KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

China 2023

KubeCon | CloudNativeCon
OPEN SOURCE SUMMIT
China 2023

CNI-agnostic network performance accelerator with eBPF

Yizhou Xu, Intel

Mengxin Liu, Alauda

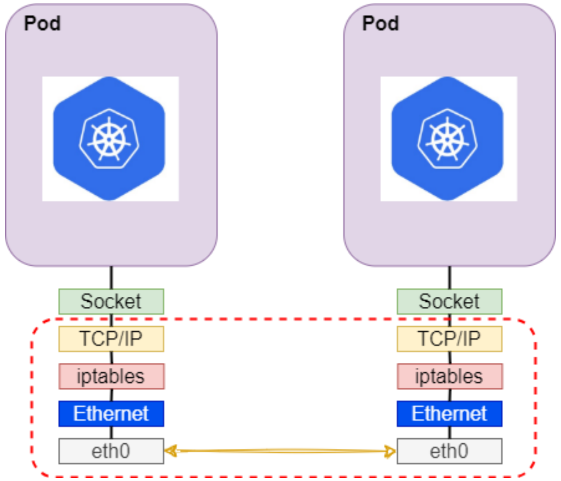
Agenda

- TCP/IP stack overhead
- eBPF background knowledge
- How to bypass Tcp/Ip with eBPF
- Performance Analysis
- Practice on KubeVirt acceleration

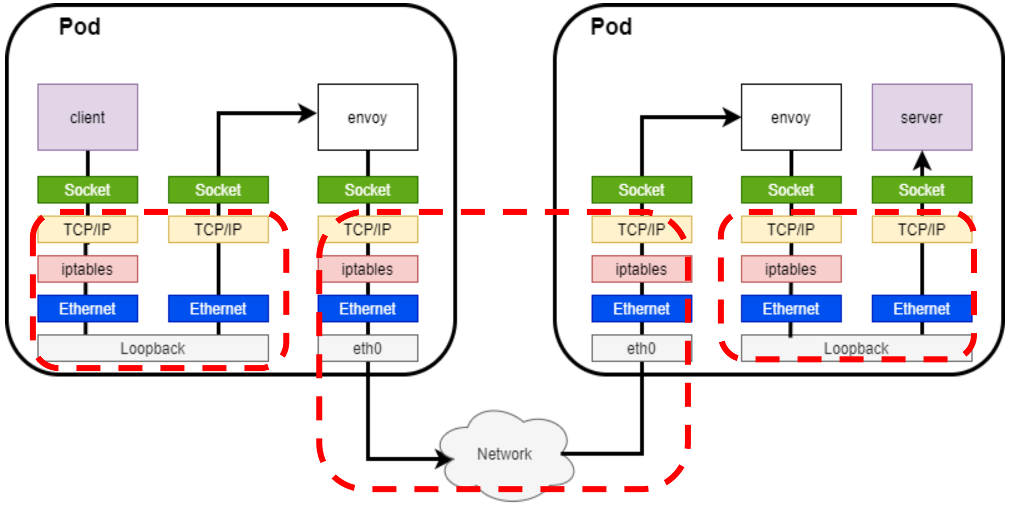
TCP/IP stack overhead

In Kubernetes, each pod has its own network stack, packet from one pod to another traverse whole stack multiple times

In particular, service mesh amplify the overhead by sidecar mode



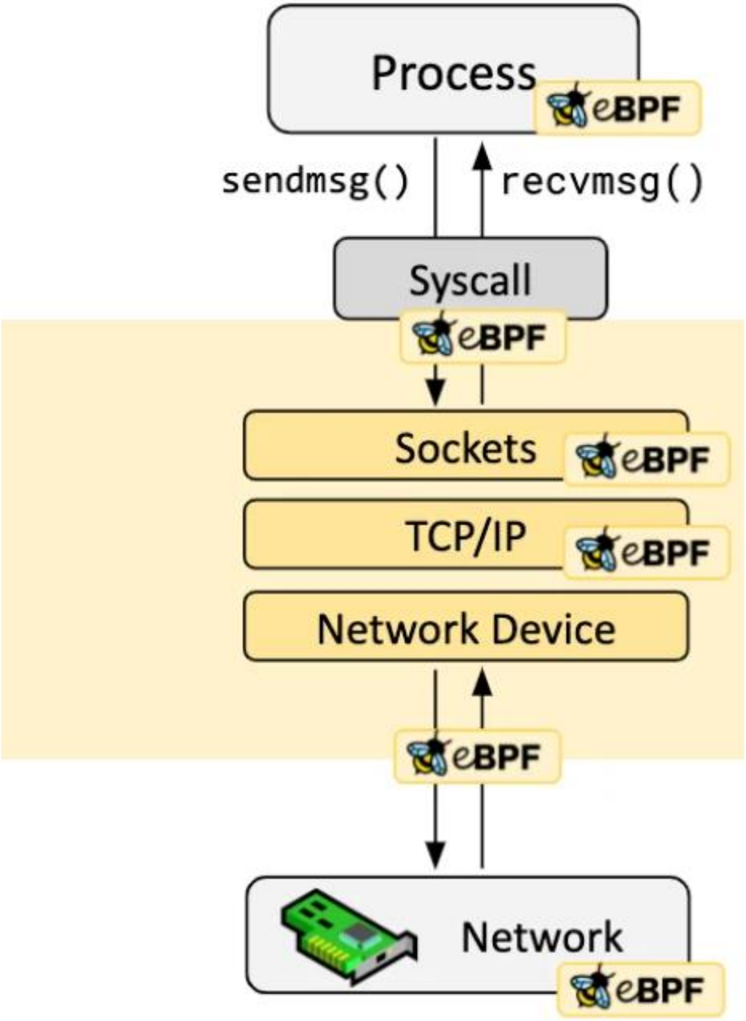
Tcp/ip overhead in Pod to Pod (same host)



Tcp/ip overhead in Service Mesh (same host)

Why use eBPF

- Work in Kernel
- Non-intrusive
- CNI agnostic
- Safety and efficient



eBPF background knowledge – Map&Prog

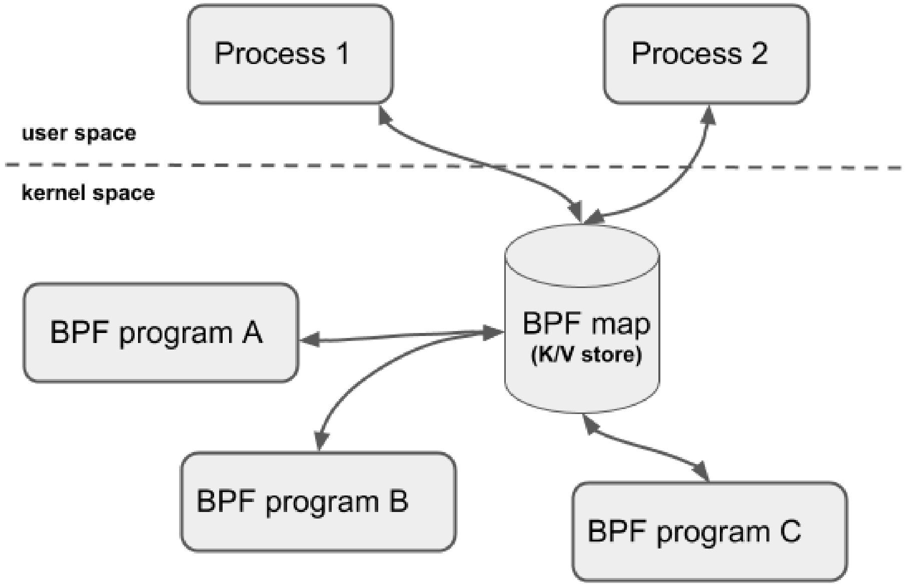
MAP

provide generic data structure for user space and kernel space communication

- ○ HASHMAP
- ○ SOCKHASH

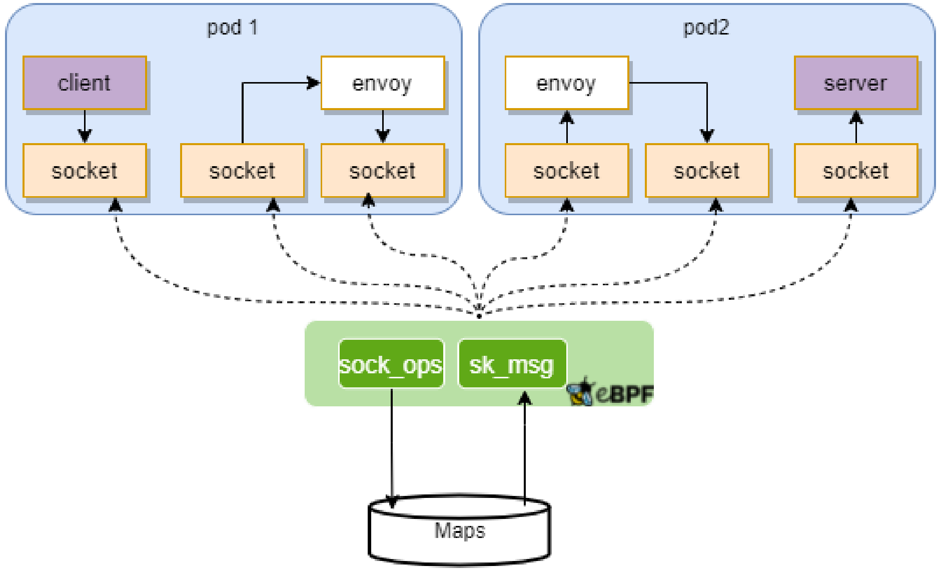
Program type

- ○ SOCK_OPS
- ○ SK_MSG

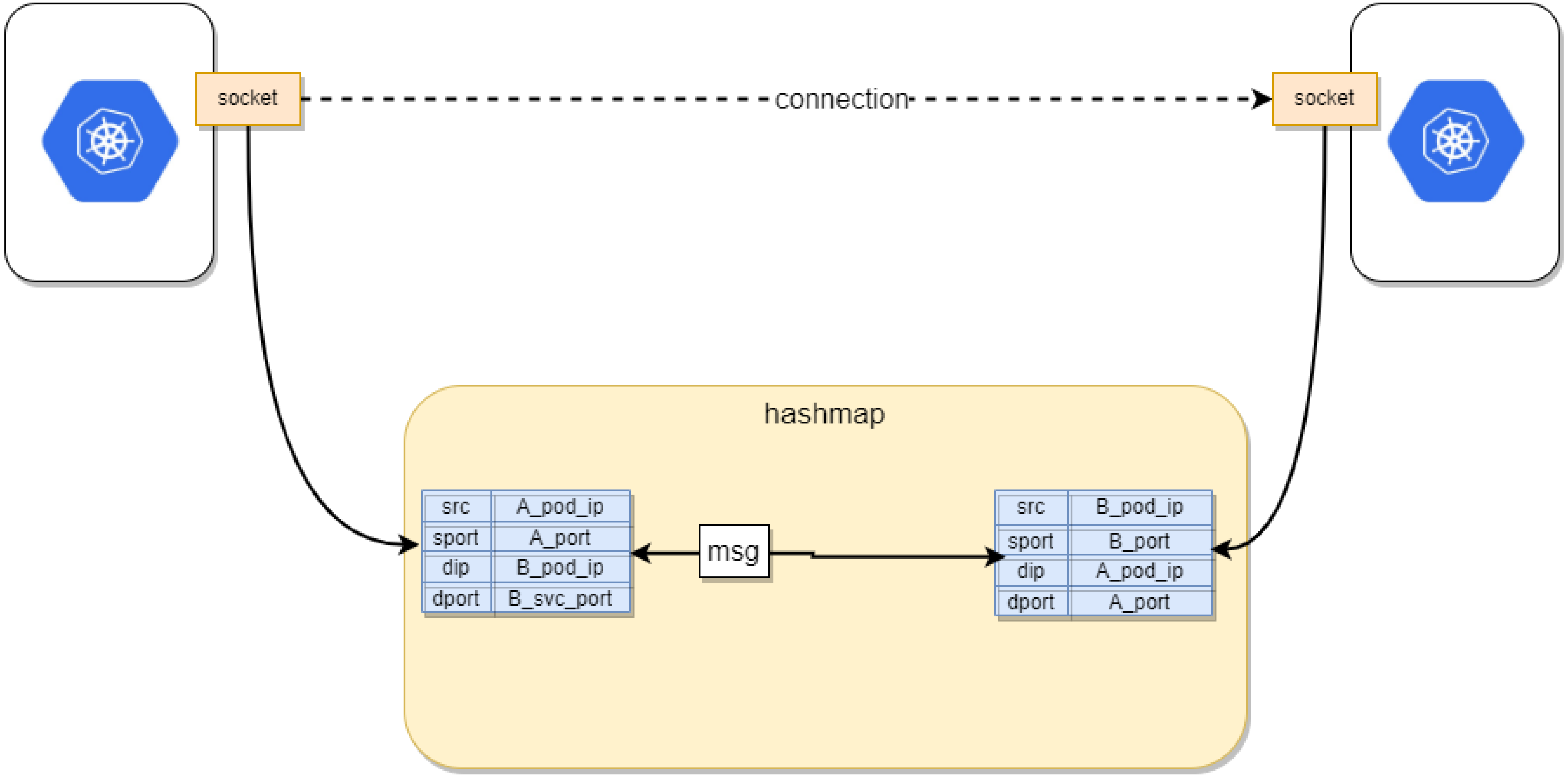


Combining MAP & Program for acceleration

- sock_ops
  - capture socket in given status, populate into map
- sk_msg
  - when there is data in socket, lookup peer socket
  - transfer data to peer



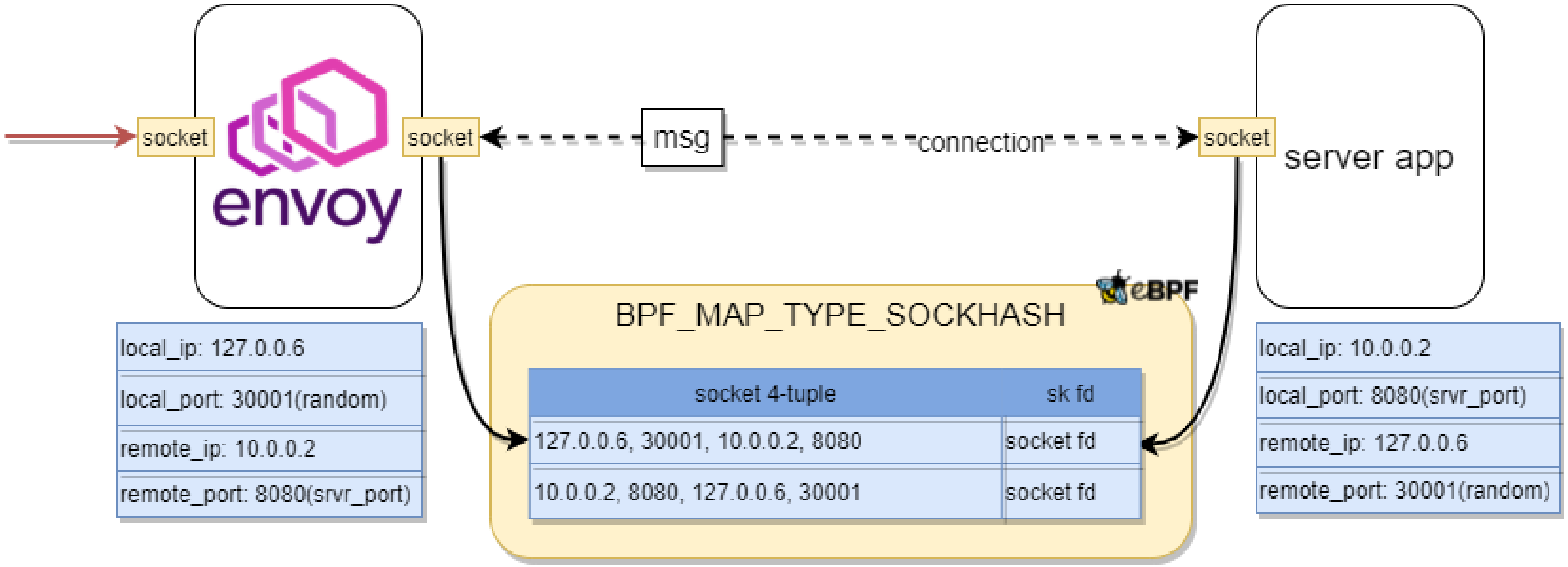
Pod to Pod acceleration(Kubernetes)



src	A_pod_ip
sport	A_port
dip	B_pod_ip
dport	B_svc_port

src	B_pod_ip
sport	B_port
dip	A_pod_ip
dport	A_port

Inbound Acceleraion(Service Mesh)



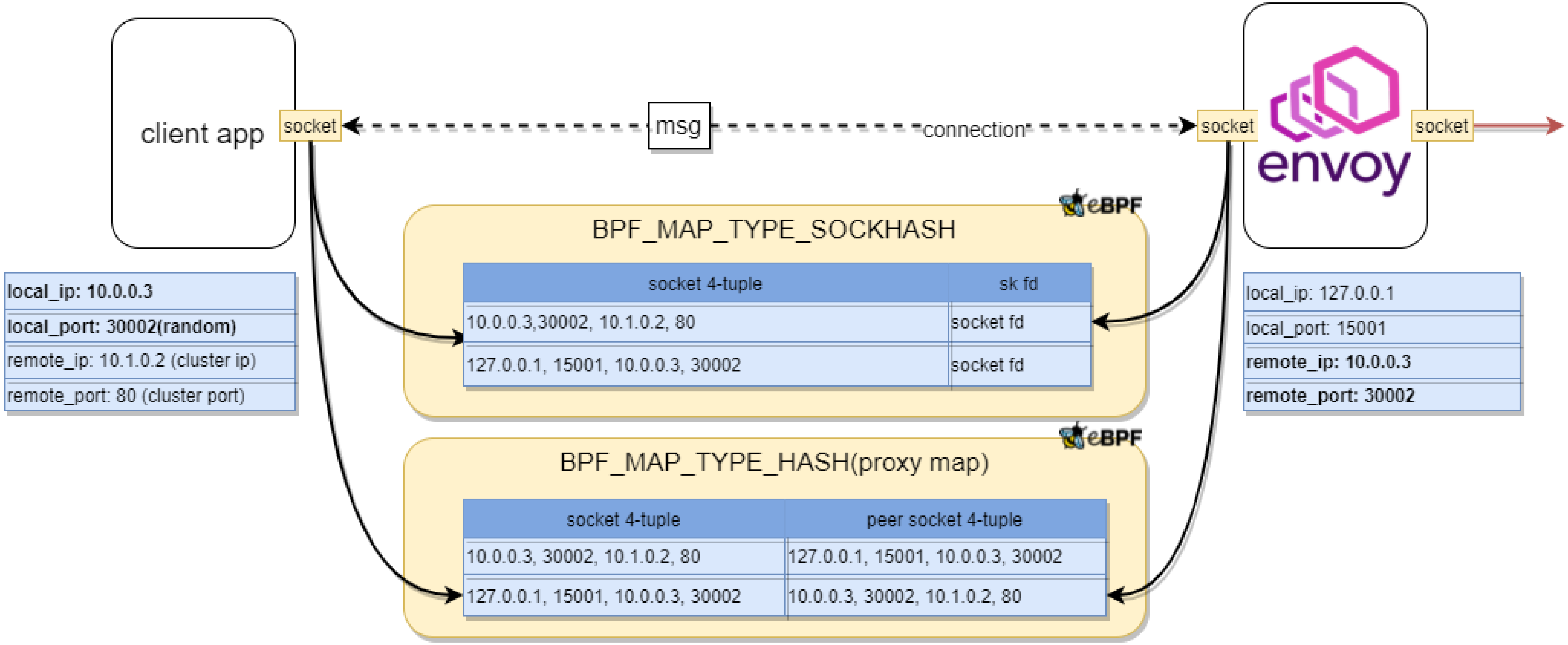
local_ip: 127.0.0.6
local_port: 30001(random)
remote_ip: 10.0.0.2
remote_port: 8080(srvr_port)

BPF_MAP_TYPE_SOCKHASH

socket 4-tuple	sk fd
127.0.0.6, 30001, 10.0.0.2, 8080	socket fd
10.0.0.2, 8080, 127.0.0.6, 30001	socket fd

local_ip: 10.0.0.2
local_port: 8080(srvr_port)
remote_ip: 127.0.0.6
remote_port: 30001(random)

Outbound Acceleration(Service Mesh)



local_ip: 10.0.0.3
local_port: 30002(random)
remote_ip: 10.1.0.2 (cluster ip)
remote_port: 80 (cluster port)

BPF_MAP_TYPE_SOCKHASH

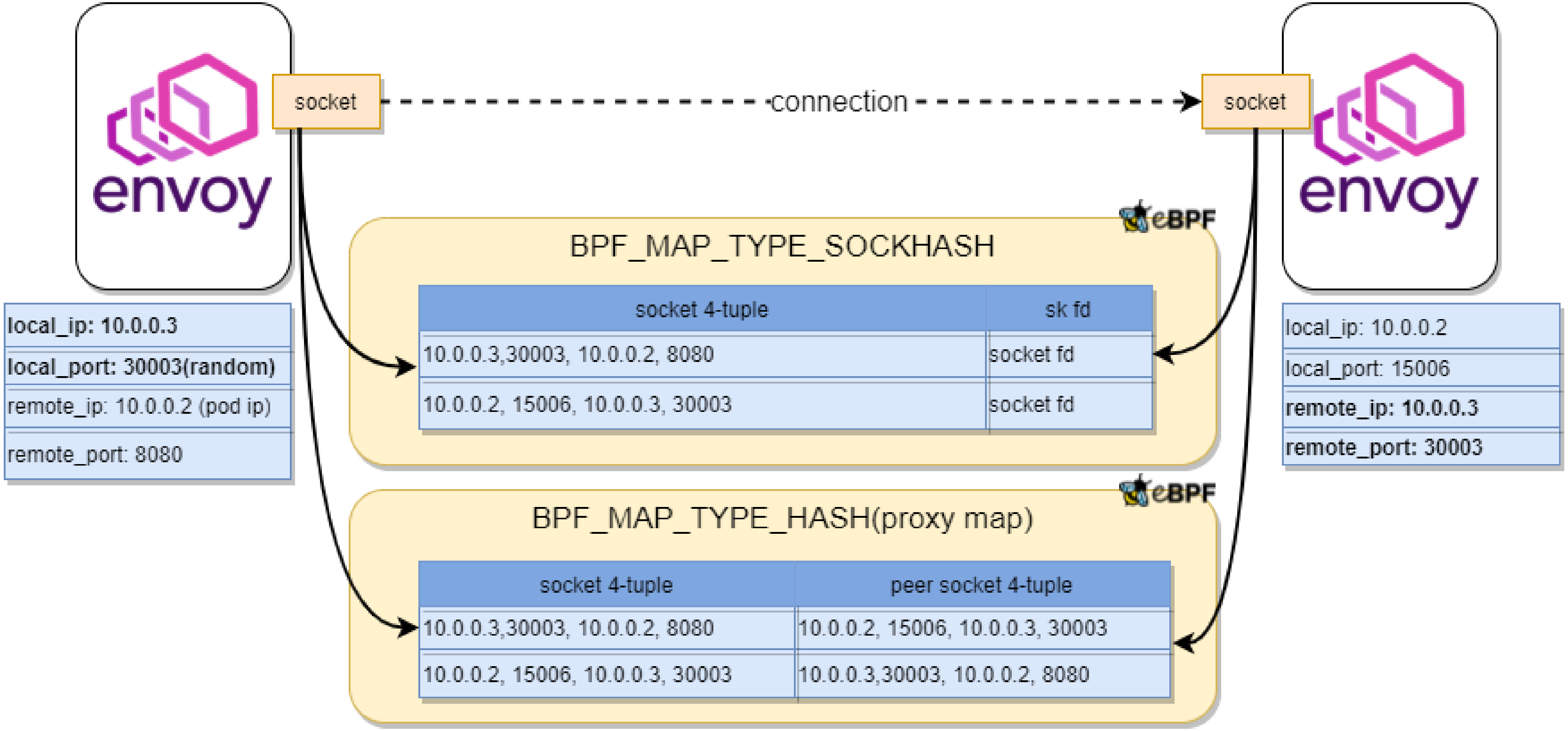
socket 4-tuple	sk fd
10.0.0.3,30002, 10.1.0.2, 80	socket fd
127.0.0.1, 15001, 10.0.0.3, 30002	socket fd

BPF_MAP_TYPE_HASH(proxy map)

socket 4-tuple	peer socket 4-tuple
10.0.0.3, 30002, 10.1.0.2, 80	127.0.0.1, 15001, 10.0.0.3, 30002
127.0.0.1, 15001, 10.0.0.3, 30002	10.0.0.3, 30002, 10.1.0.2, 80

local_ip: 127.0.0.1
local_port: 15001
remote_ip: 10.0.0.3
remote_port: 30002

Envoy to Envoy Acceleration (same host)



local_ip: 10.0.0.3
local_port: 30003(random)
remote_ip: 10.0.0.2 (pod ip)
remote_port: 8080

local_ip: 10.0.0.2
local_port: 15006
remote_ip: 10.0.0.3
remote_port: 30003

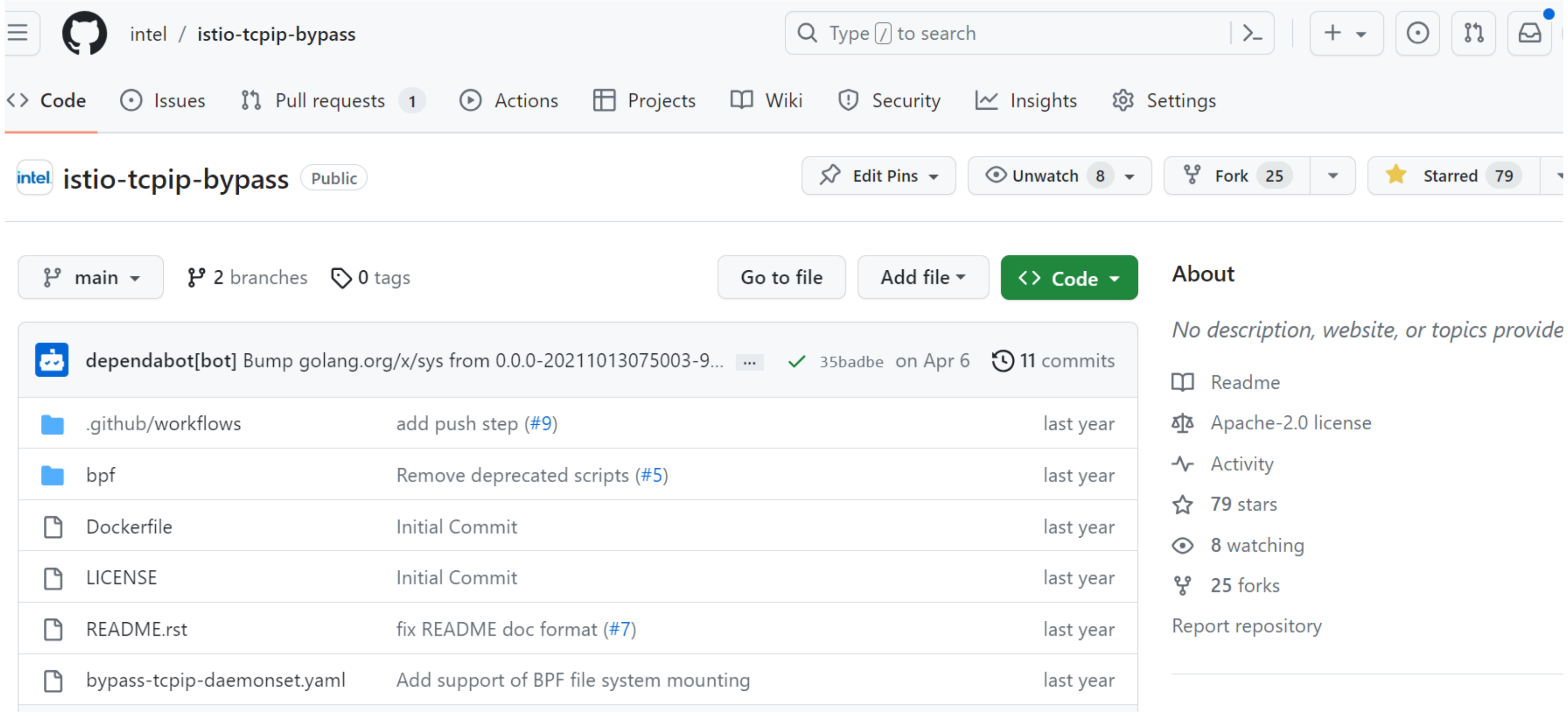
BPF_MAP_TYPE_SOCKHASH

socket 4-tuple	sk fd
10.0.0.3,30003, 10.0.0.2, 8080	socket fd
10.0.0.2, 15006, 10.0.0.3, 30003	socket fd

BPF_MAP_TYPE_HASH(proxy map)

socket 4-tuple	peer socket 4-tuple
10.0.0.3,30003, 10.0.0.2, 8080	10.0.0.2, 15006, 10.0.0.3, 30003
10.0.0.2, 15006, 10.0.0.3, 30003	10.0.0.3,30003, 10.0.0.2, 8080

It’s opensource!



dependabot[bot] Bump golang.org/x/sys from 0.0.0-20211013075003-9... ✓ 35badbe on Apr 6 11 commits

.github/workflows	add push step (#9)	last year
bpf	Remove deprecated scripts (#5)	last year
Dockerfile	Initial Commit	last year
LICENSE	Initial Commit	last year
README.rst	fix README doc format (#7)	last year
bypass-tcpip-daemonset.yaml	Add support of BPF file system mounting	last year

About

No description, website, or topics provide

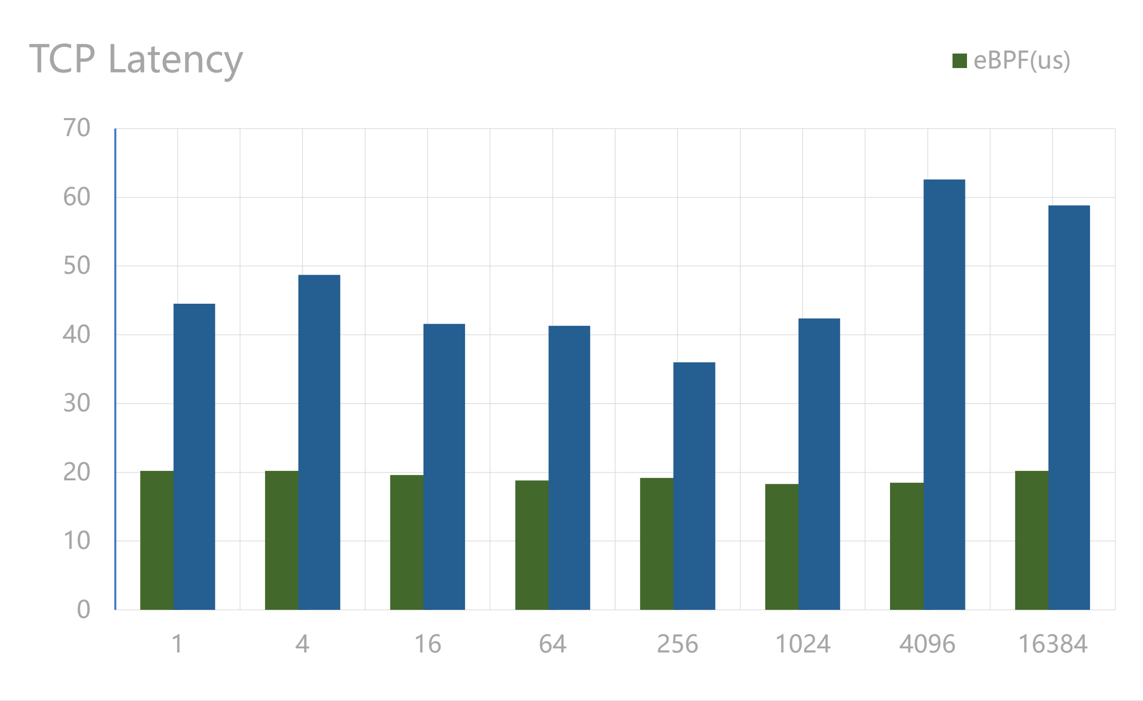
- Readme
- Apache-2.0 license
- Activity
- 79 stars
- 8 watching
- 25 forks

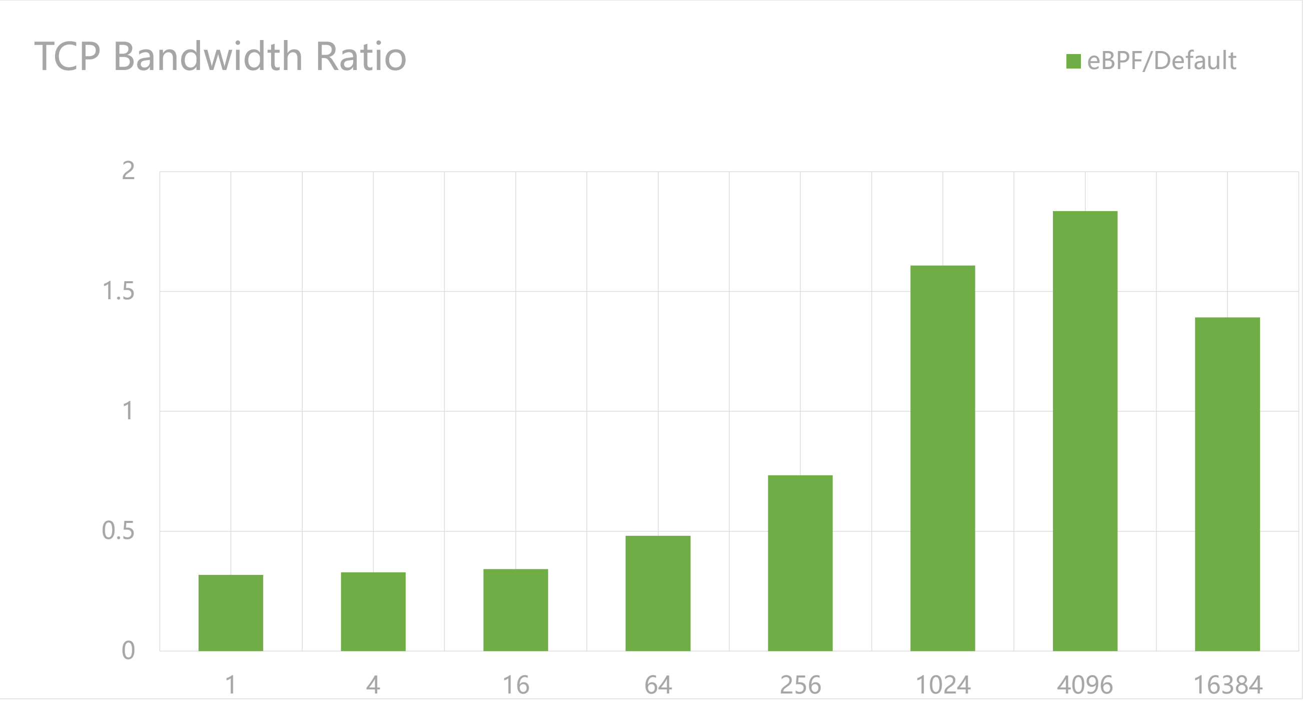
Report repository

REPO: https://github.com/intel/istio-tcpip-bypass

Benchmark

- ➢ Deploy two test Pods in the same Node
- ➢ Use qperf to benchmark TCP latency/bandwidth with packet sizes from 1byte to 16KB
  - ➢ qperf -t 60 100.64.0.3 -ub -oo msg_size:1:16K:*4 -vu tcp_lat tcp_bw
- ➢ Compare with the benchmark when the optimization is disabled



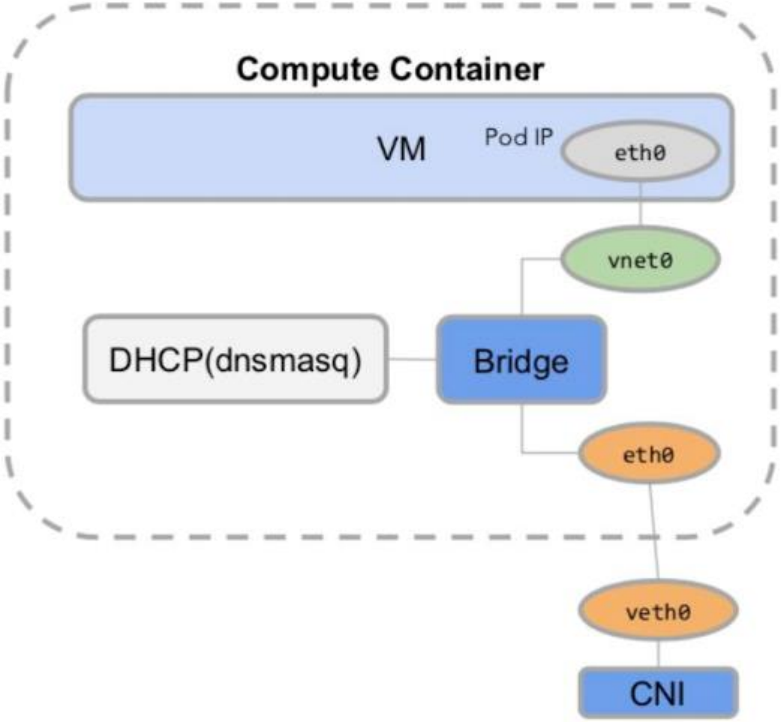


Benchmark Analysis

- TCP latency will decrease by 40% ~ 60%
- Throughput will increase by 40% ~ 80% when the packet size is greater than 1024 bytes
- When the packet size is less than 512 bytes, the throughput decrease due to per packet processing overhead

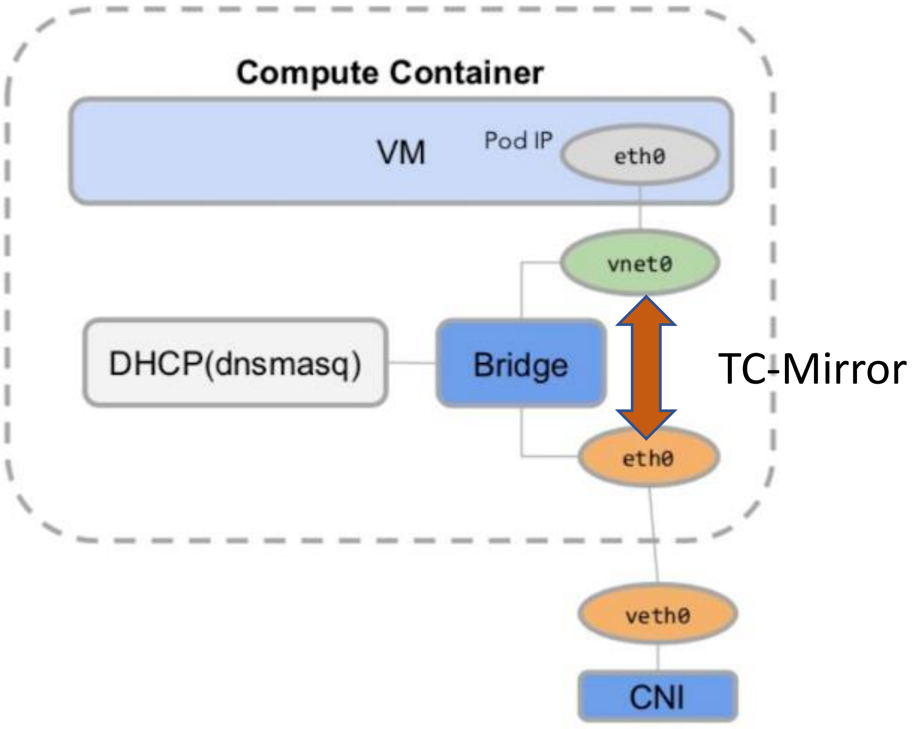
KubeVirt network performance

- Native Kubevirt Bridge networks have a much larger performance gap compared to Pods.
  - Latency increased by 60%
  - PPS decreased by 50%.
- Possible causes:
  - The consumption brought by the virtualized Linux network stack itself
  - Additional losses caused by the multiple layers of KubeVirt Bridge network.
- Kata has already utilized tc-mirror to bypass the bridge



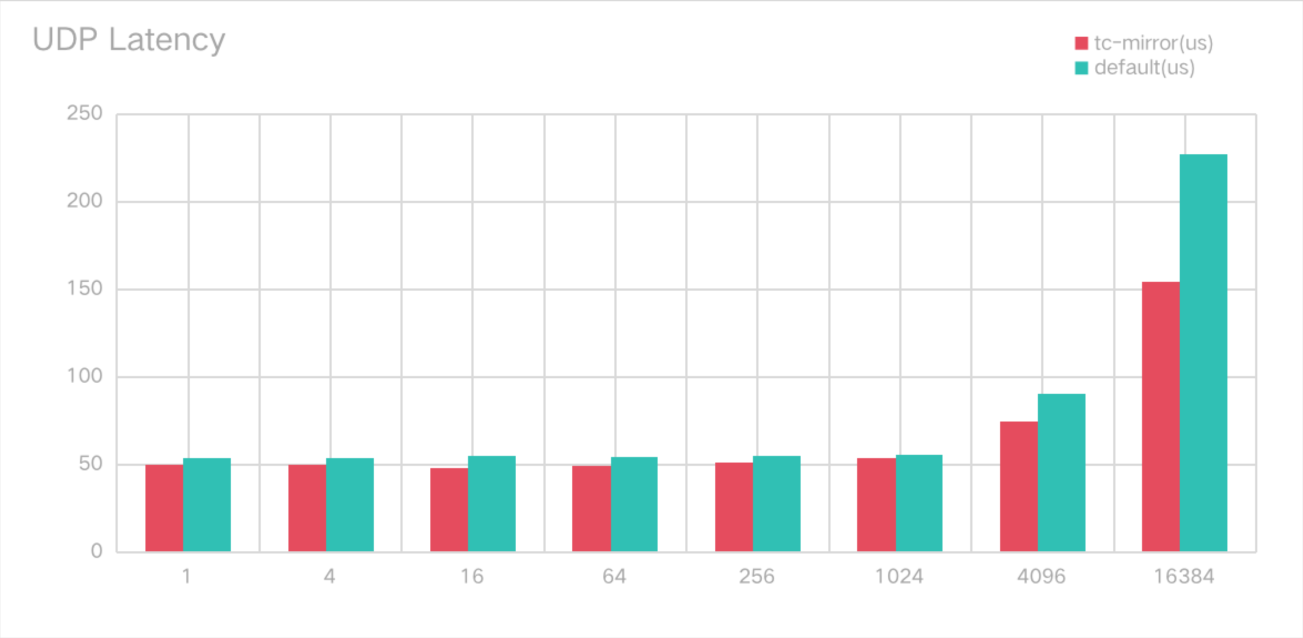
KubeVirt network acceleration

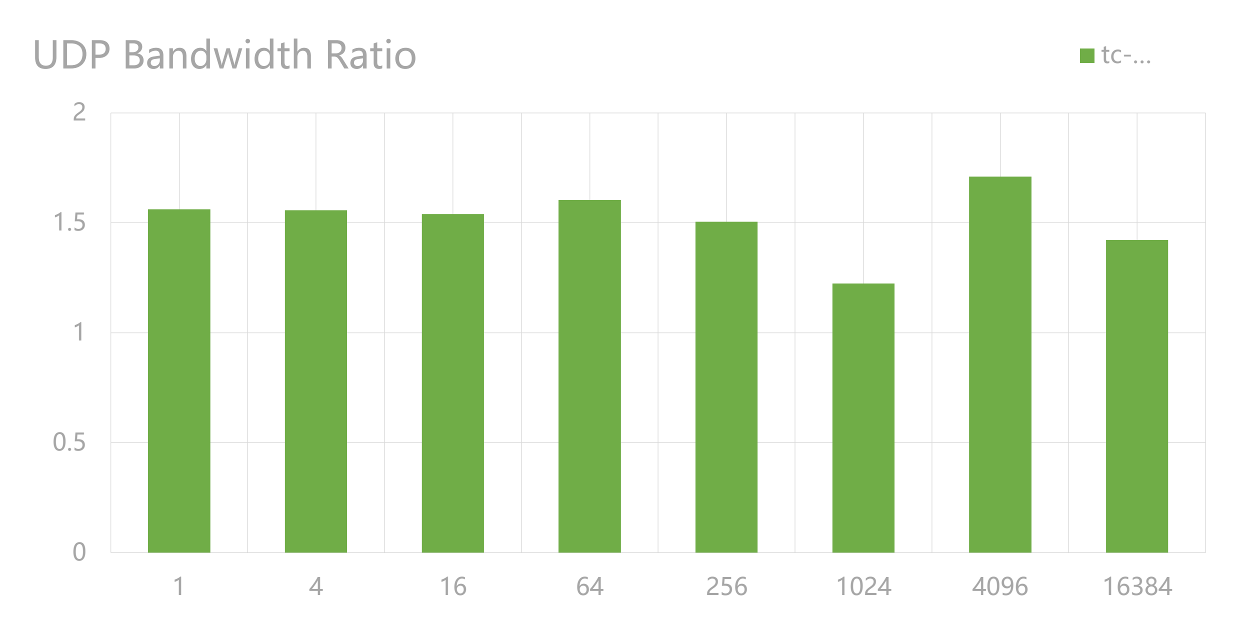
- Both tc-mirror and bpf_redirect are tried
  - tc-mirror performed slightly better than bpf_redirect in all tests
  - tc-mirror do not require high version kernel
  - We didn’t find way to redirect from VM eth0 to veth0 directly
  - Latency decrease 5% but throughput decrease about 20%
- Disable checksum
  - No need checksum for internal packet processing
  - Both rx/tx are dissabled



Benchmark

- Use qperf to benchmark UDP latency/bandwidth with packet sizes from 1byte to 16KB
  - Latency 4%~30% decrease
  - Throughput 20%~70% improvement





Feature work

- Accelerate UDP inter-node traffic
- Fine-grained control the switch between eBPF datapath and kernel network datapath
- Accelerate Kubevirt VMs traffic within the same Node

Notice and Disclaimer

Intel technologies may require enabled hardware, software or service activation.
No product or component can be absolutely secure.
Your costs and results may vary.