# Need to connect your k8s pods to multiple networks ?

## Multi-legged containers running wild with calico/vpp

Nathan Skrzypczak, Cisco Systems
(in collaboration with Mrittika Ganguli, PE, Intel Corporation)
FOSDEM 2023

# Your Speaker Today:

**Nathan Skrzypczak** - Software Engineer @ Cisco
Container networking things, Calico/VPP maintainer.

In collaboration with **Mrittika Ganguli**,
PE, Intel Corporation

And contributions from many awesome folks
@Intel, @Tigera, @Cisco

Me

# A k8s CNF ? Why ?

Our goal for this presentation was to :

- Take a packet processing application (e.g. Wireguard headend), i.e. a dynamic CNF
- Initially designed for bare-metal
- Make it run on K8s with all its benefits, but maybe not drawbacks :

| Pros | Cons |
|---|---|
| Scalability with SDN (serviceIPs, …) | Constrained Packet Ingress |
| Uniform deployments | Single interface, L3, … |
| Far from the silicon (I don't need to know) | Far from the silicon (performance) |

# Designing a solution

Application (e.g. Wireguard VPN)

CNI

routes


meta-CNI


Ipam



Dataplane

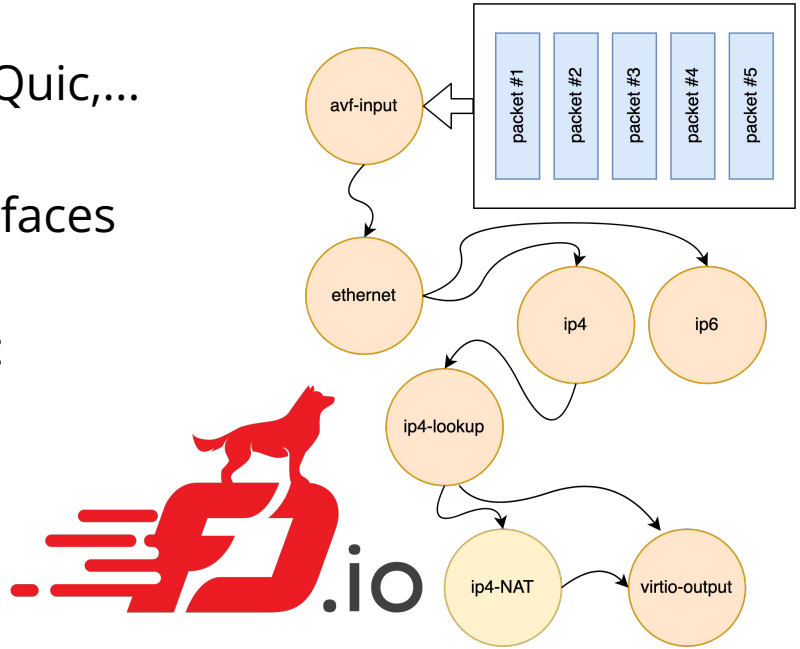
Blue boat wheel

# What is Calico?

- Open-source Kubernetes networking and network policy
- Kubernetes pods, nodes, VMs, and legacy workloads
- Rich network policy APIs
- Battle-tested: deployed in production at scale
- Support for multiple data planes

# What is VPP?

- Fast, open-source userspace networking dataplane https://fd.io/
- Feature-rich L2/L3/L4 networking Tunneling, NAT, ACL, crypto, TCP, Quic,...
- Easily extensible through plugins
- Supports virtual and physical interfaces
- Fast API >200k updates/second
- Highly optimized for performance: vectorization, cache efficiency
- Multi-architecture: x86, ARM



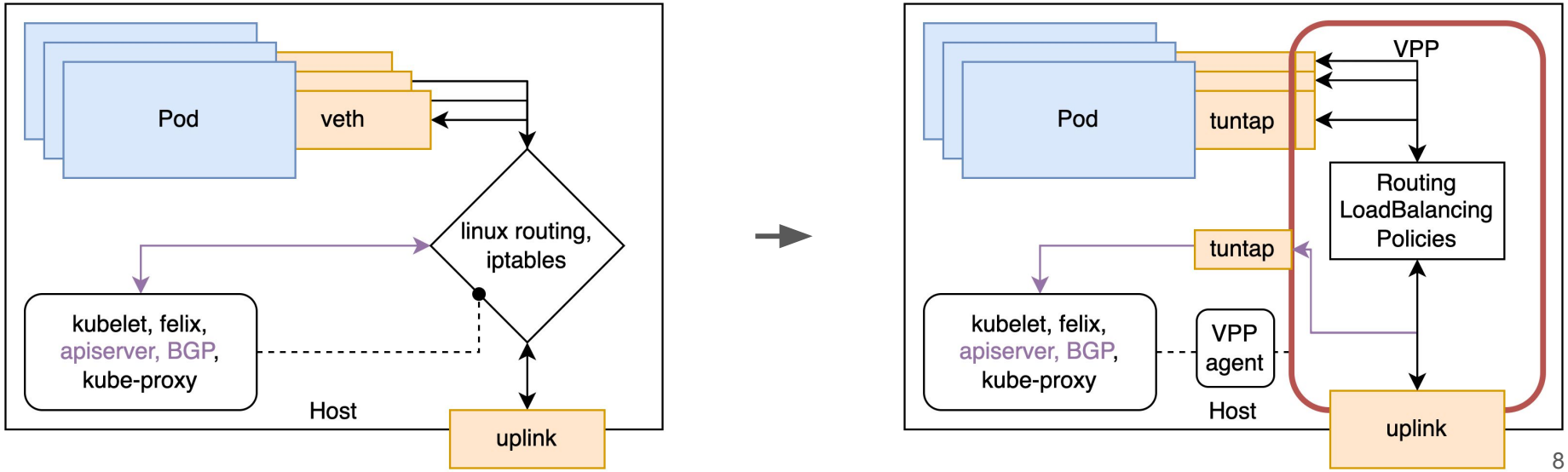
# A Calico CNI with VPP dataplane


- VPP dataplane option for Calico
  - Deployed on all nodes as a DaemonSet
  - Transparent for users (e.g. operator)
- Calico control plane configured to drive VPP
  - Dedicated NAT plugin for service load balancing
  - Dedicated plugin for fast Calico/k8s policies
  - Packet oriented interfaces support
- VPP optimized for container environments:
  - Interrupt mode, SCHED_RR scheduling
  - Lightweight (no hugepages, no dpdk, ...)
  - GRO / GSO support for container interfaces

```
# dedicated configmap for VPP settings
kind: ConfigMap
apiVersion: v1
metadata:
  name: calico-vpp-config
  namespace: calico-vpp-dataplane
data:
  # K8s service prefix. We currently cannot retrieve this from the API,
  # so it must be manually configured
  service_prefix: 10.96.0.0/12

  # Configure the name of VPP's physical interface
  vpp_dataplane_interface: eth1

  # Configures how VPP grabs the physical interface
  # available values are :
  # - ""         : will select the fastest driver among those supported for this interface
  # - avf        : use the native AVF driver
  # - virtio     : use the native virtio driver (requires hugepages)
  # - af_xdp     : use AF_XDP sock family (require at least kernel 5.4)
  # - af_packet  : use AF_PACKET sock family (slow but failsafe)
  # - none       : dont configure connectivity
  vpp_uplink_driver: ""

  # Configuration template for VPP.
  vpp_config_template: |-
    unix {
      nodaemon
      full-coredump
      cli-listen /var/run/vpp/cli.sock
      pidfile /run/vpp/vpp.pid
      exec /etc/vpp/startup.exec
    }
```

# How does it work ?

- VPP inserts itself between the host and the network
  - Uplink consumed with optimised drivers : DPDK / VPP-native drivers / AF_XDP
  - Pure layer 3 network model (no ARP/mac address in the pods)



# Designing a solution

Multi-legged containers

meta-CNI

MULTUS

WHERE ABOUTS

Ipam

Application (e.g. Wireguard VPN)

Blue boat wheel

"Regular" CNI deployment

CNI

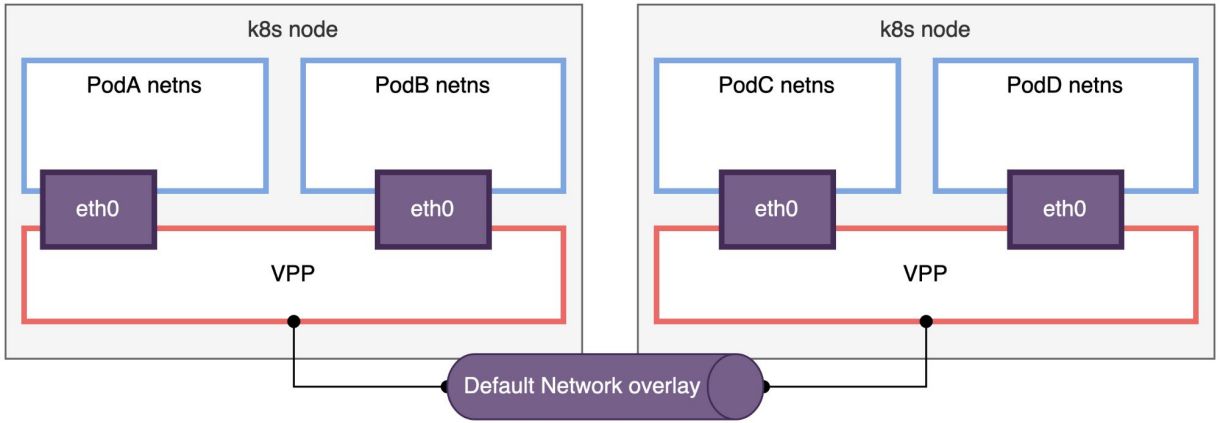
routes

GoBGP

.io

Dataplane

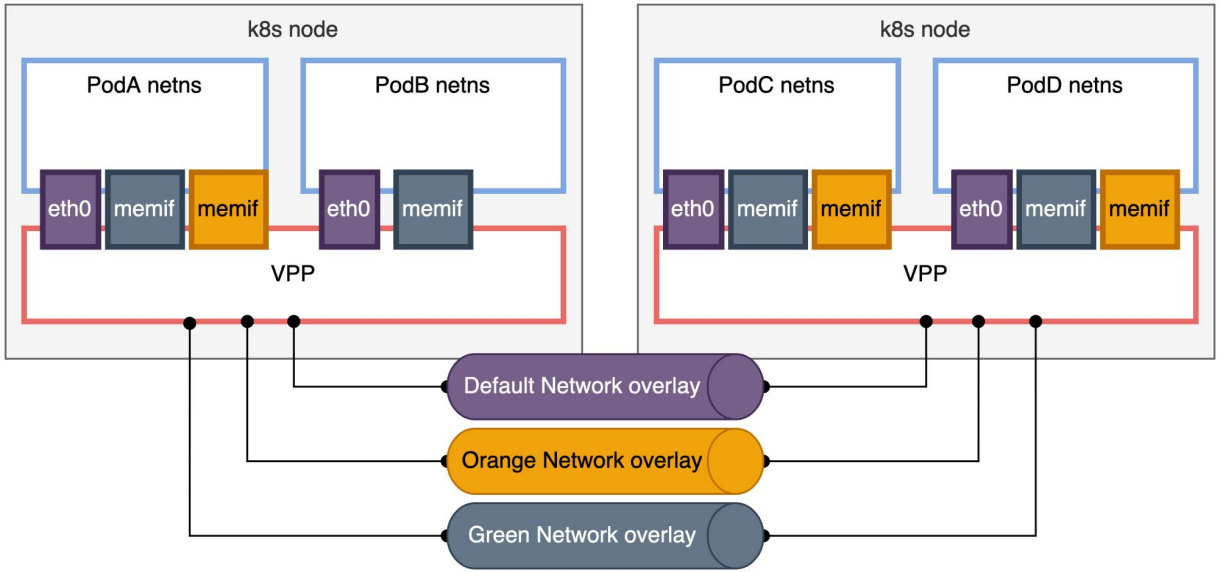
# Multiple pod networks




**Typical k8s networking**

- Single pod interface
- Optional overlay / encap to hide pod addressing

# Multiple pod networks




**Additional K8S networks**

- Provide isolation (akin to multiple clusters on top of each other)
- Pods optionally attach to networks
- Node to node isolation with encap (e.g. Vxlan)

# Multiple pod networks

- The CNI interface calls Calico once per pod
- Multus allows us to call Calico multiple times per pod
- VPP agent does the magic, making multiple networks out of those calls


- Using a dedicated IPAM (whereabouts) to support overlapping IPs

# Specification

```yaml
apiVersion: projectcalico.org/v3
kind: Network
metadata:
  name: blue
spec:
  vni: 1234
  range: "192.168.0.0/16"
```


```yaml
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: network-blue-conf
spec:
  config: |-
    {
      "name": "network-blue",
      "plugins": [
        {
          "type": "calico",
          "ipam": {
            "type": "whereabouts",
          }
          …
          "dataplane_options": {
            "network_name": "blue"
          }
        }
      ]
    }
```

**A network catalog**

- Defined as CRDs (standardized in KEP #3700)
- Carry the VxLan overlay spec

**Still need NetworkAttachmentDefs**

- 1:1 mapping to networks
- Consumed by multus

# Specification


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: samplepod
  annotations:
    k8s.v1.cni.cncf.io/networks: |-
      network-blue-conf@eth1,
      network-red-conf@eth2
spec:
  containers:
  ...
```

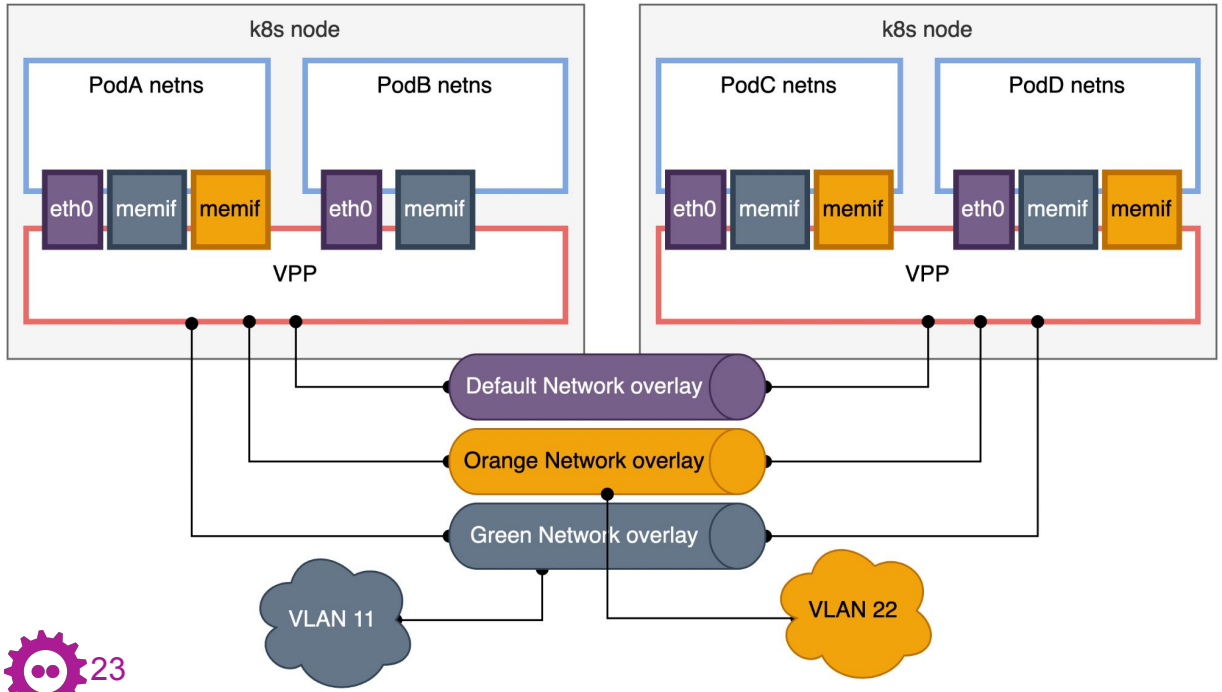

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
  annotations:
    extensions.projectcalico.org/selector: "app=MyApp"
    extensions.projectcalico.org/network: "blue"
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

**Mapping to applications**

- Annotations in pods
- Support for interface sizing (#queues, queue depth)
- Multiple interface types (L2, L3, memory interfaces…)

- Supports Services & Policies

# Multiple pod networks

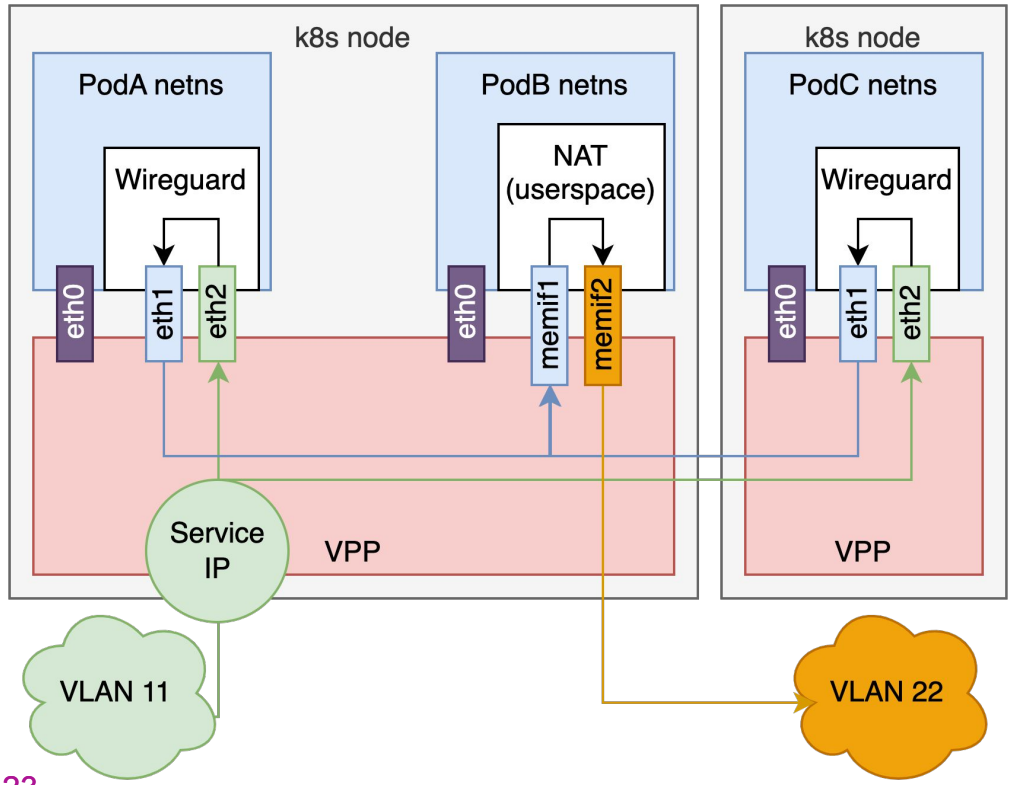



**Ingress / egress**

- BGP advertisement
- E.g. Fabric routing networks to the outside

*Still work in progress*

# Back to the application



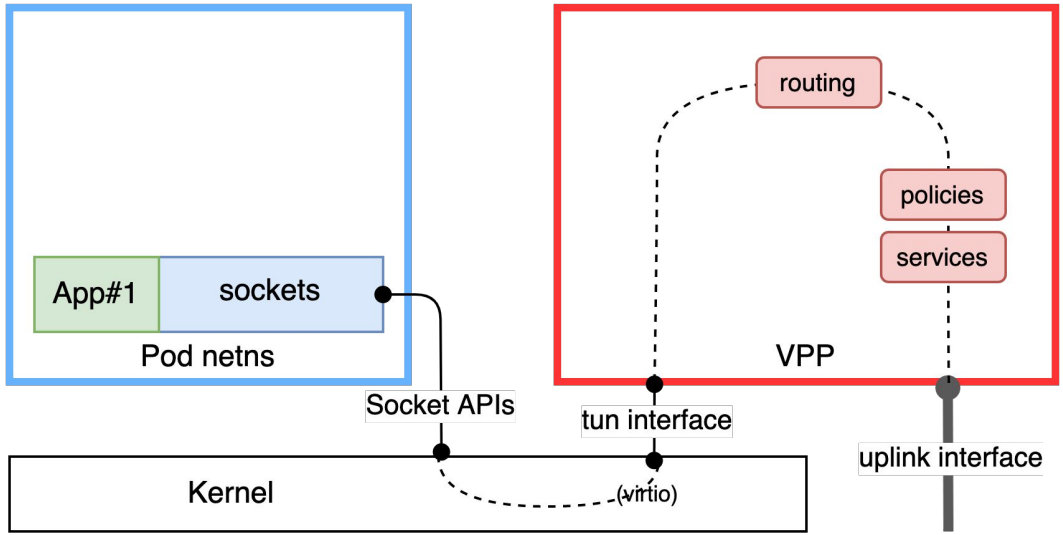

**Back to the application,**
building a VPN gateway

- The **Wireguard pod** decrypts, encapsulate in e.g. Geneve and forwards
- The **NAT pod** decapsulates Geneve srcNATs and sends
- ServiceIPs provide load-balancing (e.g. Maglev)

# Closer to the application

Now applications living in pods can consume multiple interfaces, what do they get ? By default Socket APIs.

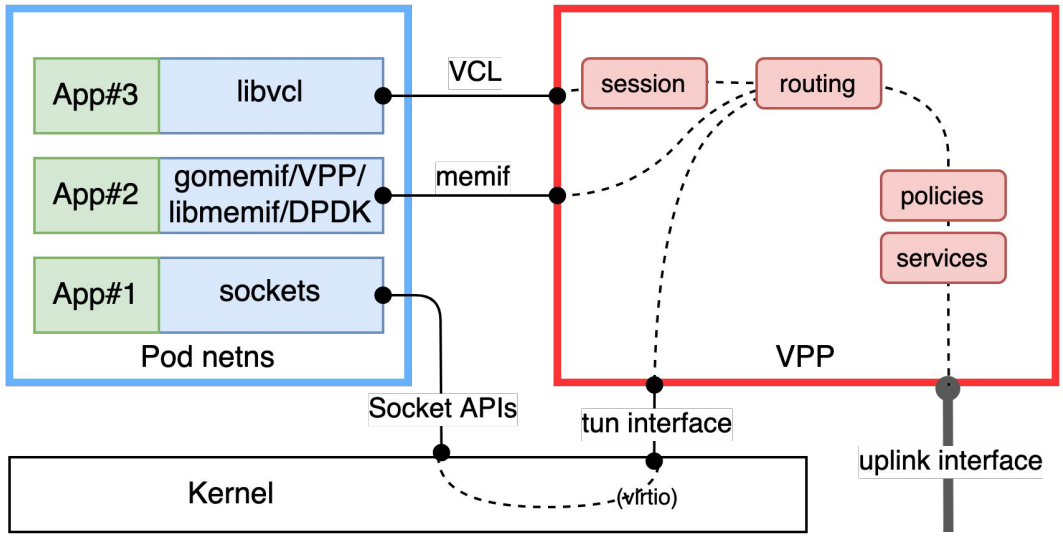
- Standard for apps
- But goes through the kernel
- Socket APIs were not designed for performance levels of modern apps
- Slower network (TCP, pps...) & crypto stack (hence GSO)
- Does two copies (VPP & socket)



# Optimizing the data path

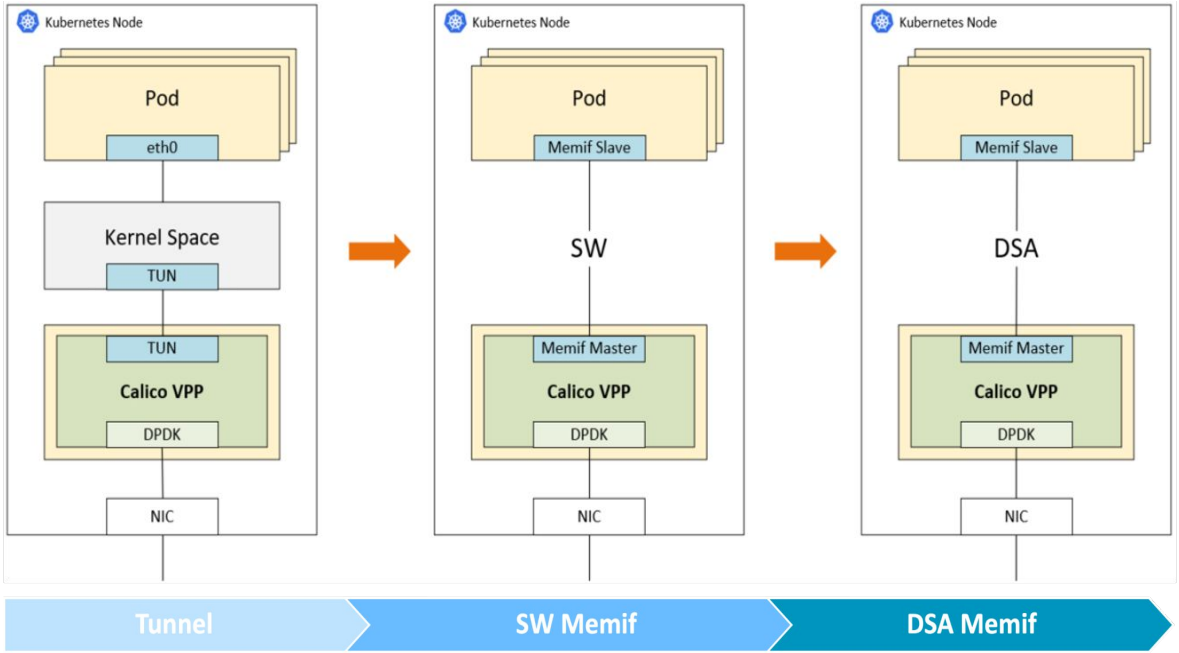
Going straight from VPP to the application ?

- If the application handles packets: **memif interfaces**
- If the application terminates L4+ connections: **VPP host stack**
- Exposed via pod annotations




- Full userspace networking
- Zero copy APIs
- Regular sockets still work (e.g. DNS)

# Re-enters the silicon

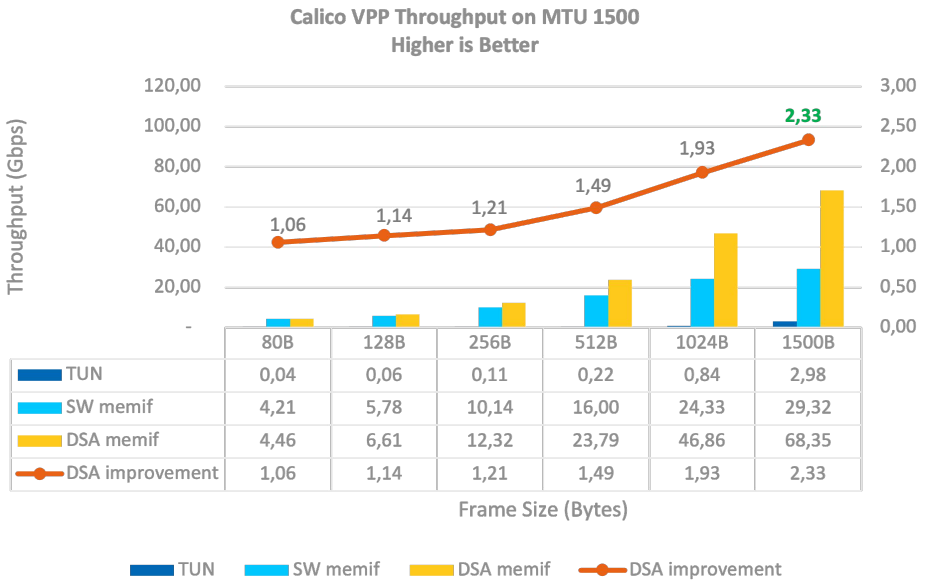



**memifs can be accelerated**

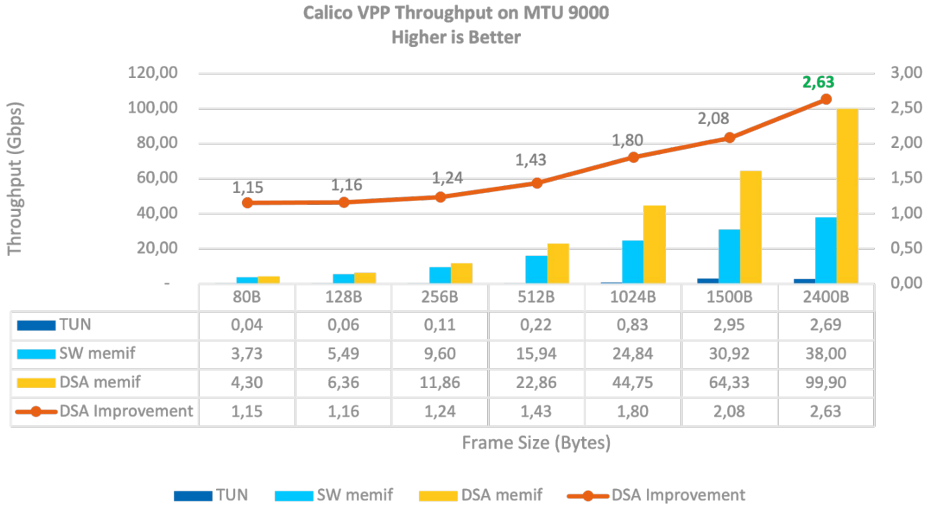
- With Intel(R) Data Streaming Accelerator (DSA) on 4th Generation Intel Xeon
- Exposed as a K8s plugin

- Transparent to the app
- Faster packet copies between VPP & the app

# Single Core Throughput Frame Size Scaling




| | 80B | 128B | 256B | 512B | 1024B | 1500B |
|---|---|---|---|---|---|---|
| TUN | 0,04 | 0,06 | 0,11 | 0,22 | 0,84 | 2,98 |
| SW memif | 4,21 | 5,78 | 10,14 | 16,00 | 24,33 | 29,32 |
| DSA memif | 4,46 | 6,61 | 12,32 | 23,79 | 46,86 | 68,35 |
| DSA improvement | 1,06 | 1,14 | 1,21 | 1,49 | 1,93 | 2,33 |

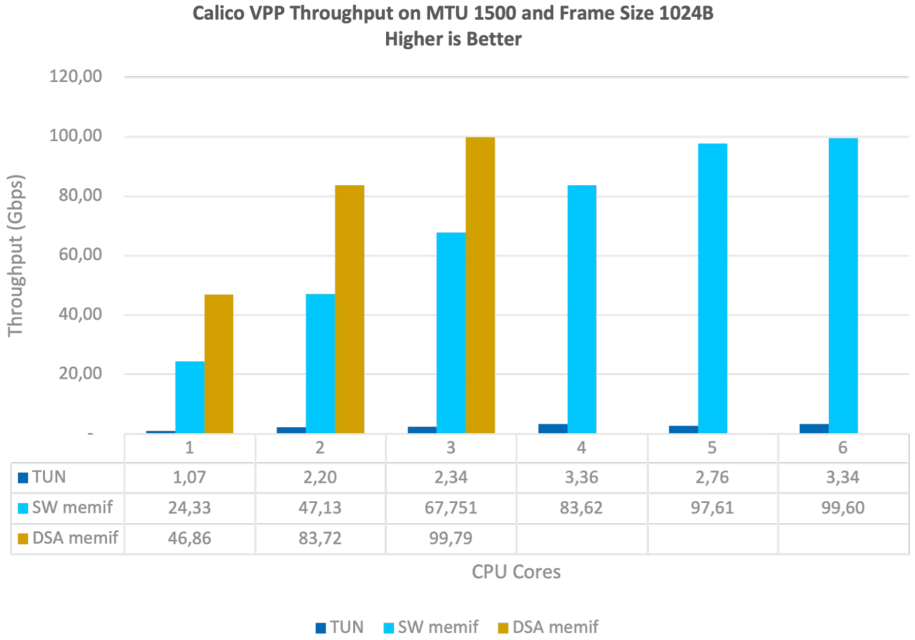


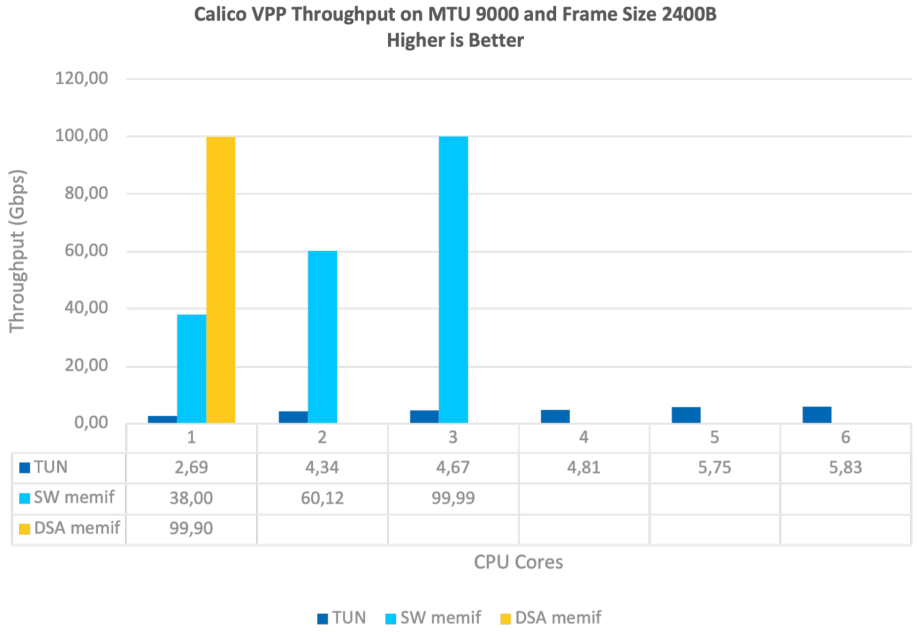

| | 80B | 128B | 256B | 512B | 1024B | 1500B | 2400B |
|---|---|---|---|---|---|---|---|
| TUN | 0,04 | 0,06 | 0,11 | 0,22 | 0,83 | 2,95 | 2,69 |
| SW memif | 3,73 | 5,49 | 9,60 | 15,94 | 24,84 | 30,92 | 38,00 |
| DSA memif | 4,30 | 6,36 | 11,86 | 22,86 | 44,75 | 64,33 | 99,90 |
| DSA Improvement | 1,15 | 1,16 | 1,24 | 1,43 | 1,80 | 2,08 | 2,63 |

```
Calico VPP Core Number: 1
VPP L3FWD Core Number : 1
Protocol              : TCP
Methodology : Maximum Receive Rate (MRR)
Operating Core Frequency : 3.8 GHz
Turbo                    : Enabled
NIC                      : 100 Gbps
DSA: 1 instance, 4 engines, 4 work queues
```

- Calico VPP performance with DSA memif is ~2.33 times SW memif when MTU is 1500 and ~2.63 times when MTU is 9000
- MTU 9000 has better performance for big packet flows
- TUN interface has low throughput across all frame size
- Calico VPP CPU usage is 100%

# Multiple Cores Throughput




```
Calico VPP Core Number: 1/2/3/4/5/6
VPP L3FWD Core Number: 1/2/3/4/5/6
Protocol: TCP
Methodology: Maximum Receive Rate (MRR)
Operating Core Frequency: 3.8 GHz
Turbo: Enabled
NIC: 100 Gbps
DSA: 1 instance, 4 engines, 4 work queues
```

- DSA memif needs 1-3 CPU cores to achieve the max **throughput**, while SW memif needs 3-6 cores, DSA memif can save up to 2-3 CPU cores

23

# Wrapping up

Many thanks to everybody who contributed to this work !

- Contributions welcome! https://github.com/projectcalico/vpp-dataplane
- Join us on the Calico Users Slack #VPP channel https://calicousers.slack.com/archives/C017220EXU1
- v3.25 is soon out, in beta, now aiming at GA

# About This Document

The performance measurement and analysis of an embedded platform for communication and security processing can be very challenging due to the diverse applications and workload inherent in the platform. The Network and Edge Group (NEX) at Intel is dedicated to performing lab measurements which will assist customers in understanding the performance of combinations of Intel® architecture microprocessors and chipsets

This report includes performance of Calico-VPP, measured on Intel® 4th Gen Xeon® Scalable Processor

This document publishes a set of indicative performance data for selected Intel® processors and chipsets. However, the data should be regarded as reference material only and the reader is reminded of the important Disclaimers that appear in this document

# Notices & Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

# Calico Benchmark Configurations

**8480**+: 1-node, pre-production platform with 2x Intel(R) Xeon(R) Platinum 8480+ on Intel M50FCP2SBSTD with 512 GB (16 slots/ 32GB/ DDR5 4800) total memory, ucode 0x9000051, HT on, Turbo on, Ubuntu 22.04 LTS, 5.15.0-48-generic, 1x 894.3G Micron_5300_MTFD, 3x Ethernet Controller E810-C for QSFP, 2x Ethernet interface, Calico VPP Version 3.23.0, VPP Version 22.02, gcc 8.5.0, DPDK Version 21.11.0, Docker Version 20.10.18, Kubernetes Version
1.23.12, IXIA Traffic Generator 9.20.2112.6, NIC firmware 3.20 0x8000d83e 1.3146.0, ice 5.18.19-051819-generic, Calico VPP Core Number: 1/2/3/4/5/6, VPP L3FWD Core Number: 1/2/3/4/5/6, Protocol: TCP, DSA: 1 instance, 4 engines, 4 work queues, test by Intel on 10/26/2022

# Thanks