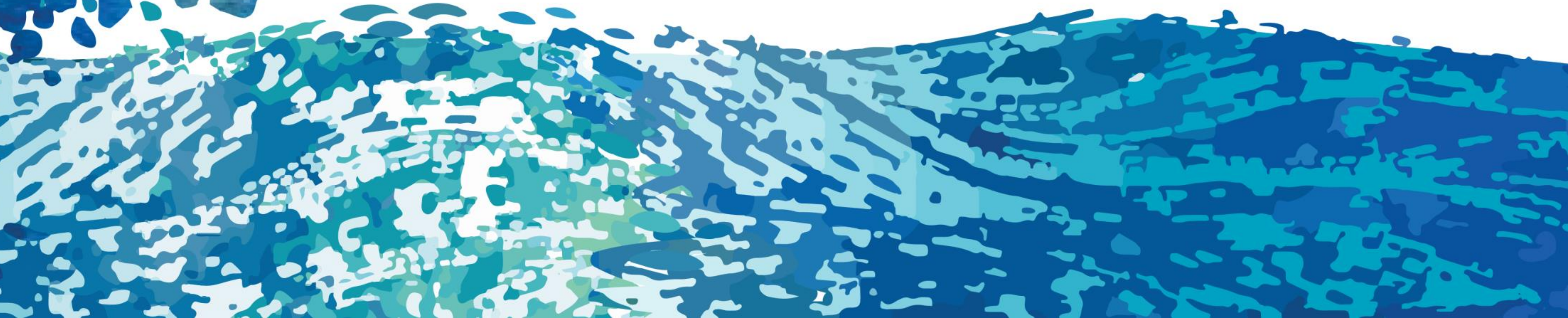
KubeCon | CloudNativeCon

Europe 2019



KubeCon | CloudNativeCon

Europe 2019

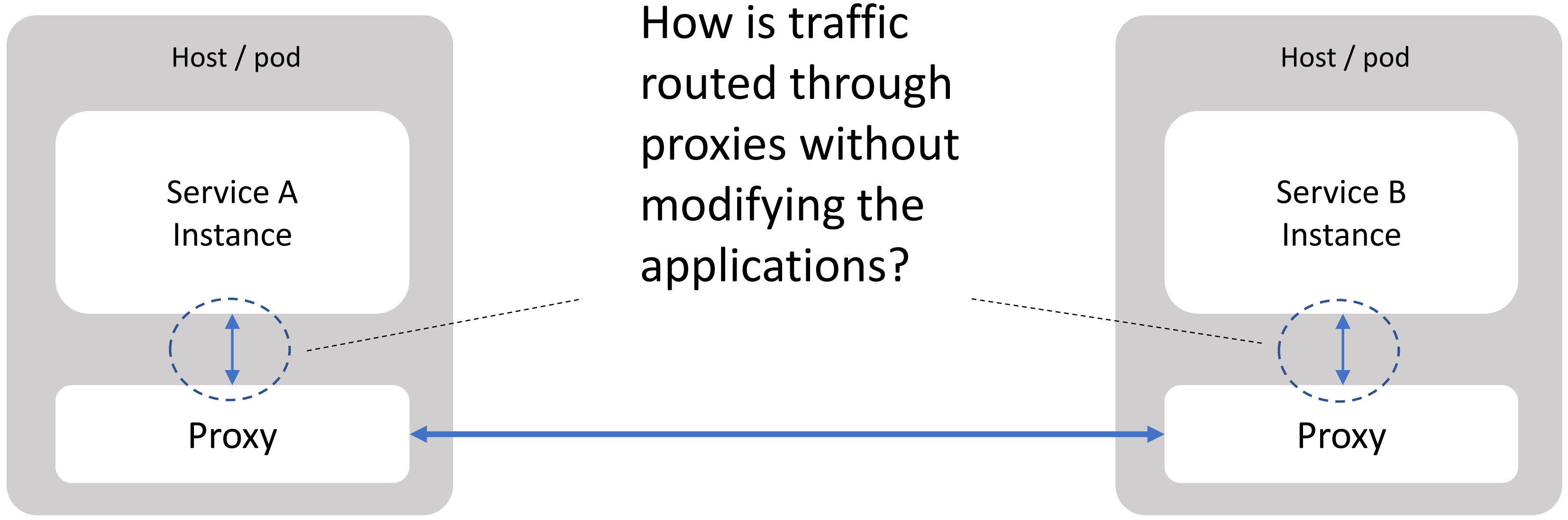
Networking the service mesh proxy

Where we are, where we’re going

Agenda

- Problem: Networking the Proxies
- Review of Kubernetes Container Network Interface (CNI)
- Istio-cni deep dive
- Istio-cni future work
- Istio-cni relationship to other projects and work
- NSM summary and comparison to Istio-cni
- Cilum’s eBPF summary and comparison to Istio-cni
- References and how to contribute

Problem: Networking the Proxies



How is traffic routed through proxies without modifying the applications?

Problem: Networking the Proxies

- Injecting the Service Mesh proxies in the network datapath of applications requires actions specific to the hosting environment.
- Networking the proxies ends up being coupled with proxy lifecycle management & orchestration.
- Various approaches exist with advantages/disadvantages
  - Ubiquity
  - Performance
  - Integration with application & proxy orchestration

Traffic Redirect Approaches

Traffic control options – the dataplane

- iptables/ip6tables REDIRECT
- eBPF—transparent proxy
- eBPF—socket redirect
- Host vswitch—e.g. fd.io/OVS

Control/Orchestration Options

- K8s pod NET_ADMIN init container
- CNI plugin
- NSM network service
- Node-agent

Proxy Lifecycle Approaches

Approach	Description	Pros	Cons
K8s Admission Control mutating webhook	Webhook modifies App's k8s pod specs to inject proxy as a sidecar container	• Simple k8s pod lifecycle management • Depends entirely on k8s API server	• Pod proxy not ready to network initContainers • Up/down-grade proxy tied to pod lifecycle • Potential sequencing problems with Admission control and pod security policy webhooks
CNI	CNI starts proxy in network namespace and manages proxy lifecyle based on netns lifecycle	• Pod proxy network is ready when any containers start • Up/down-grade totally under CNI control (independent of pod lifecycle	• Not leveraging k8s lifecycle management for proxy • More complicated proxy resource accounting with k8s scheduler
Network Service Mesh (NSM)	Proxy instantiated by network service create method when app pod created	• Separation of roles for network service management from application mesh. • Easy tie in with other network functions.	• Flexibility requires "solution" level integration for an application mesh. (cross project)
Node-agent	VM/baremetal use-case. Proxy lifecycle and config is controlled via a node-agent.	• Ease-of-use. Integrated with specific application service mesh type.	• Host OS dependencies. • Requires ability to install in host OS.

Review K8s CNI

Review of Kubernetes CNI

- More complete description available here: https://github.com/containernetworking/cni/blob/master/README.md
- Definition

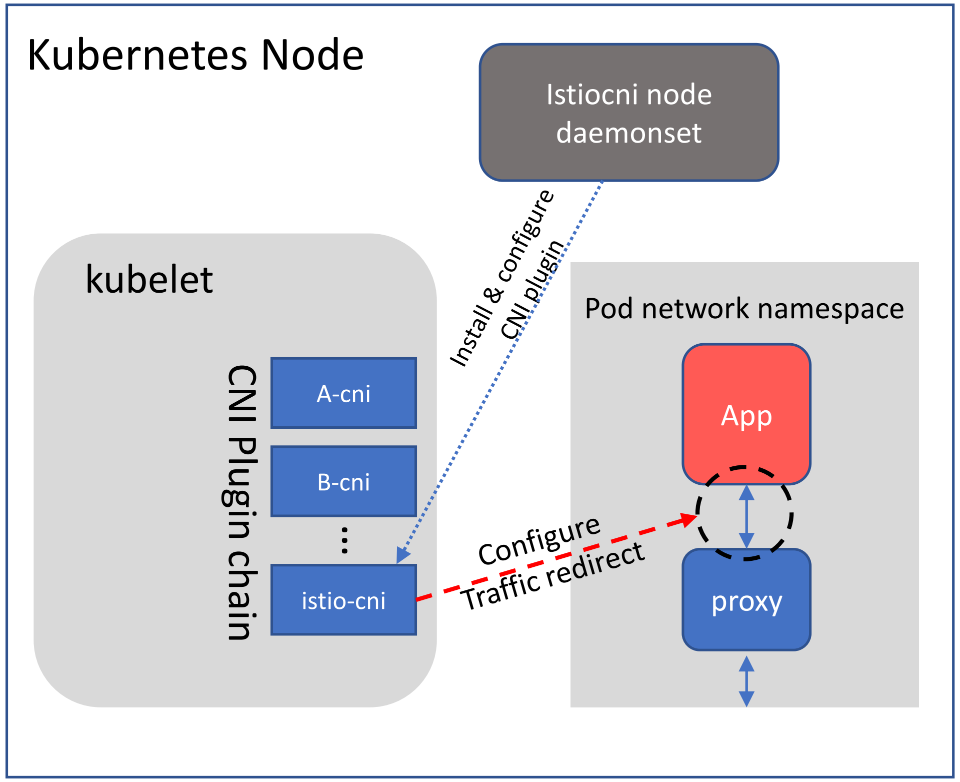
  > "CNI (Container Network Interface), a Cloud Native Computing Foundation project, consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins. CNI concerns itself only with network connectivity of containers and removing allocated resources when the container is deleted. Because of this focus, CNI has a wide range of support and the specification is simple to implement."
- The CNI is a specification not an implementation
- CNI plugins adhering to the specification are responsible for plumbing a container to the network for communications to other containers and endpoints within the cluster and possibly endpoints beyond the cluster.
- Many 3rd party CNI plugins exist: https://github.com/containernetworking/cni/blob/master/README.md#3rd-party-plugins
- The CNI plugins can be chained to allow multiple plugins to coexist and perform different aspects of plumbing the network connection.
- Different installation models are possible but a daemonset running on each node is most prevalent
- Istio-cni is a plugin written to address the unique requirements of plumbing containers to networks in environments utilizing a service Mesh

Istio-cni Deep Dive

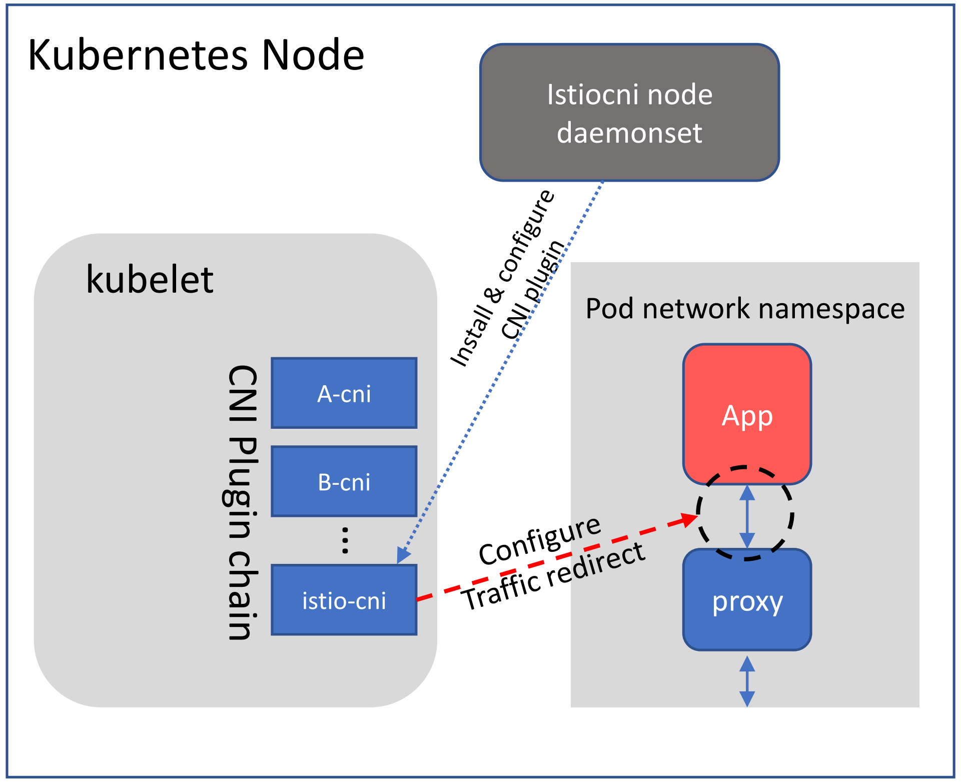
Istio-cni deep dive

- The Istio-cni project was spawned to address the problems we discussed a few minutes back.
- It is currently tightly aligned with the Istio project
  - Features and roadmap dictated by Istio needs and coordinated through Istio community
  - Reviewers and contributors from Istio community
  - Most testing is coupled with Istio components
  - In its own repo which can allow for separate evolution, release cadence and governance
- It takes advantage of the CNI chaining properties so that it runs after the other plugins have plumbed the pod to the network.
- Its job is to ensure that all appropriate traffic is first diverted to the Istio side car proxy instead of the application container traffic directly accesses the network.
- Under the hood it will setup iptable rules in the netns of the pod to ensure all required traffic is diverted through the proxy sidecar
- Installs via daemonset on each node

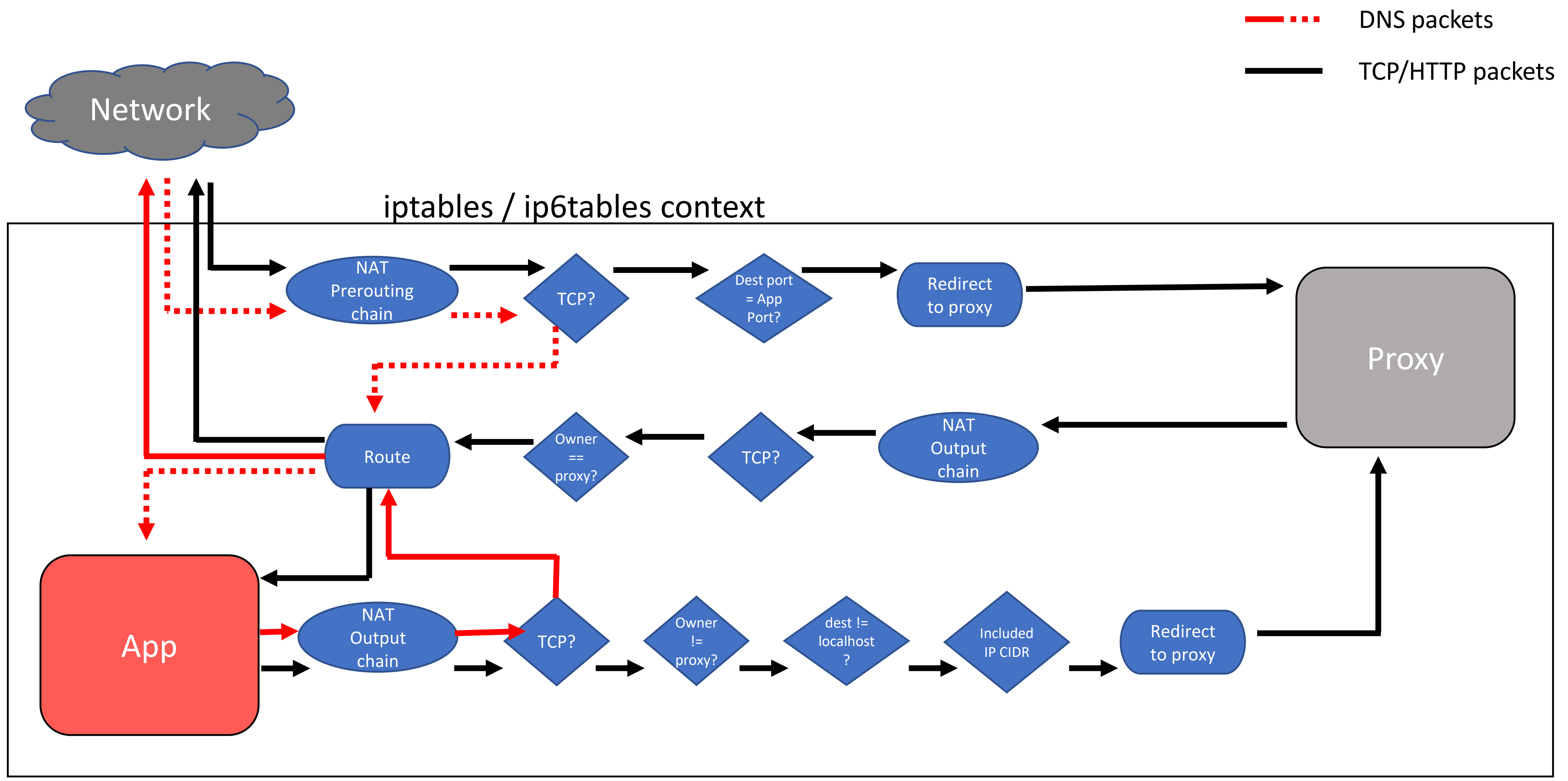
Istio-cni


Kubernetes Node
Istiocni node daemonset
Install & configure CNI plugin
kubelet
CNI Plugin chain
A-cni
B-cni
⋮
istio-cni
Configure Traffic redirect
Pod network namespace
App
proxy

...


Kubernetes Node
Istiocni node daemonset
Install & configure CNI plugin
kubelet
CNI Plugin chain
A-cni
B-cni
⋮
istio-cni
Configure Traffic redirect
Pod network namespace
App
proxy

Istio-cni life of a packet



Istio-cni Features

- Feature parity with istio-init container redirect
  - iptables redirect
    - ip6tables support in 1.2
  - TPROXY support
- Separately installed and administered from other Istio components
  - Installable via Helm and new istio/installer
- Tested on numerous public clouds
- Tested with a number of other CNI plugins: Calico, Weave, Flannel
  - Not a standalone CNI – must be used with other CNI plugins
- Compliant to K8s CNI specification
- Support for Istio parameters via application pod annotations
- Configurable bin and conf directories
- Configurable logging level

Istio-cni Future Work

Istio-cni future work

- Proxy injection via CNI
  - Implementation proposed by Marko Luksa from RH Openshift team
  - Pros:
    - Proxy totally within the control of cluster administration
      - Decoupled proxy lifecycle management from application pods
    - Proxy & networking established prior to any K8s pod execution—e.g. initContainers
    - Avoids sequencing issues with k8s Admission-controller/pod-security-policy webhooks
  - Cons:
    - K8s is not performing proxy lifecycle management
    - Resource accounting

NSM Summary

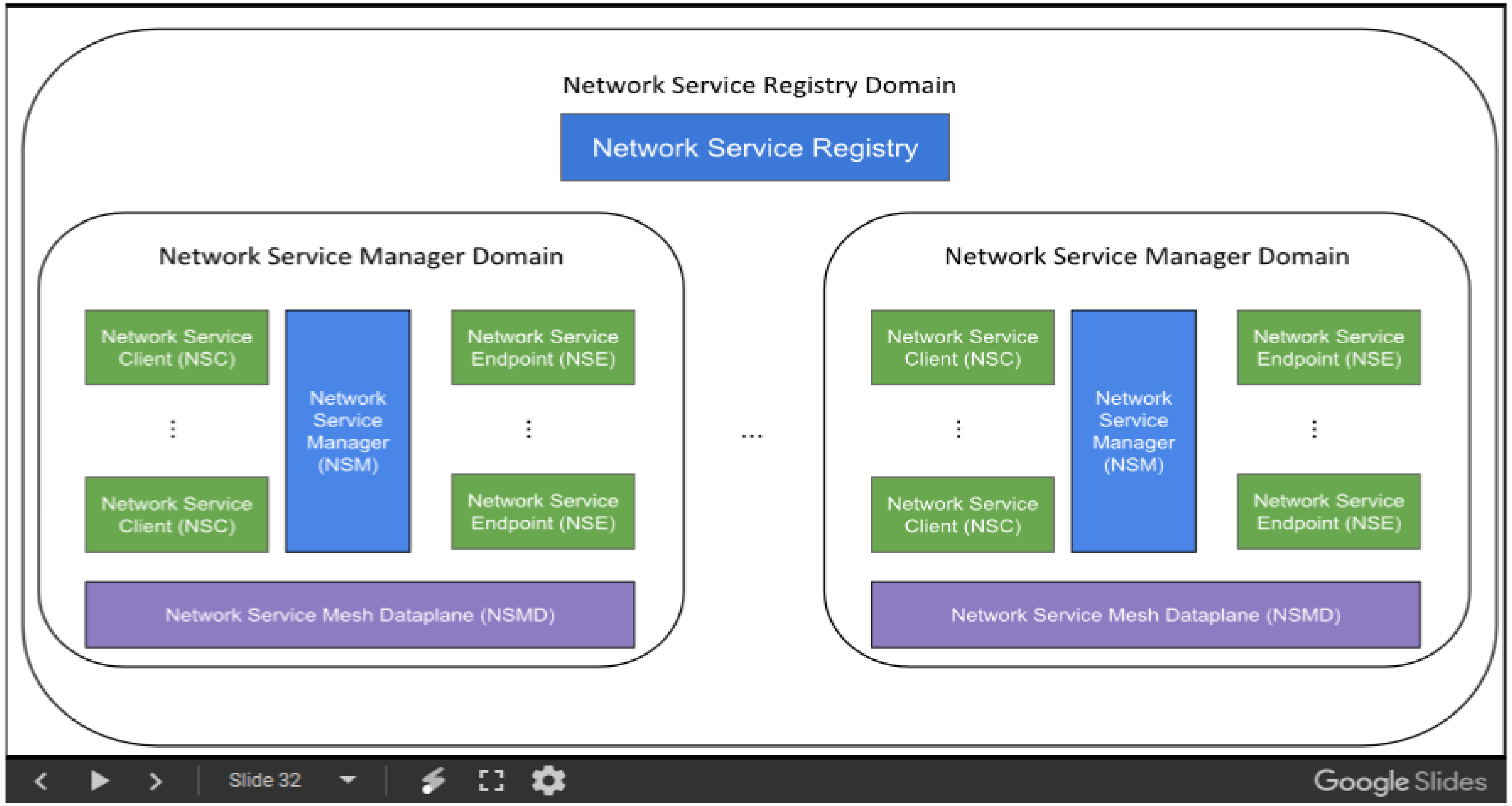
Istio-cni relationship to other projects

- Linkerd relationship
  - Linkerd and Istio share the service mesh architecture and thus Linkerd community shares the same set of problems with sidecar traffic redirection.
  - Linkerd community has begun to support an experimental CNI option to handle traffic redirection to the proxy: https://linkerd.io/2/features/cni/
  - The Linkerd CNI model is identical to Istio's and shares code
- Network Service Mesh (NSM)
  - A Kubernetes incubation project - https://networkservicemesh.io/
  - Doesn't rely on a sidecar proxy model like Istio or Linkerd
  - Would be possible to move sidecar management to NSM
- Cilium & eBPF
  - https://cilium.io/
  - https://prototype-kernel.readthedocs.io/en/latest/bpf/
  - An alternative to iptables to handle the redirection
  - Istio-cni could allow for easier adoption of eBPF

NSM summary and comparison to Istio-cni

- Network service mesh != Istio or Linkerd service mesh
- Concentrates more on the network level and how to connect network endpoints together
- The control plane manages connecting the network endpoints with any required network functions (e.g. firewalls, VPNs)
- The set of required network functions is based on interaction with K8s API server and user or admin configuration
- Co-exists peacefully with CNI based pod networking
- No tight binding to any particular dataplane implementation
  - Most community activity is vswitch oriented

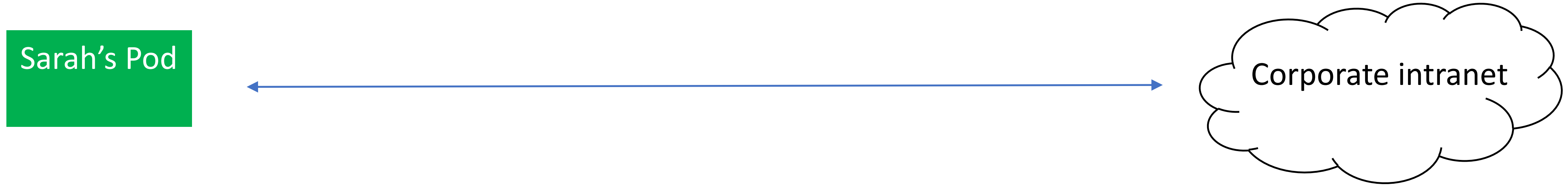
Network Service Mesh Architecture


Network Service Registry Domain
Network Service Registry
Network Service Manager Domain
Network Service Client (NSC)
Network Service Manager (NSM)
Network Service Endpoint (NSE)
Network Service Client (NSC)
Network Service Endpoint (NSE)
Network Service Mesh Dataplane (NSMD)
...
Network Service Manager Domain
Network Service Client (NSC)
Network Service Manager (NSM)
Network Service Endpoint (NSE)
Network Service Client (NSC)
Network Service Endpoint (NSE)
Network Service Mesh Dataplane (NSMD)
Slide 32
Google Slides

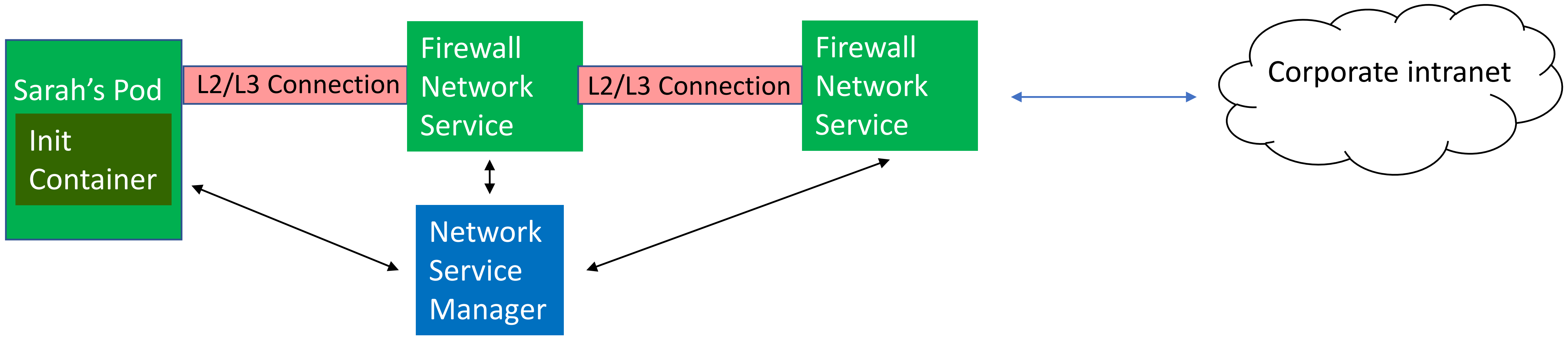
Borrowed from NSM deep dive documentation: https://networkservicemesh.io/docs/concepts/deepdive/

NSM service chaining example

Sarah simply wants a connection to the corporate Internet



NSM's Orchestrates the result to include required network services



NSM and Istio-cni integration

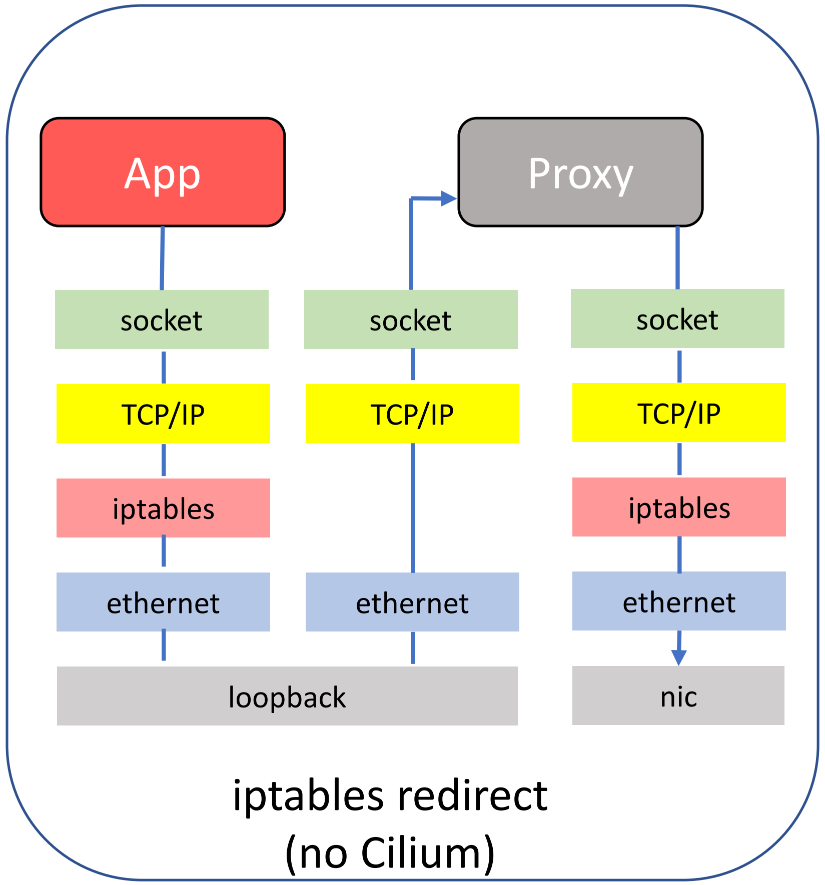
- At a high level Istio-cni and NSM are performing similar functions
  - Both manage how pod traffic should be diverted to meet business needs/requirements
  - Istio-CNI relatively static and simple most/all traffic to proxy side-car
  - Istio-CNI view constrained to individual pod
  - NSM more dynamic and based on policy and configuration
  - NSM view not constrained to individual pod
- NSM abstracts the proxy as just another network function
- NSM manages proxy network functions like any other network functions.

Cilium Summary

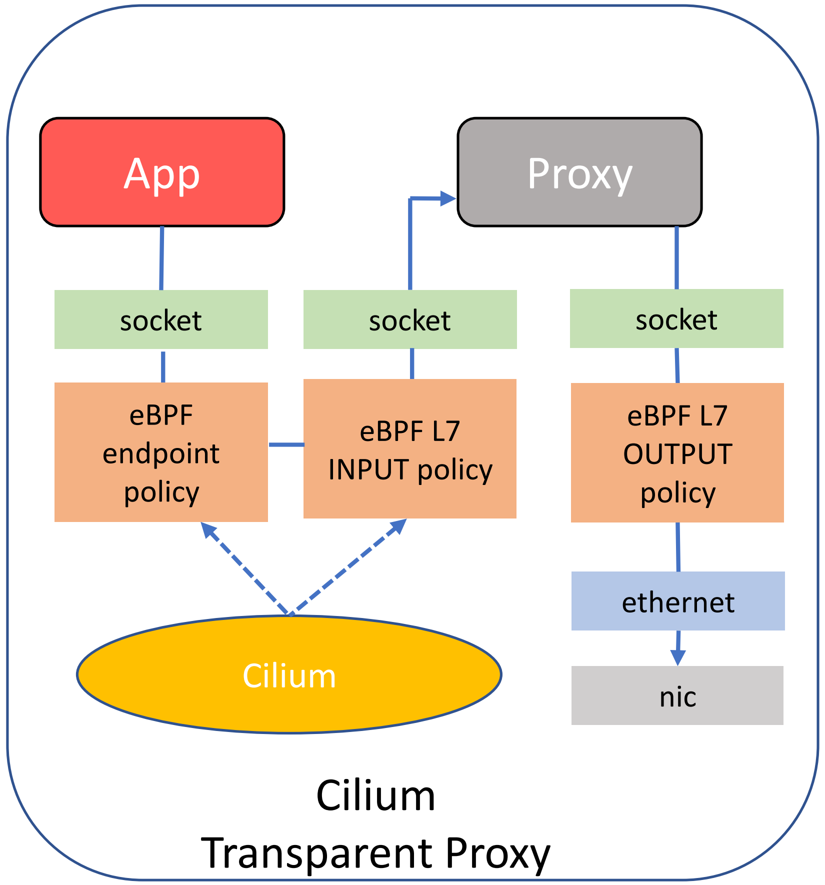
Cilium: Networking the Proxy

- General idea: Perf/scale! Utilize eBPF to integrate policy & identity more optimally while processing pkts (L3/4 & L7 policy)
- Depends on the host kernel version & settings
- Supports 2 modes of app<->proxy networking
  - Transparent proxy—Cilium Traffic Control connects proxy
  - Socket level redirect—Cilium eBPF directly connect app & proxy sockets
    - Requires kernel version 4.19+

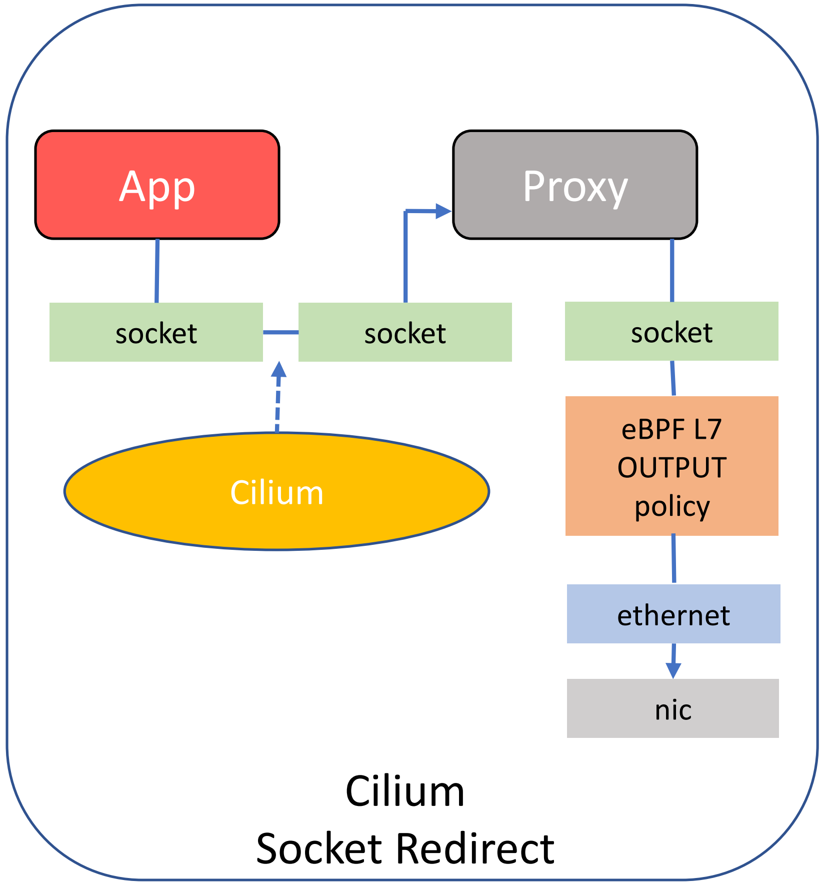
Cilium: Networking the Proxy



iptables redirect
(no Cilium)



Cilium
Transparent Proxy



Cilium
Socket Redirect

References and Contributing

References and how to contribute

- CNI project: https://github.com/containernetworking/cni/blob/master/README.md
- Network Service Mesh (NSM): https://networkservicemesh.io/
- Linkerd CNI plugin: https://linkerd.io/2/features/cni/ https://github.com/linkerd/linkerd2/tree/master/cni-plugin
- How to contribute: https://github.com/istio/cni/blob/master/CONTRIBUTING.md#contribution-guidelines
- Cilium datapath with proxy: https://docs.cilium.io/en/v1.5/architecture/