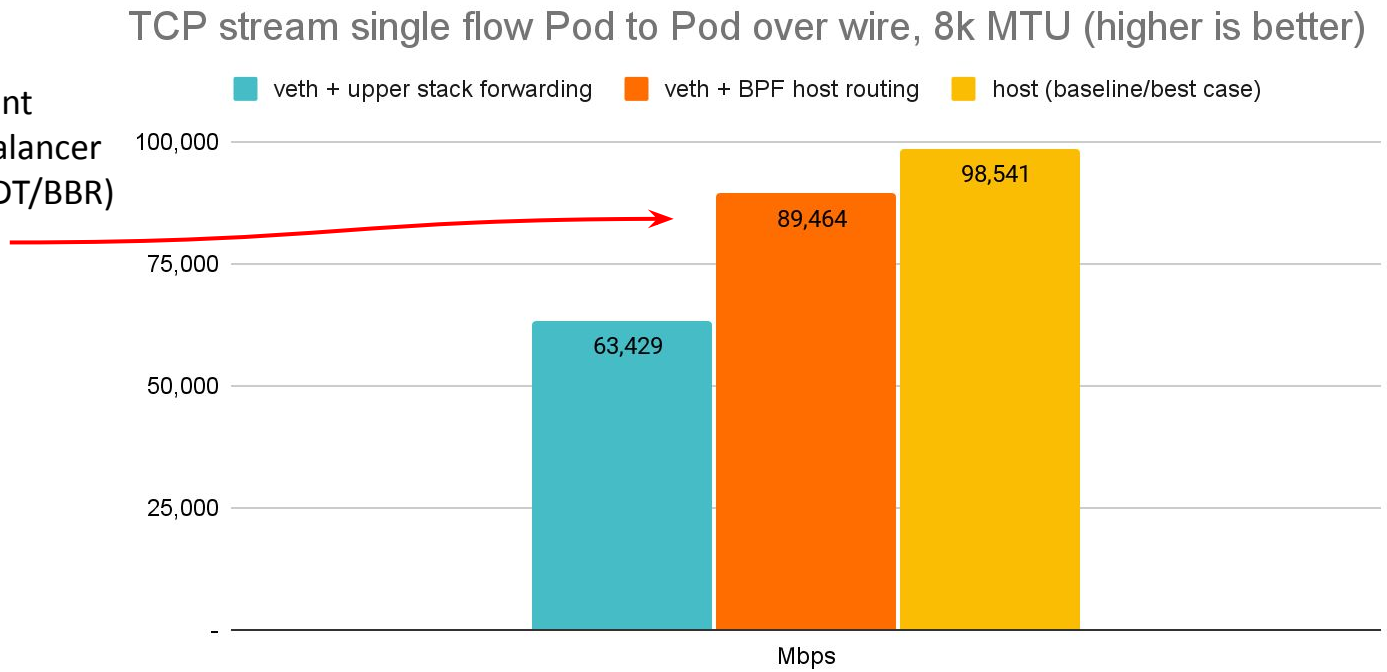
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
  - ↪ Routing only via tc BPF layer
  - ↪ Fast netns switch on ingress
  - ↪ Helper for fib + dynamic neighbor resolution on egress



Kernel 5.10: bpf_redirect_peer and bpf_redirect_neigh helpers (Daniel Borkmann)

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing

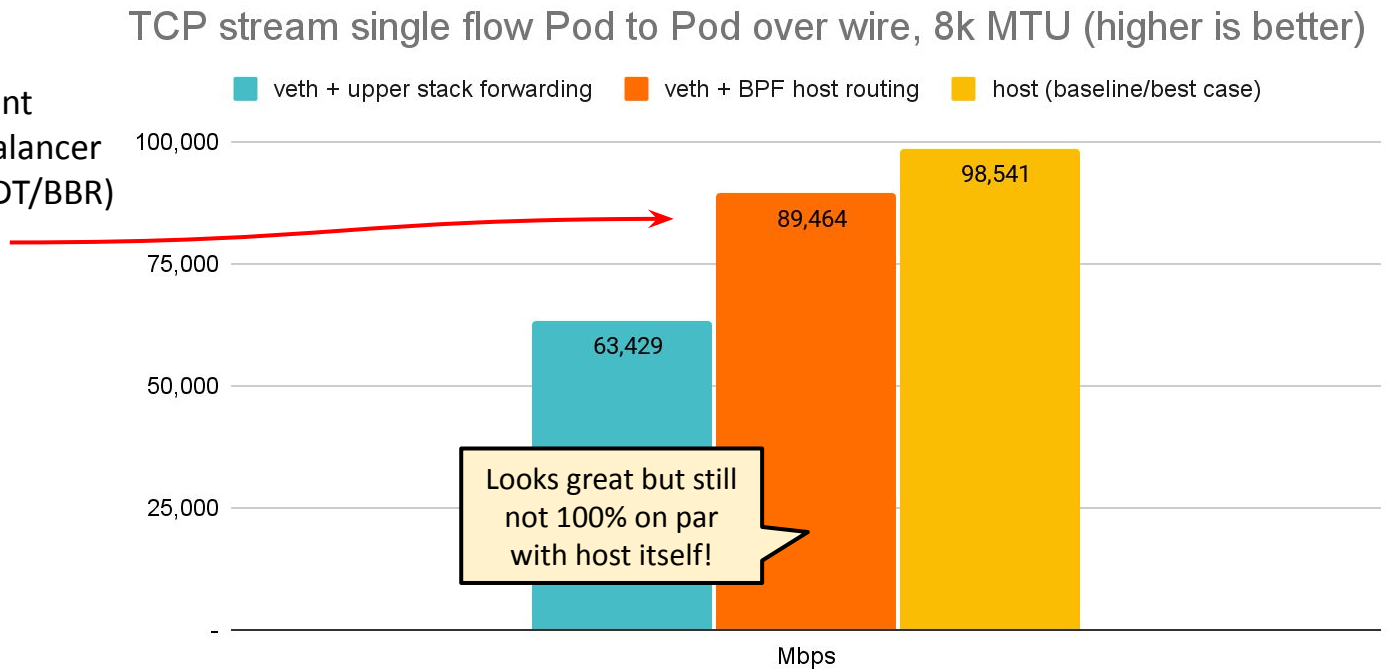


* 8264 MTU for data page alignment in GRO

Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 8264 MTU
Receiver: taskset -a -c <core> tcp_mmap -s (non-zerocopy mode), Sender: taskset -a -c <core> tcp_mmap -H <dst host>

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing

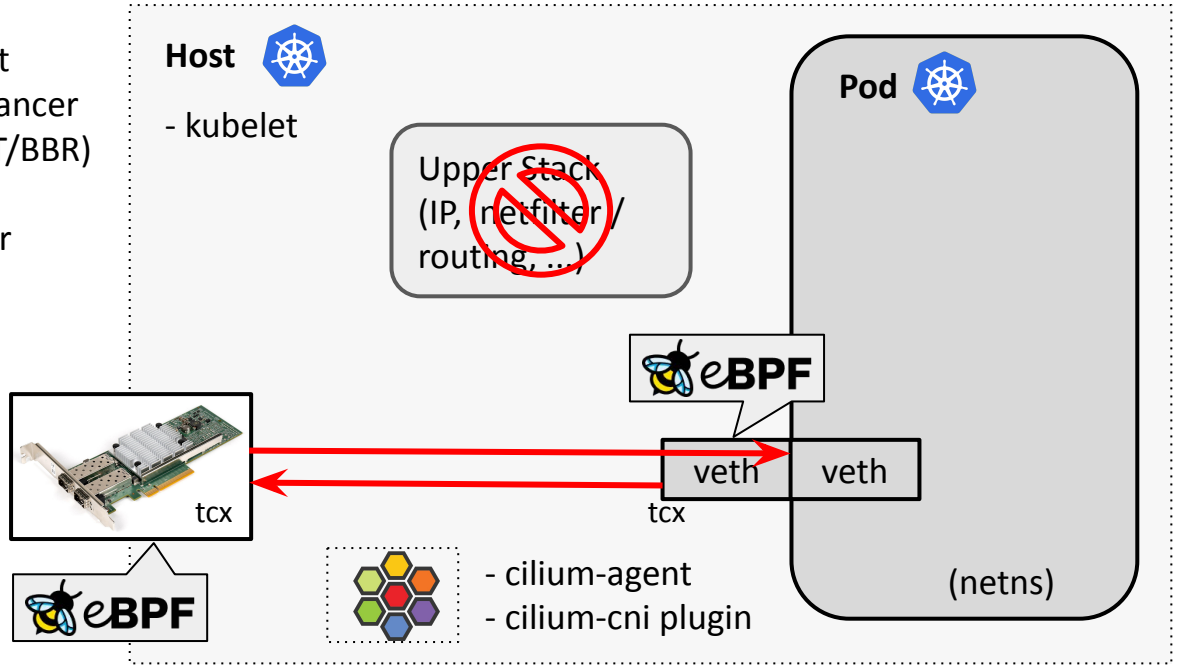


* 8264 MTU for data page alignment in GRO

Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 8264 MTU
Receiver: taskset -a -c <core> tcp_mmap -s (non-zerocopy mode), Sender: taskset -a -c <core> tcp_mmap -H <dst host>

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
- tcx-based BPF datapath layer



Kernel 6.6: tcx infrastructure and BPF link support (Daniel Borkmann)

Brief Deep Dive: tcx

tcx (tc express) layer for BPF:

- Full rework of tc BPF data and control path, approx. 1 year effort
- Merged and released with Linux kernel v6.6 onwards
- Modernizing the underlying foundation of Cilium's networking data path in the kernel:
  - More efficient, binary-compatible fast-path for tc BPF programs
  - Better robustness in control path with BPF link support
  - Before/After dependency directives for multi-program users

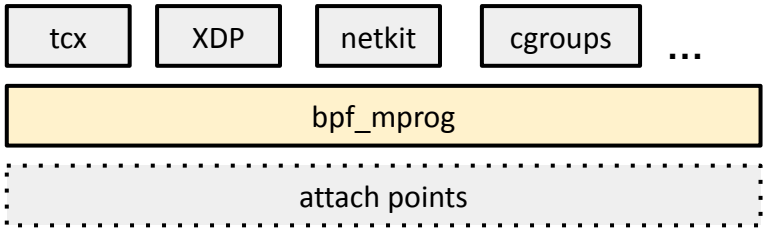
```
author      Daniel Borkmann <daniel@iogearbox.net>     2015-03-01 12:31:48 +0100
committer   David S. Miller <davem@davemloft.net>      2015-03-01 14:05:19 -0500
commit      e2e9b6541dd4b31848079da80fe2253daaafb549 (patch)
tree        dd4de9c9faa662e188285fa5db20663859139f55 /net/sched/cls_bpf.c
parent      24701ecea76b0b93bd9667486934ec310825f558 (diff)
download    linux-e2e9b6541dd4b31848079da80fe2253daaafb549.tar.gz

cls_bpf: add initial eBPF support for programmable classifiers

This work extends the "classic" BPF programmable tc classifier by
extending its scope also to native eBPF code!
```

```
author      Daniel Borkmann <daniel@iogearbox.net>     2023-07-19 16:08:52 +0200
committer   Alexei Starovoitov <ast@kernel.org>        2023-07-19 10:07:27 -0700
commit      e420bed025071a623d2720a92bc2245c84757ecb (patch)
tree        fa8c7e0b31d755ada58465dba12ffdd82a92bc4c
parent      053c8e1f235dc3f69d13375b32f4209228e1cb96 (diff)
download    linux-e420bed025071a623d2720a92bc2245c84757ecb.tar.gz

bpf: Add fd-based tcx multi-prog infra with link support

This work refactors and adds a lightweight extension ("tcx") to the tc BPF
ingress and egress data path side for allowing BPF program management based
on fds via bpf() syscall through the newly added generic multi-prog API.
The main goal behind this work which we also presented at LPC [0] last year
and a recent update at LSF/MM/BPF this year [3] is to support long-awaited
BPF link functionality for tc BPF programs, which allows for a model of safe
ownership and program detachment.
```

Brief Deep Dive: tcx

tcx (tc express) layer for BPF:

- For tcx we also built a new bpf_mprog framework for BPF program dependency management
- Provides a common “look-and-feel” for all networking attach points and beyond
- Future work is to integrate this into XDP as well to support multi-program attachment for the latter



author	Daniel Borkmann <daniel@iogearbox.net> 2023-07-19 16:08:51 +0200
committer	Alexei Starovoitov <ast@kernel.org> 2023-07-19 10:07:27 -0700
commit	053c8e1f235dc3f69d13375b32f4209228e1cb96 (patch)
tree	5ceaf956ca383f496c703fd8700dca5bf06c000f
parent	3226e3139dfe02d5892562976a649a54ada12a13 (diff)
download	linux-053c8e1f235dc3f69d13375b32f4209228e1cb96.tar.gz

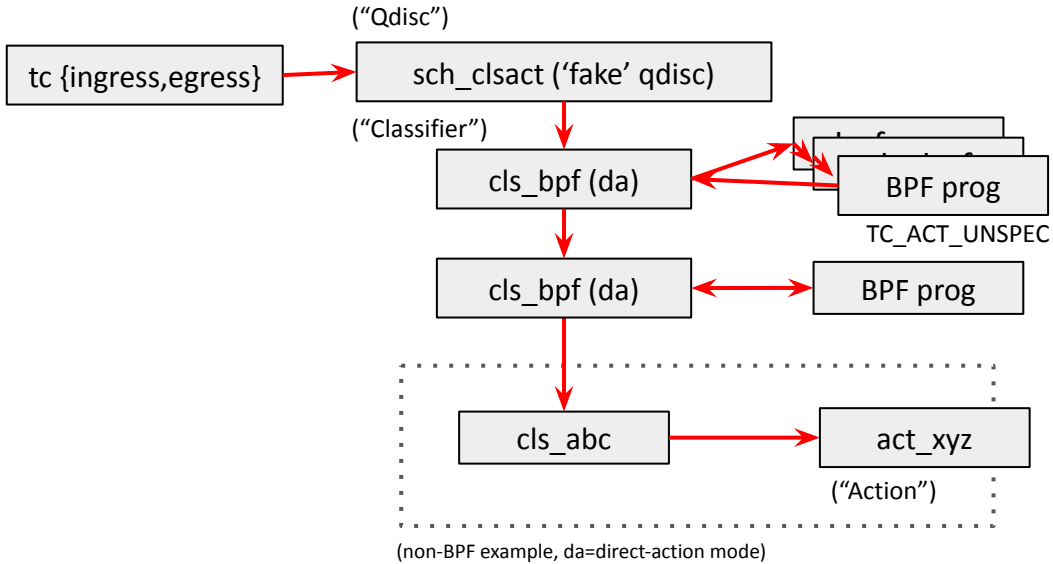
bpf: Add generic attach/detach/query API for multi-progs

```
This adds a generic layer called bpf_mprog which can be reused by different
attachment layers to enable multi-program attachment and dependency resolution.
In-kernel users of the bpf_mprog don't need to care about the dependency
resolution internals, they can just consume it with few API calls.
```

Brief Deep Dive: tcx

tcx (tc express) layer for BPF:

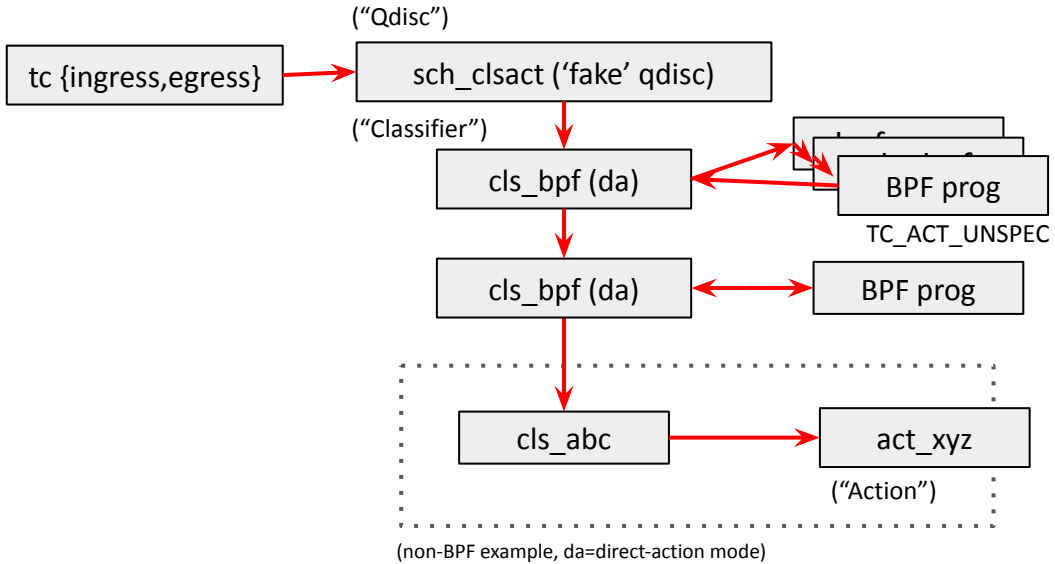
Old style (up to v6.6): cls_bpf



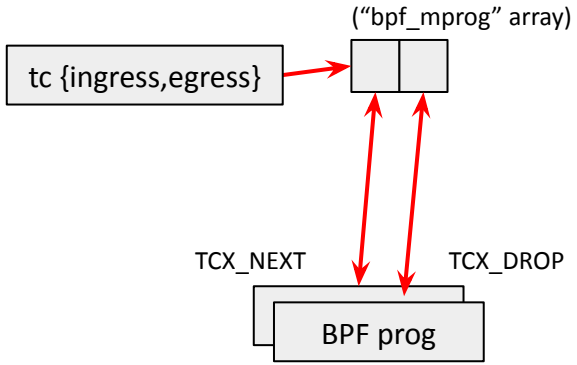
Brief Deep Dive: tcx

tcx (tc express) layer for BPF:

Old style (up to v6.6): cls_bpf



New style (v6.6 and onwards): tcx



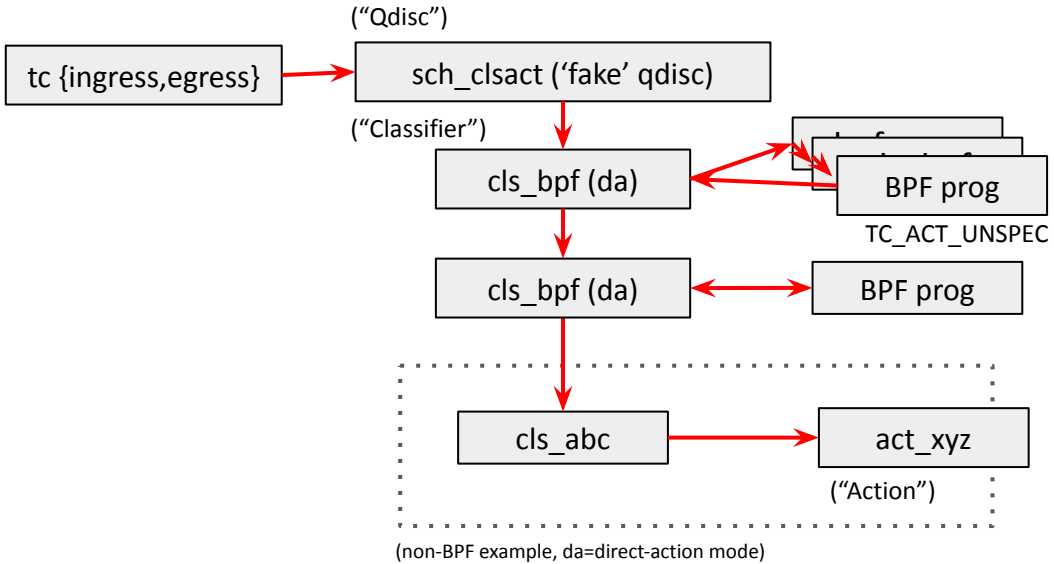
Brief Deep Dive: tcx

tcx (tc express) layer for BPF:

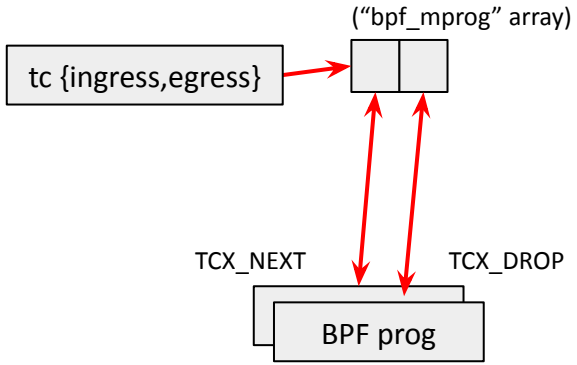
Point to BPF entry µbenchmark (cache-hot):

59 cycles → 33 cycles

Old style (up to v6.6): cls_bpf

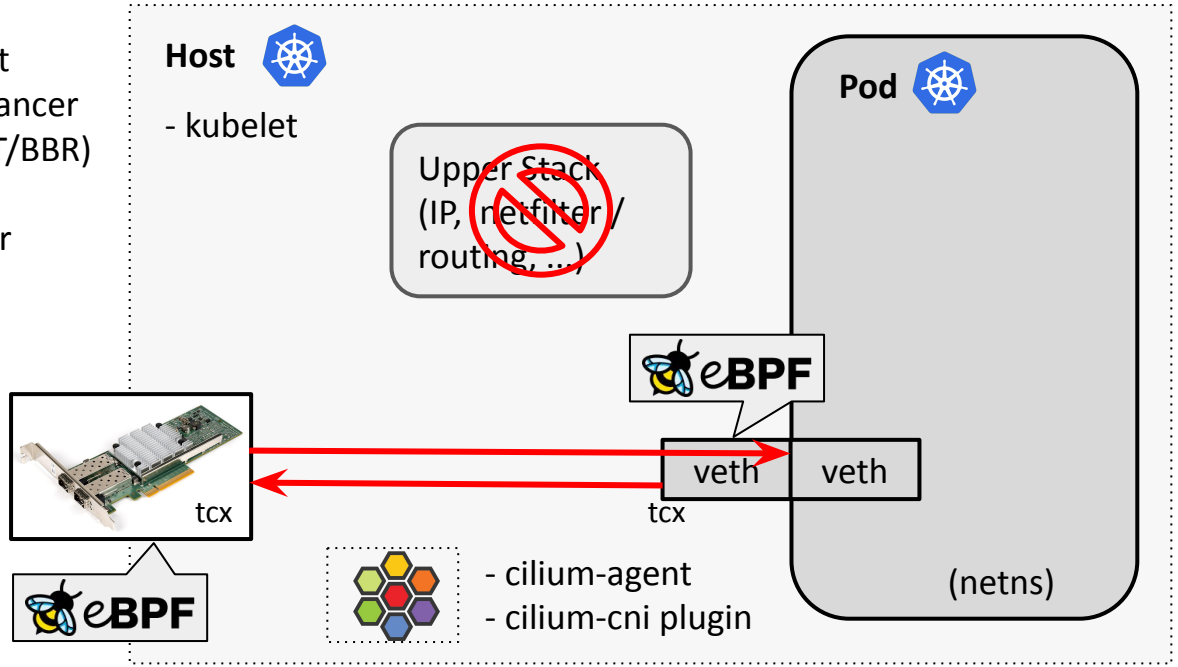


New style (v6.6 and onwards): tcx



Cilium Datapath Architecture (journey 2019 - today):

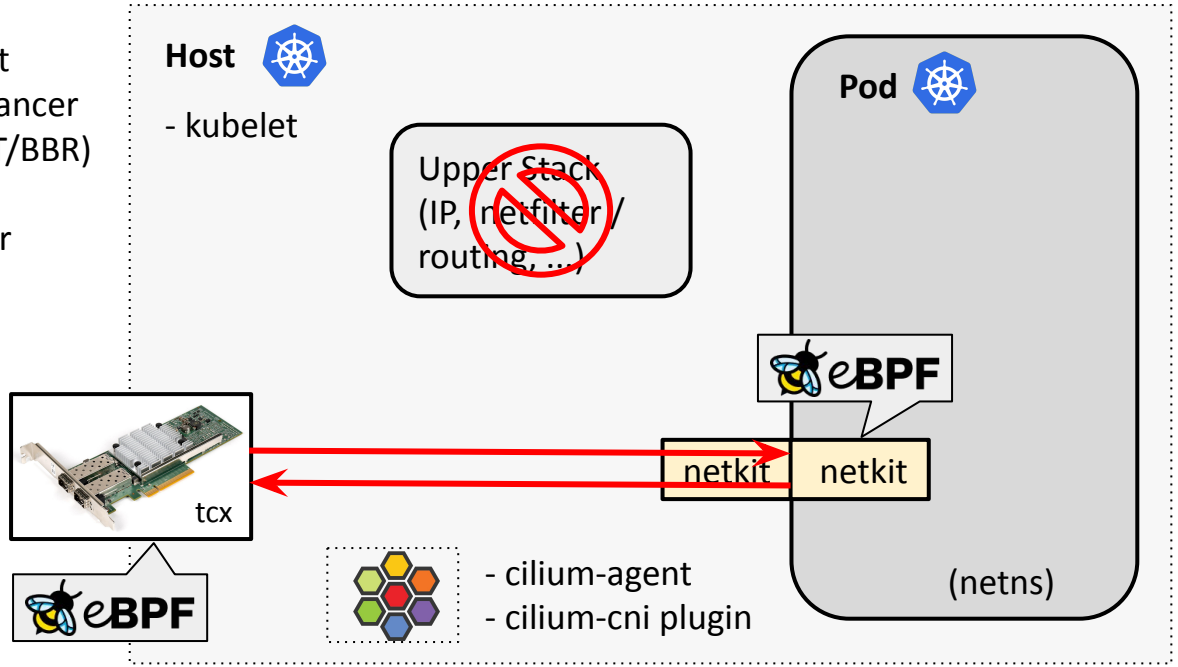
Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
- tcx-based BPF datapath layer



Kernel 6.6: tcx infrastructure and BPF link support (Daniel Borkmann)

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
- tcx-based BPF datapath layer
- netkit devices for Pods

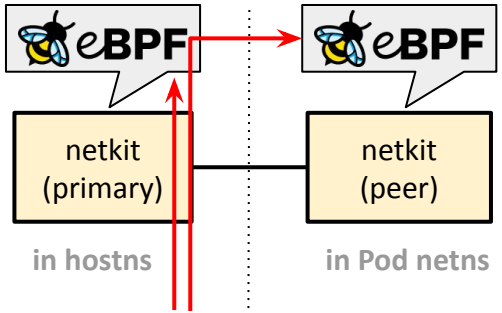


Kernel 6.7: netkit devices (Daniel Borkmann, Nikolay Aleksandrov), **LWN coverage:** The BPF-programmable network device

Brief Deep Dive: veth-replacement for Pods

netkit programmable virtual devices for BPF:

- Going forward, Cilium's CNI code will set up netkit devices for Pods instead of veth !
- Merged and will be released with Linux kernel v6.7 onwards
- BPF program via bpf_mprog is part of the driver's xmit routine, allowing fast egress netns switch
- Configurable as L3 device (default) or L2 device plus default drop-all if no BPF is attached
- Currently in production testing at Meta & Bytedance



Manages BPF programs on primary and peer device

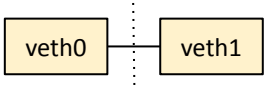
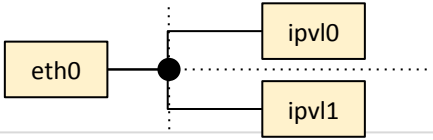
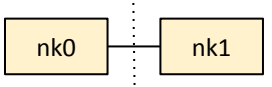
BPF programs inside the Pod inaccessible, only configurable via primary device

author	Daniel Borkmann <daniel@iogearbox.net> 2023-10-24 23:48:58 +0200
committer	Martin KaFai Lau <martin.lau@kernel.org> 2023-10-24 16:06:03 -0700
commit	35dfaad7188cdc043fde31709c796f5a692ba2bd (patch)
tree	53a88f1799ac38892434318a47278de4a255bc19
parent	42d31dd601fa43b9afdf069d1ba410b2306a4c76 (diff)
download	linux-35dfaad7188cdc043fde31709c796f5a692ba2bd.tar.gz

netkit, bpf: Add bpf programmable net device

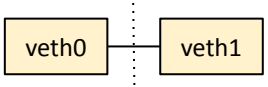
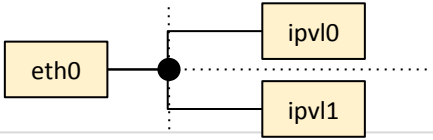
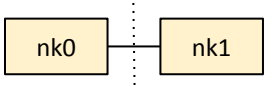
```
This work adds a new, minimal BPF-programmable device called "netkit"
(former PoC code-name "meta") we recently presented at LSF/MM/BPF. The
core idea is that BPF programs are executed within the drivers xmit routine
and therefore e.g. in case of containers/Pods moving BPF processing closer
to the source.
```

Brief Deep Dive: veth-replacement for Pods

	veth	ipvlan	netkit
Operation mode:	L2	L3 (or L2)	L3 (or L2)
Device “legs”:	pair (e.g. 1 host, 1 Pod) 	1 “master” device (e.g. physical device), n “slave” devices 	pair (e.g. 1 host, 1 Pod) with “primary” and “peer” device 
BPF programming:	tc(x) BPF on host device*	In host with tc(x) via “master” device (only entity in host) *	In Pod, BPF is native part of “peer” device internals
Routing:	L2 gateway (+ Host’s FIB)	ipvlan internal FIB + kernel FIB	kernel FIB e.g. bpf_fib_lookup
Problems:	Needs L2 neigh resolution, Higher overhead due to per-CPU backlog queue, native XDP support but very slow and hard to use .	Inflexible for multiple physical devices & troubleshooting, cumbersome to program BPF on “master”. ipvlan needs to be operated in L3/private mode for Pod policy enforcement.	Still one device per Pod inside host, for some use-cases the host device can be removed fully (wip).

(* It needs to be inside host so that BPF programs cannot be detached from app inside Pod)

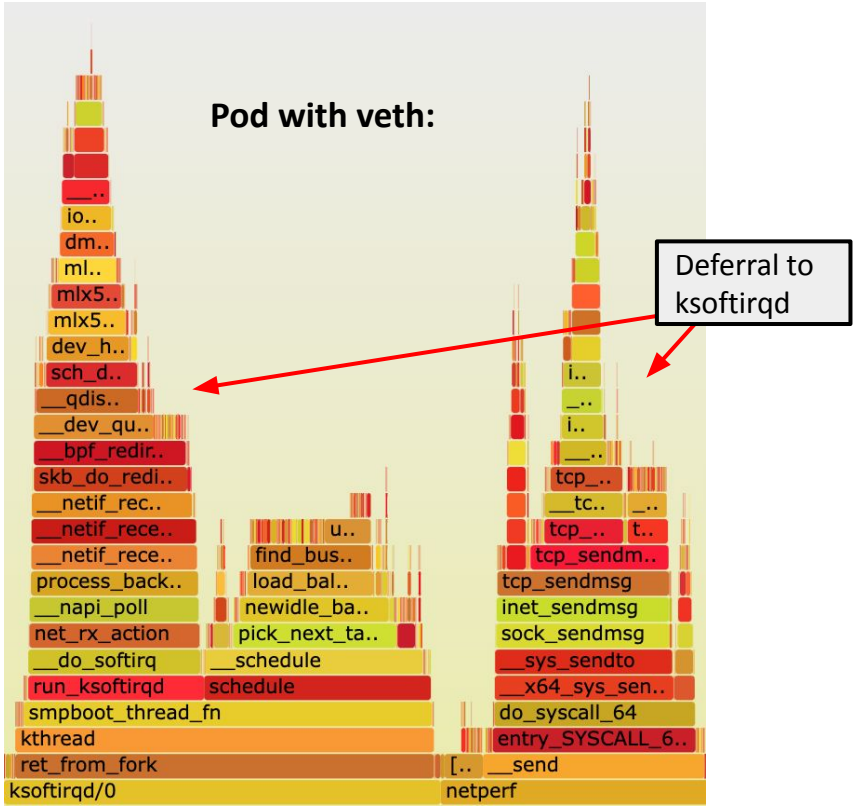
Brief Deep Dive: veth-replacement for Pods

	veth	ipvlan	netkit
Operation mode:	L2	L3 (or L2)	
Device “legs”:	pair (e.g. 1 host, 1 Pod) 	1 “master” device (e.g. phys device), n “slave” devices 	host, 1 Pod) with “primary” and “peer” device 
BPF programming:	tc(x) BPF on host device*	In host with tc(x) via “master” device (only entity in host) *	In Pod, BPF is native part of “peer” device internals
Routing:	L2 gateway (+ Host’s FIB)	ipvlan internal FIB + kernel FIB	kernel FIB e.g. bpf_fib_lookup
Problems:	Needs L2 neigh resolution, Higher overhead due to per-CPU backlog queue, native XDP support but very slow and hard to use .	Inflexible for multiple physical devices & troubleshooting, cumbersome to program BPF on “master”. ipvlan needs to be operated in L3/private mode for Pod policy enforcement.	Still one device per Pod inside host, for some use-cases the host device can be removed fully (wip).

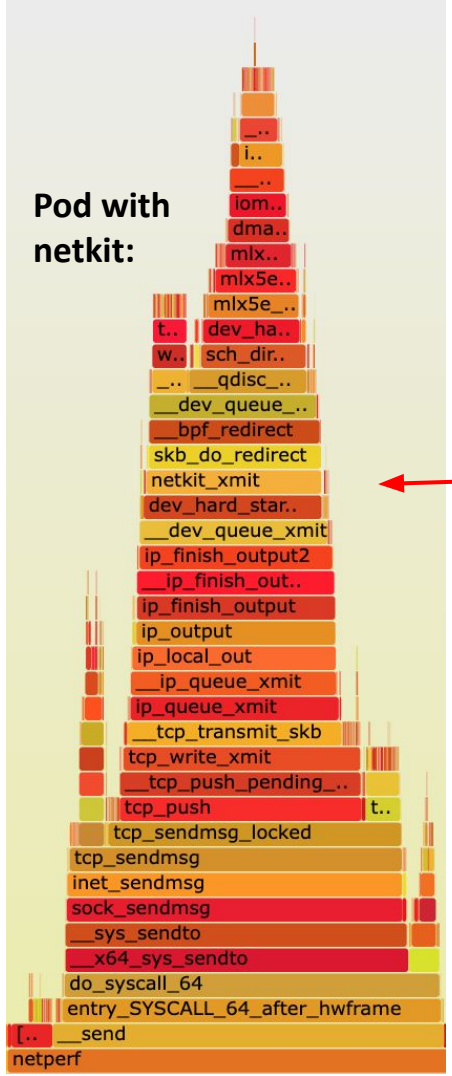
tl;dr: Takes best of both worlds!

(* It needs to be inside host so that BPF programs cannot be detached from app inside Pod)

veth vs netkit: backlog queue



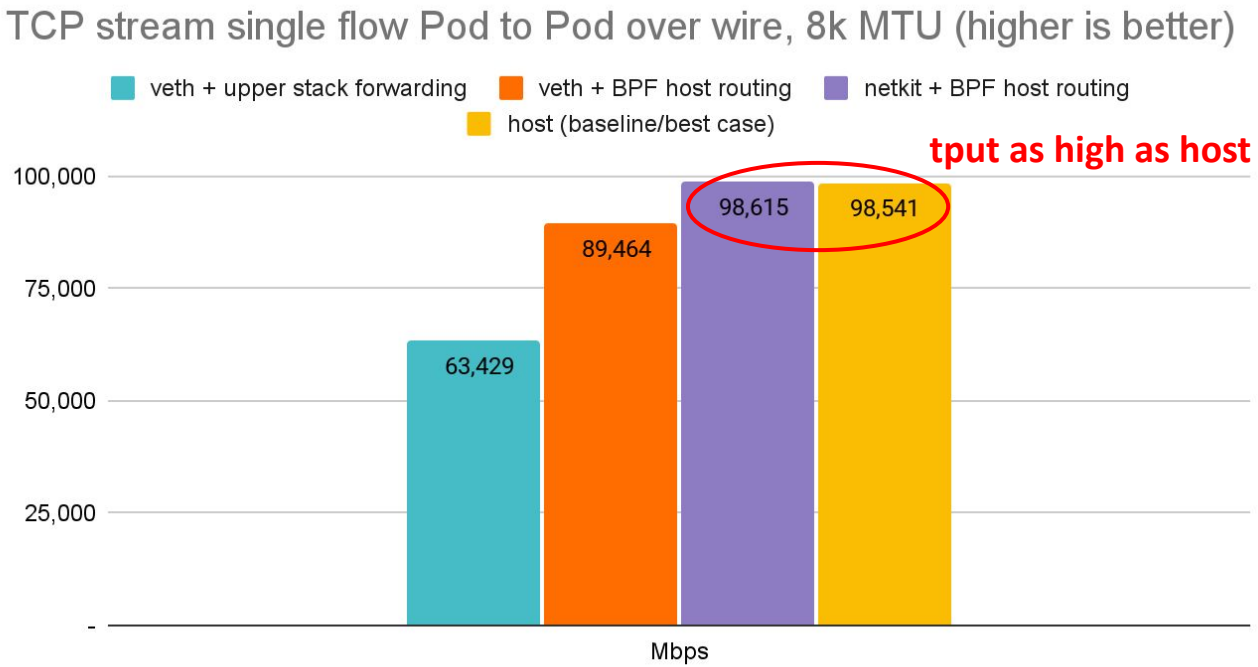
Deferral to ksoftirqd



Remains in process context all the way, leading to better process scheduler decisions.

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
- tcx-based BPF datapath layer
- netkit devices for Pods

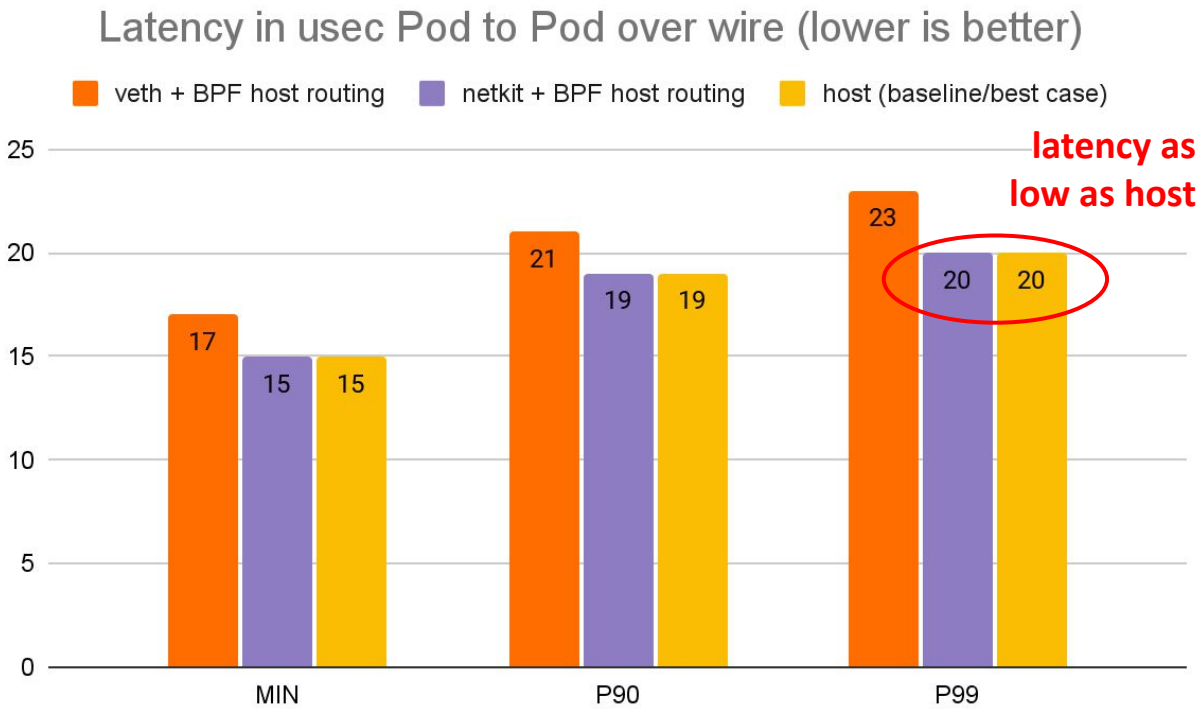


* 8264 MTU for data page alignment in GRO

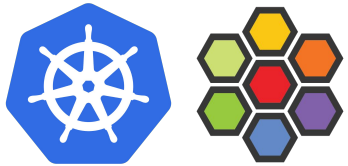
Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 8264 MTU
Receiver: taskset -a -c <core> tcp_mmap -s (non-zerocopy mode), Sender: taskset -a -c <core> tcp_mmap -H <dst host>

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
- tcx-based BPF datapath layer
- netkit devices for Pods



Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off
netperf -t TCP_RR -H <remote pod> -- -O MIN_LATENCY,P90_LATENCY,P99_LATENCY,THROUGHPUT

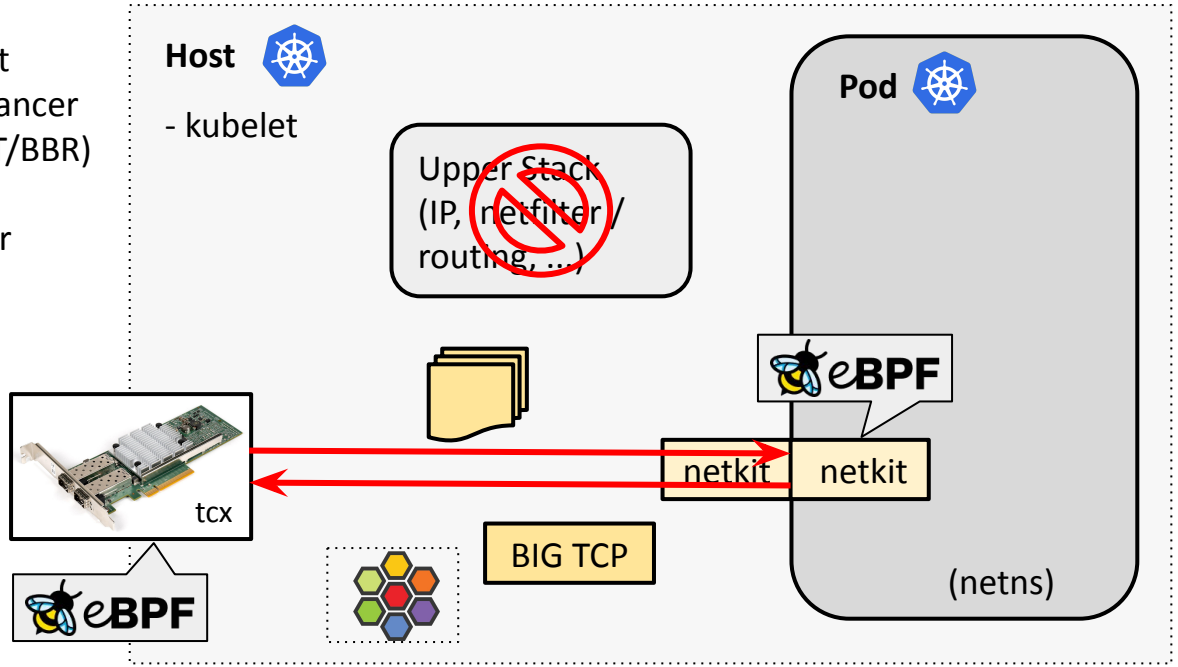


Now with zero-overhead Pods, can the networking stack generally push even further?

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
- tcx-based BPF datapath layer
- netkit devices for Pods

Pushing even further:
- BIG TCP (IPv4/IPv6)



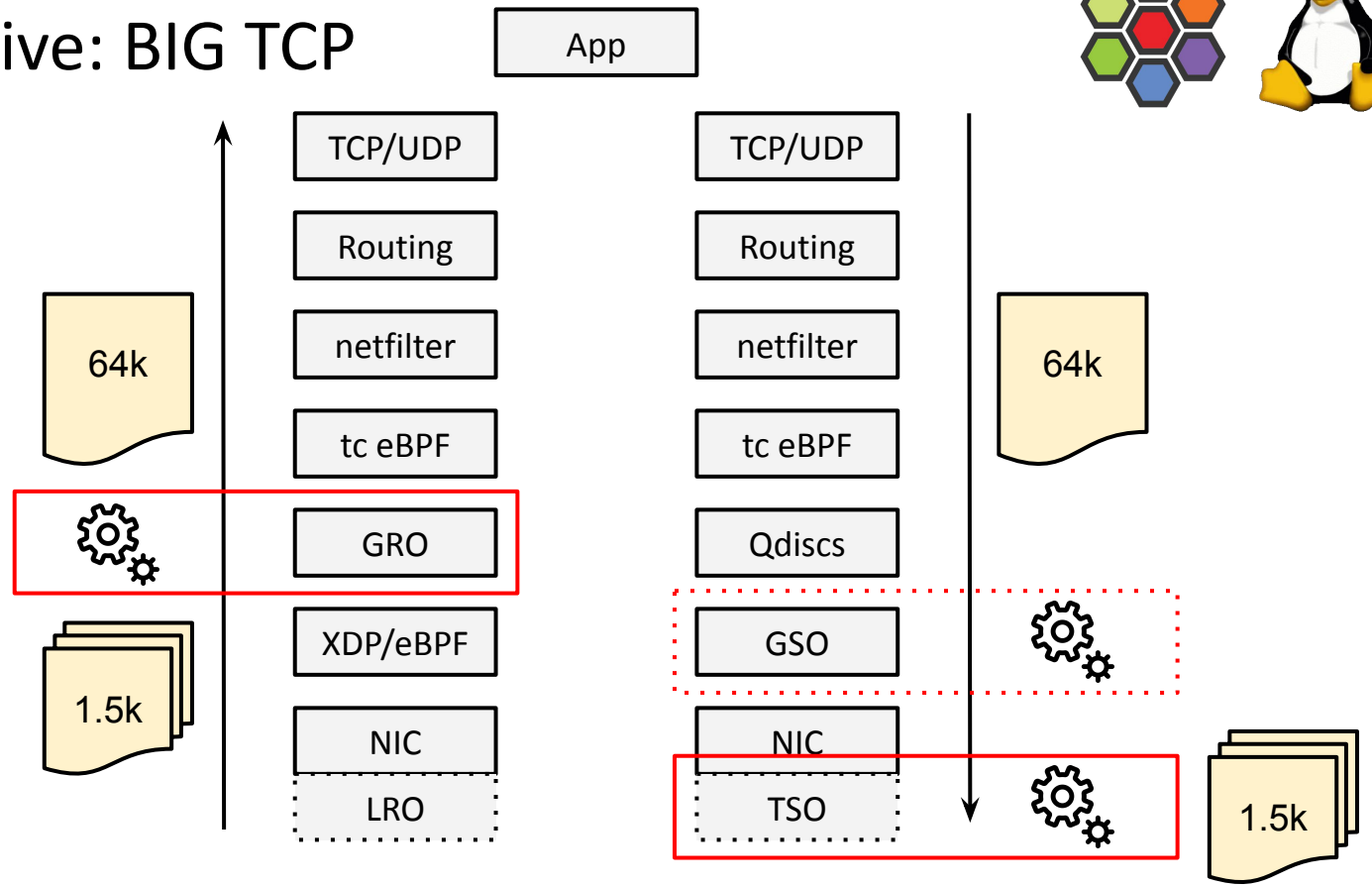
Kernel 5.19 & 6.3: BIG TCP for IPv6 and IPv4 (Eric Dumazet, Coco Li, Xin Long)

Brief Deep Dive: BIG TCP

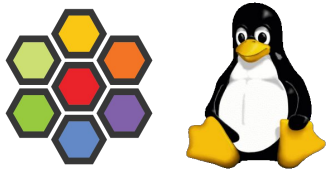
tldr: “TCP is slooooow, use BIG packets!”

- Developed by Google to prepare the Linux kernel’s TCP stack for 200/400+ Gbit/s NIC speeds
- BIG TCP for IPv6 merged in v5.19, for IPv4 merged in v6.3 kernel
- Deployed in Google fleet in production for IPv6 traffic
- Cilium supports BIG TCP for both address families, probes drivers and configures all Cilium managed devices/Pods
- No changes to the network such as MTU needed, this affects only local host (GSO/GRO engine)

Brief Deep Dive: BIG TCP

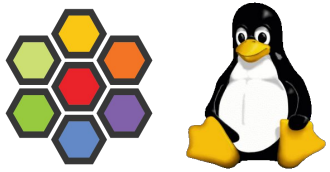


Brief Deep Dive: BIG TCP

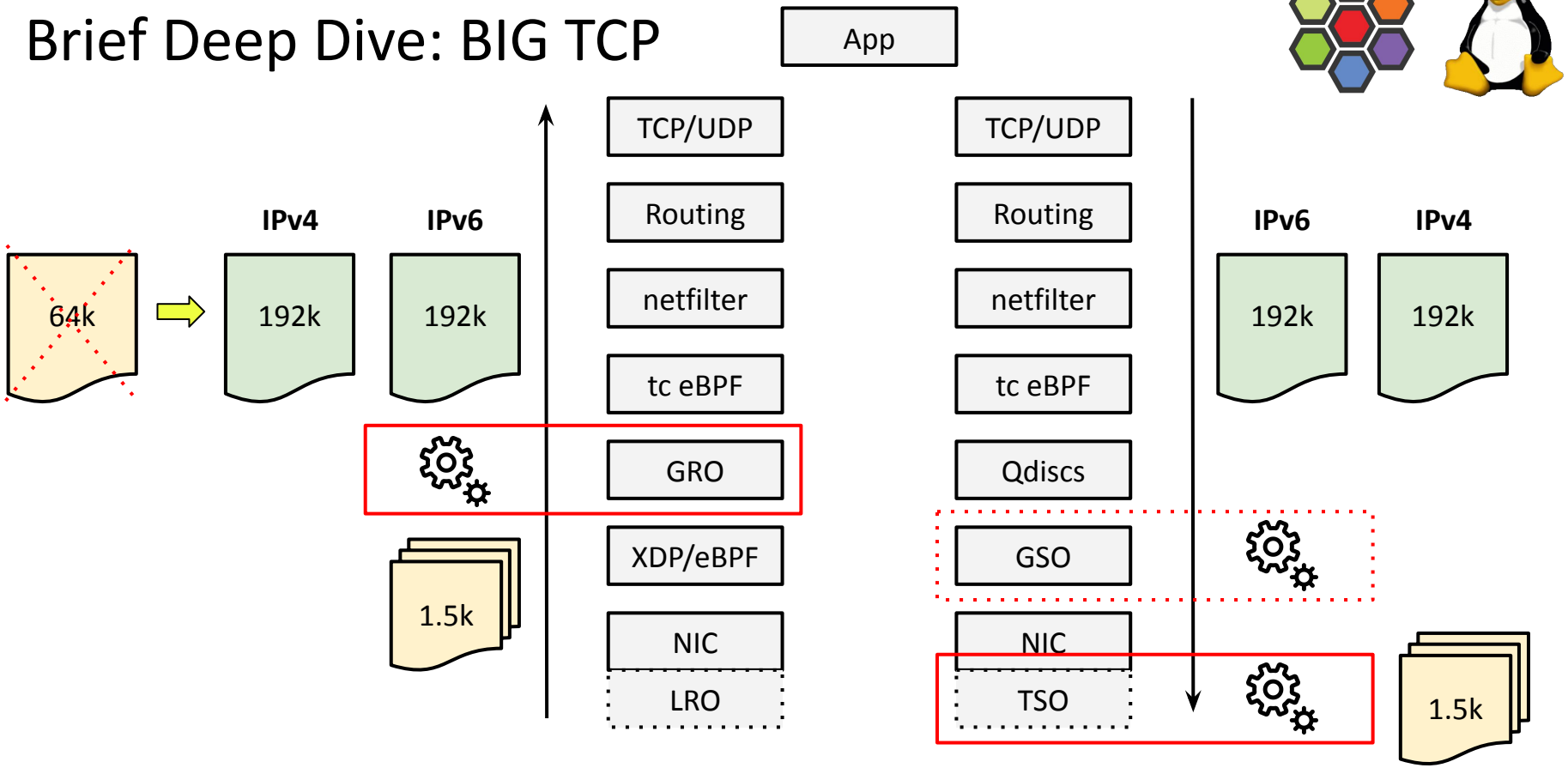


TSO segments TCP super-sized packet in NIC/HW, and GRO on receiver gets packet train, reconstructs super-sized packet.

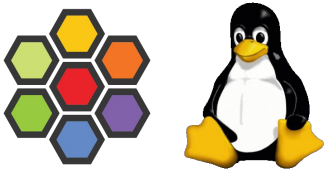
Brief Deep Dive: BIG TCP



Brief Deep Dive: BIG TCP



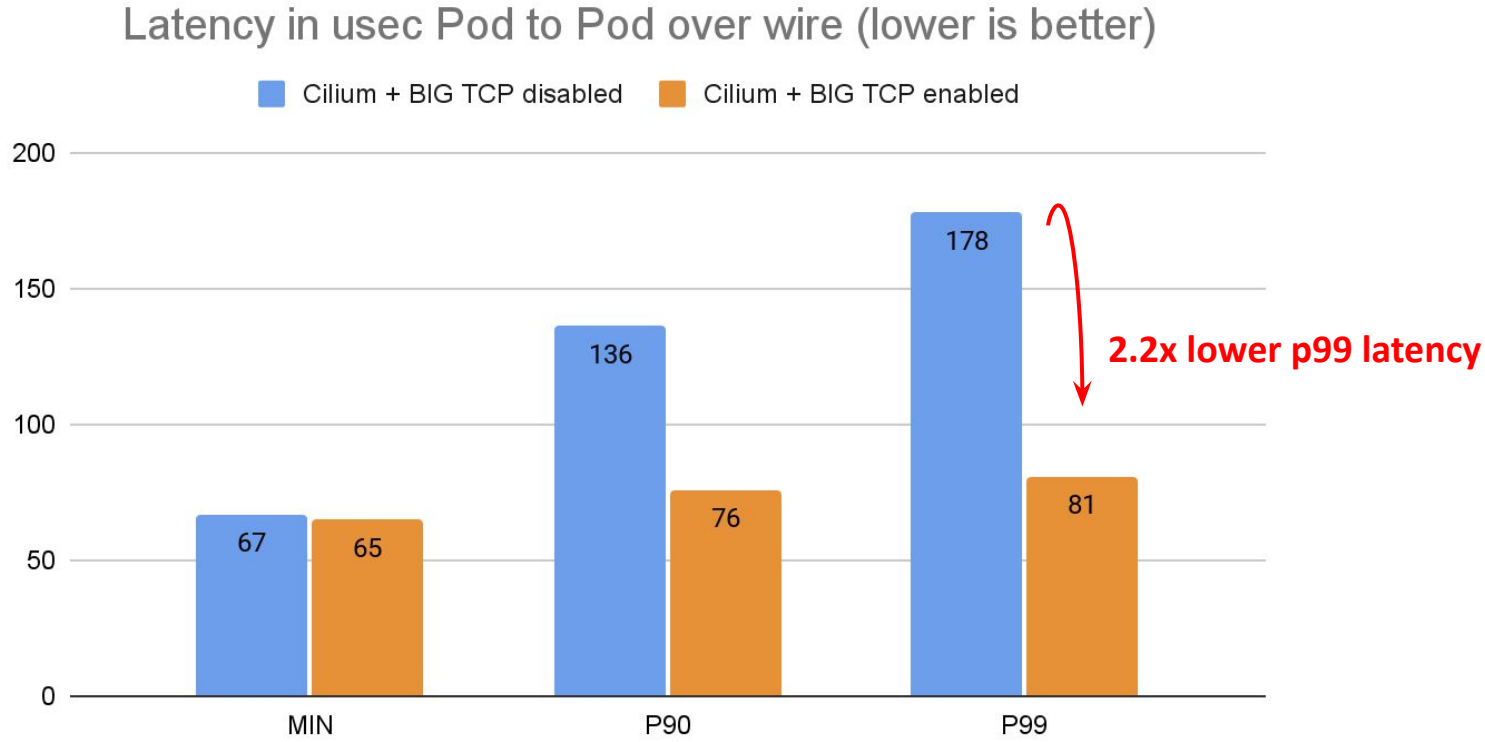
Brief Deep Dive: BIG TCP



Cilium 1.14 & BIG TCP:

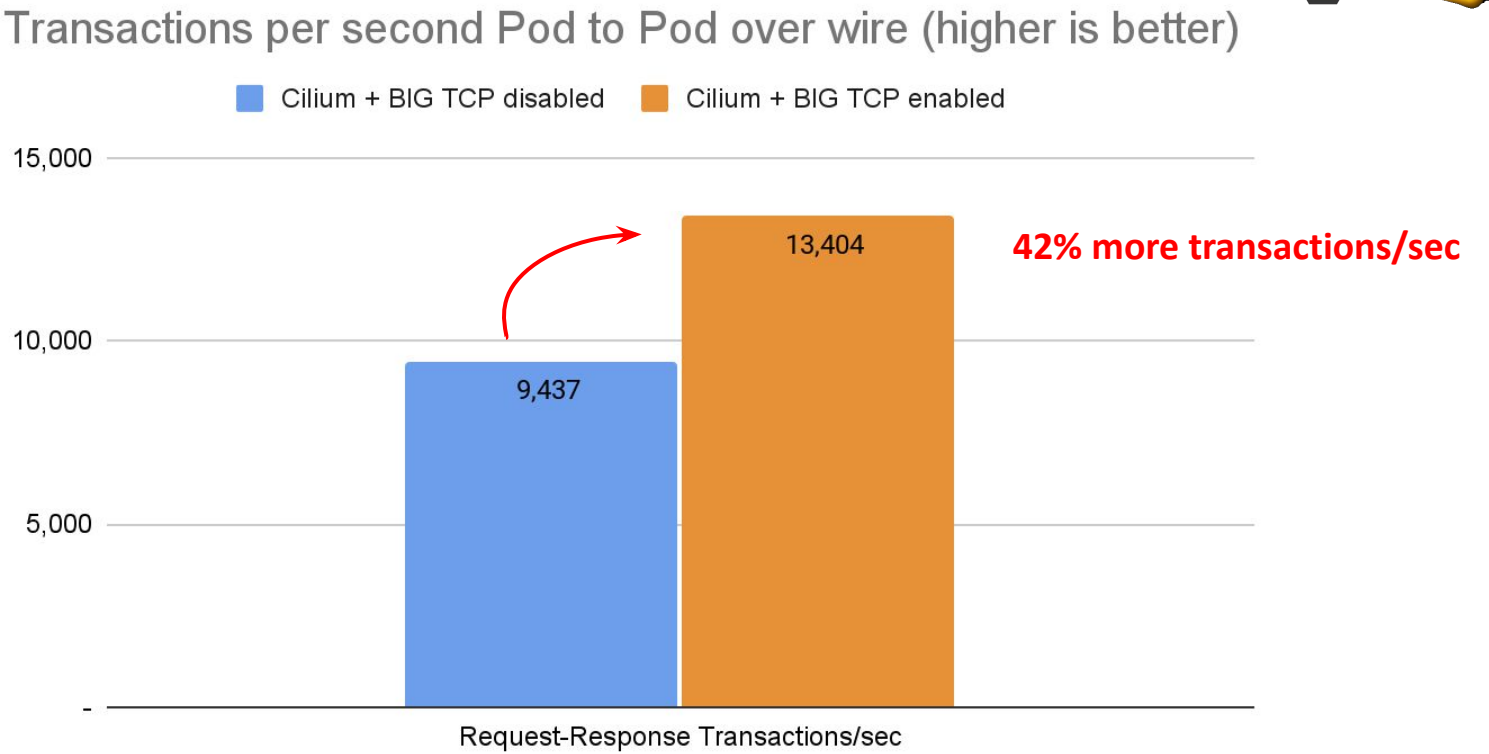
- Supports BIG TCP for both IPv4 & IPv6
- Sets GSO/GRO max limit to 48 pages (192k) which we found to be performance sweet-spot
- Implements max TSO probing for drivers not supporting 192k, e.g. ice has 128k (32 pages)

Cilium & BIG TCP



Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver
netperf -t TCP_RR -H <remote pod> -- -r 80000,80000 -O MIN_LATENCY,P90_LATENCY,P99_LATENCY,THROUGHPUT

Cilium & BIG TCP

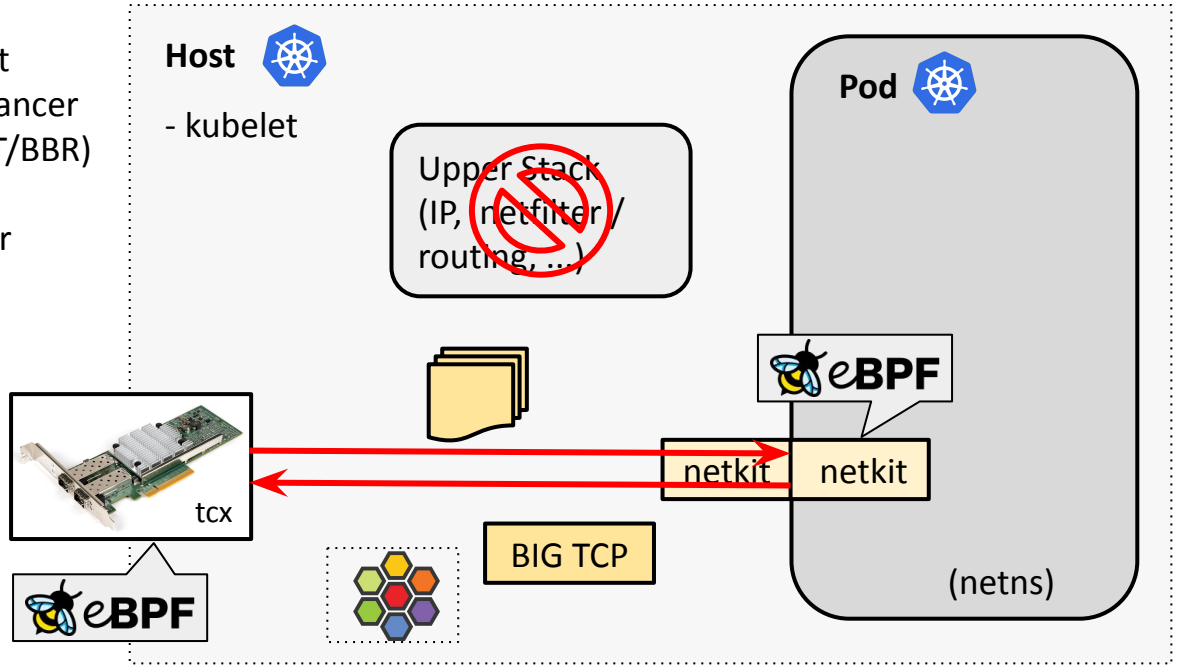


Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver
netperf -t TCP_RR -H <remote pod> -- -r 80000,80000 -O MIN_LATENCY,P90_LATENCY,P99_LATENCY,THROUGHPUT

Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
- tcx-based BPF datapath layer
- netkit devices for Pods

Pushing even further:
- BIG TCP (IPv4/IPv6)

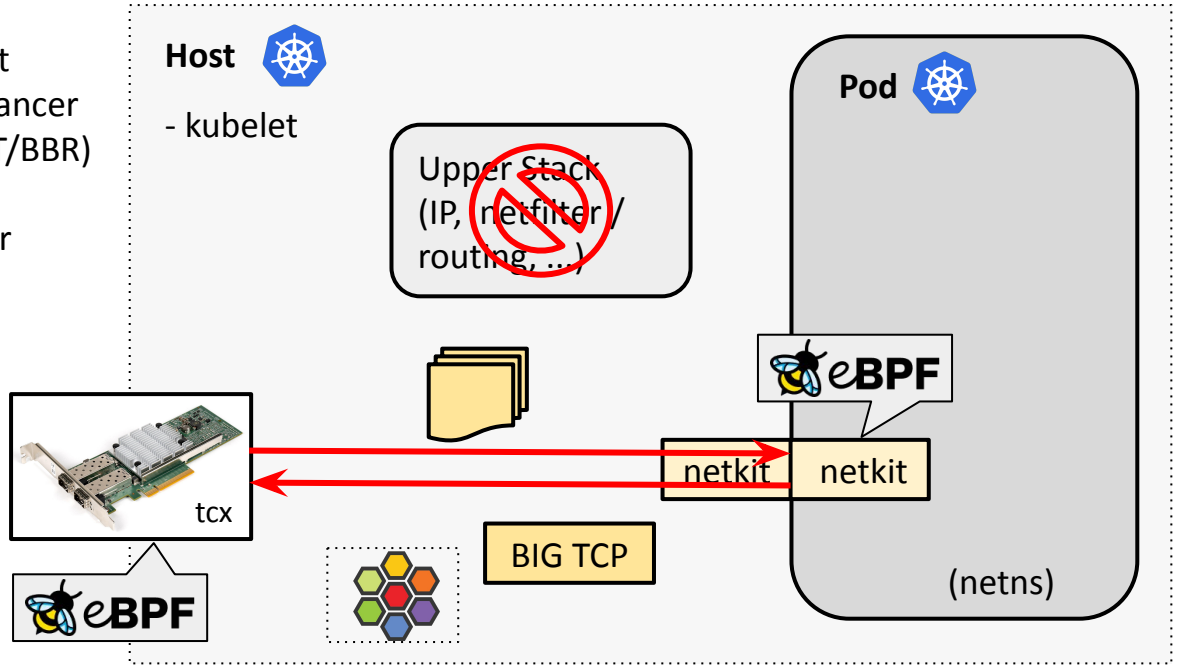


Cilium Datapath Architecture (journey 2019 - today):

Building Blocks:
- BPF kube-proxy replacement
- XDP-based Service Load-Balancer
- Bandwidth Manager (fq/EDT/BBR)
- BPF Host Routing
- tcx-based BPF datapath layer
- netkit devices for Pods

Pushing even further:
- BIG TCP (IPv4/IPv6)

Future integration:
- TCP usec resolution (v6.7)
- BBRv3 (once upstream)



Back to our Experiment

Conclusions:

- Significant performance gains can be achieved with our recent eBPF & Cilium work to **completely remove** a Pod's **netns networking** data path **overhead**

Back to our Experiment

Conclusions:

- Significant performance gains can be achieved with our recent eBPF & Cilium work to **completely remove** a Pod's **netns networking** data path **overhead**
- **BIG TCP and Cilium**'s integration enable K8s clusters to **better deal with >100G NICs**
  - Without application or network MTU changes necessary
  - Notable efficiency improvements also for <= 100G NICs

Back to our Experiment

Conclusions:

- Significant performance gains can be achieved with our recent eBPF & Cilium work to **completely remove** a Pod's **netns networking** data path **overhead**
- **BIG TCP and Cilium**'s integration enable K8s clusters to **better deal with >100G NICs**
  - Without application or network MTU changes necessary
  - Notable efficiency improvements also for <= 100G NICs
- To achieve even higher throughput, application changes to utilize **TCP zero-copy** are necessary and there is **still ongoing kernel work**.

Recent discussions at netconf 2023 workshop:
- NIC & kernel support for header/data split
- BIG TCP & TCP zero-copy support
- TCP device memory for direct GPU data placement

ISOVALENT

Thank you! Questions?

github.com/cilium/cilium

cilium.io

ebpf.io

Bandwidth Manager

BPF Host Routing

BIG TCP for IPv4/IPv6

tcx as a new tc BPF datapath

netkit devices as veth replacement