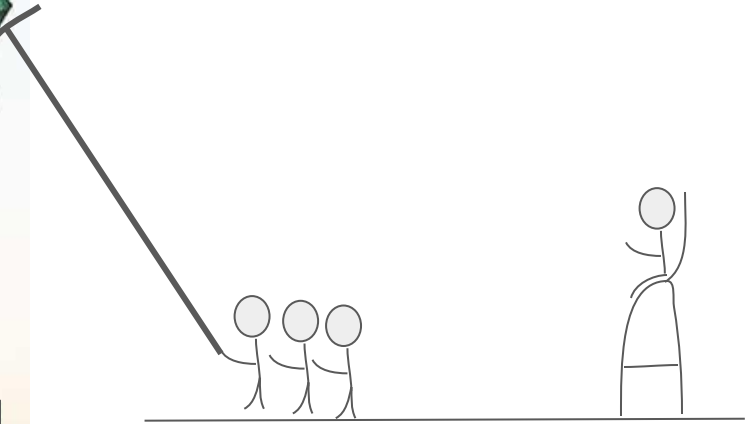
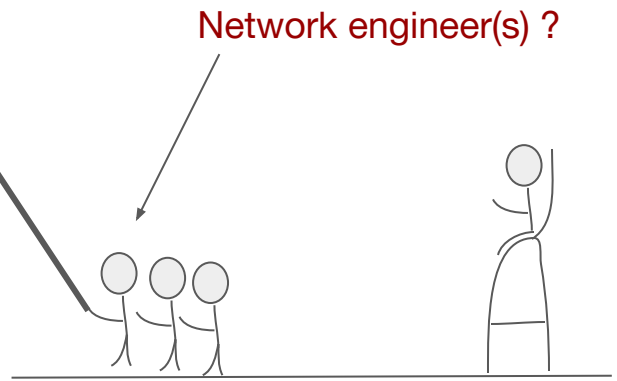
Kubernetes Networking

Marian Babik, Spyridon Trigazis
CERN


Container Platform of Choice




Container Platform of Choice

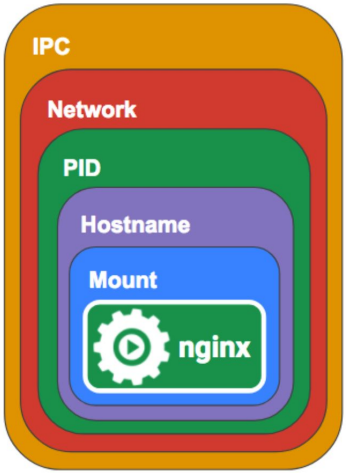

Network engineer(s) ?

Containers and Pods

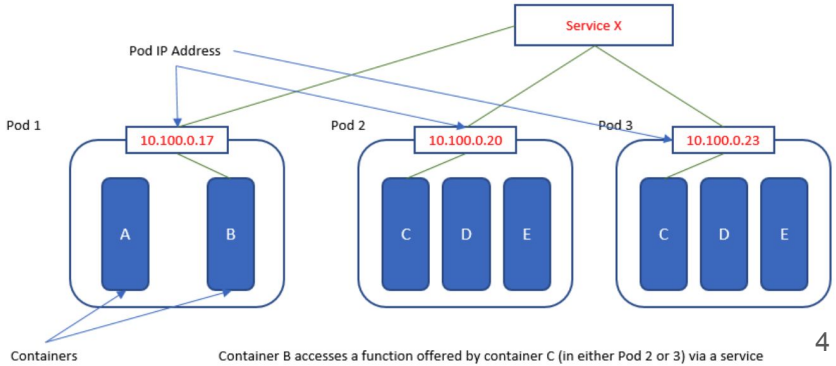
Container - regular process created by using two Linux Kernel features - *namespaces* and *cgroups*

Namespaces - provides means of isolation to a process, e.g. IPC, Network, PID, UTS, Users and Mount

Cgroups - way to manage/share resources for a collection of processes (CPU, Memory, Network, etc.)



Pod - groups multiple containers.
Each pod has a unique IP address;
containers in a pod can communicate via localhost;
containers betweens the pods must use IPs

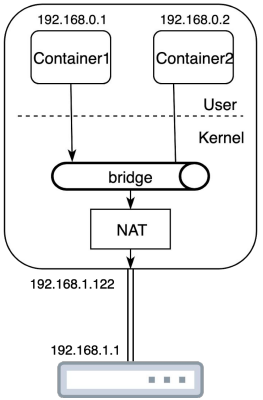


Container Networking - Single-host

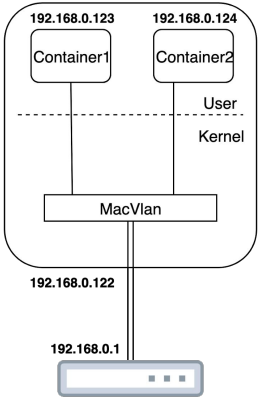
Typically two options: bridge or macvlan

Bridge is usually the default approach, connects all containers to the bridge, which acts as layer-2 bridge, and connects to the external world via NAT

MacVlan is the other option, it's layer-2 virtual network interface, which is connected to a physical interface. Kernel assigns to each MacVlan interface a unique MAC address and then uses it to change the MAC address of the outgoing packet and vice versa to match the correct container



(a) Container networking using NAT

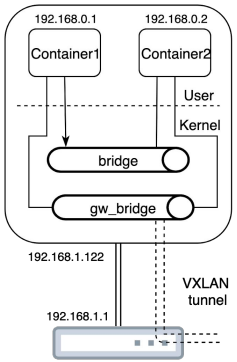


(b) Container networking using MacVlan

Container Networking - Multi-host

Basically there are two options:

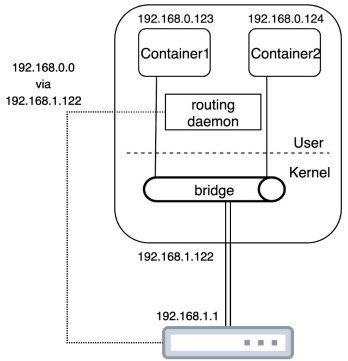
Overlay - usually VXLAN - each VTEP is connected to server and tunnel is established between the bridges
Coordination of IP address and tunnels is done by the orche-strator



(a) Container networking using VXLAN

Direct Routing - layer-3 solutions - uses single-host container network driver (usually bridge) and connects multiple container via routing

Usually a routing daemon runs on each server and announces container IP addresses or bridge subnets via BGP



(b) Container networking using direct routing

Container Network Interface (CNI)

Kubernetes uses a pluggable support for network “drivers” via so called Container Network Interface (CNI) - started in CoreOS to abstract networks for the container runtime

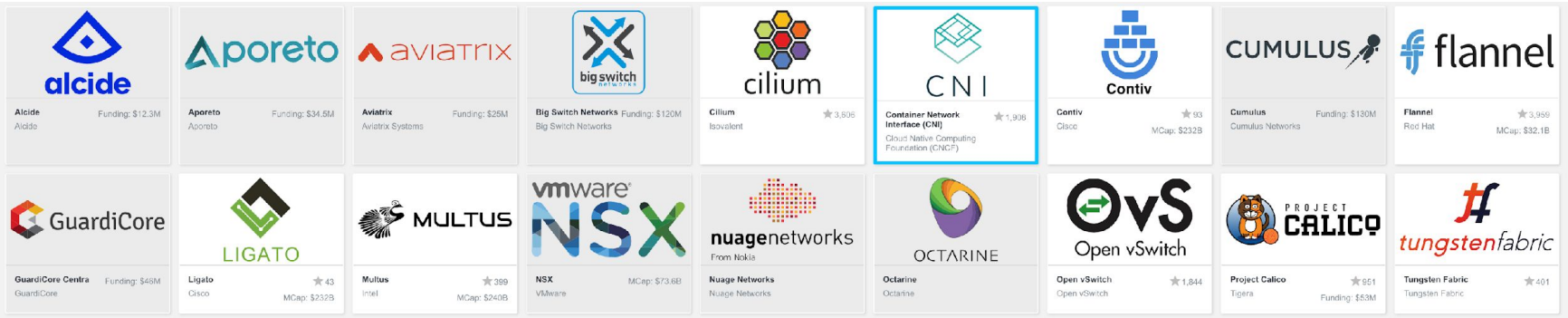
CNI plugin requirements:

- Expected to use no NAT for communication between the pods
- All nodes should communicate with all containers without NAT (and vice versa)
- IP that a container sees itself is the same IP that others see it as

There are many existing CNI implementations, which are part of the Cloud-Native Networking landscape

Cloud Native Networking

Range of approaches, both open-source (white) and commercial (grey) exist, tracked by the [Cloud Native Computing Foundation]



In open source there are currently three different types of projects:

- Hardware switches/NOS - open-source frameworks running on white boxes/ODMs (Cumulus)
- Software switches - running on servers/hypervisors (OVS, Tungsten)
- Linux kernel network extensions - **Flannel**, **Calico, Contiv, Cilium**, **WeaveNet**, etc.

Flannel

Flannel is the simplest way to configure a layer 3 network fabric for Kubernetes.

Usually serves as the default approach

Provides layer 3 IPv4-only network between multiple nodes using several different backends.

The recommended choice is the VXLAN (where only one vxlan network is created), host-gw (for direct routing, remote gw must be reachable via layer-2), UDP (for debugging purposes)

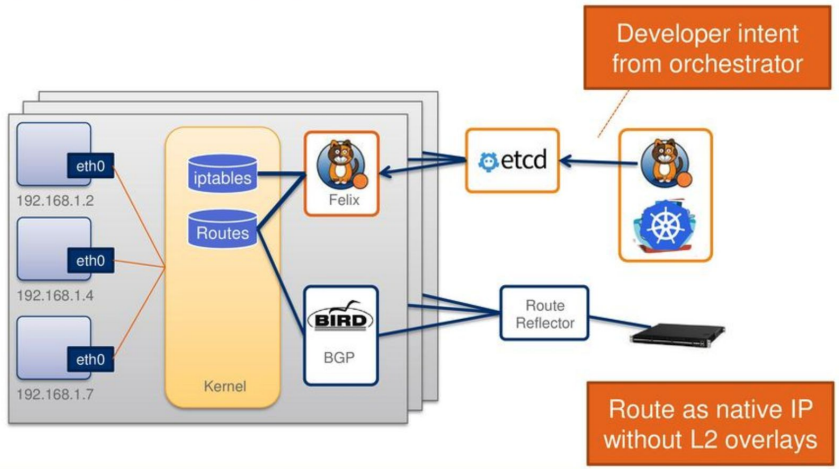
Experimental backeds: AWS, GCE and AliCloud VPC backends

Tigera's Calico

TIGERA

IP routed fabric that can work both on top layer-2 or layer-3 only networks and uses BGP peering to integrate
Four basic components:

- Orchestrator plugin (CNI)
- Data store (etcd)
- Felix - agent that runs on each serv
- BIRD - BGP client that distributes routes (optionally also BGP reflector)



Contiv/VPP

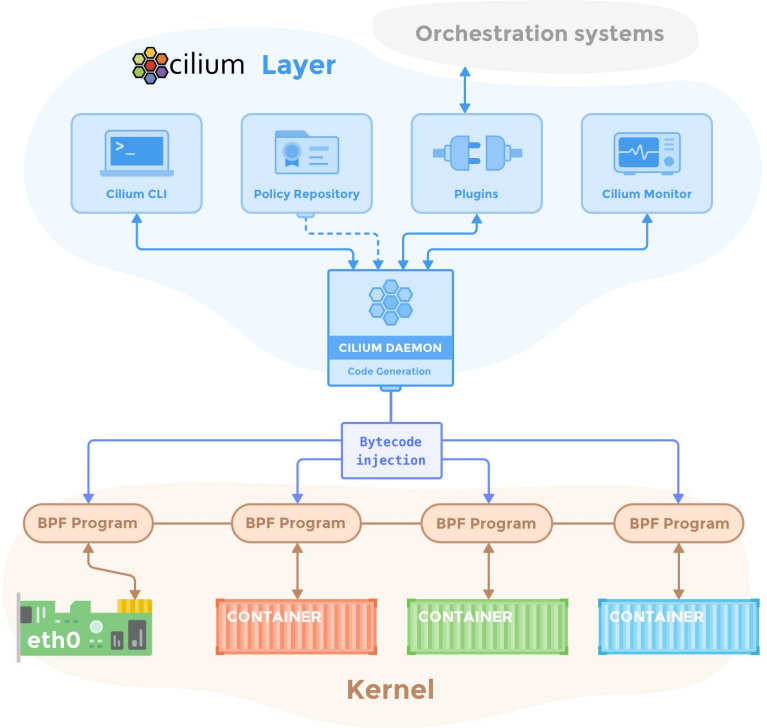
Contiv is a networking solution that directly integrates with Cisco Application Centric Infrastructure, it supports Kubernetes and OpenShift only.

- Using VLANs directly configured on the switches to interconnect the containers
  - with integration at the container level networking - docker network/linux bridge
- eBGP peering for layer-3
  - Communication between containers on different hosts runs natively using VLANs
  - Communication between container and non-containers is done by BGP peering
- FD.io VPP extension
  - Open source implementation of the **Cisco's Vector Packet Processing (VPP)** technology.
  - Accelerated by DPDK - software switch runs as another container in Kubernetes

Cilium

Cilium operates at Layer 3/4 for networking and Layer 7 for applications.

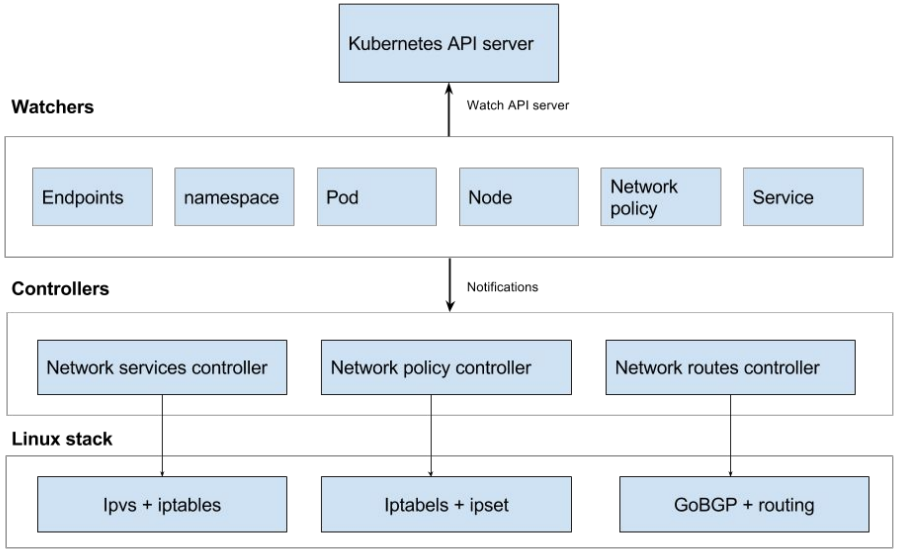
- Uses vxlan by default
- Uses eBPF programs
- Can be used with kube-router for bgp peering
- Can do service-bases LB With eBPF
- Security enforcement



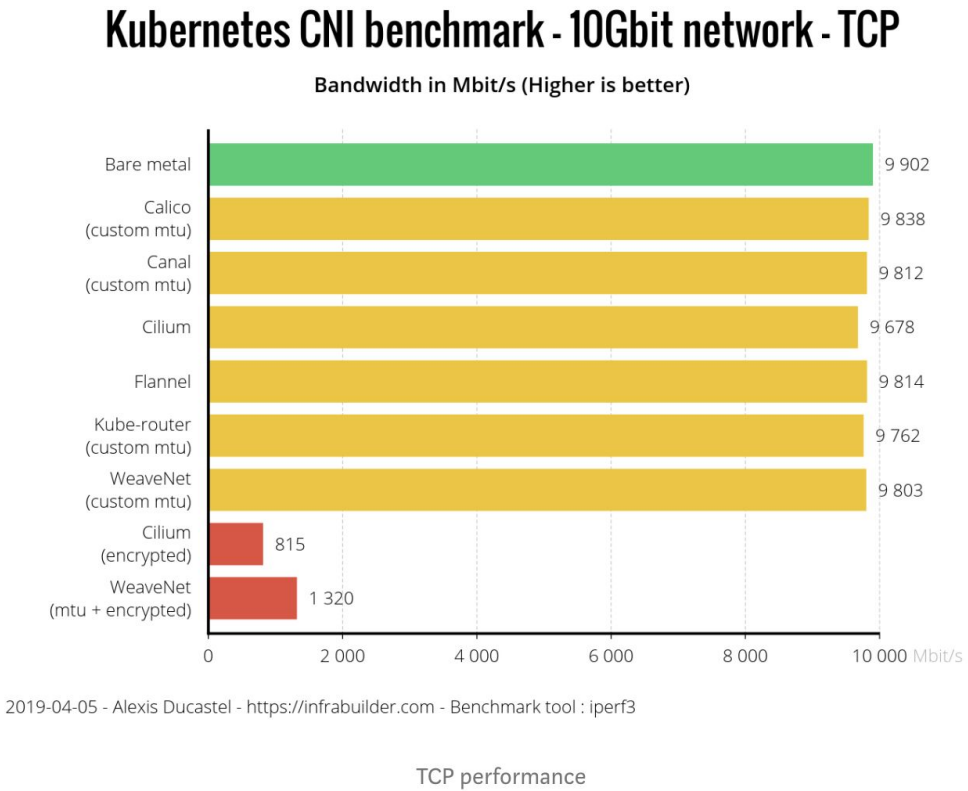
Kube-router

Kube-router uses iptables, ipvs/lvs, ipset and iproute2, not new CNI plugin.

- Can provide service proxy, firewall and pod networking
- Can replace kube-proxy
- Full BGP mesh
- overlay across subnets



Performance evaluation



TCP performance

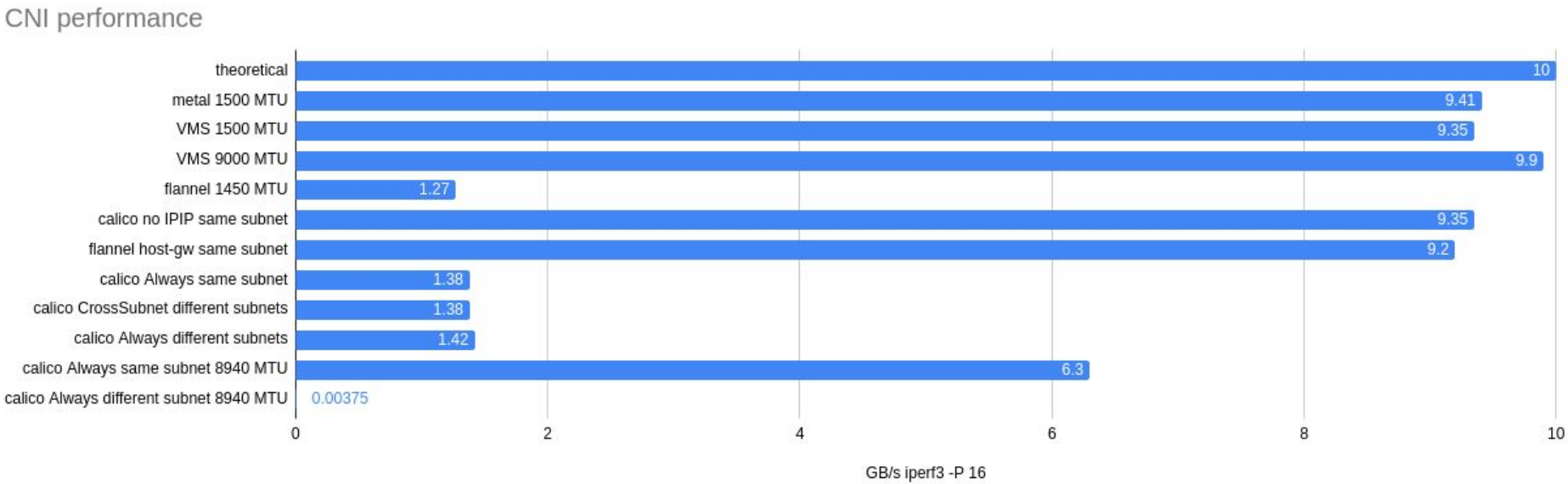
https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-april-2019-4a9886efe9c4

Performance evaluation @CERN (specs)

Motivation: Use cases like accessing storage systems (eos, cephfs) or DBs

- iperf tcp test between two nodes
- Compare 10Gbit, baremetal, virtual machines, pods
- Baremetal nodes connected on the same switch
- Virtual Machines IP in same or different subnet
- Flannel with host-gw or vxlan
- Calico with no encapsulation and IP-in-IP
- Different MTU configurations

Performance evaluation @CERN (results)



Linux packet processing

Linux kernel bypass models - improve networking performance by going around the Linux networking stack (needs modified device drivers), e.g.

- Intel Data Plane Development Kit (DPDK) -
- VPP (FD.io) - open source version of the Cisco's vector packet processing

But also, e.g OpenDataPlane, OpenFastPath, netmap, Snabb, pf_ring, etc.

Linux kernel fast path models, which try to process as much data early on the data path as possible, in Linux driver code or on NIC itself:

- eBPF - extended Berkeley Packet Filter (BPF)
- XDP - eXpress Data Path -
    - uses eBPF programs and performs processing RX packet-pages directly before the driver. It can run as native or offloaded (BPF in NIC, or via DPDK).

Linux kernel hardware offloading models (smartNICs)

Summary

Kubernetes and other container technologies will have significant impact on DC networking

Collaboration between compute and network engineers essential to achieve good performance

Our initial results not as optimistic as what was reported by others

Crossing subnets appears to have a serious performance impact

Plan is to try additional tests/benchmarks and publish our results (in a blog)

- Try Cilium and Kube-router, test accessing remote filesystem and mysql

References

[0] Cilium webinar https://www.cncf.io/webinars/how-cilium-uses-bpf-to-supercharge-kubernetes-networking-security/

[1] Cilium docs https://docs.cilium.io/en/v1.6/concepts/networking/

[2] Calico docs https://docs.projectcalico.org/v3.11/introduction/

[3] Flannel docs https://github.com/coreos/flannel#documentation

[4] CNI benchmark https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-36475925a560

[5] kube-router docs https://www.kube-router.io/docs/user-guide/#try-kube-router-with-cluster-installers

HEPiX NFV WG report
https://docs.google.com/document/d/1w7XUPxE23DJXn--j-M3KvXlfXHUnYgsVUhBpKFjyjUQ/edit?usp=sharing