KubeCon | CloudNativeCon

North America 2023

KubeCon | CloudNativeCon
North America 2023

High Performance, Low Latency Networking for Edge & Telco


Dan Daly, Intel


Nupur Jain, Intel


Vipin Jain, AMD


Ian Coolidge, Google


Nabil Bitar, Bloomberg

High Performance, Low Latency Networking for Edge & Telco

Abstract: Traditional edge & telco deployments are often containerized and use Kubernetes, however applications written for these environments struggle to be cloud native as their network intensive workloads require SR-IOV and kernel bypass to maximize bandwidth and minimize latency/jitter. In this panel we will discuss the different approaches for supporting Kubernetes Network Infrastructure Offload, an implementation agnostic solution for providing high performance, low latency network connections using standard Kubernetes networking. As a follow-up to our panel last year, we will update on the standardization and open-source developments for offloading Kubernetes networking operations such as endpoint discovery, pod connectivity, service scale, load balancing, and network policy. This offload does not require end-users to make code changes to their CNFs or VNFs and can simplify deployment and management by removing the need to run SR-IOV in the cluster.

Telco: Network Sensitive Workload

Abstract
Velocity
Ease-of-Use
Scale Out

Optimize
Speed
Lower Latency
Scale Up

5G Data: Need for **Speed**
Voice & Video: **Latency** Sensitive
Big Conferences in Chicago: Need for **Scale**!

Optimize Under the Abstraction

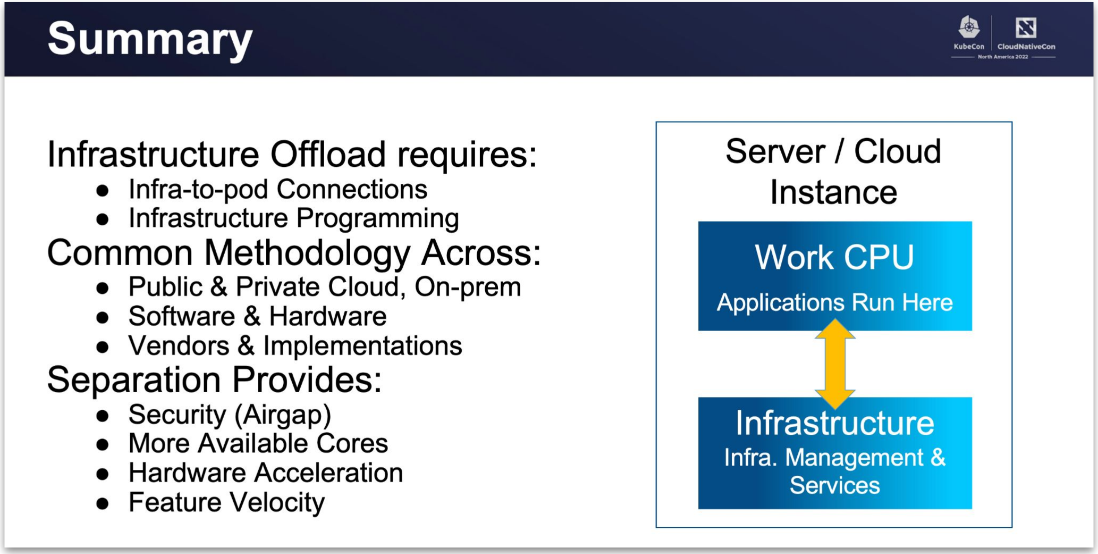
[Kubernetes Networking Infrastructure Offload - Dan Daly & Nupur Jain, Intel; Nabil Bitar, Bloomberg; Moshe Levi, Nvidia; Vytautas (Valas) Valancius, Google]

No changes to applications or to end users other than improved performance

Higher Bandwidth, Lower Latency Using DPUs, IPUs, & Optimized Software

Standardize Approach through CNCF & Open Programmable Infrastructure (OPI)

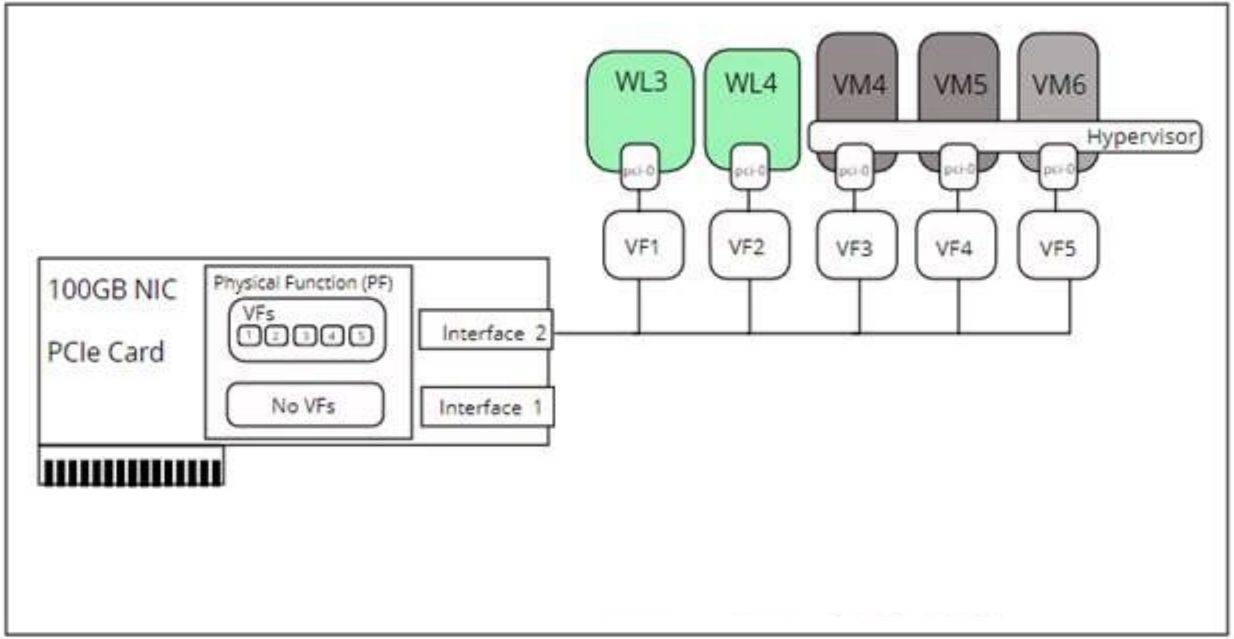
How can we apply this to
Network Sensitive Workloads like Telco



Our Panel Today

Ian Coolidge, Google	Nabil Bitar, Bloomberg	Vipin Jain, AMD	Nupur Jain, Intel
Motivating Use Case: Telco Edge	Value of Offload	Standardized Offloads Multi-Vendor Support	Working Example

SR-IOV: Introduction



- Split a NIC into “virtual functions”, to be distributed to containers or VMs
- Direct access to HW queues for bypassing kernel network stack for high throughput and low latency (typically via DPDK)

SR-IOV Pain Points: Hardware Abstraction

Hardware *details*, not *capability*, leak up to workload orchestration

```
// SriovNetworkNodePolicySpec defines the desired state of SriovNetworkNodePolicy
type SriovNetworkNodePolicySpec struct {
        // SRIOV Network device plugin endpoint resource name
        ResourceName string `json:"resourceName"`
        // NodeSelector selects the nodes to be configured
        NodeSelector map[string]string `json:"nodeSelector"`
        // +kubebuilder:validation:Minimum=0
        // +kubebuilder:validation:Maximum=99
        // Priority of the policy, higher priority policies can override lower ones.
        Priority int `json:"priority,omitempty"`
        // +kubebuilder:validation:Minimum=1
        // MTU of VF
        Mtu int `json:"mtu,omitempty"`
        // +kubebuilder:validation:Minimum=0
        // Number of VFs for each PF
        NumVfs int `json:"numVfs"`
        // NicSelector selects the NICs to be configured
        NicSelector SriovNetworkNicSelector `json:"nicSelector"`
        // +kubebuilder:validation:Enum=netdevice;vfio-pci
        // The driver type for configured VFs. Allowed value "netdevice", "vfio-pci". Defaults to netdevice.
        DeviceType string `json:"deviceType,omitempty"`
        // RDMA mode. Defaults to false.
        IsRdma bool `json:"isRdma,omitempty"`
        // mount vhost-net device. Defaults to false.
        NeedVhostNet bool `json:"needVhostNet,omitempty"`
        // +kubebuilder:validation:Enum=eth;ETH;ib;IB
        // NIC Link Type. Allowed value "eth", "ETH", "ib", and "IB".
        LinkType string `json:"linkType,omitempty"`
        // +kubebuilder:validation:Enum=legacy;switchdev
        // NIC Device Mode. Allowed value "legacy","switchdev".
        EswitchMode string `json:"eSwitchMode,omitempty"`
        // +kubebuilder:validation:Enum=virtio;vhost
        // VDPA device type. Allowed value "virtio", "vhost"
        VdpaType string `json:"vdpaType,omitempty"`
        // Exclude device's NUMA node when advertising this resource by SRIOV network device plugin. Default to false.
        ExcludeTopology bool `json:"excludeTopology,omitempty"`
        // don't create the virtual function only allocated them to the device plugin. Defaults to false.
        ExternallyManaged bool `json:"externallyManaged,omitempty"`
}
```

SR-IOV Pain Points: Network Policy & Services LB

Since SR-IOV VF interfaces are not managed via Kubernetes, you must roll your own network policy and service load balancing

SR-IOV Pain Points: Continued

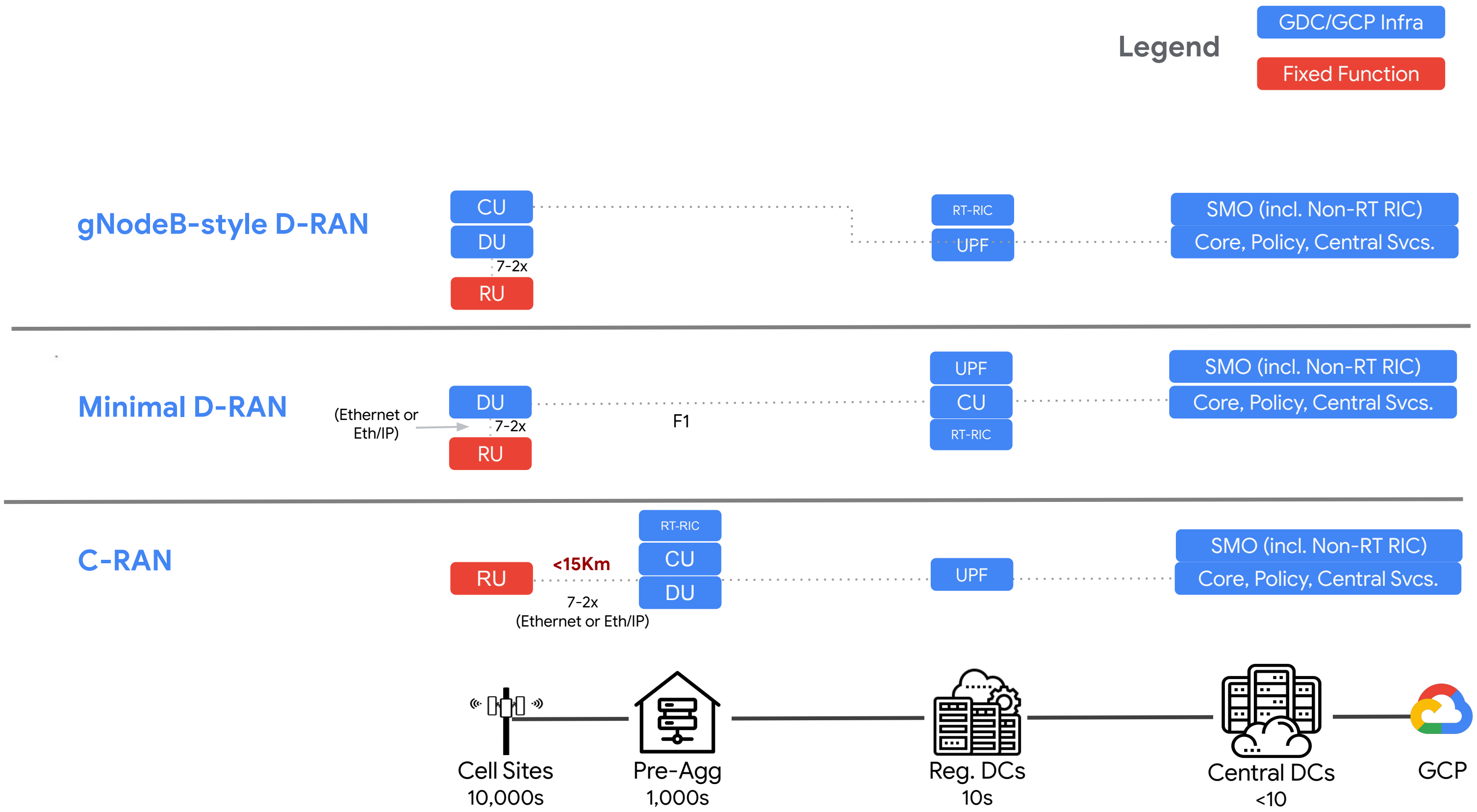
Draining

Workloads are drained from a node during reconfiguration since SR-IOV reconfiguration and VF usage are asynchronous

Enabling SR-IOV

Some NICs require a command line tool and reboot, others require a BIOS option ROM setting and reboot

Telco Cloud: Varied Deployments



Telco Cloud: Fungibility Goal

- Workloads can be deployed anywhere easily
- Minimize coupling to the hardware (NICs, routers)
- Scale up can happen in public cloud when necessary

Kubernetes Networking looks and feels the same on premises and in public clouds!

Multi-Network takes this even further! Now multiple interfaces can be used with Kubernetes Networking.

Outline

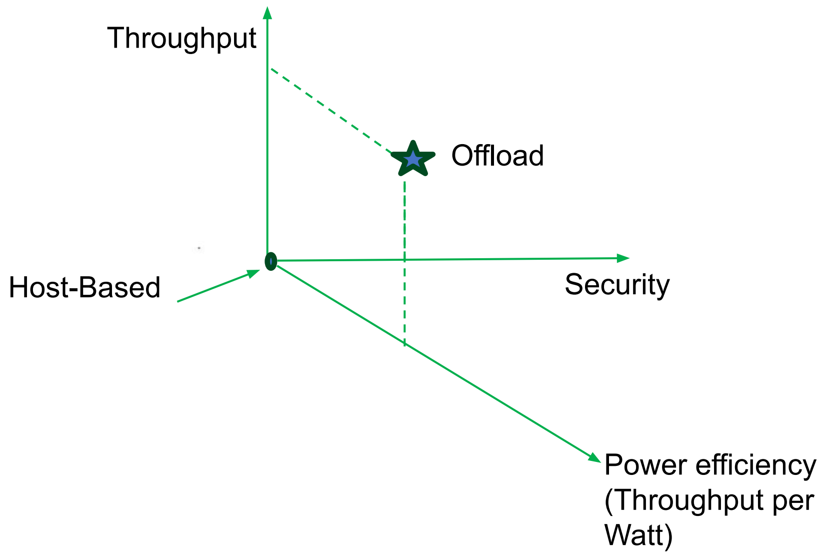
- Offload - definition and drivers
- Control plane - target and implementation state
- Some offload experimental results
- Goals

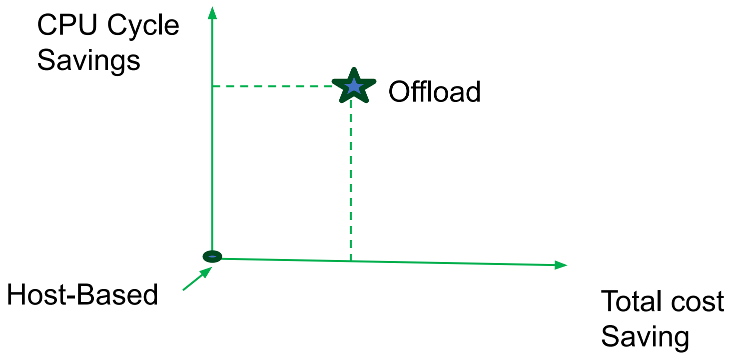
DPU/IPU Networking Offload - Refresher

- Packet processing and control offload from software on the host CPU to network interface card (NIC) hardware. Examples:
  - Network access policy enforcement
  - Encryption (e.g., TLS, IPsec)
  - NVMe/TCP
  - Load balancing
  - Flow based action (allowed, denied) and statistics
  - Networking Control plane (e.g., Calico)
- Solutions in Market Place (DPU/IPU) - Examples:
  - Intel IPU
  - AMD Pensando DPU
  - Nvidia DPU (Bluefield)

Why offload – The Drivers/Anticipations

Transcending centralized and Edge cloud

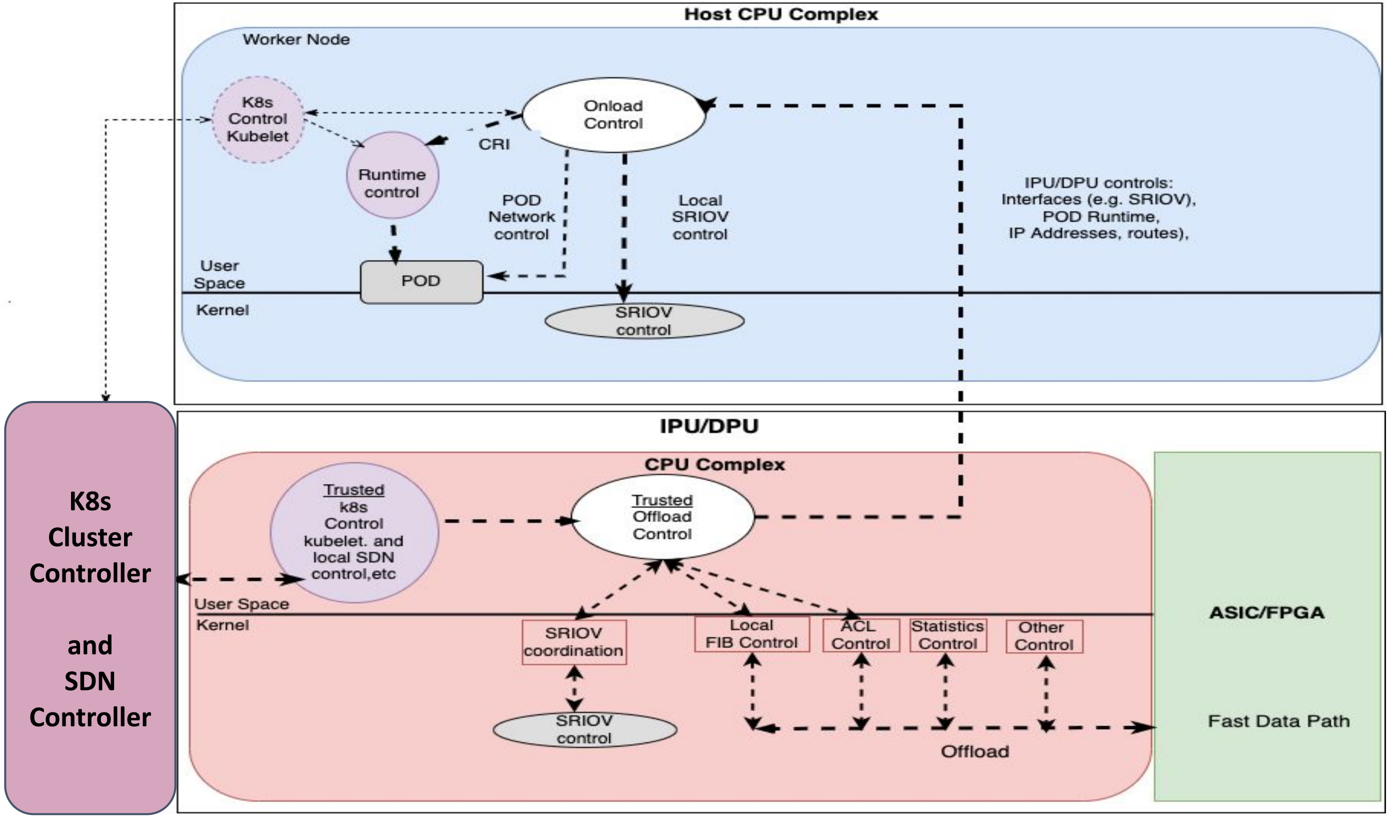




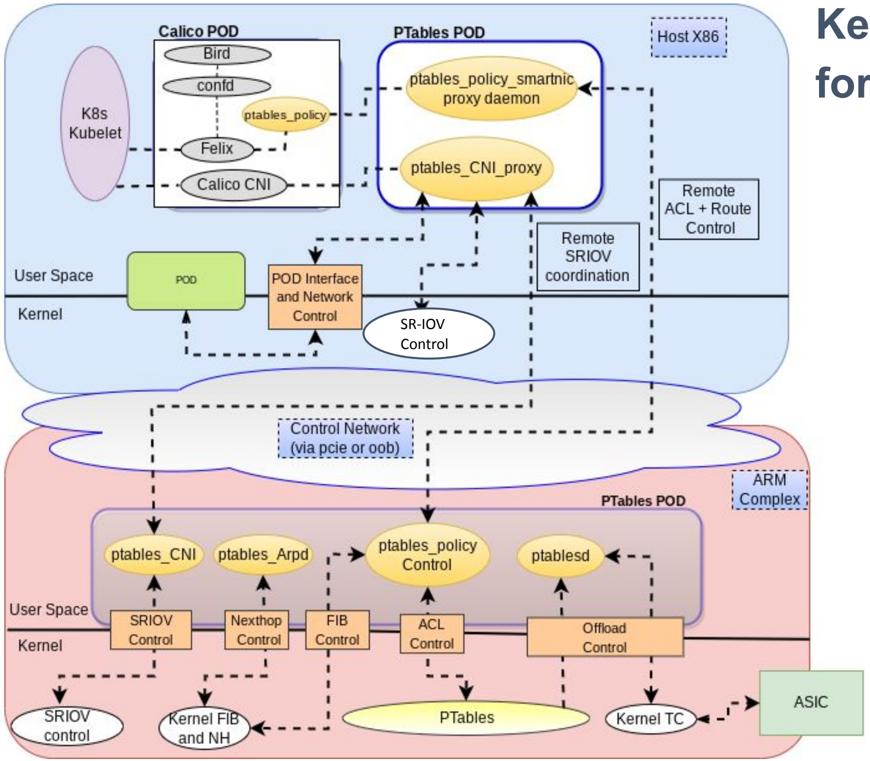
Some anticipated offload advantages to be further vetted

Target Architecture - Offload and Security

Functional distribution between host processor and DPU/IPU



What we implemented: Calico Integration With a DPU - a milestone in the journey



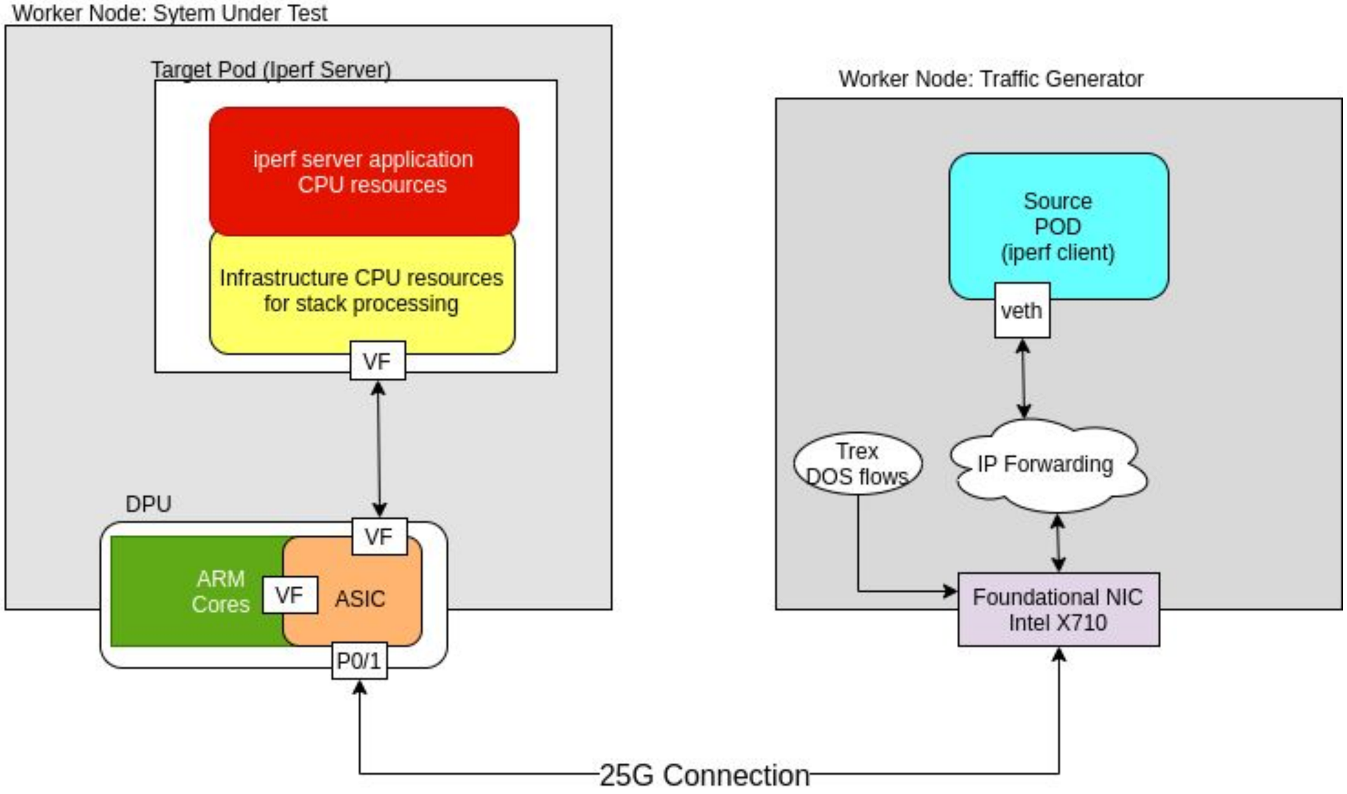
Kept Calico CNI and modified Felix part of Calico for new datapath → Felix-ptables

7 agents total and associated interfaces

1. *ptables_proxy*
   - transform Felix API into TCP network API
2. *ptables_cni_proxy and ptables_cni*
   - Coordinates SR-IOV on host and DPU
   - Handles networking config on POD
3. *ptables_policy_smartnic* and *ptables_policy*
   - Injects routes into DPU
   - Injects ACLs into policy tables Ebpf kernel
4. *ptables_arpd*
   - Handles POD ARP requests
5. *ptablesd*
   - Handles offloading handed by ptables eBPF kernel

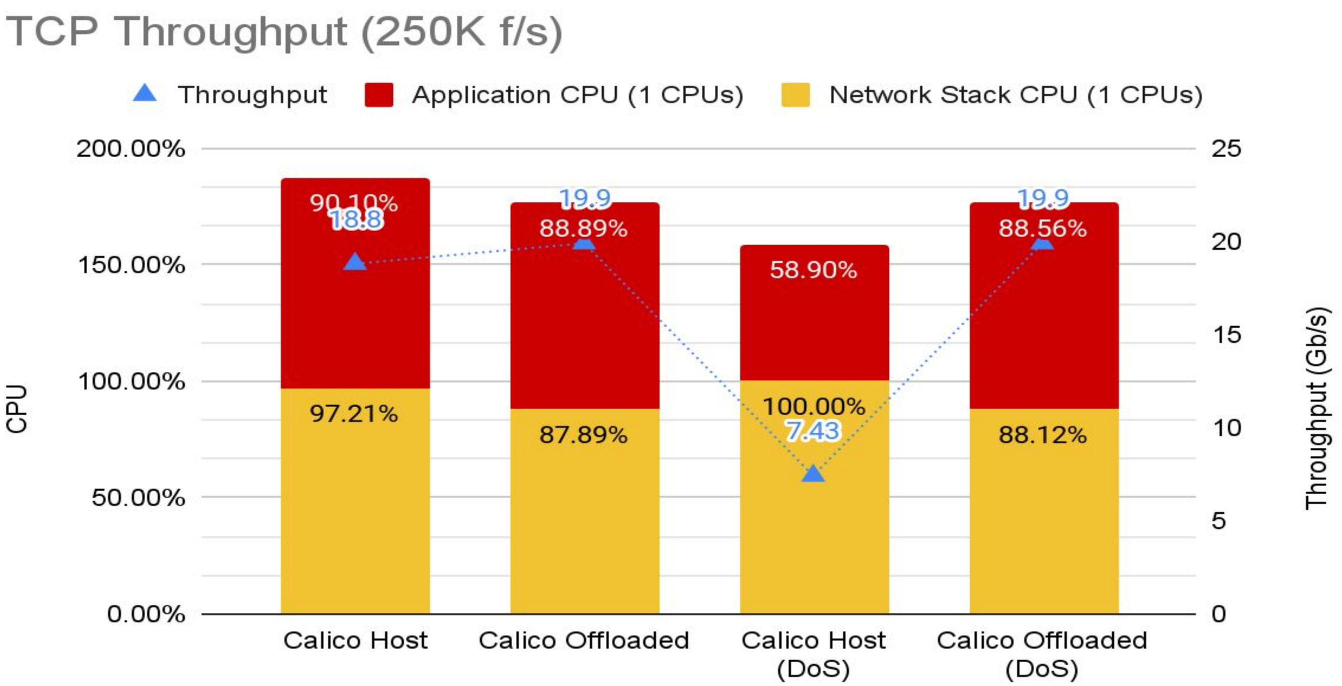
L3/L4 Policy Enforcement Offload - Setup

Basic Experiment Setup



L3/L4 Policy Enforcement Offload - Results

Throughput of a single flow in normal situation and with host undergoing attack by denied connections

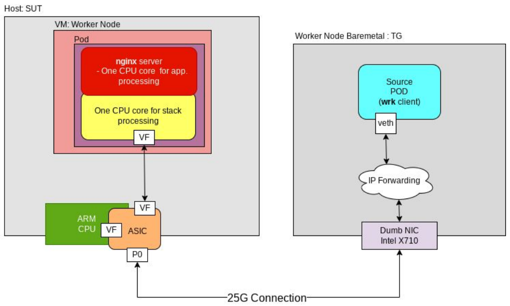


L3/L4 Policy Enforcement - Observations

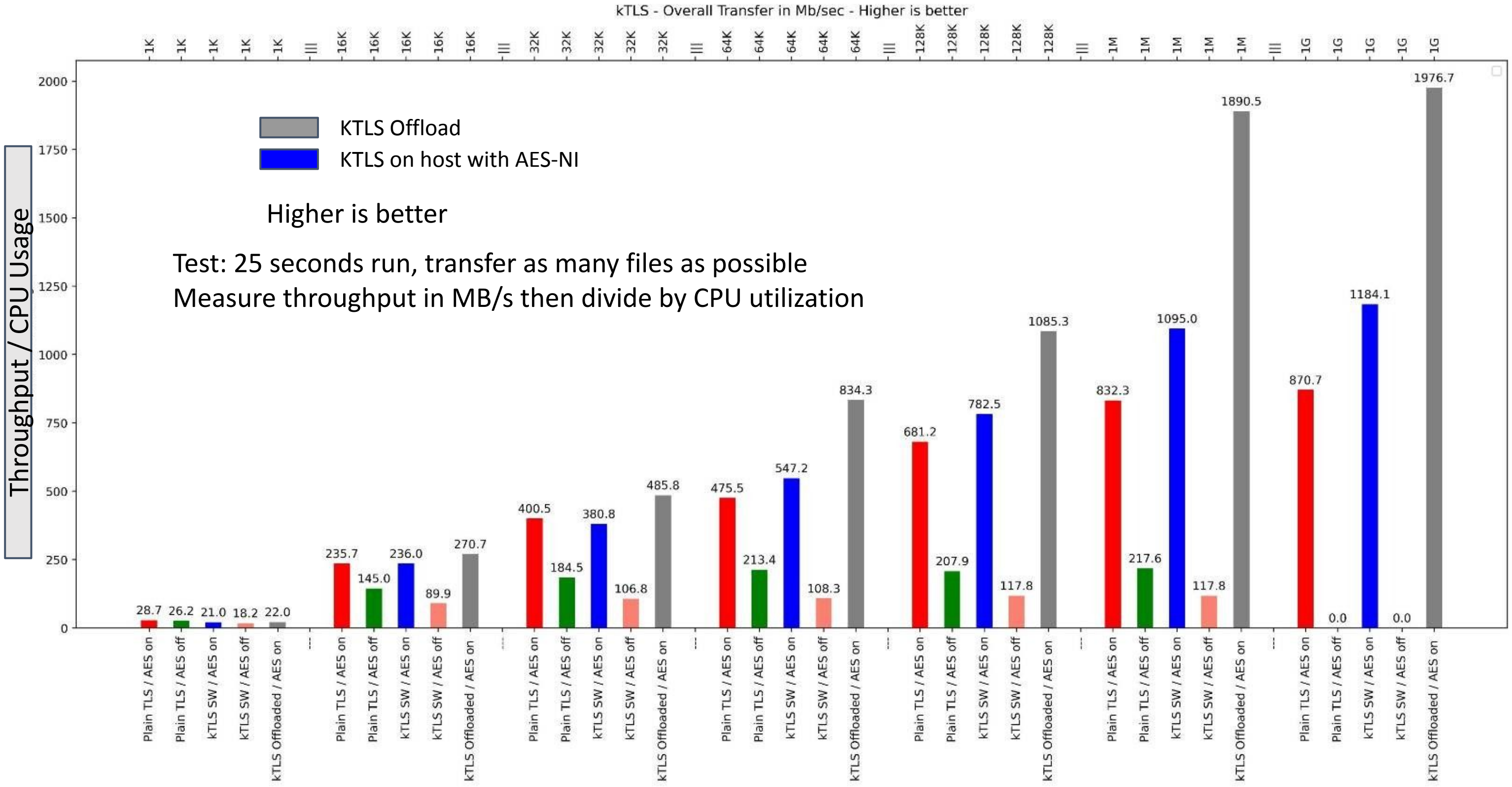
- Consistent offload results regardless of
  - Packets/second ingressing (attack or legitimate) traffic
  - Number of flows/sec
- Efficient utilization of host CPU
- Control-driven L3/L4 policy instantiation in the data path ahead of packet flows (e.g., Calico) as opposed to packet-driven approaches (e.g., OVS)
  - Better performance
  - Lower host CPU and DPU/IPU-CPU utilization

Encryption Performance - Setup

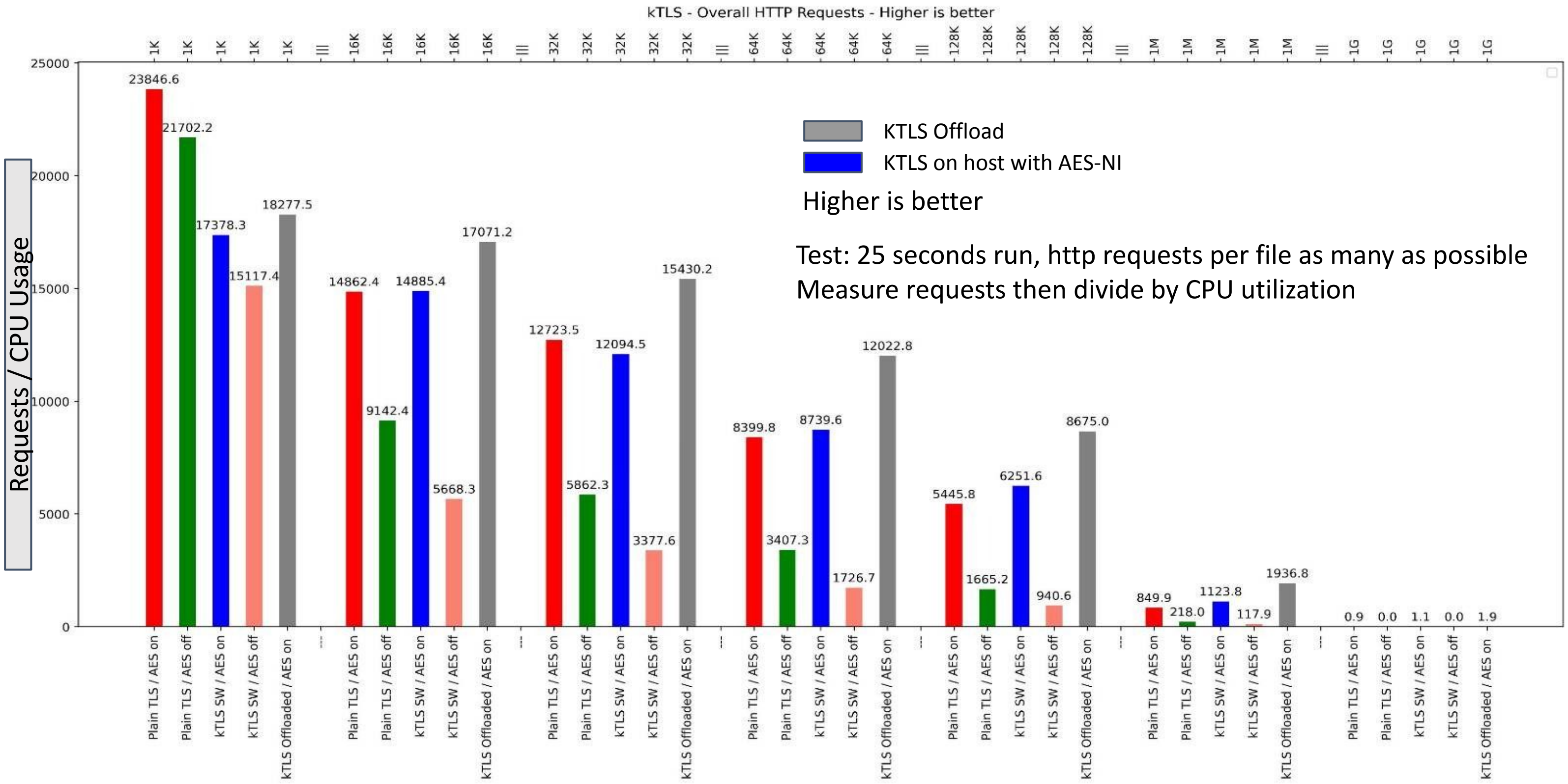
- System under test: VM with a single k8s POD running nginx server
- Client: POD with traffic generator
  - Open two https connections
  - Request files of different sizes (for each test) - 1K, 16K, 32K, 64K, 128K, 1M, 1G
  - 1000 requests complete for each test - Close/open again and again within a 25-second (25s) window
- 3 test runs, each 25s duration, to measure
  - Throughput (bps) relative to CPU utilization
  - Transaction rate (requests per second) relative to CPU utilization



Throughput Testing - Encryption Offload Effect



Transaction Testing - Encryption Offload Effect



Goals/Requirements

- Compute and networking control to be optimally designed for DPU/IPU
  - Security/trust
  - Performance
  - Offload of control plane functions and coordination with control functions on host CPU - consider control path
  - Policy programmability in hardware - Control driven rather than packet driven

- Support data plane and control plane for various compute endpoints
  - Bare metal
  - Virtual machines
  - Containers on bare metal hosts and on VMs as worker nodes

Standardization Efforts

- Consistent Lifecycle Management of DPU/IPUs across vendors
  - https://github.com/opiproject/opi-prov-life
- GRPC APIs for DPU/IPUs (OPI)
  - Telco Cloud https://github.com/opiproject/opi-api/tree/main/network/evpn-gw
- DPDK RTE_FLOW offload APIs
  - DPDK based CNFs: https://doc.dpdk.org/guides/prog_guide/rte_flow.html
  - Open vSwitch DPDK Dataplane
- P4TC (Providing P4 natively on Linux, offload when hardware is present)
  - https://github.com/p4tc-dev
  - More on P4 language (programmable fast datapath) - https://github.com/p4lang

Where Do We Go From Here?

Kubernetes Enhancement Proposals (KEP)

3698-multi-network

https://github.com/topics/k8s-sig-architecture

+

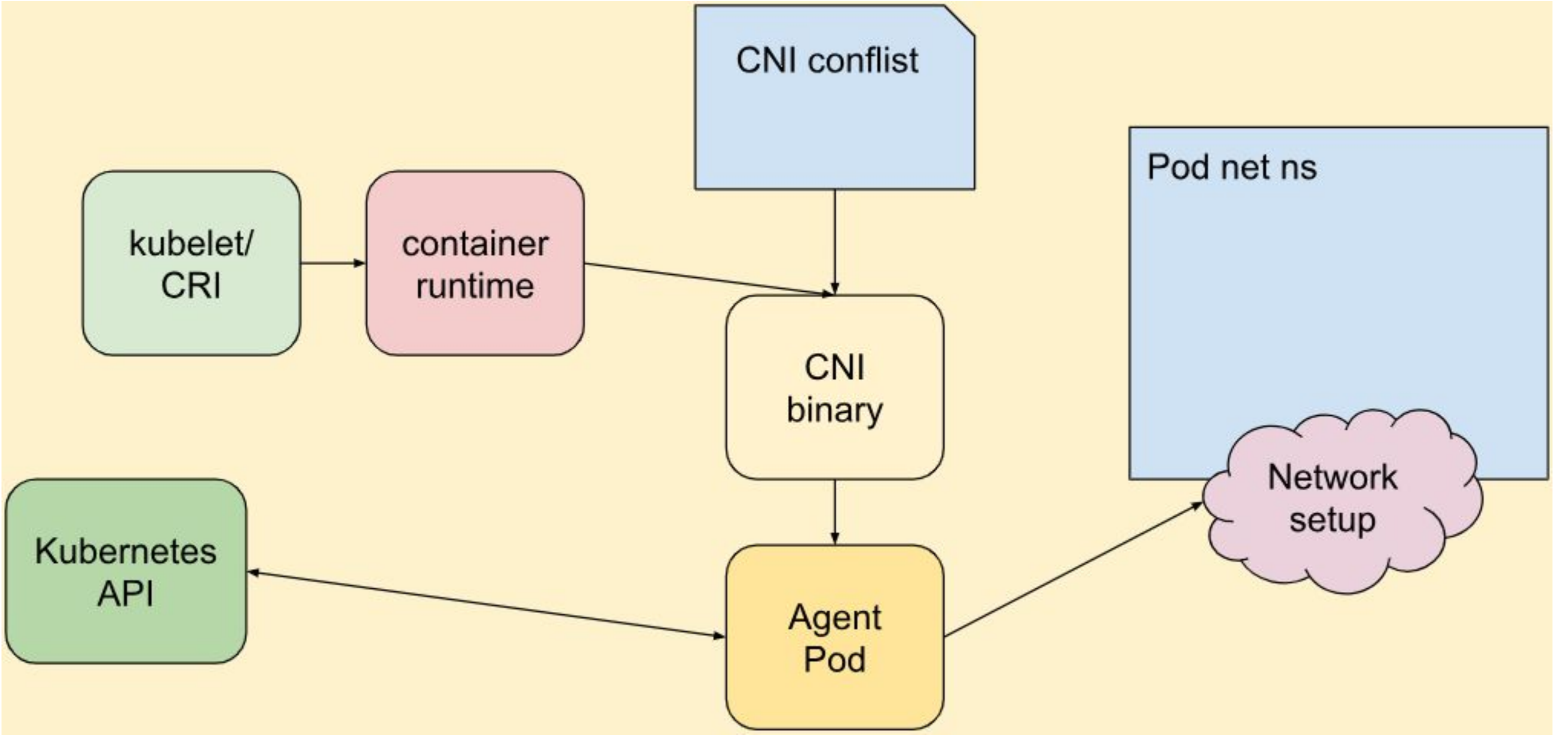
https://github.com/ipdk-io/k8s-infra-offload

ipdk

OPEN PROGRAMMABLE INFRASTRUCTURE PROJECT

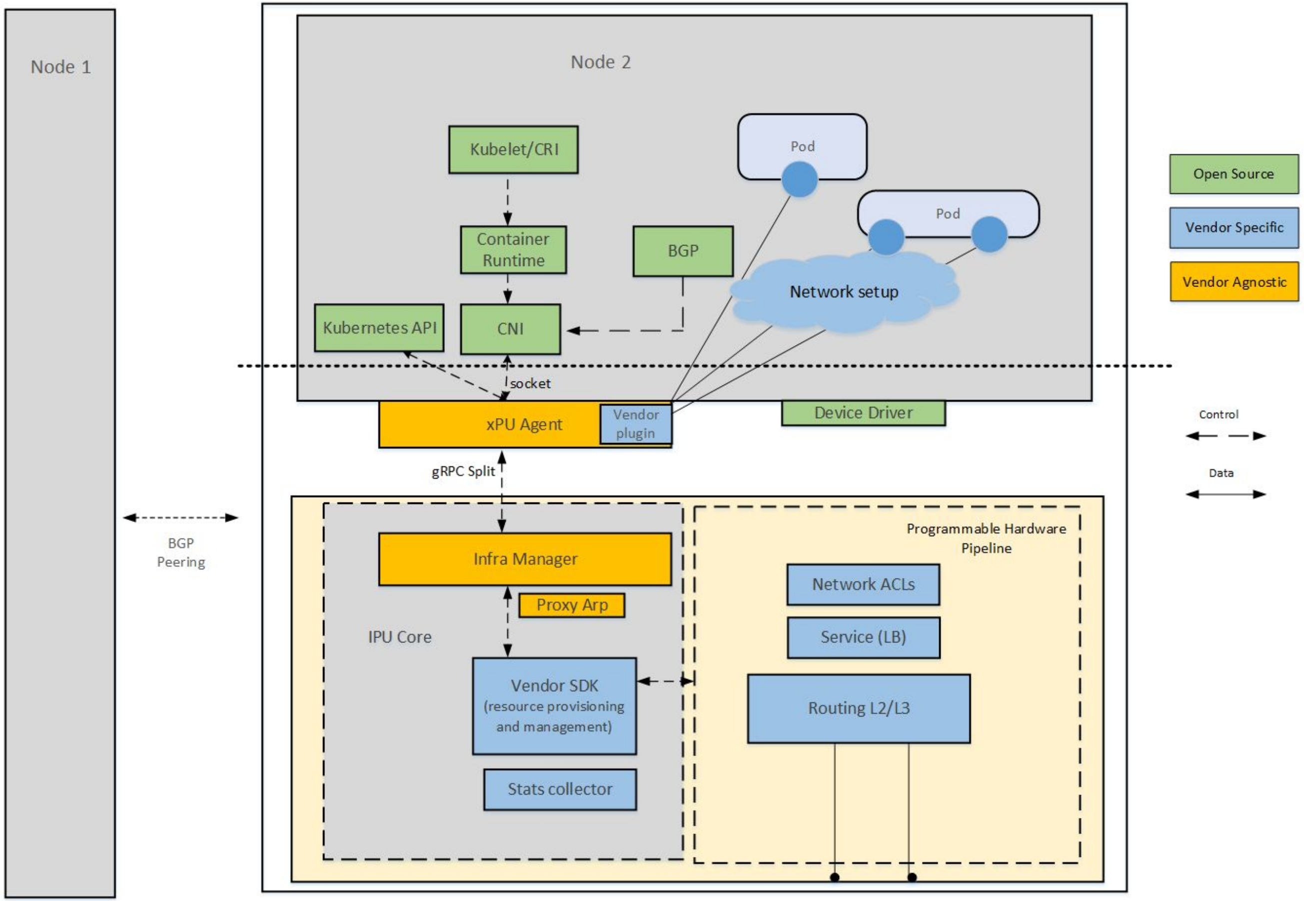
Working Example:
IPDK Kubernetes Networking Offload

KEP - Multi network



https://docs.google.com/document/d/17LhyXsEgjNQ0NWtvqvtgJwVqdJWreizsgAZHWflgP-A/edit#heading=h.mrwz1ucj09yg

IPDK.IO K8S Infra Offload (OPI Project)



Networking Abstraction Goals

Goals	Enhanced K8s	IPDK K8s Recipe	Notes
Multiple Networks	Multi-Net	SR-IOV	
Draining	Multi-Net	Open	
End-to-End Security	Partially Implemented	Partially Implemented	
High Performance	Green	Implemented	
Low Latency	Green	Implemented	
Crypto	-	Implemented	IPsec Recipe
Standard Agent Model	Multi-Net	Calico CNI	Dataplane Plug-in

Call For Contributions

Kubernetes Enhancement Proposals (KEP)

3698-multi-network

https://github.com/topics/k8s-sig-architecture

Multi-network pods

CNIs with agent model

+

Working Example: IPDK Kubernetes Networking Offload

https://github.com/ipdk-io/k8s-infra-offload

ipdk

OPEN PROGRAMMABLE INFRASTRUCTURE PROJECT

open source development

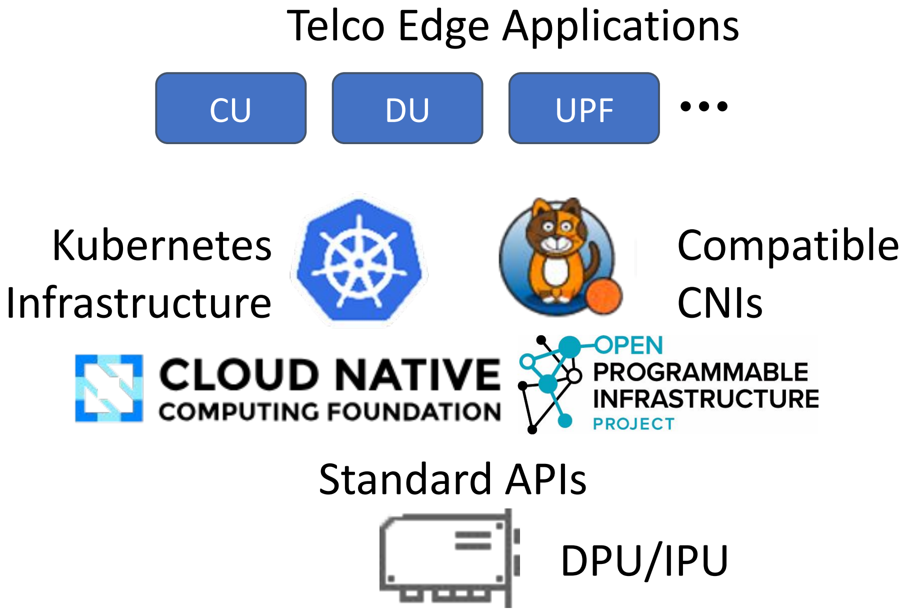
community standardization

Summary

Telco Applications can use Kubernetes CNI for High Performance Networking when Offloaded in the Infrastructure

Kubernetes Network Offload:

- Preserves User’s Abstractions
- Standards Based, Using Existing CNIs
- Multi-Vendor, Multi-Implementation



**Please scan the QR Code above
to leave feedback on this session**