CLOUD NATIVE eBPF DAY NORTH AMERICA

A Beginner's Guide to eBPF Programming for networking

Liz Rice

Chief Open Source Officer, Isovalent

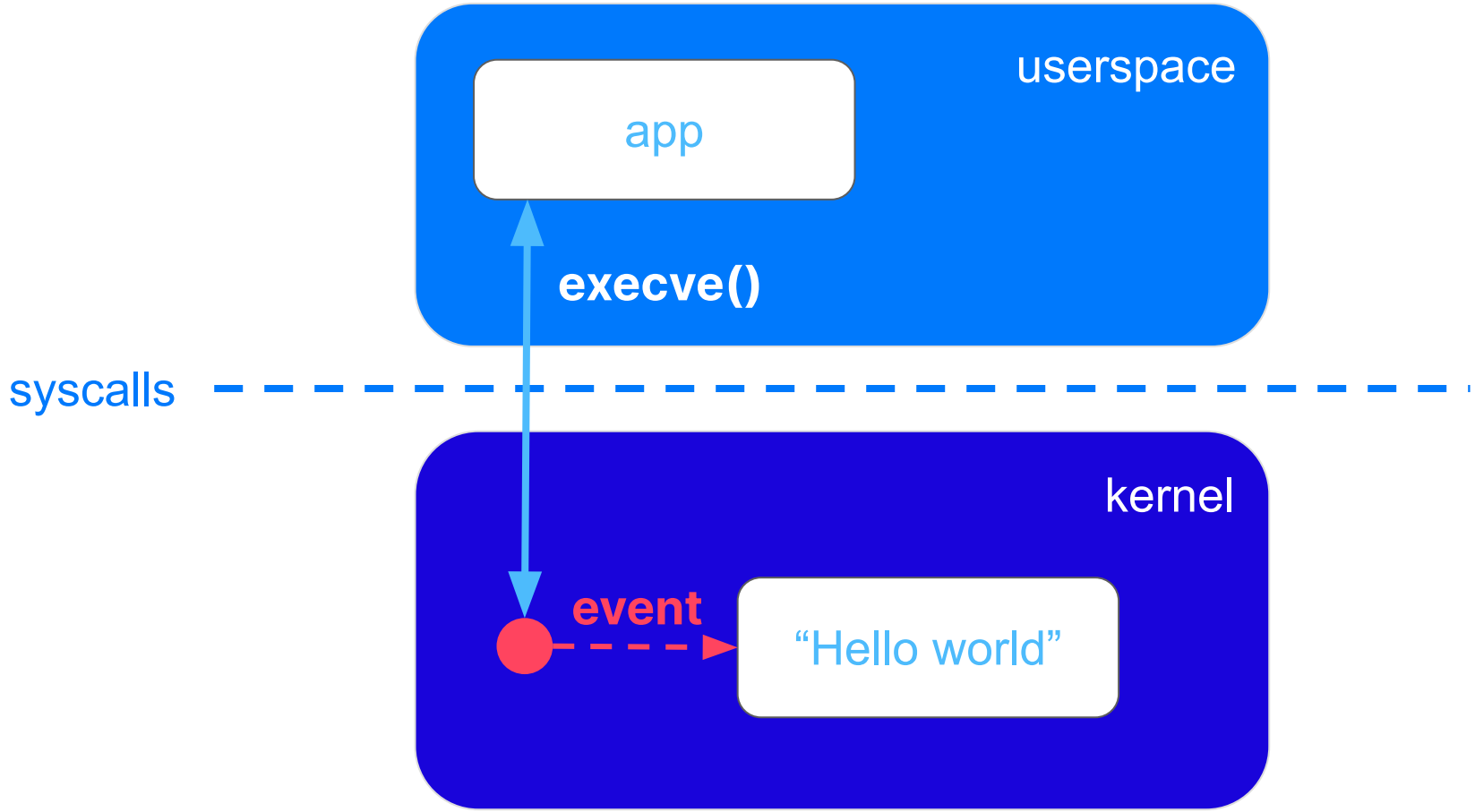
@lizrice

eBPF lets you run custom code in the kernel

Attaching eBPF to events

eBPF programs are event-driven and are run when the kernel or an application passes a certain hook point. Pre-defined hooks include system calls, function entry/exit, kernel tracepoints, network events, and several others.

eBPF Hello World



eBPF Hello World

```
SEC("kprobe/sys_execve")
int hello(void *ctx)
{
    bpf_printk("I'm alive!");
    return 0;
}
```

+ userspace code to load eBPF program

Info about process that called execve syscall

```
$ sudo ./hello
   bash-20241   [004] d... 84210.752785: 0: I'm alive!
   bash-20242   [004] d... 84216.321993: 0: I'm alive!
   bash-20243   [004] d... 84225.858880: 0: I'm alive!
```

Program types

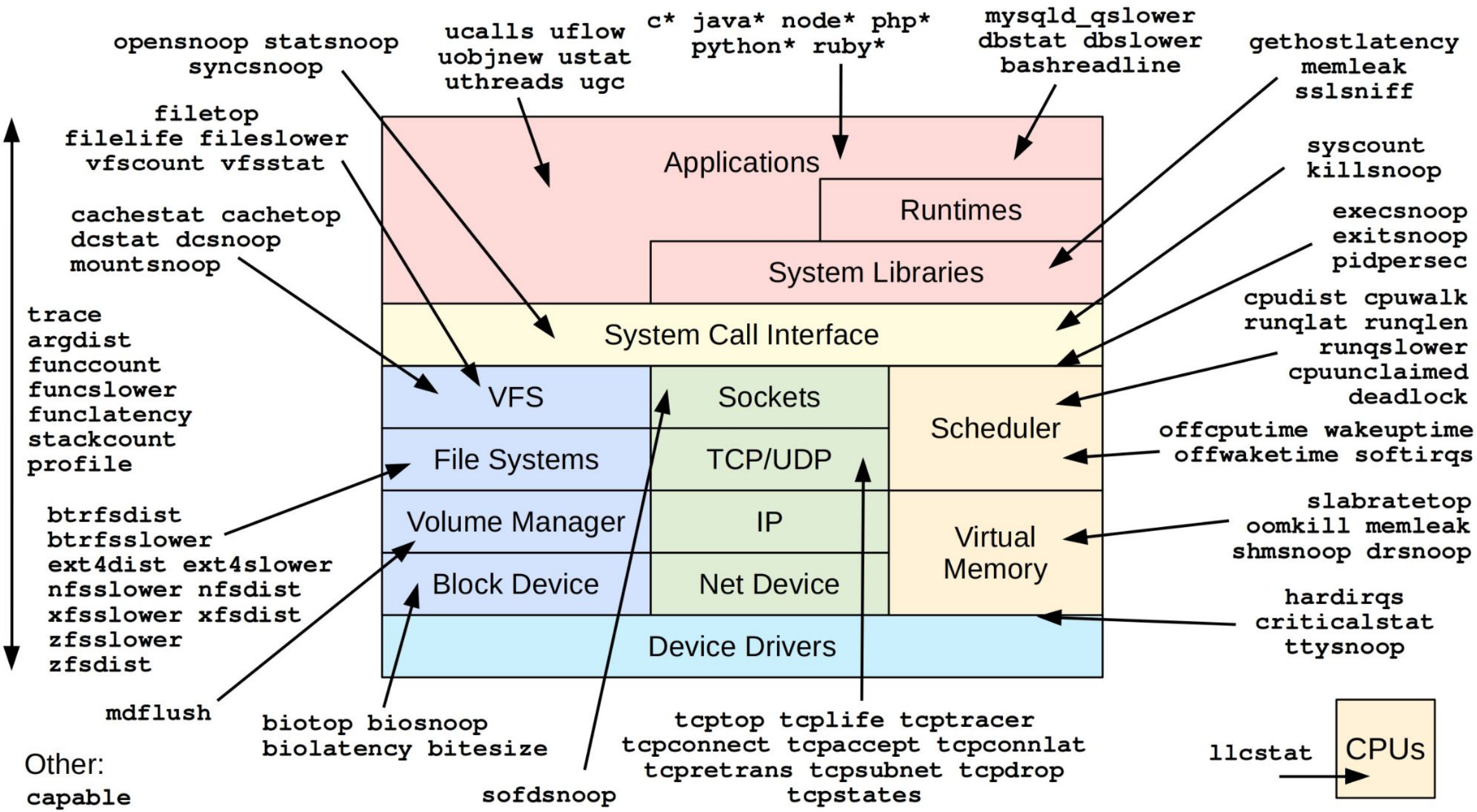
```
enum bpf_prog_type {
    BPF_PROG_TYPE_UNSPEC,
    BPF_PROG_TYPE_SOCKET_FILTER,
    BPF_PROG_TYPE_KPROBE,
    BPF_PROG_TYPE_SCHED_CLS,
    BPF_PROG_TYPE_SCHED_ACT,
    BPF_PROG_TYPE_TRACEPOINT,
    BPF_PROG_TYPE_XDP,
    BPF_PROG_TYPE_PERF_EVENT,
    BPF_PROG_TYPE_CGROUP_SKB,
    BPF_PROG_TYPE_CGROUP_SOCK,
    BPF_PROG_TYPE_LWT_IN,
    BPF_PROG_TYPE_LWT_OUT,
    BPF_PROG_TYPE_LWT_XMIT,
    BPF_PROG_TYPE_SOCK_OPS,
    BPF_PROG_TYPE_SK_SKB,
    BPF_PROG_TYPE_CGROUP_DEVICE,
    BPF_PROG_TYPE_SK_MSG,
    BPF_PROG_TYPE_RAW_TRACEPOINT,
    BPF_PROG_TYPE_CGROUP_SOCK_ADDR,
    BPF_PROG_TYPE_LWT_SEG6LOCAL,
    BPF_PROG_TYPE_LIRC_MODE2,
    BPF_PROG_TYPE_SK_REUSEPORT,
    BPF_PROG_TYPE_FLOW_DISSECTOR,
    /* See /usr/include/linux/bpf.h for
       the full list. */
};
```

Program types

```
enum bpf_prog_type {
    BPF_PROG_TYPE_UNSPEC,
    BPF_PROG_TYPE_SOCKET_FILTER,
    BPF_PROG_TYPE_KPROBE,
    BPF_PROG_TYPE_SCH
    BPF_PROG_TYPE_SCH
    BPF_PROG_TYPE_TRA
    BPF_PROG_TYPE_XDP
    BPF_PROG_TYPE_PER
    BPF_PROG_TYPE_CGROUP_SKB,
    BPF_PROG_TYPE_CGROUP_SOCK,
    BPF_PROG_TYPE_LWT_IN,
    BPF_PROG_TYPE_LWT_OUT,
    BPF_PROG_TYPE_LWT_XMIT,
    BPF_PROG_TYPE_SOCK_OPS,
    K_SKB,
    GROUP_DEVICE,
    K_MSG,
    AW_TRACEPOINT,
    GROUP_SOCK_ADDR,
    BPF_PROG_TYPE_LWT_SEG6LOCAL,
    BPF_PROG_TYPE_LIRC_MODE2,
    BPF_PROG_TYPE_SK_REUSEPORT,
    BPF_PROG_TYPE_FLOW_DISSECTOR,
    /* See /usr/include/linux/bpf.h for
       the full list. */
};
```

eBPF - not just for syscalls!

Linux bcc/BPF Tracing Tools



https://github.com/iovisor/bcc#tools 2019

Also, many perf events

```
sudo perf list
```

Network events
(a very non-comprehensive guide)

Program types

```
enum bpf_prog_type {
    BPF_PROG_TYPE_UNSPEC,
    BPF_PROG_TYPE_SOCKET_FILTER,
    BPF_PROG_TYPE_KPROBE,
    BPF_PROG_TYPE_SCHED_CLS,
    BPF_PROG_TYPE_SCHED_ACT,
    BPF_PROG_TYPE_TRACEPOINT,
    BPF_PROG_TYPE_XDP,
    BPF_PROG_TYPE_PERF_EVENT,
    BPF_PROG_TYPE_CGROUP_SKB,
    BPF_PROG_TYPE_CGROUP_SOCK,
    BPF_PROG_TYPE_LWT_IN,
    BPF_PROG_TYPE_LWT_OUT,
    BPF_PROG_TYPE_LWT_XMIT,
    BPF_PROG_TYPE_SOCK_OPS,
    BPF_PROG_TYPE_SK_SKB,
    BPF_PROG_TYPE_CGROUP_DEVICE,
    BPF_PROG_TYPE_SK_MSG,
    BPF_PROG_TYPE_RAW_TRACEPOINT,
    BPF_PROG_TYPE_CGROUP_SOCK_ADDR,
    BPF_PROG_TYPE_LWT_SEG6LOCAL,
    BPF_PROG_TYPE_LIRC_MODE2,
    BPF_PROG_TYPE_SK_REUSEPORT,
    BPF_PROG_TYPE_FLOW_DISSECTOR,
    /* See /usr/include/linux/bpf.h for
       the full list. */
};
```

Kprobes / kretprobes

Entry to / exit from a kernel function

Lots of kernel functions relate to networking

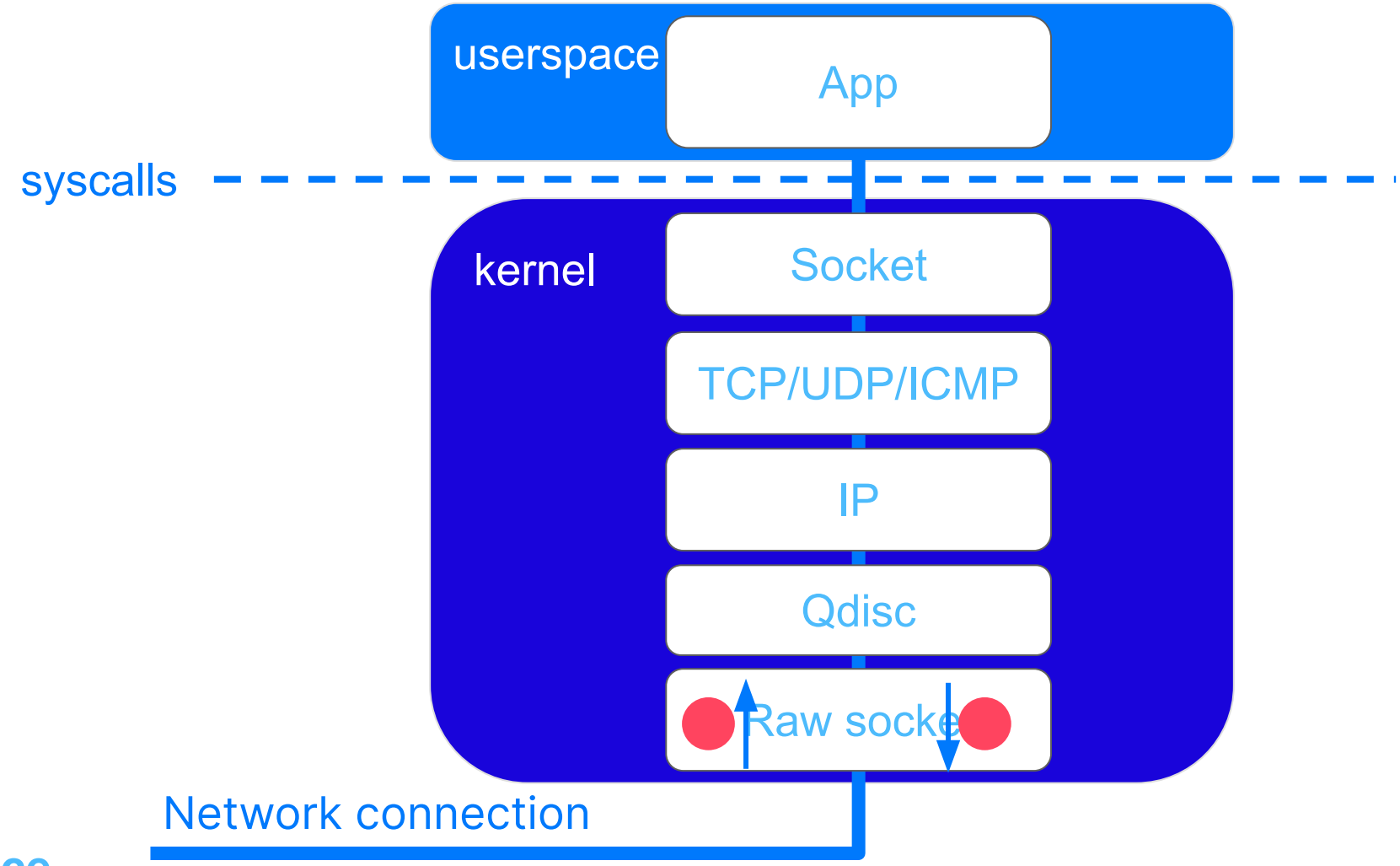
example

tcp_v4_connect() kernel function

Program types

```
enum bpf_prog_type {
    BPF_PROG_TYPE_UNSPEC,
    BPF_PROG_TYPE_SOCKET_FILTER,
    BPF_PROG_TYPE_KPROBE,
    BPF_PROG_TYPE_SCHED_CLS,
    BPF_PROG_TYPE_SCHED_ACT,
    BPF_PROG_TYPE_TRACEPOINT,
    BPF_PROG_TYPE_XDP,
    BPF_PROG_TYPE_PERF_EVENT,
    BPF_PROG_TYPE_CGROUP_SKB,
    BPF_PROG_TYPE_CGROUP_SOCK,
    BPF_PROG_TYPE_LWT_IN,
    BPF_PROG_TYPE_LWT_OUT,
    BPF_PROG_TYPE_LWT_XMIT,
    BPF_PROG_TYPE_SOCK_OPS,
    BPF_PROG_TYPE_SK_SKB,
    BPF_PROG_TYPE_CGROUP_DEVICE,
    BPF_PROG_TYPE_SK_MSG,
    BPF_PROG_TYPE_RAW_TRACEPOINT,
    BPF_PROG_TYPE_CGROUP_SOCK_ADDR,
    BPF_PROG_TYPE_LWT_SEG6LOCAL,
    BPF_PROG_TYPE_LIRC_MODE2,
    BPF_PROG_TYPE_SK_REUSEPORT,
    BPF_PROG_TYPE_FLOW_DISSECTOR,
    /* See /usr/include/linux/bpf.h for
       the full list. */
};
```

Socket filter



Socket filter

“The filtering actions include dropping packets (if the program returns 0) or trimming packets (if the program returns a length less than the original). … Note that **we're not trimming or dropping the original packet** which would still reach the intended socket intact; we're working with a **copy of the packet metadata** which raw sockets can access for observability. “

https://blogs.oracle.com/linux/post/bpf-a-tour-of-program-types

Socket filter

Network packet data **copy**

Filters what gets sent to userspace, for performant observability

example

attach_raw_socket()

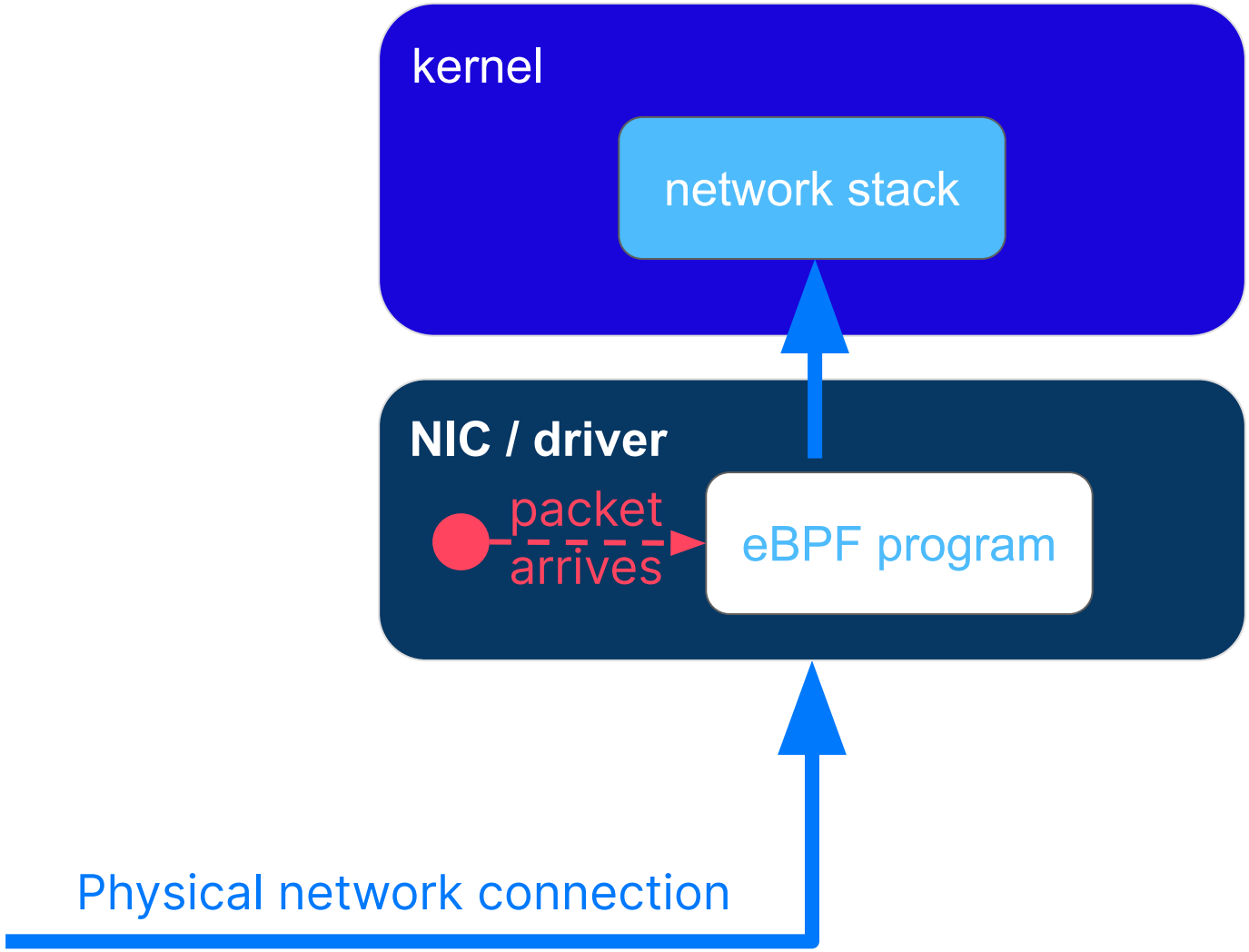
Program types

```
enum bpf_prog_type {
    BPF_PROG_TYPE_UNSPEC,
    BPF_PROG_TYPE_SOCKET_FILTER,
    BPF_PROG_TYPE_KPROBE,
    BPF_PROG_TYPE_SCHED_CLS,
    BPF_PROG_TYPE_SCHED_ACT,
    BPF_PROG_TYPE_TRACEPOINT,
    BPF_PROG_TYPE_XDP,
    BPF_PROG_TYPE_PERF_EVENT,
    BPF_PROG_TYPE_CGROUP_SKB,
    BPF_PROG_TYPE_CGROUP_SOCK,
    BPF_PROG_TYPE_LWT_IN,
    BPF_PROG_TYPE_LWT_OUT,
    BPF_PROG_TYPE_LWT_XMIT,
    BPF_PROG_TYPE_SOCK_OPS,
    BPF_PROG_TYPE_SK_SKB,
    BPF_PROG_TYPE_CGROUP_DEVICE,
    BPF_PROG_TYPE_SK_MSG,
    BPF_PROG_TYPE_RAW_TRACEPOINT,
    BPF_PROG_TYPE_CGROUP_SOCK_ADDR,
    BPF_PROG_TYPE_LWT_SEG6LOCAL,
    BPF_PROG_TYPE_LIRC_MODE2,
    BPF_PROG_TYPE_SK_REUSEPORT,
    BPF_PROG_TYPE_FLOW_DISSECTOR,
    /* See /usr/include/linux/bpf.h for
       the full list. */
};
```

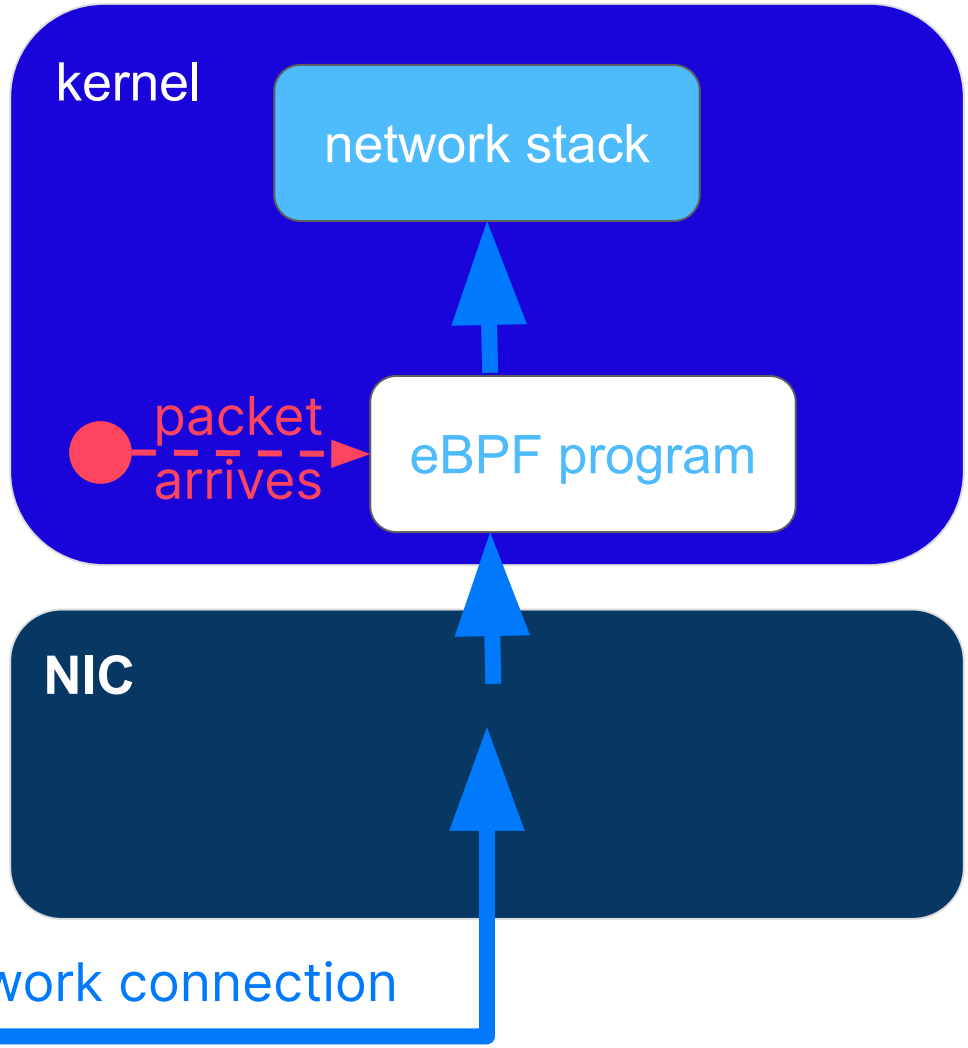
XDP - express data path

“What if we could run eBPF on the network interface card?”

XDP

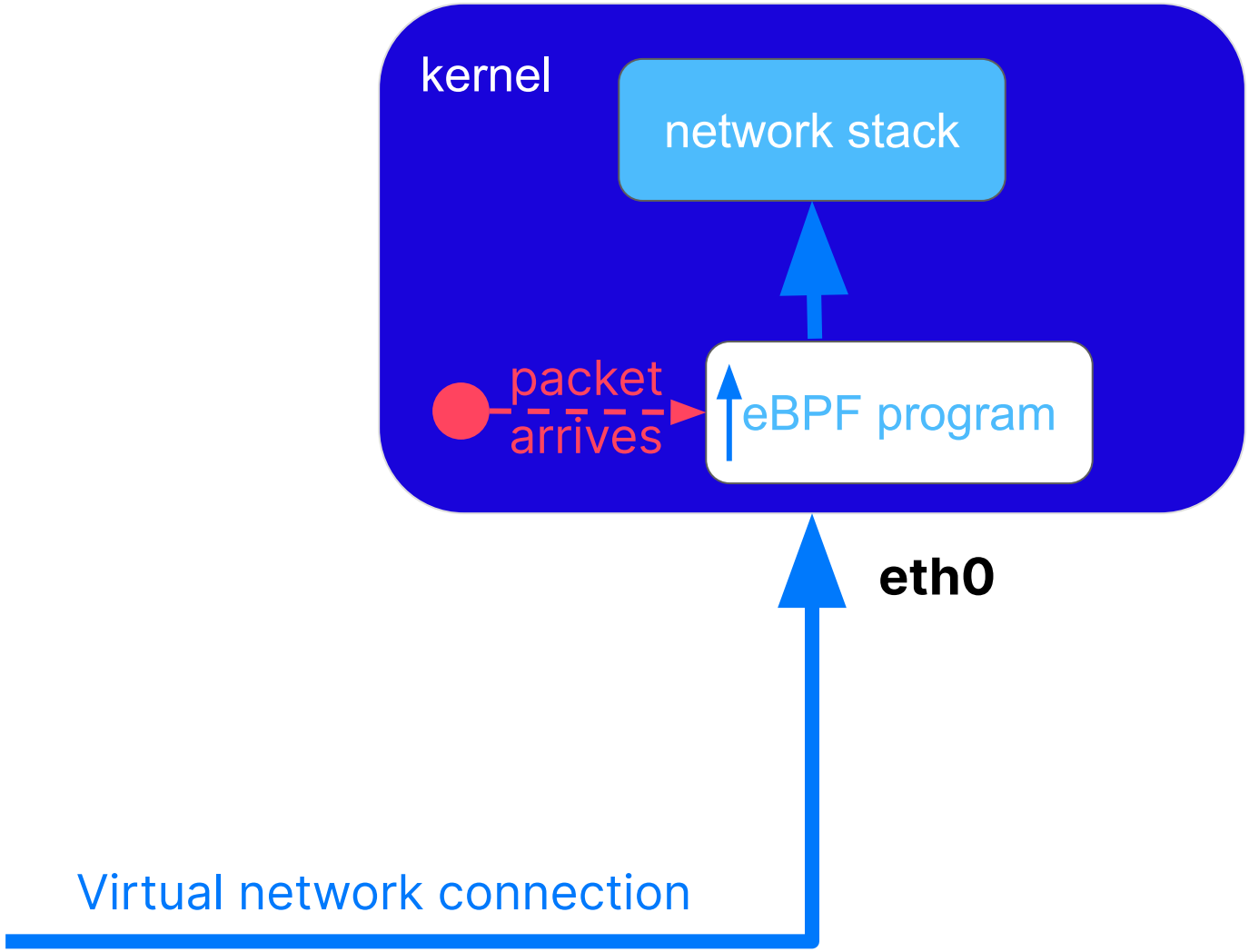

kernel
network stack
NIC / driver
packet arrives
eBPF program
Physical network connection

XDP



Only some NICs / drivers support XDP

XDP


kernel
network stack
packet arrives
eBPF program
eth0
Virtual network connection

XDP - express data path

Inbound packets

Pass / drop / manipulate / redirect packets

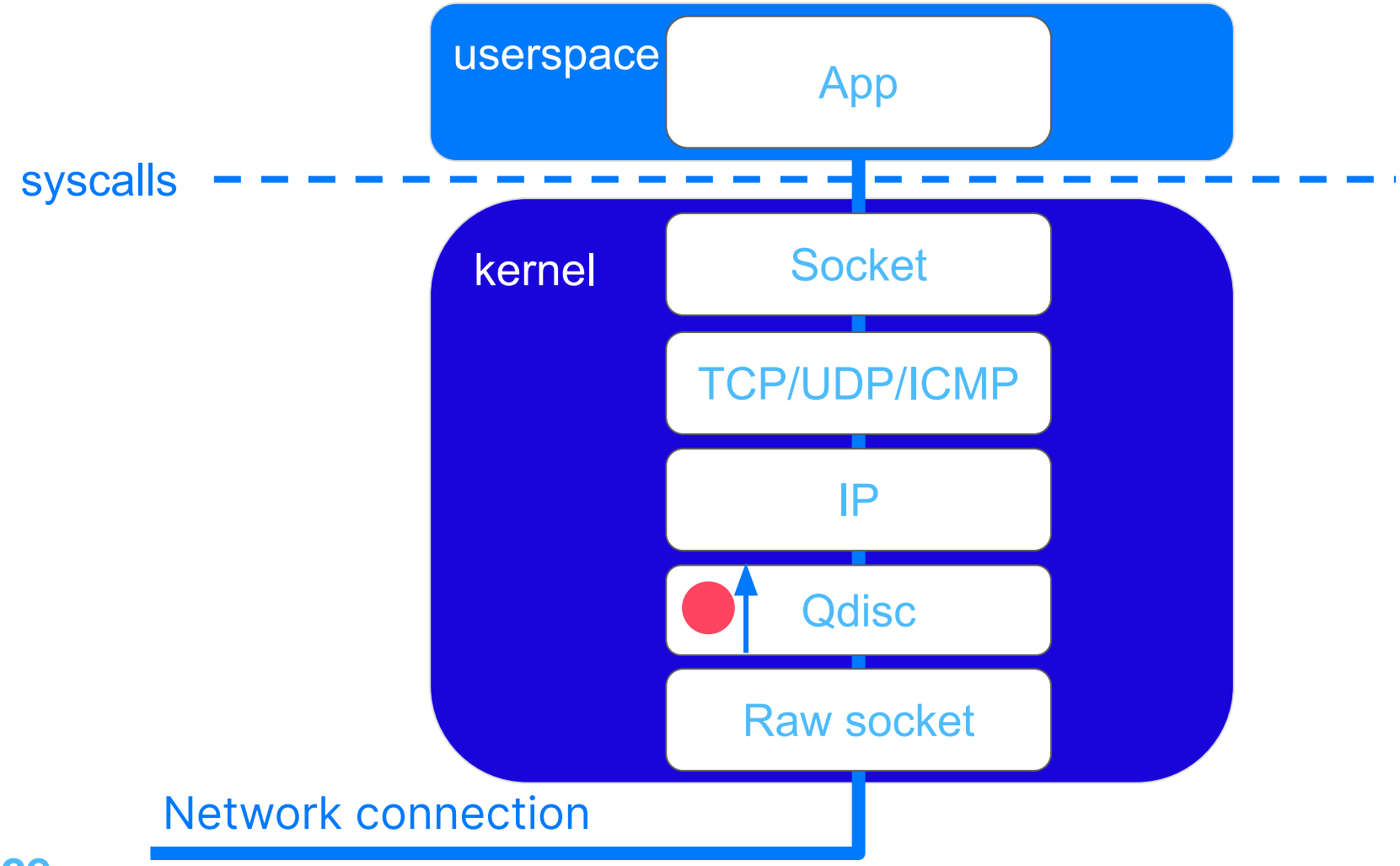
example

attach_xdp()

Program types

```
enum bpf_prog_type {
    BPF_PROG_TYPE_UNSPEC,
    BPF_PROG_TYPE_SOCKET_FILTER,
    BPF_PROG_TYPE_KPROBE,
    BPF_PROG_TYPE_SCHED_CLS,
    BPF_PROG_TYPE_SCHED_ACT,
    BPF_PROG_TYPE_TRACEPOINT,
    BPF_PROG_TYPE_XDP,
    BPF_PROG_TYPE_PERF_EVENT,
    BPF_PROG_TYPE_CGROUP_SKB,
    BPF_PROG_TYPE_CGROUP_SOCK,
    BPF_PROG_TYPE_LWT_IN,
    BPF_PROG_TYPE_LWT_OUT,
    BPF_PROG_TYPE_LWT_XMIT,
    BPF_PROG_TYPE_SOCK_OPS,
    BPF_PROG_TYPE_SK_SKB,
    BPF_PROG_TYPE_CGROUP_DEVICE,
    BPF_PROG_TYPE_SK_MSG,
    BPF_PROG_TYPE_RAW_TRACEPOINT,
    BPF_PROG_TYPE_CGROUP_SOCK_ADDR,
    BPF_PROG_TYPE_LWT_SEG6LOCAL,
    BPF_PROG_TYPE_LIRC_MODE2,
    BPF_PROG_TYPE_SK_REUSEPORT,
    BPF_PROG_TYPE_FLOW_DISSECTOR,
    /* See /usr/include/linux/bpf.h for
       the full list. */
};
```

Traffic control (ingress)



Traffic control

Traffic filters, attached to queueing disciplines

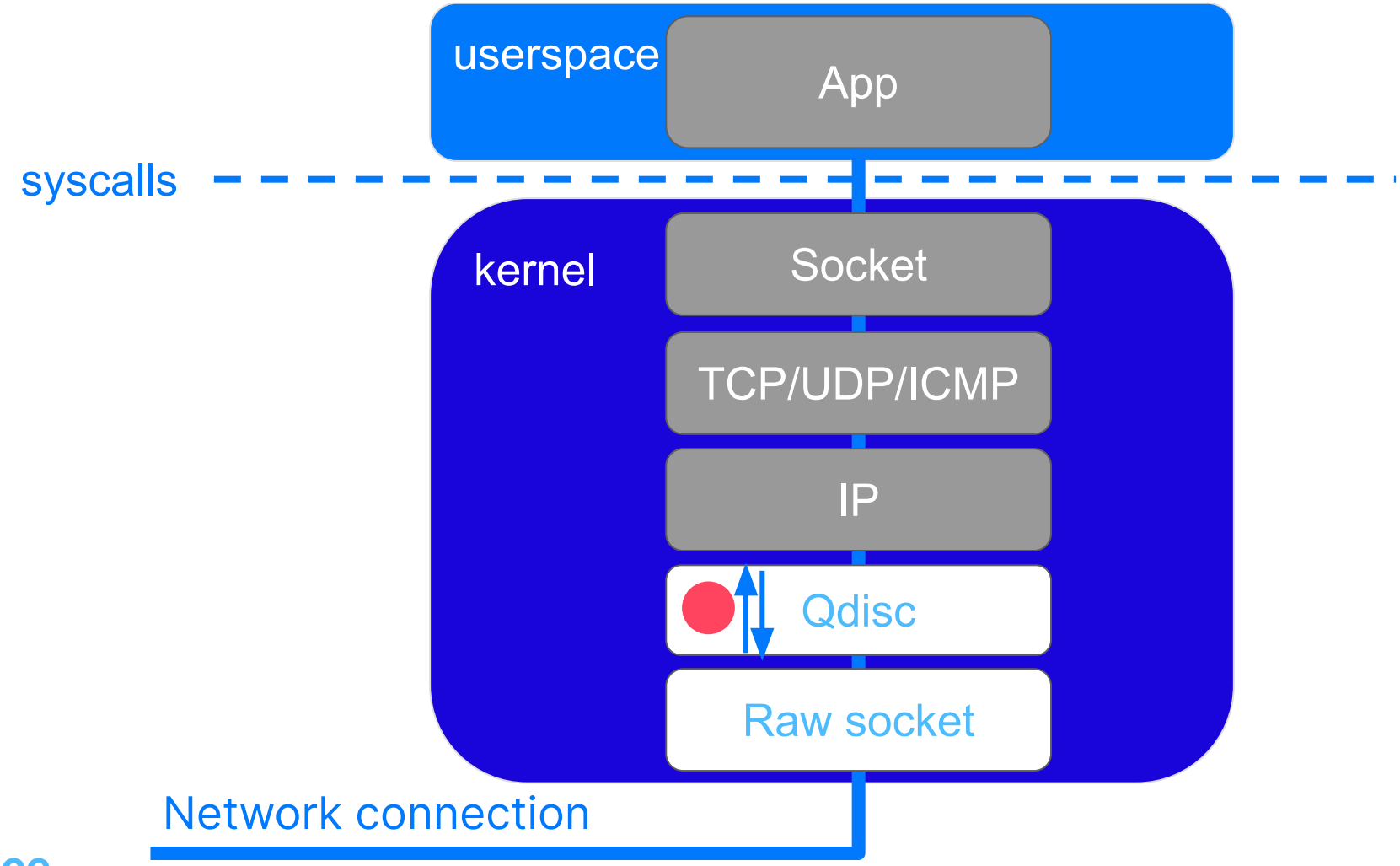
Ingress / egress (separately)

Pass / drop / manipulate / redirect packets

example

tc(“add-filter)

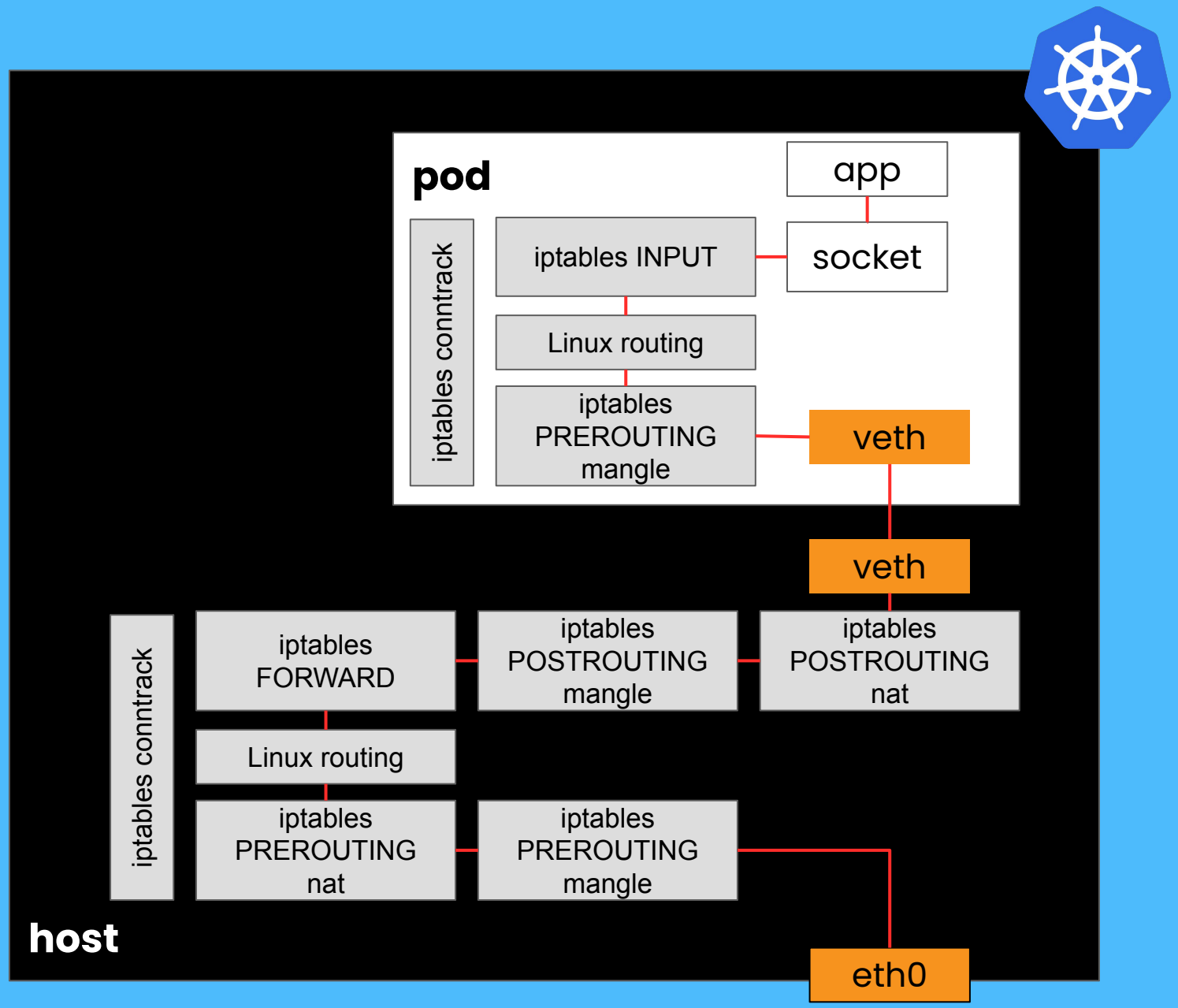
Traffic control ingress - ping reply

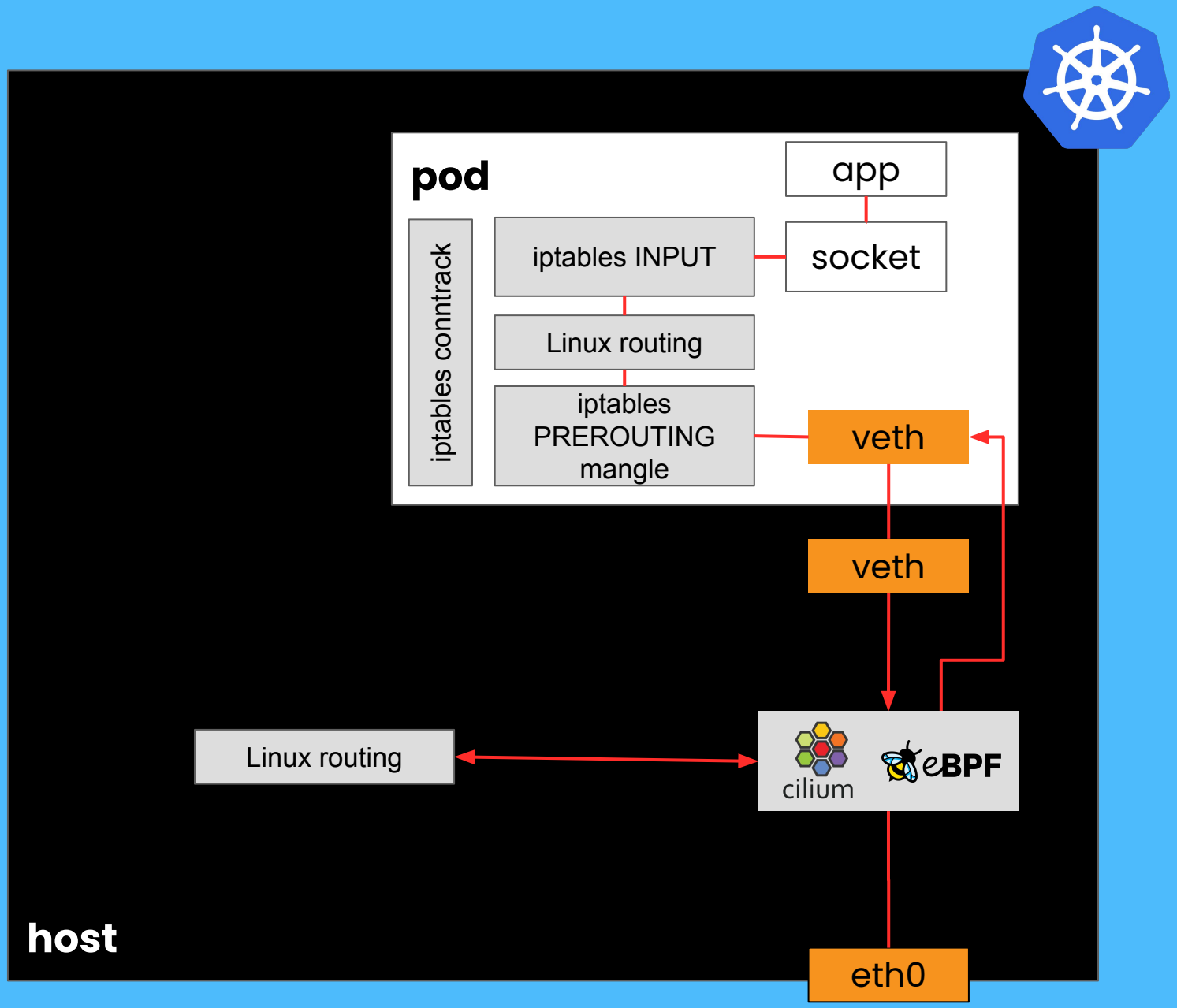


Fewer perf events using TC pingpong

```
sudo perf trace -e “net:*” ping -c1 <addr>
```

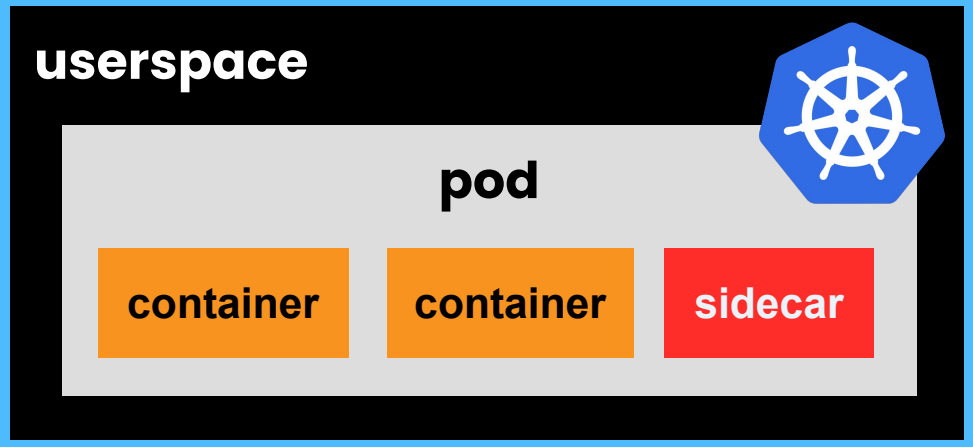
eBPF networking
enables efficiency & high performance


pod
app
socket
iptables conntrack
iptables INPUT
Linux routing
iptables PREROUTING mangle
veth
veth
iptables conntrack
iptables FORWARD
iptables POSTROUTING mangle
iptables POSTROUTING nat
Linux routing
iptables PREROUTING nat
iptables PREROUTING mangle
host
eth0


pod
app
socket
iptables conntrack
iptables INPUT
Linux routing
iptables PREROUTING mangle
veth
veth
Linux routing
cilium
eBPF
host
eth0

eBPF can instrument apps without any app or config changes

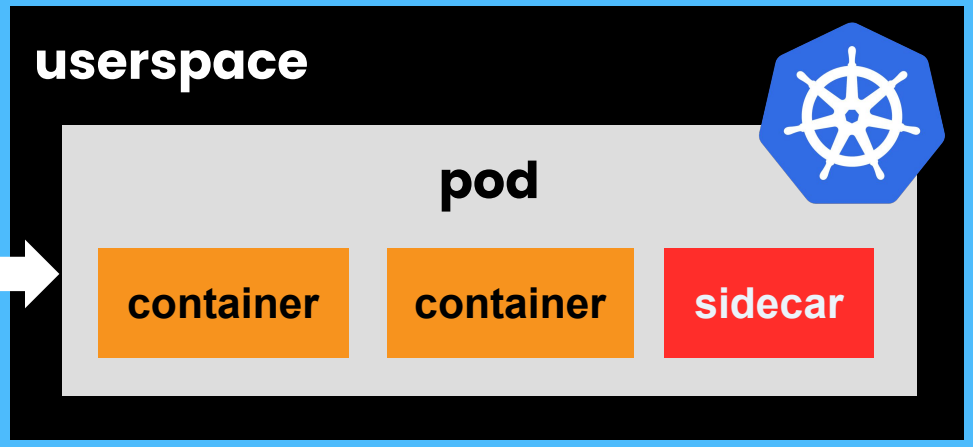
A sidecar has a view across one pod



Sidecars need YAML

my-app.yaml

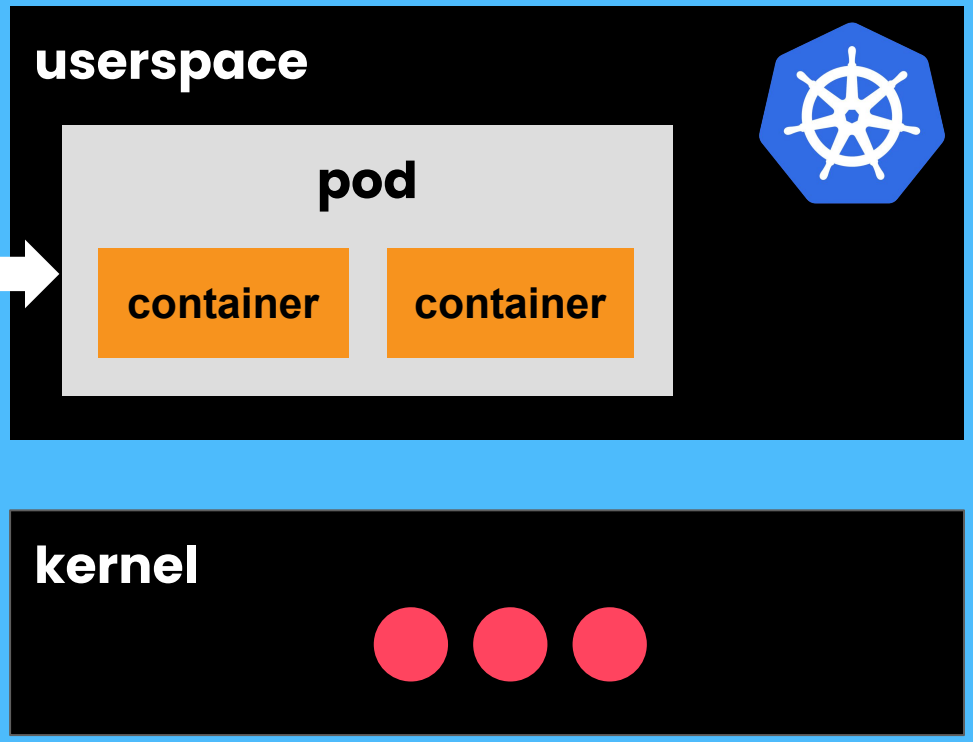
```
containers:
 -   name: my-app
     ...
 -   name: my-app-init
     …
 -   name: my-sidecar
     ...
```



eBPF does not need app changes

my-app.yaml

```
containers:
 -   name: my-app
     ...
 -   name: my-app-init
     …
```



eBPF-enabled networking capabilities

Inspect packets → **Observability**

Identity-aware data flows, message parsing, security forensics...

Drop or modify packets → **Security**

Network policies, encryption...

Redirect packets → **Networking functions**

Load balancing, routing, service mesh...

eBPF enables next-gen service mesh

high performance
without any app or config changes

Thank you

github.com/lizrice/ebpf-beginners

ebpf.io | cilium.io | isovalent.com