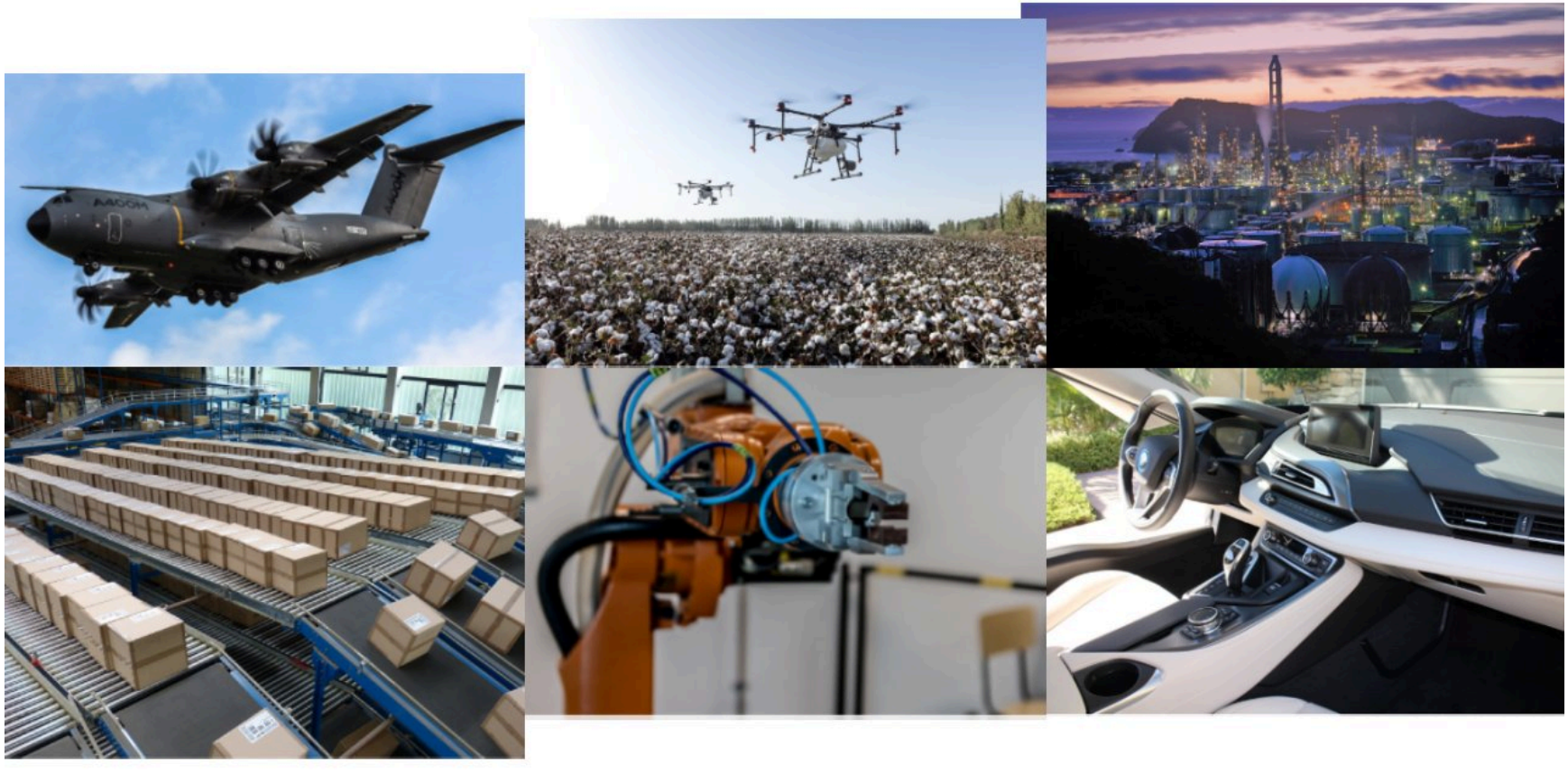
Enable Time-Sensitive Applications in Kubernetes with Container Network Interface Plugin Agnostic Metadata Proxy

Ferenc Orosi, **Ferenc Fejes**

Ericsson Research



source: pixabay.com

Context

- Microservices: main direction of application deployment since the past decade
- This trend emerged in the field of industrial applications as well
- *Kubernetes* has become the de-facto orchestration platform
- There are challenges however...

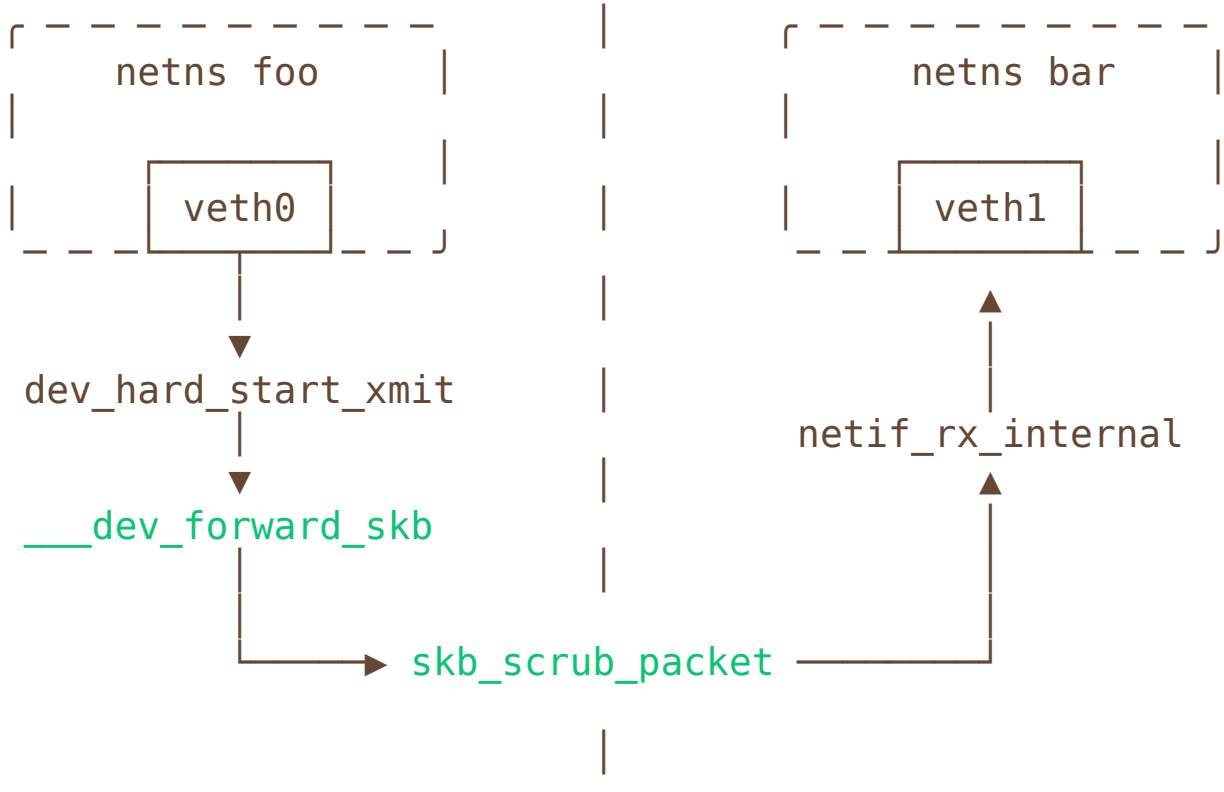
```
struct sk_buff {    // unrelated members omitted

    u32   priority;
    union {
        ktime_t   tstamp;
        u64       skb_mstamp_ns;  //for TCP
    }
}
```

- `skb->tstamp` and `skb->priority` are used by Time-Sensitive Networking (TSN) Qdiscs like **mqprio**, **taprio** and **etf**
- `skb->priority` also used for VLAN priority tagging

- Application can set these metadata fields from user space via socket API
  - `skb->tstamp`: `SO_TXTIME` control message **v4.19**
  - `skb->priority`: `SO_PRIORITY`
    - Socket-wide setting with `setsockopt` **v0.98** (released in 1992)
    - Per-packet setting with control message **v6.14**

Network namespace isolation brief



```
static __always_inline int ____dev_forward_skb(...)
{
    // [...]
    // bool xnet = !net_eq(dev_net(dev), dev_net(skb->dev);

    skb_scrub_packet(skb, xnet); // metadata removal
    skb->priority = 0;    // removing priority too
    return 0;
}
```

```
void skb_scrub_packet(struct sk_buff *skb, bool xnet)
{
    // clearing some skb fields
    [...]
    if (!xnet)
        return;

    skb->mark = 0;
    skb_clear_tstamp(skb);
}
```

- `skb->mark` discussed in details previously
  - *Traits: Rich Packet Metadata*: https://netdevconf.info/0x19/26
- `skb->tstamp` cleared conditionally

```
static inline void skb_clear_tstamp(struct sk_buff *skb)
{
    if (skb->tstamp_type)
        return;

    skb->tstamp = 0;
}
```

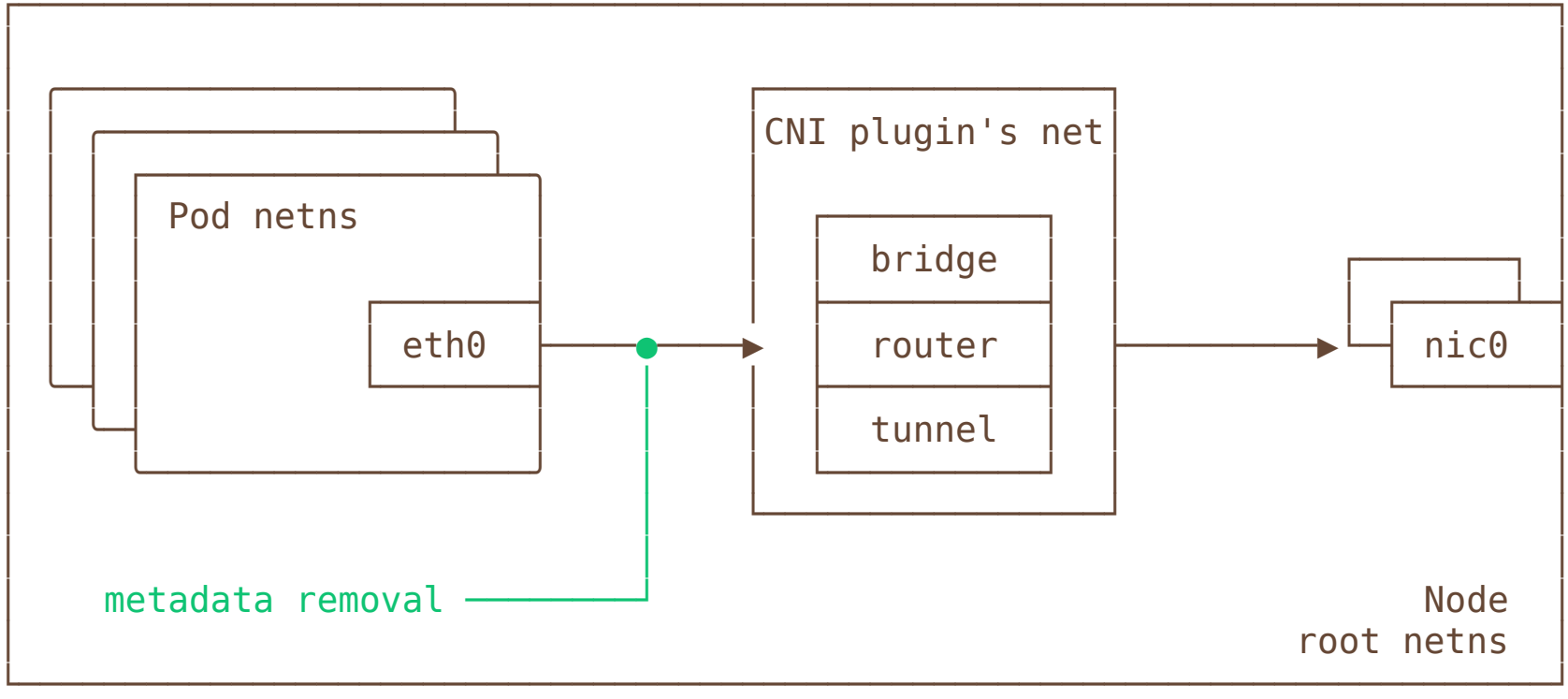
- Motivation discussed in *BPF datapath extensions for K8s workloads* LPC '21 talk (part 2).
- `skb->tstamp_type` values are defined in `enum skb_tstamp_type`
  - `SKB_CLOCK_{REALTIME,MONOTONIC,TAI}`

Summarizing what we know so far

- For TSN we need `tstamp` and `priority` to be preserved
- They cleared by network namespace isolation
- If `CLOCK_{MONOTONIC,TAI}` specified, `tstamp` is preserved
  - That works for **taprio** and **etf** Qdiscs

Kubernetes networking

- Networking implemented by *Container Network Interface* (CNI) plugins
  - Popular CNI plugins: Calico, Cilium, Flannel, Weave Net, Kindnet, Canal
- Each implement the cluster network differently
- Most of them rely on GNU/Linux tools and APIs to do it


Pod netns
eth0
CNI plugin's net
bridge
router
tunnel
nic0
metadata removal
Node
root netns

Main challenges for TSA deployment

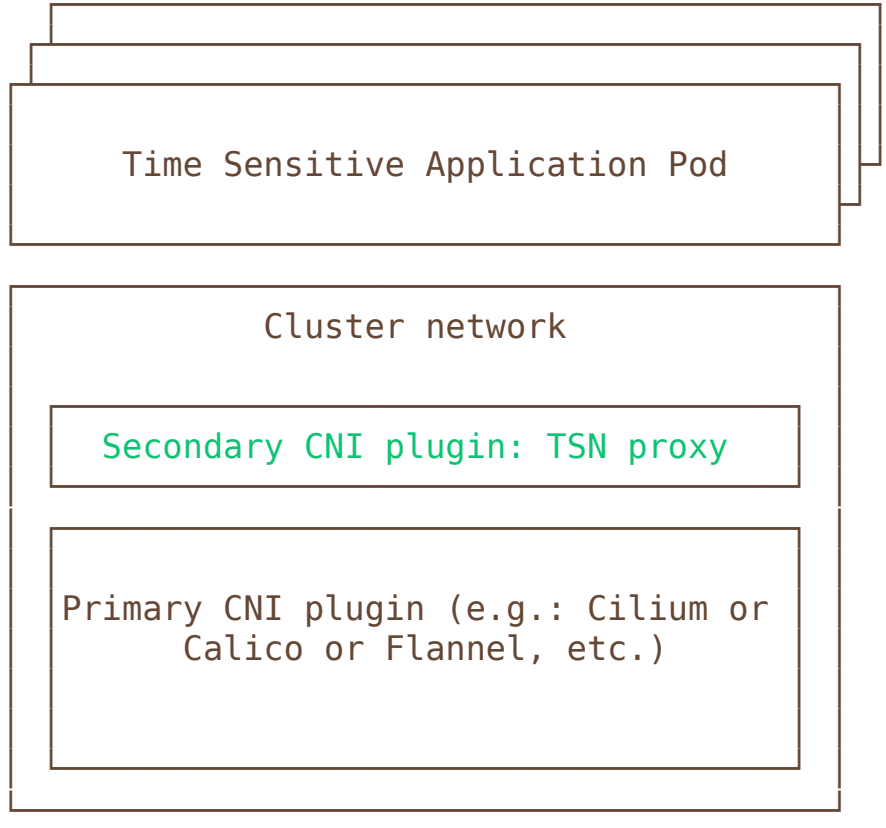
- CNI plugin diversity: different strategies to implement the network
- Must be transparent to the application, no API modifications
- Linux namespace isolation removes the necessary metadata
  - no option to configure a set of metadata to exclude from the isolation process*

Proposal: TSN Metadata Proxy CNI plugin

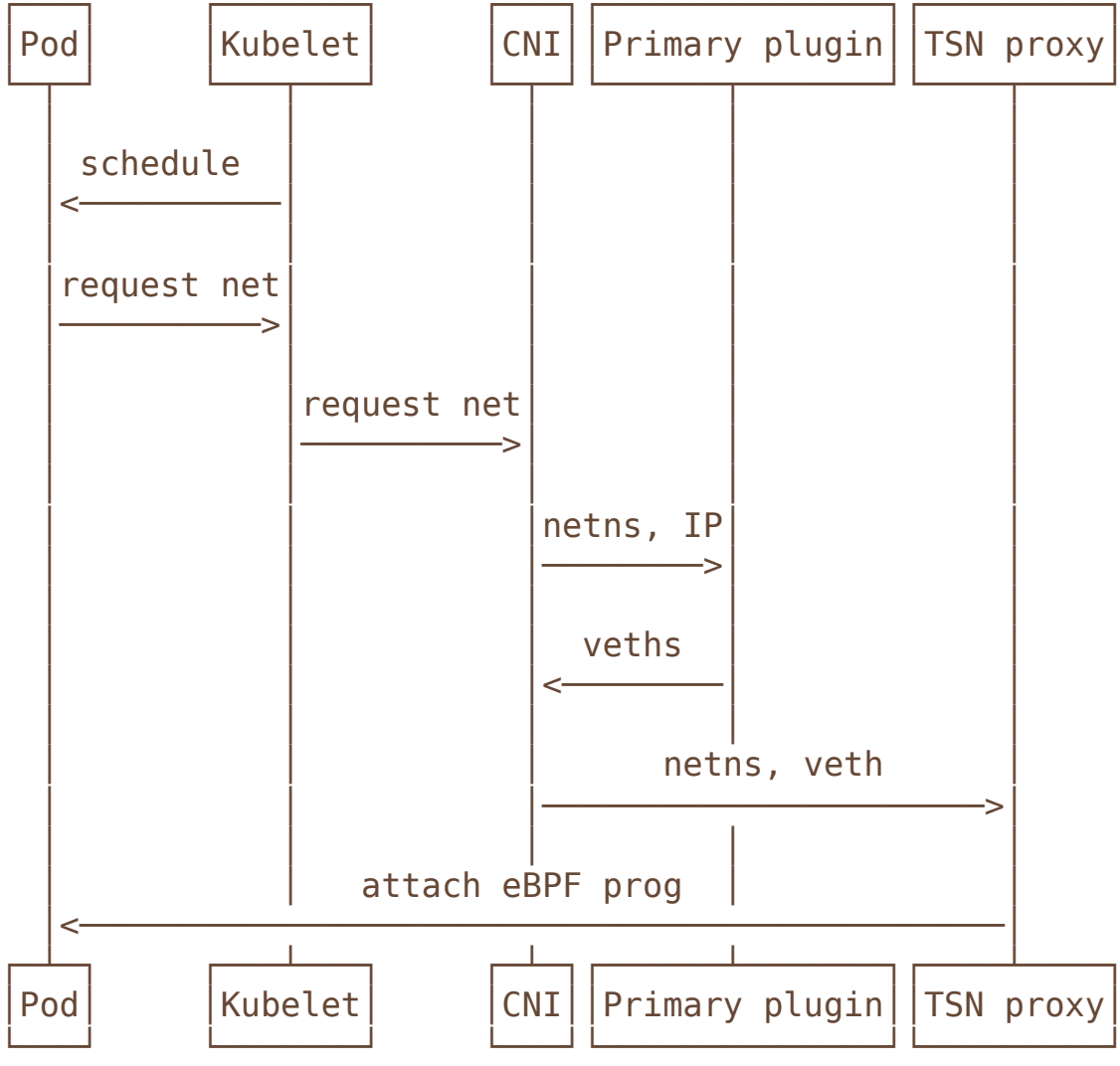
(TSN proxy for short)

TSN proxy

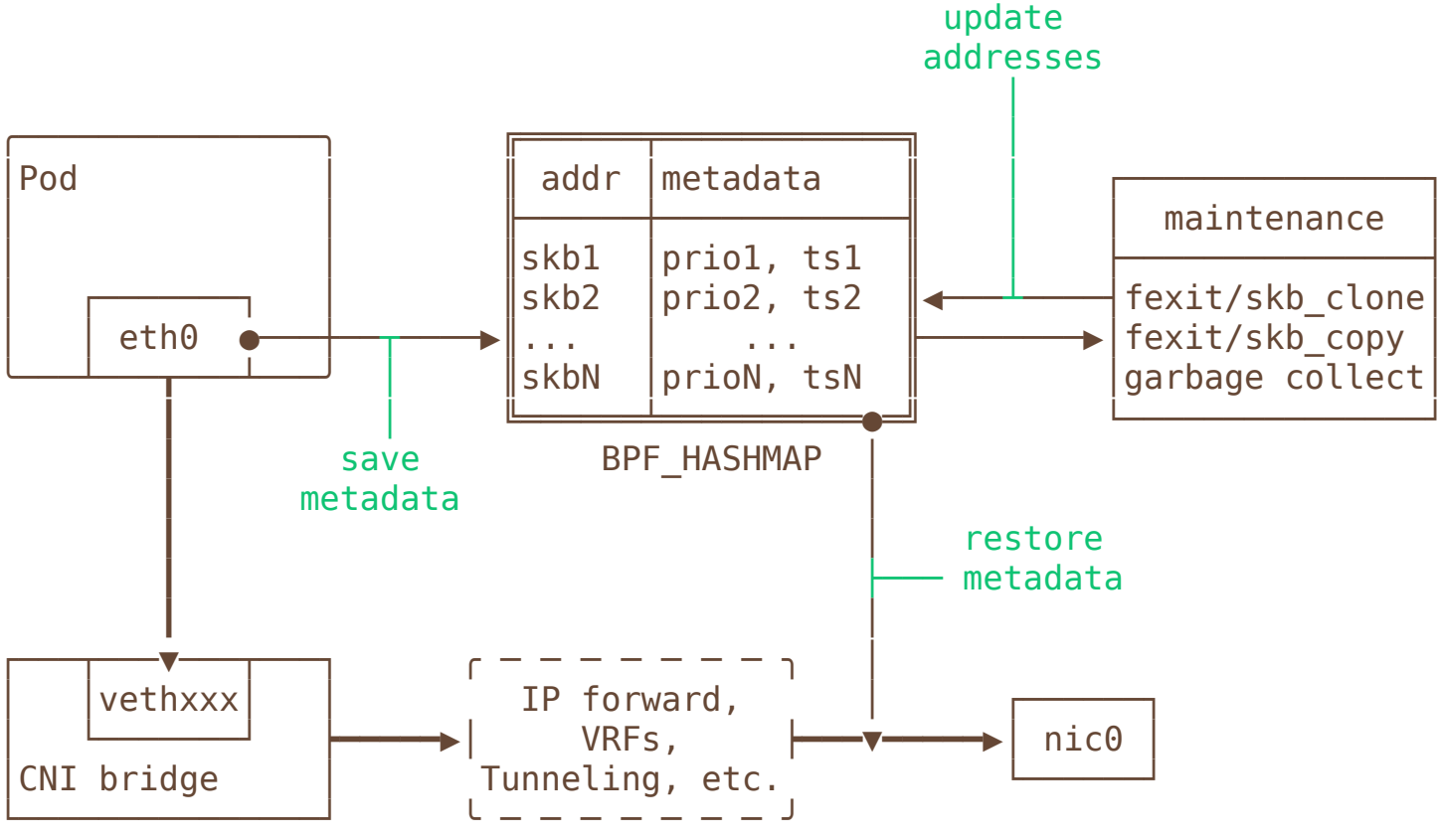
- Equip CNI plugins with TSN capability
- Does not require TSA microservice modification, transparent
- No Linux kernel bypass required
- Does not require custom kernel


Time Sensitive Application Pod
Cluster network
Secondary CNI plugin: TSN proxy
Primary CNI plugin (e.g.: Cilium or Calico or Flannel, etc.)

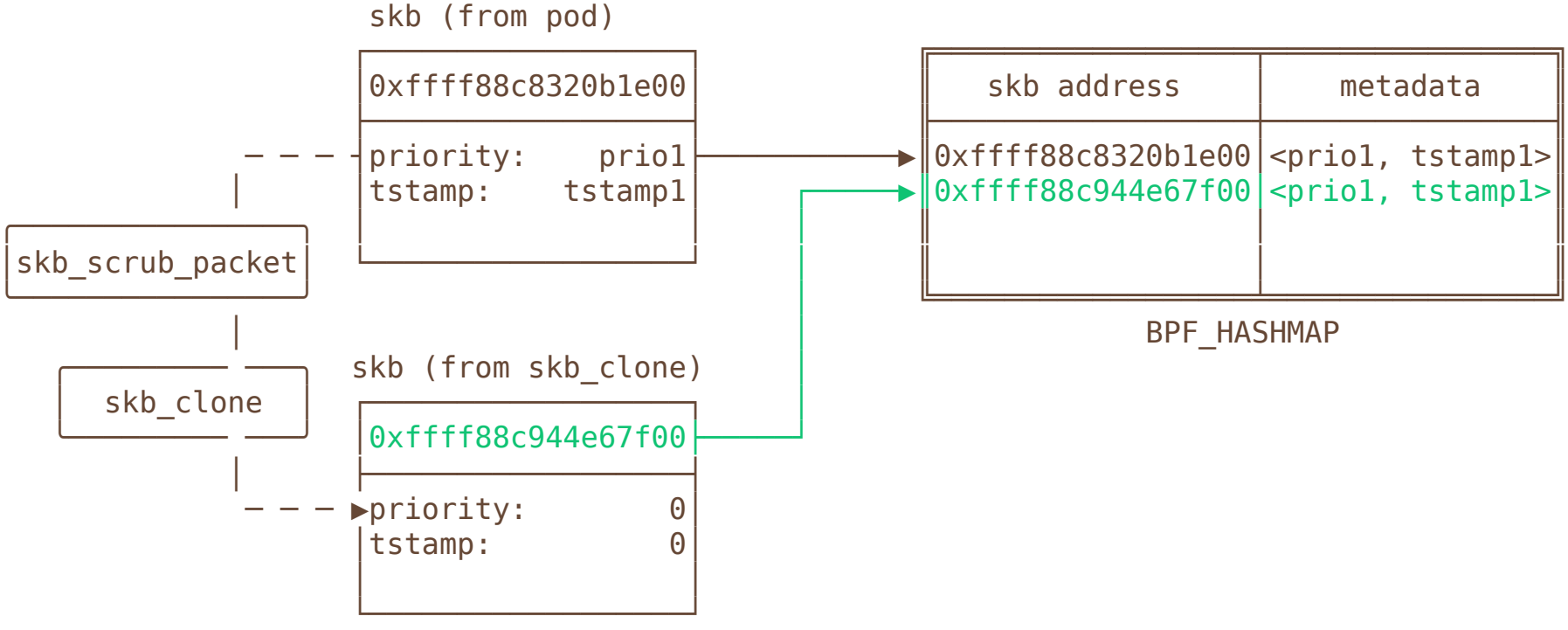
Pod initialization



TSN proxy operation



Packet tracking



Garbage collection

- Periodically check and remove the old entires from `BPF_HASHMAP`
- Implemented with `bpf_timer`
- Loading:

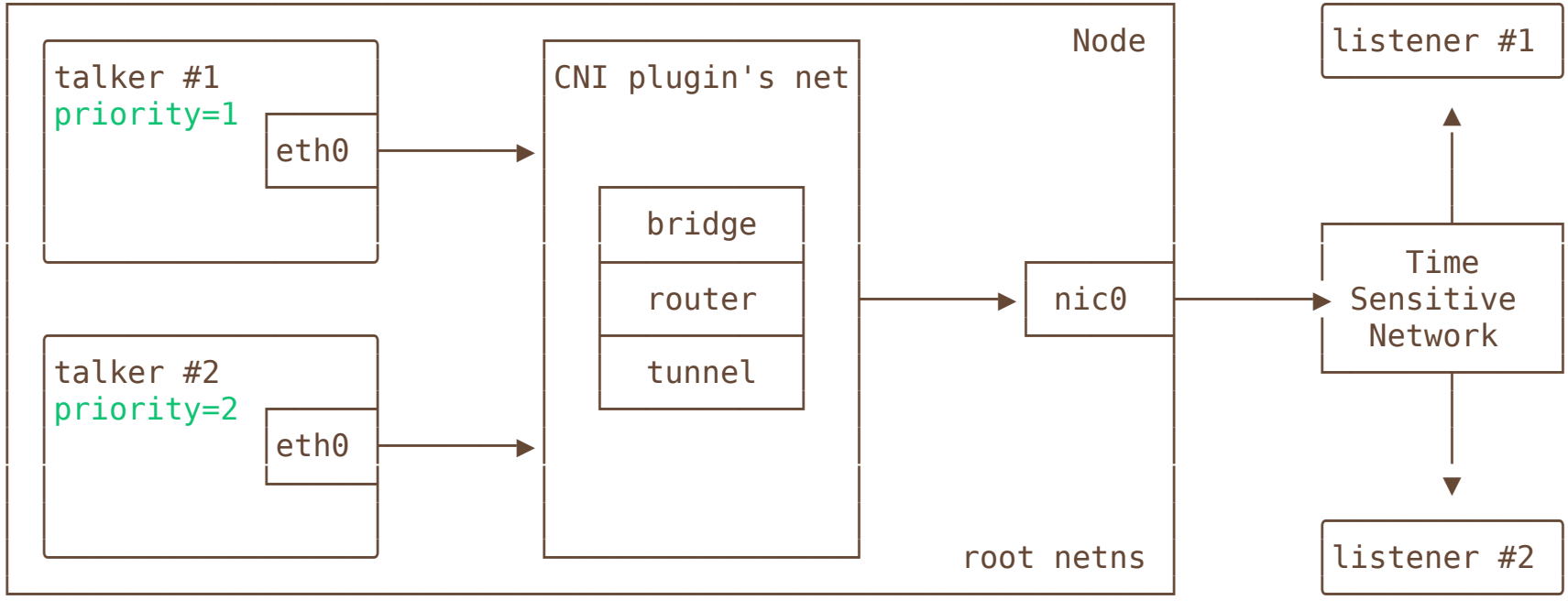
```
bpftool prog load garbage_collector.bpf.o /sys/fs/bpf/...
```

- Initialized from userspace (once)

```
bpftool prog run name garbage_collector data_in unused
```

- Timer callback rearm the timer indefinitely

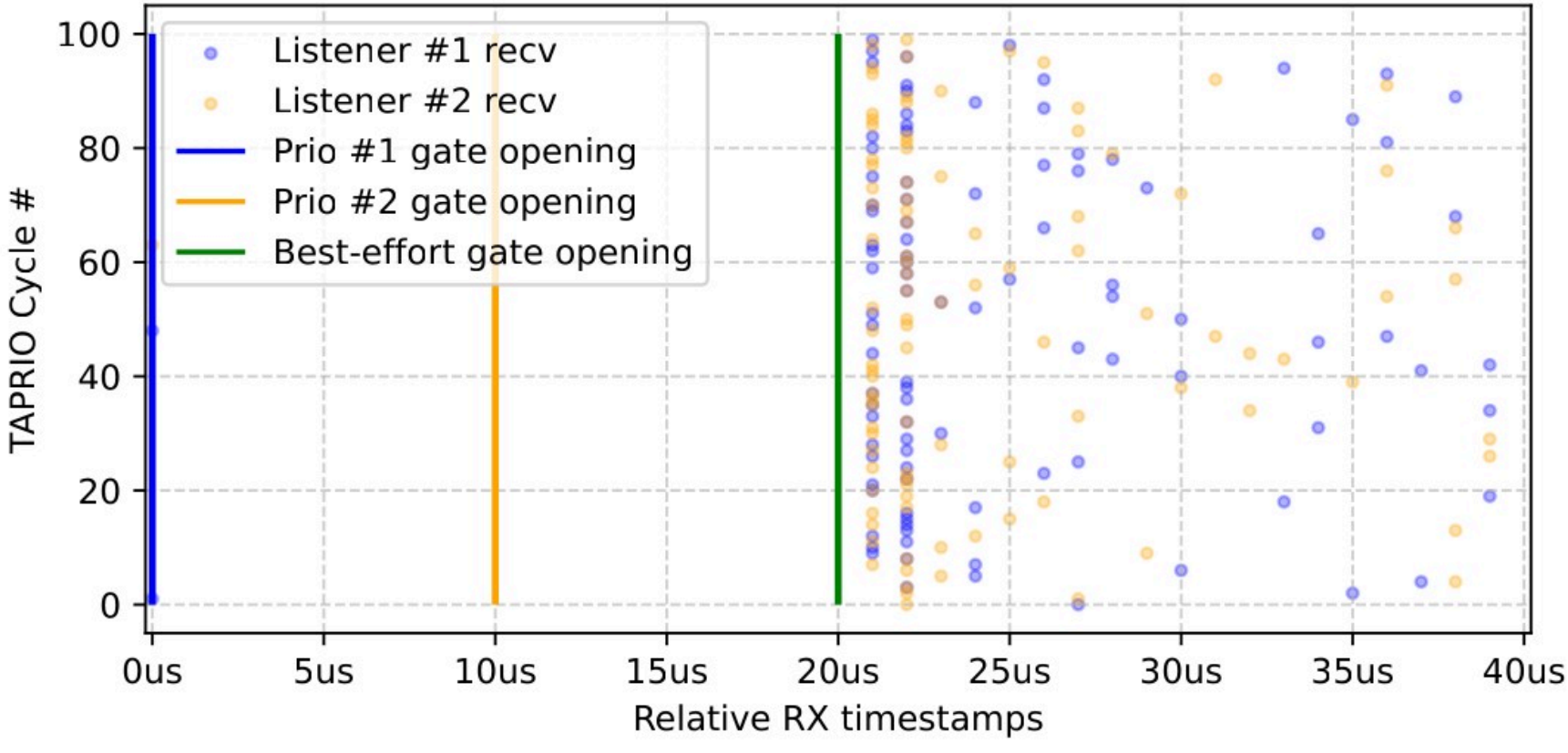
Example scenario



```
tc qdisc add dev tsn0 parent root handle 100 taprio \
  num_tc 4 \
  map 0 1 2 3 \
  queues 1@0 1@1 1@2 1@3 \
  base-time 1693996801300000000 \
  sched-entry S 03 10000 \
  sched-entry S 05 10000 \
  sched-entry S 01 20000 \
  clockid CLOCK_TAI
```

- `nic0` configured with `taprio` Qdisc
  - 1st slot: priority=1 packets, length is 10 usec
  - 2nd slot: priority=2 packets, length is 10 usec
  - 3rd slot: best effort, length is 20 usec

Without TSN proxy



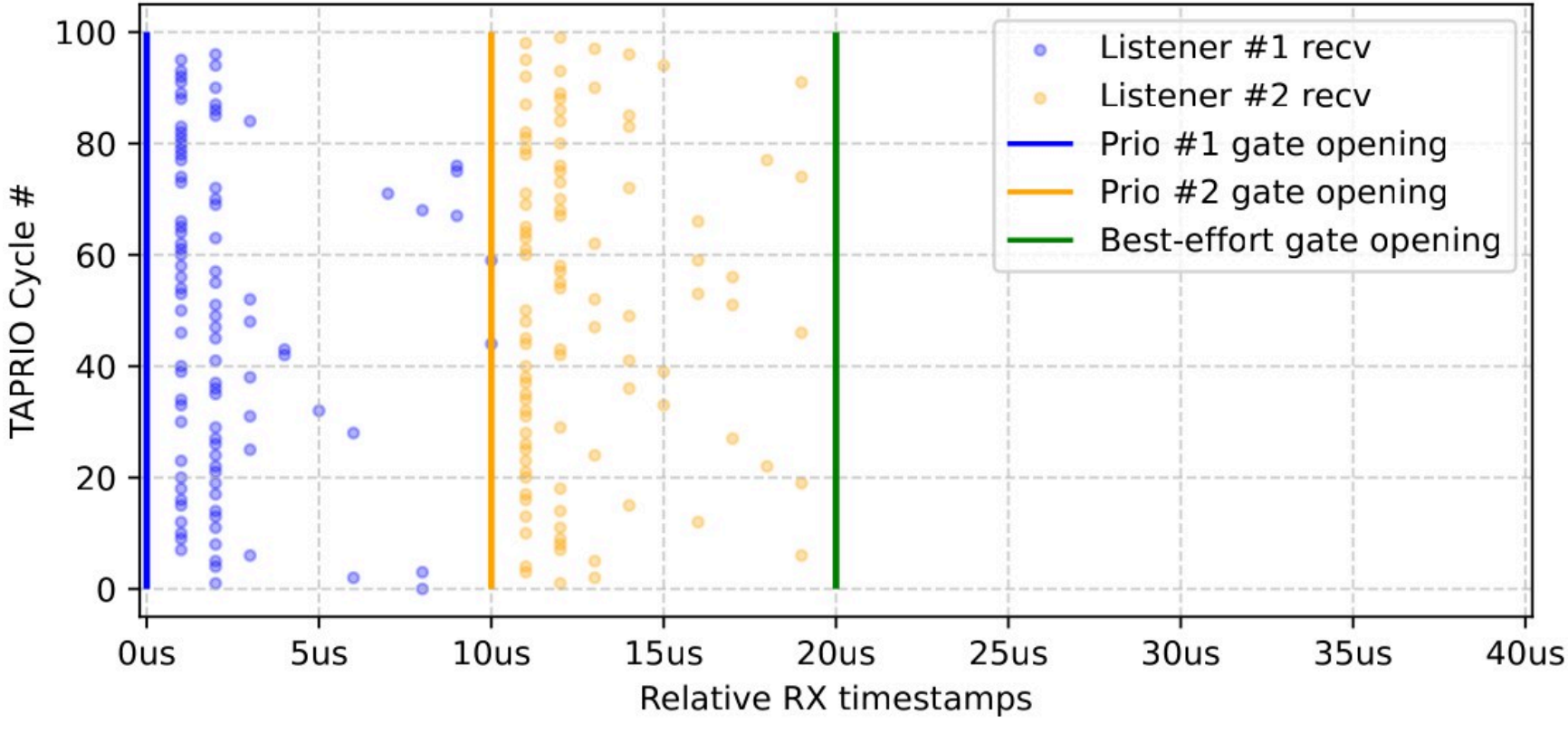
Packets missing their timeslots: skb->priority == 0

Installation

`kubectl apply -f tsn-proxy.yaml`

- Container image with artifacts ~9Mb
  - eBPF prog objects
  - static compiled `bpftool`
  - CNI plugin executable (bash script)
- No eBPF userspace
  - `bpftool` is good enough
  - not much state to manage

With TSN proxy



Packets scheduled according to their priority

Alternatives

- Netkit **v6.7**
  - Like veth but uses eBPF program as xmit callback
  - Packet scrubbing can be turned off with `ethtool`!
- Packet traits **RFC**
  - Per-packet KV store for metadata
  - Proper solution, no tracking needed
- Multus or kernel bypass
  - Multiple primary plugins
  - Configuration overhead, extra network

Limitations

- Cilium? Netkit?
- Tracking across XDP processing missing
  - `pwru` proved it is possible
- Performance analysis missing
  - Delay spikes?
  - Scalability?
- Tracking through userspace proxies not possible

Thank you!

Code: github.com/EricssonResearch/tsn-proxy

Slides & paper: netdevconf.info/0x19/15

E-mail: ferenc.fejes@ericsson.com

This work was supported by the European Union's Horizon 2020 research and innovation programme through DETERMINISTIC6G project under Grant Agreement no. 101096504

https://deterministic6g.eu/

1) Pod -> Kubelet: request net
2) Kubelet -> Pod: schedule
Kubelet -> CNI: request net
CNI -> Primary plugin: netns, IP
Primary plugin -> CNI: veths
CNI -> TSN proxy: netns, veth
TSN proxy -> Pod: attach eBPF prog

Pod: 1>2
Kubelet: