NODUS

Goal: To understand K8s Networking and Nodus advancement

Overview

Contacts: kuralamudhan.ramakrishnan@intel.com

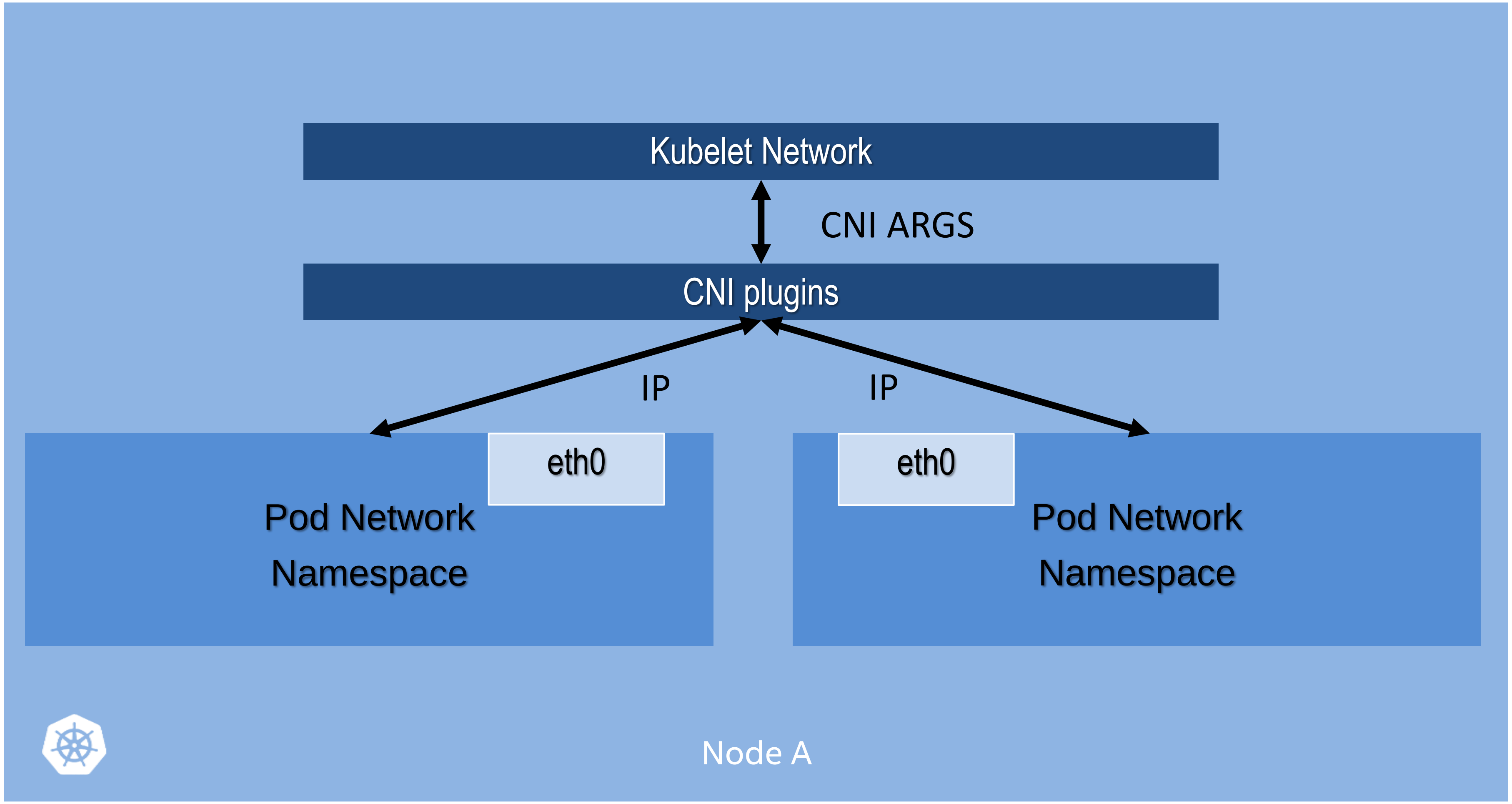
Agenda

- What is Kubernetes Network?
- Nodus
- Nodus - Networking CR
- Service Function chaining model

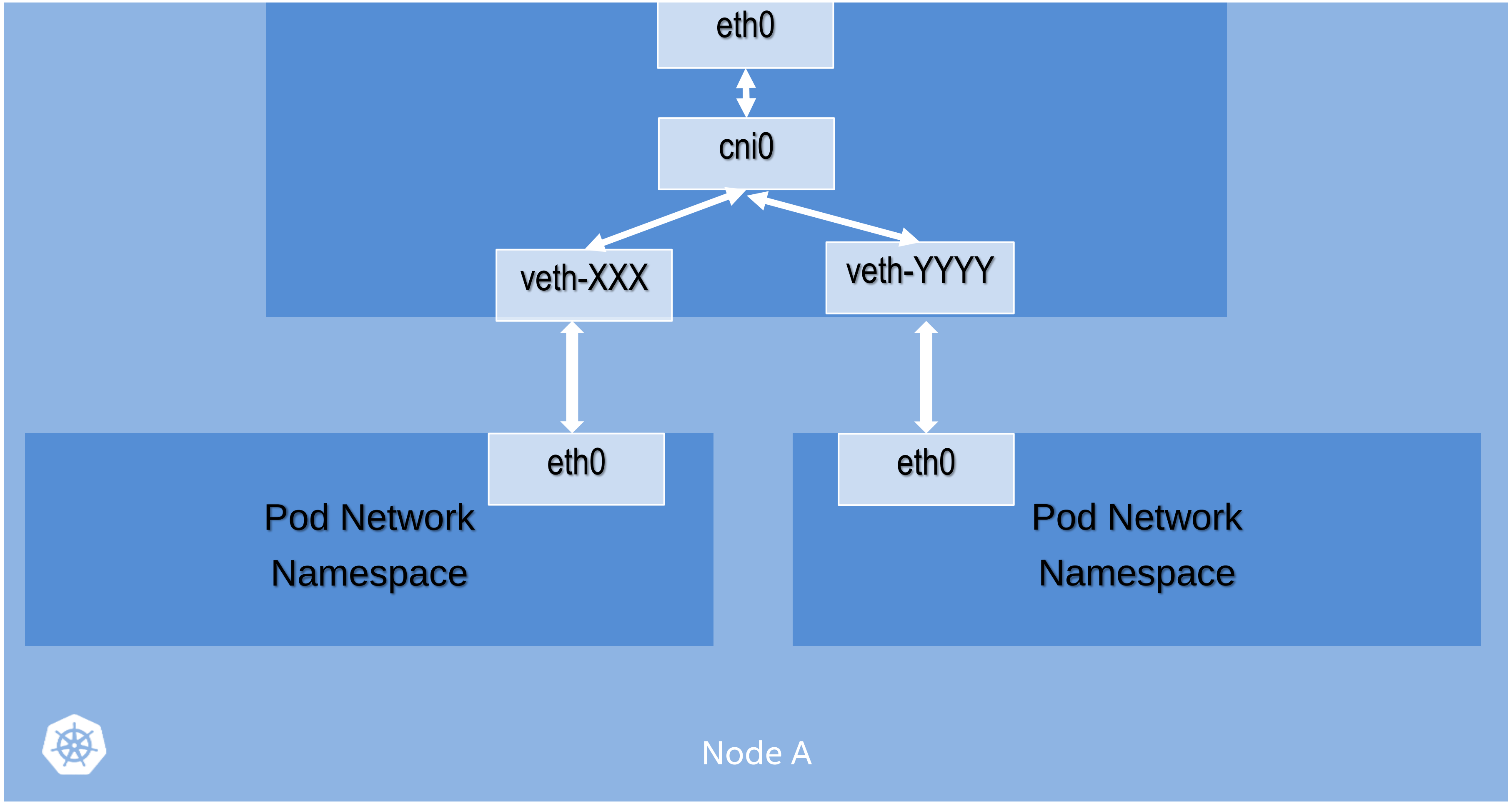
What is CNI ? How is it related to Kubernetes ?

- CNI stands for Container Networking interfaces
- The community is started by CoreOS with Bridge plugins and ipam plugins
- The community works on the CNI spec. The standard way to communicate to a container Network namespaces
- The CNI spec has basic commands such as ADD, DEL and CHECK. Output of CNI spec should has Interfaces, ips, routes and dns
- The plugins in CNI community follows CNI spec create and delete an interface. Currently CNI community host 18 plugins starting from dhcp, bridge, ipvlan and macvlan etc and 26th 3rd Party CNI plugins
- Kubernetes Network doesn’t handle network at the first place, it call CNI plugin to handle the network.

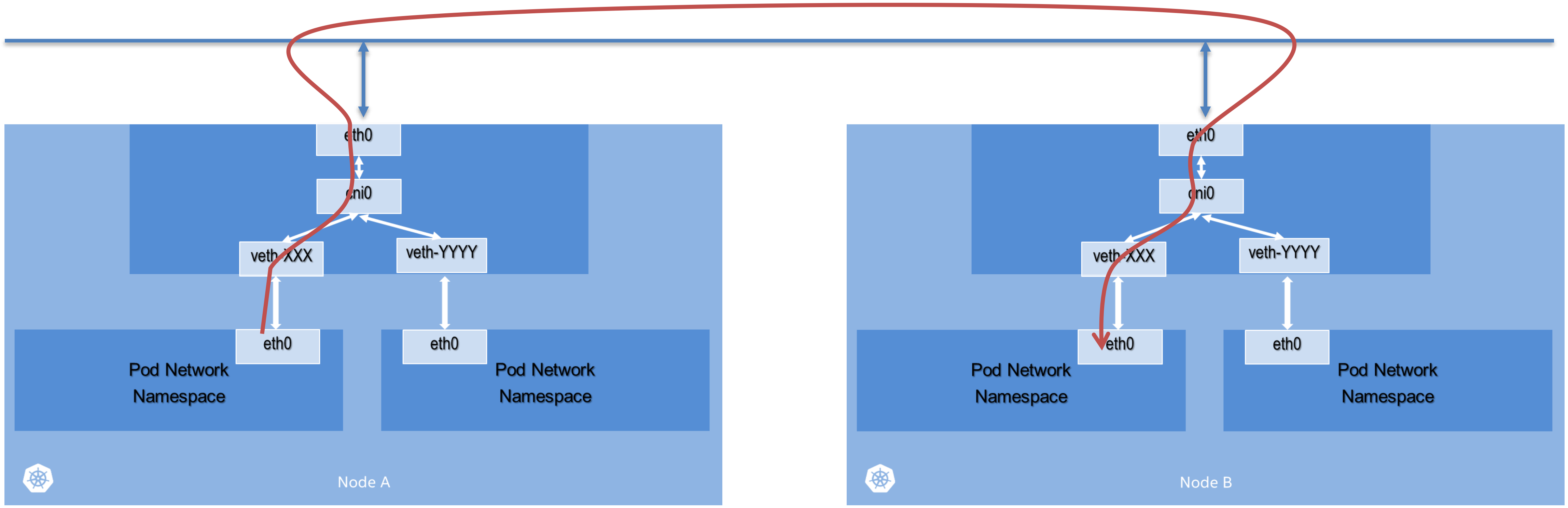
Pod Network in Kubernetes



Pod Network in Kubernetes – example flannel



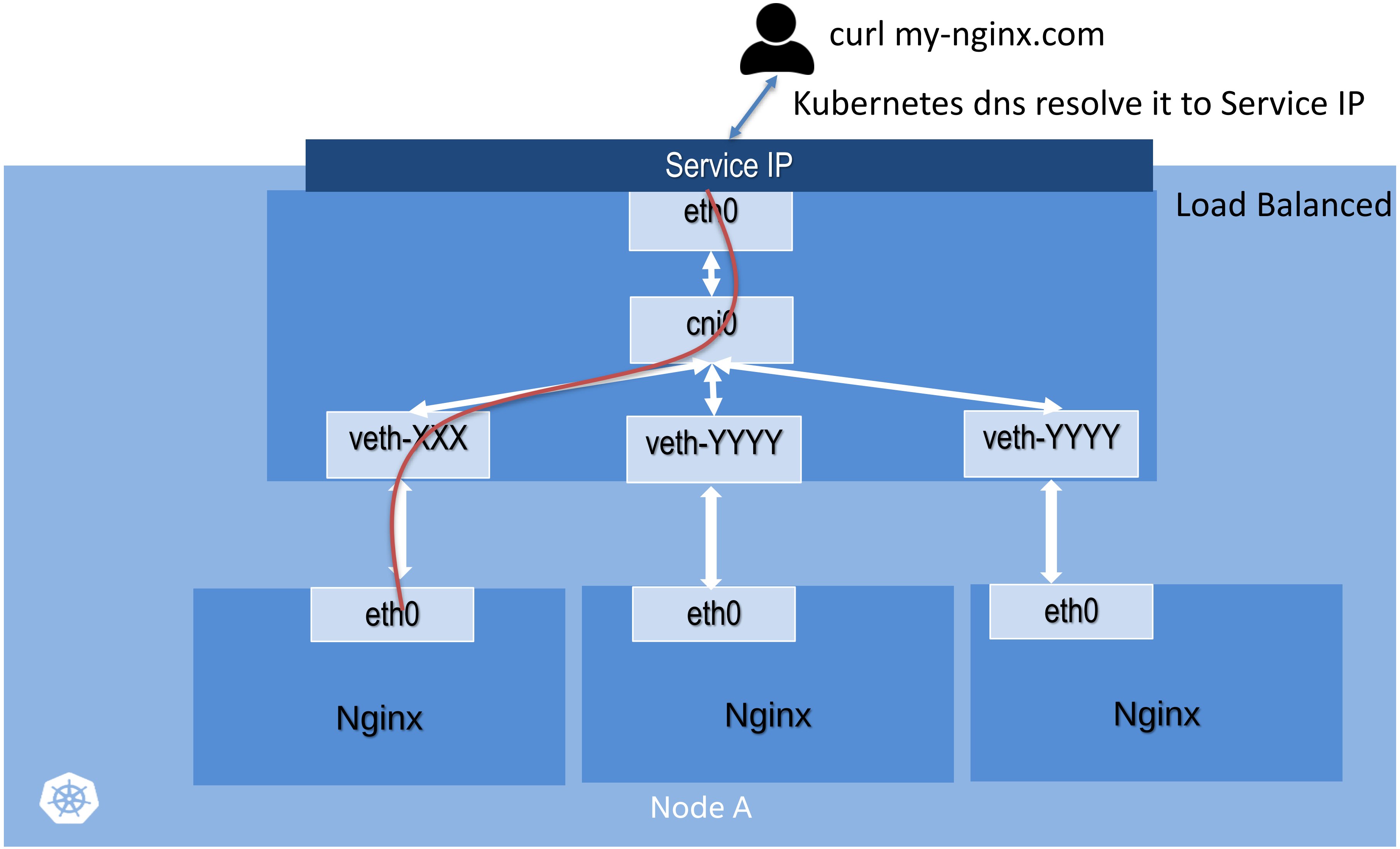
Pod Network in Kubernetes



Pod Network in Kubernetes - services

- A group of pods that work together grouped by a label selector
- Publishes how to access the service
  - DNS name
  - DNS SRV records for ports (well known ports work, too)
  - Kubernetes Endpoints API
- Defines access policy
- Load-balanced: name maps to stable virtual IP
- “Headless”: name maps to set of pod IPs
- Hides complexity - ideal for non-native apps
- Decoupled from Pods and Replication Controllers

Pod Network in Kubernetes - services



curl my-nginx.com

Kubernetes dns resolve it to Service IP

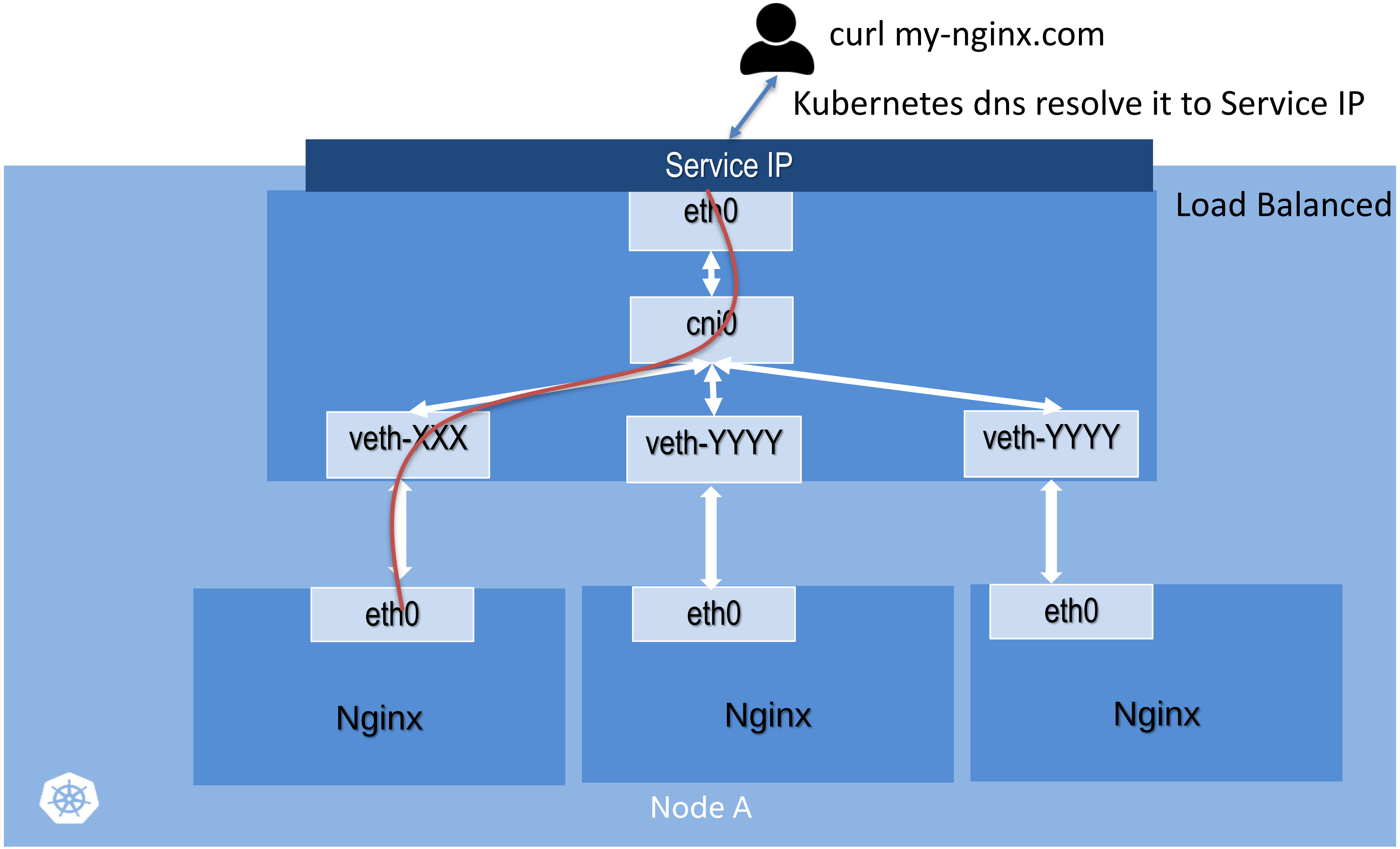
Load Balanced using Iptables/IPVS

Pod Network in Kubernetes

curl my-nginx.com

Kubernetes dns resolve it to Service IP

Load Balanced using Iptables/IPVS



Service IP

eth0

cni0

veth-XXX

veth-YYYY

veth-YYYY

eth0

eth0

eth0

Nginx

Nginx

Nginx

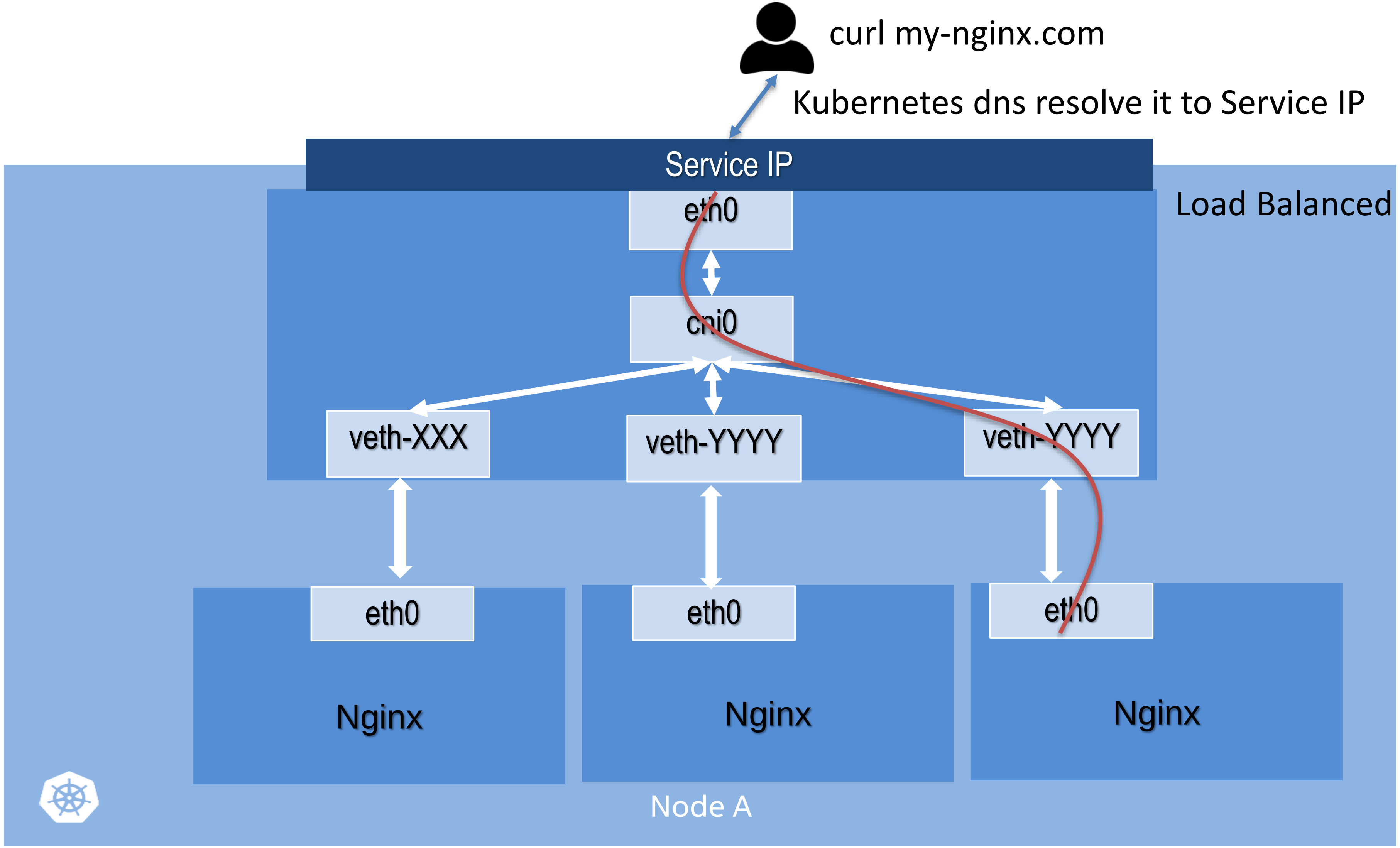
Node A

Pod Network in Kubernetes

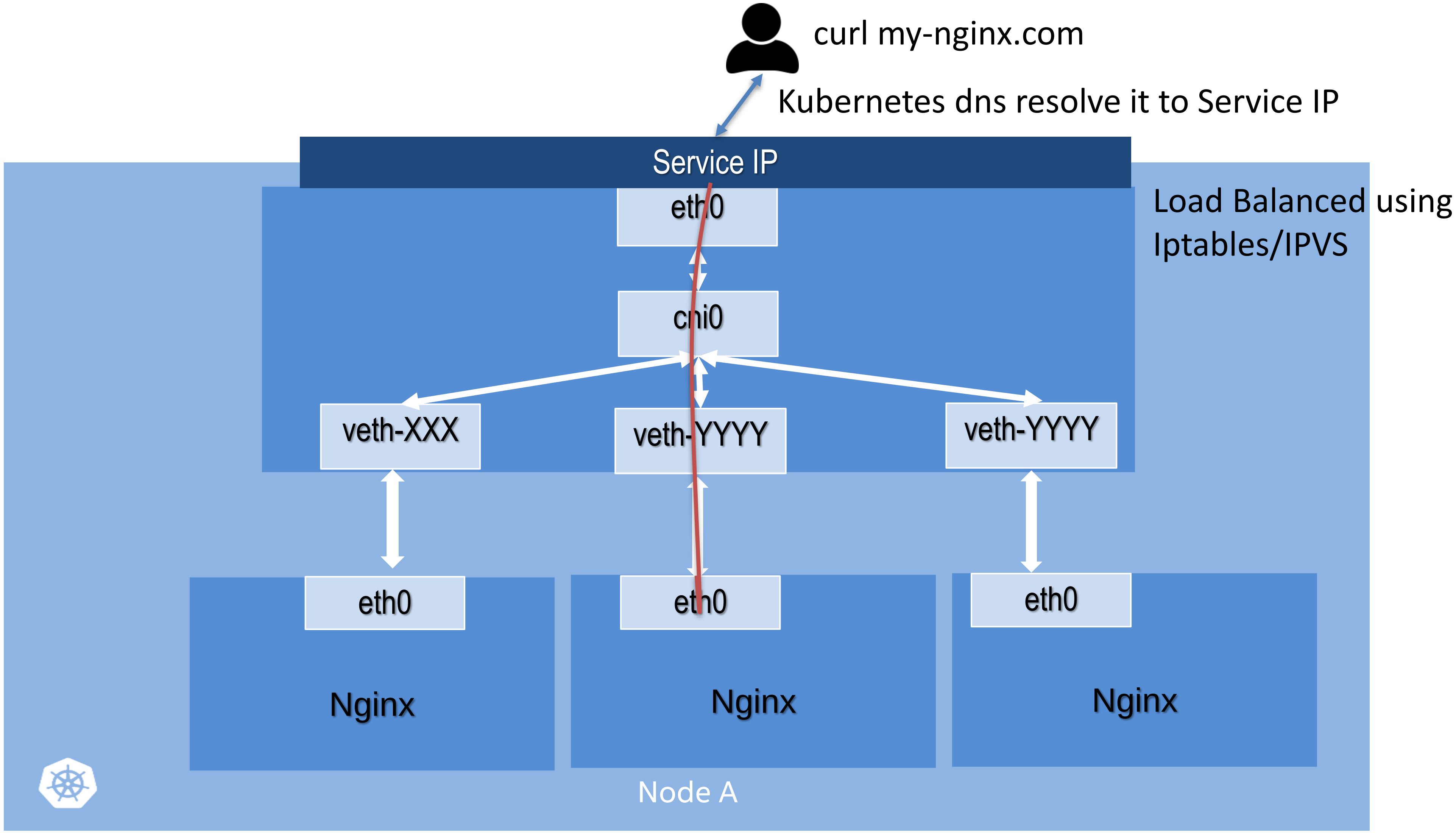
curl my-nginx.com

Kubernetes dns resolve it to Service IP

Load Balanced using Iptables/IPVS



Pod Network in Kubernetes



curl my-nginx.com

Kubernetes dns resolve it to Service IP

Load Balanced using Iptables/IPVS

What is Kubernetes Networking ?

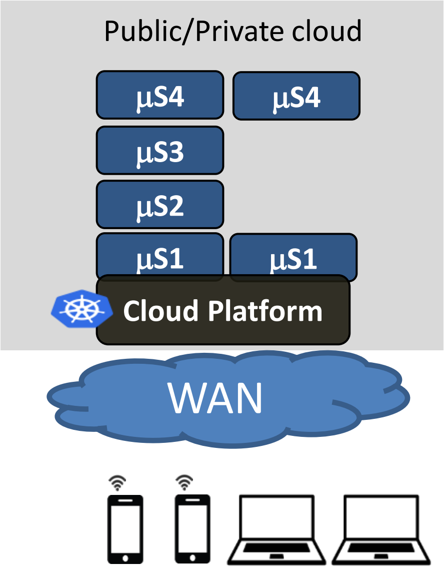
- Kubernetes Network doesn’t handle network at the first place, it call CNI plugin to handle the network.
- CNI stands for Container Networking interfaces. This community works on the CNI spec. The standard way to communicate to a container Network namespaces
- Networking in Kubernetes remains as the out-of-scope components
- Kubernetes Networking doesn’t address Multiple network Interfaces, Multiple Network managements and identities network as entity, we must depend on the CNI plugins to do heavy lifting here.
- Things not yet discussed are more in Kubernetes – Edge Networking, Network as an entity to be configured by Kubernetes, Virtual and Provider Networks, SDN, Service Function chaining

What is Nodus ?

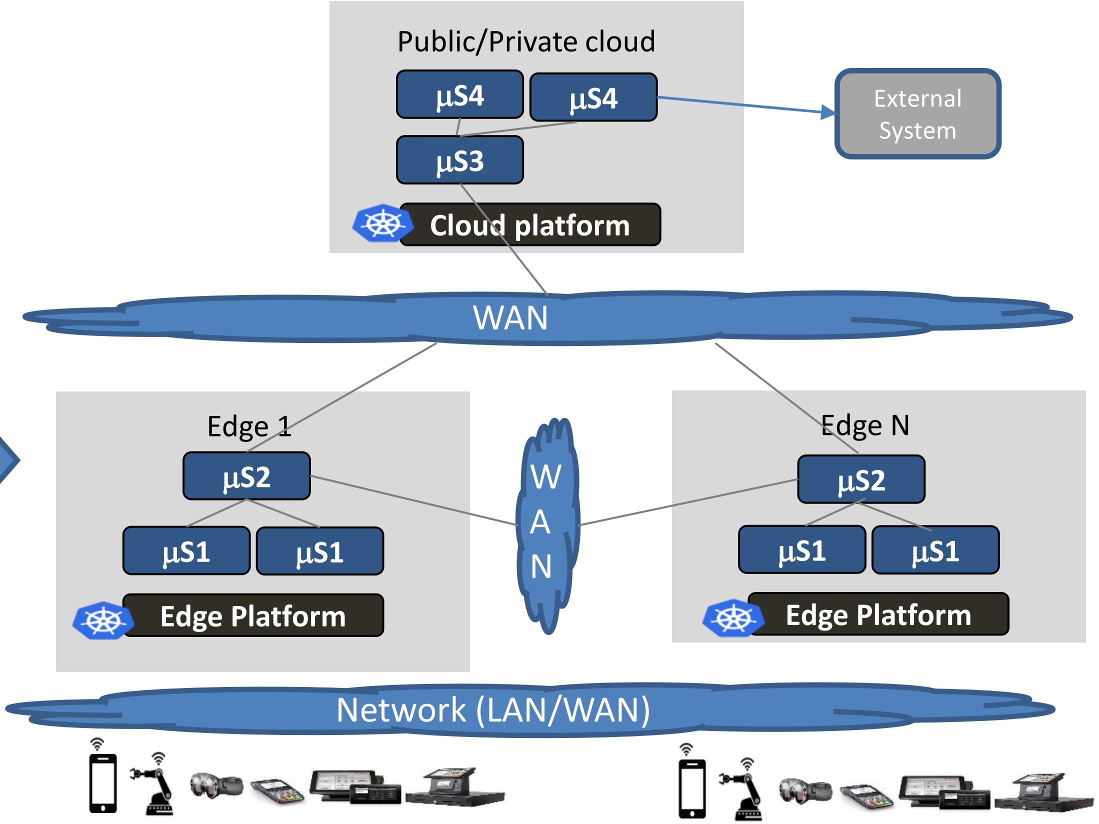
- Nodus is Latin word for “a Knot”.
- As the name suggested, the Nodus is the network controller in Kubernetes
- It act like a knot that perfectly converge the NFV networking concept and uses the Kubernetes labels to implement the Service Function Chaining.
- Nodus is answer for Software Defined Networking in Kubernetes, it take care of Edge networking solutions, support containers and VMs Service Function chaining

Application and Network Transformation in Edges

(AR/VR apps, Gaming, Analytics and Even traditional applications due to sovereignty and context)



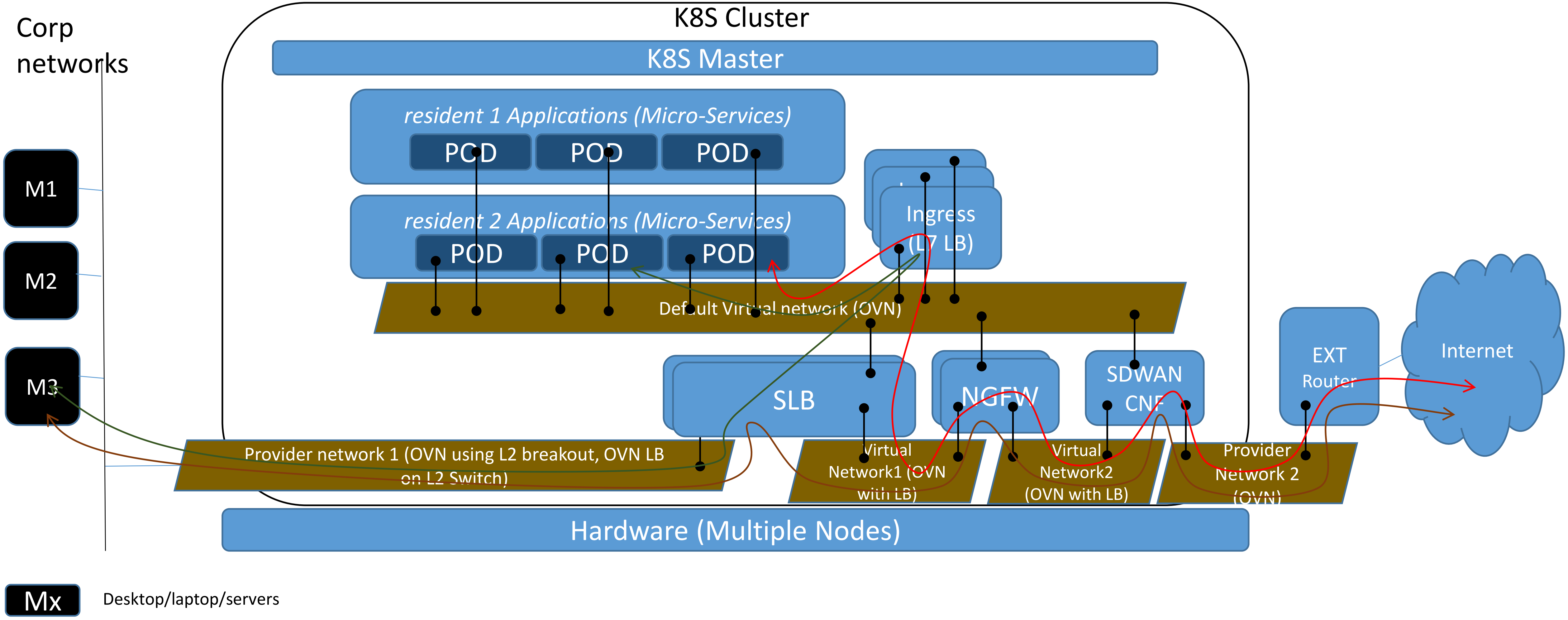
- Proximity
- Data sovereignty
- Economics
- Context



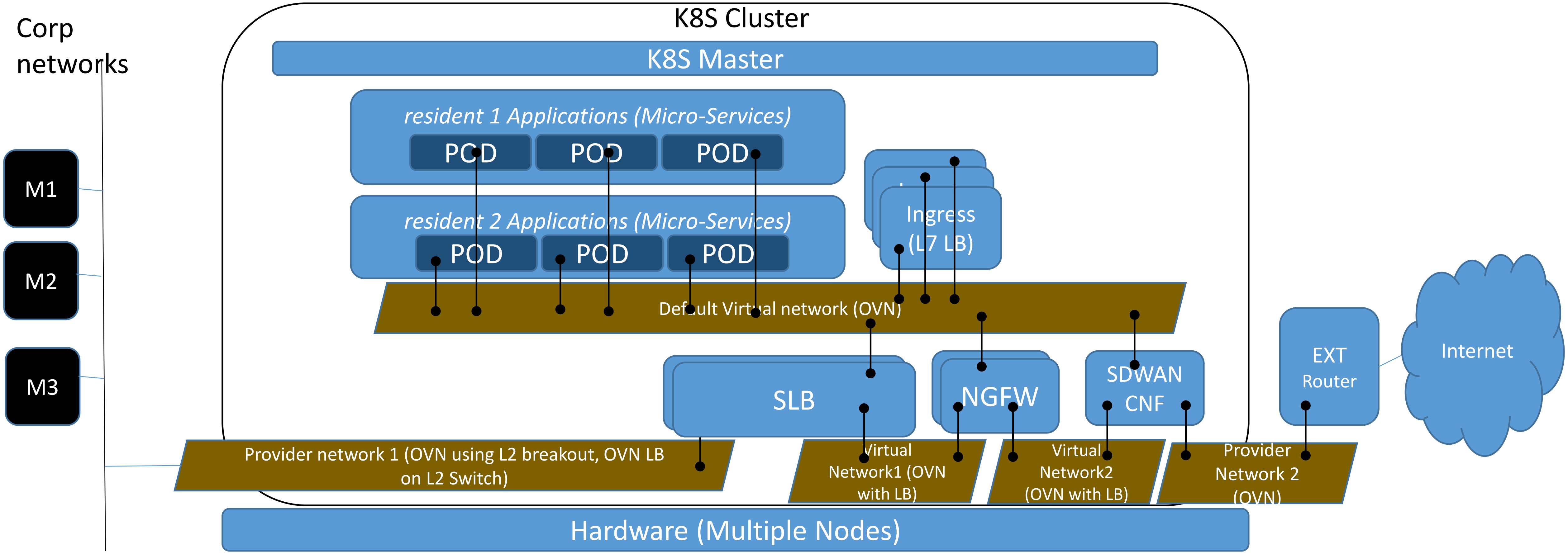
Centralized computing to Geo distributed computing

An App consisting of four Micro-services
ms1 talks to ms2, ms2 to ms3 and ms3 to ms4
ms1" is user facing service
"ms1", "ms2" are expected to be there together
"ms2" is stateful and hence need to talk to each other

How does NFV based deployment with Cloud-native applications look like (Taking SDWAN with security NFs as an example)


Corp networks
K8S Cluster
K8S Master
resident 1 Applications (Micro-Services)
POD
POD
POD
resident 2 Applications (Micro-Services)
POD
POD
POD
Ingress (L7 LB)
M1
M2
M3
Default Virtual network (OVN)
SLB
NGFW
SDWAN CNF
EXT Router
Internet
Provider network 1 (OVN using L2 breakout, OVN LB on L2 Switch)
Virtual Network1 (OVN with LB)
Virtual Network2 (OVN with LB)
Provider Network 2 (OVN)
Hardware (Multiple Nodes)
Mx
Desktop/laptop/servers

What is Nodus ?


Corp networks
M1
M2
M3
K8S Cluster
K8S Master
resident 1 Applications (Micro-Services)
POD
POD
POD
resident 2 Applications (Micro-Services)
POD
POD
POD
Ingress (L7 LB)
Default Virtual network (OVN)
SLB
NGFW
SDWAN CNF
EXT Router
Internet
Provider network 1 (OVN using L2 breakout, OVN LB on L2 Switch)
Virtual Network1 (OVN with LB)
Virtual Network2 (OVN with LB)
Provider Network 2 (OVN)
Hardware (Multiple Nodes)

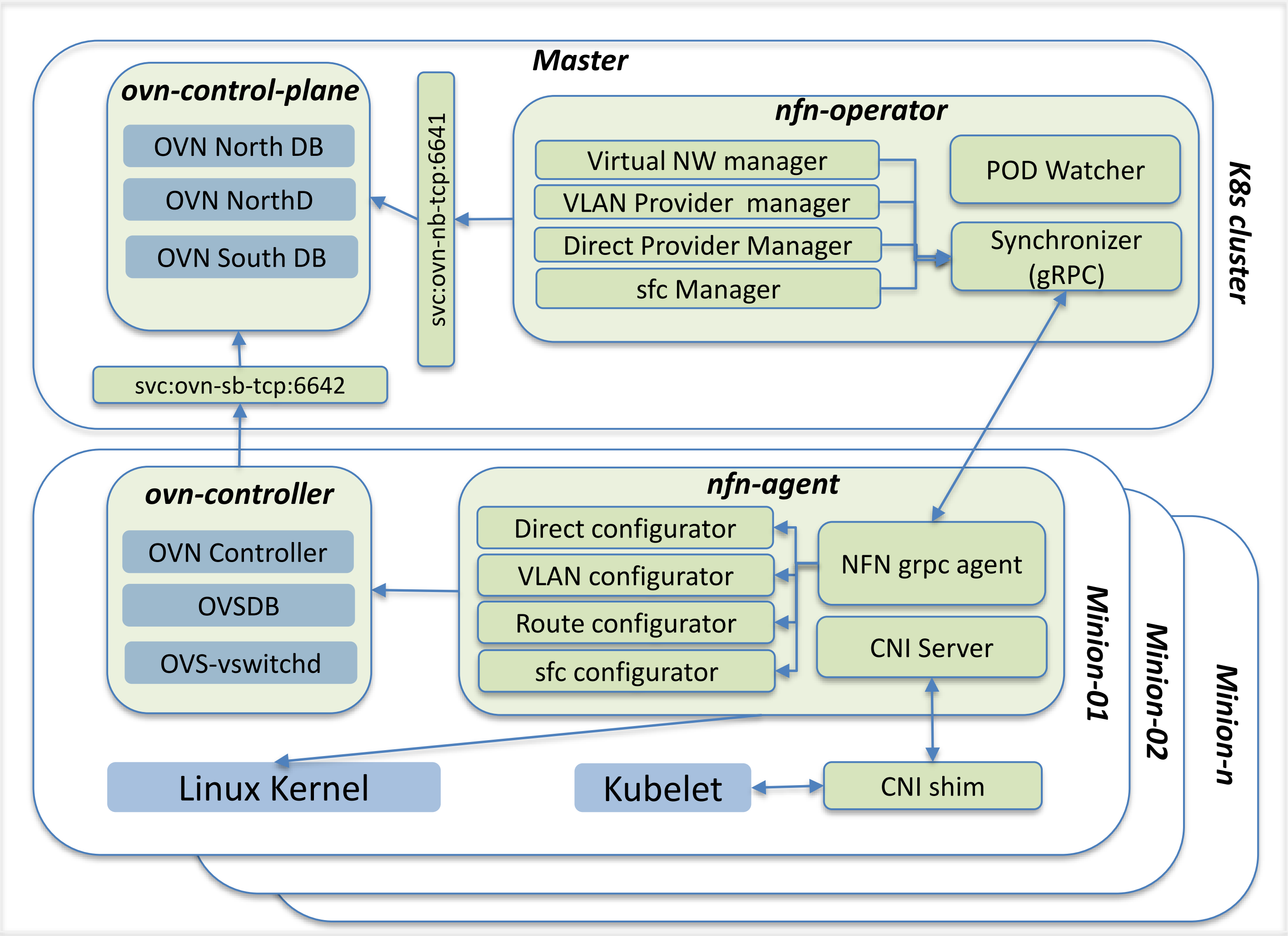
NODUS

Feature Reqmts	Dynamic virtual Networks	Provider networks	Multiple interfaces	Network function chaining	Network function load balancing
Considerations	No changes to NFs	No changes to Apps	Configuration via operators	Finite network SRIOV Overlay networking	Smart NIC friendly & AF_XDP for packet processing NFs

Why did we choose OVN for Nodus?

- One of the best programmable controller
- Hides OVS complexity
- Broader eco-system
- L2 CNI – Support for unicast, multicast, broadcast applications
- One site level IPAM – No IP address restriction with number of nodes
- Possible to implement critical features with table based pipeline
  (Firewall, Routing, Switching, Load balancing)
- SmartNIC friendly

Nodus Architecture blocks



NFN Operator:

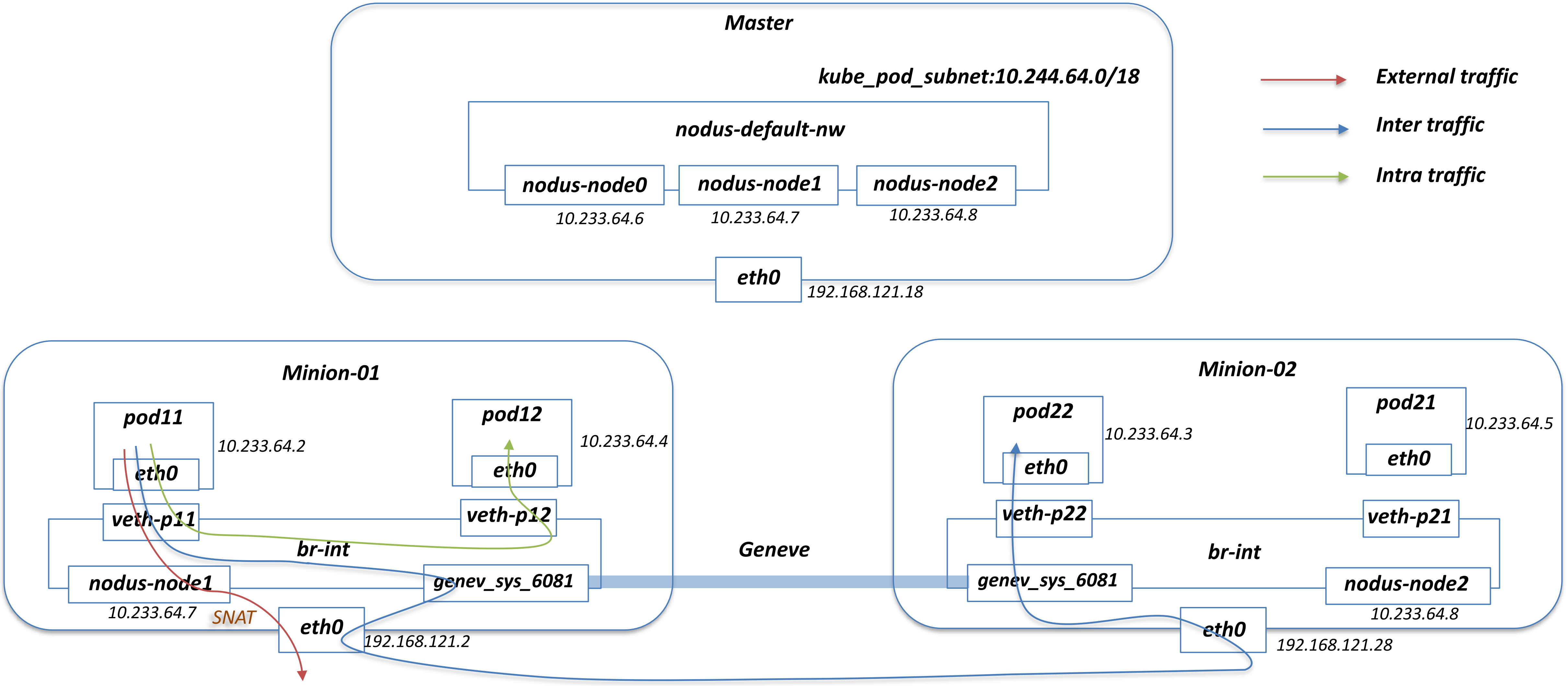
- Exposes virtual, provider, chaining CRDs to external world.
- Programs OVN to create L2 switches.
- Watches for PODs being coming up
  - Assigns IP addresses for every network of the deployment.
  - Looks for replicas and auto create routes for chaining to work.
  - Create LBs for distributing the load across CNF replicas.

NFN agent:

- Performs CNI operations.
- Configures VLAN and Routes in Linux kernel (in case of routes, it could do it in both root and network namespaces)
- Communicates with OVSDB to inform of provider interfaces. (creates ovs bridge and creates external-ids:ovn-bridge-mappings)

https://github.com/akraino-edge-stack/icn-nodus

Nodus - Network traffic between pods



Virtual Network CR

```yaml
apiVersion: k8splugin.opnfv.org/v1alpha1
kind: Network
metadata:
  name: ovn-priv-net
spec:
  cniType: Ovn4nfv
  ipv4subnets:
  - subnet: 172.16.33.0/24
    name: subnet1
    gateway: 172.16.33.1/24
    excludeIps: 172.16.33.2 172.16.33.5..172.16.33.10
```

Creates OVN Switch with this configuration

Revisited standards -
Draft_Kubernetes_Software_defined_Network_Custom_Resource_Definition_De-facto_Standard_out_of_CNI_scope

Dynamic Multiple Network Interfaces

Pod Annotation

```
k8splugin.opnfv.org/nfn-network: '{ "type": "ovn4nfv", "interface": [
               { "name": “ovn-priv-net”, "interfaceRequest": "eth1" },
               { "name":  “ovn-prot-net”, "interfaceRequest": "eth2" }
        ]}’
```

- Assumes primary/first interface provided by another CNI
- Supports Static IP addresses

Revisited standards -
Draft_Kubernetes_Software_defined_Network_Custom_Resource_Definition_De-facto_Standard_out_of_CNI_scope

Provider Network CR

```
apiVersion:  k8splugin.opnfv.org/v1alpha1
kind: OvnProviderNetwork
metadata:
  name: ovn-provider-net
spec:
  cniType: Ovn4nfv
  ipv4subnets:
  - subnet: 172.16.33.0/24
    name: subnet1
    gateway: 172.16.33.1/24
    excludeIps: 172.16.33.2 172.16.33.5..172.16.33.10
  providerNetworkType: vlan
  vlan:
    vlanId: 100
    providerInterfaceName: eth0
    Node: node1,node2
    logicalInterfaceName: eth0.100
```

Create OVN Switch and configures nodes

Revisited standards -

[Draft_Kubernetes_Software_defined_Network_Custom_Resource_Definition_De-facto_Standard_out_of_CNI_scope]

Provider Network Functionality

- CR creates OVN Switch
- Per Node (can be list of nodes, “all” nodes or “any” node)
  - Creates VLAN interfaces
  - Creates OVS Bridge and attaches VLAN interface
  - Configure ovs external-ids:ovn-bridge-mappings

- Pod annotation for attaching Provider network to a Pod

```
k8splugin.opnfv.org/nfn-network: '{ "type": "ovn4nfv", "interface": [
                 { "name": “ovn-provider-net”, "interfaceRequest": “net0" }
        ]}’
```

Revisited standards -
Draft_Kubernetes_Software_defined_Network_Custom_Resource_Definition_De-facto_Standard_out_of_CNI_scope

Network Chaining CR

```
apiVersion: k8s.plugin.opnfv.org/v1alpha1
kind: NetworkChaining
metadata:
  name: example-networkchaining
spec:
  # Add fields here
  chainType: "Routing"
  routingSpec:
    namespace: "default"
    networkChain: "net=virutal-net1,app=slb,net=dync-net1,app=ngfw,net=dync-net2,app=sdewan,net=virutal-net2"
    left:
    - networkName: "left-pnetwork"
      gatewayIp: "172.30.10.2"
      subnet: "172.30.10.0/24"
      podSelector:
        matchLabels:
          sfc: head
      namespaceSelector:
        matchLabels:
          sfc: head
    right:
    - networkName: "right-pnetwork"
      gatewayIp: "172.30.20.2"
      subnet: "172.30.20.0/24"
      podSelector:
        matchLabels:
          sfc: tail
      namespaceSelector:
        matchLabels:
          sfc: tail
```

Revisited standards -
Draft_Kubernetes_Software_defined_Network_Custom_Resource_Definition_De-facto_Standard_out_of_CNI_scope

NODUS

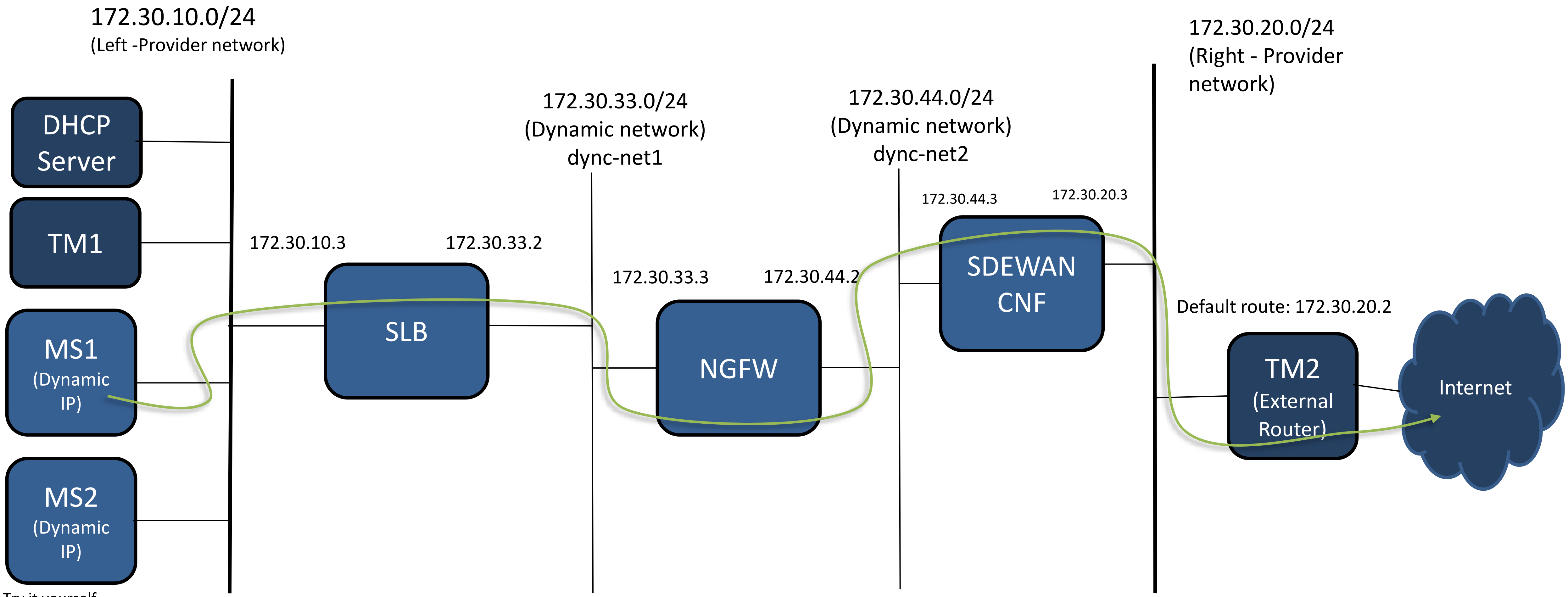
SFC Model Demo in Kubernetes

Goal: Labels eliminates Pod annotations

Overview

Contacts: kuralamudhan.ramakrishnan@intel.com

Nodus Demo - Traffic from pod within the cluster with sfc



172.30.10.0/24
(Left -Provider network)

172.30.33.0/24
(Dynamic network)
dync-net1

172.30.44.0/24
(Dynamic network)
dync-net2

172.30.20.0/24
(Right - Provider network)

DHCP Server

TM1

MS1 (Dynamic IP)

MS2 (Dynamic IP)

172.30.10.3

172.30.33.2

SLB

172.30.33.3

172.30.44.2

NGFW

172.30.44.3

172.30.20.3

SDEWAN CNF

Default route: 172.30.20.2

TM2 (External Router)

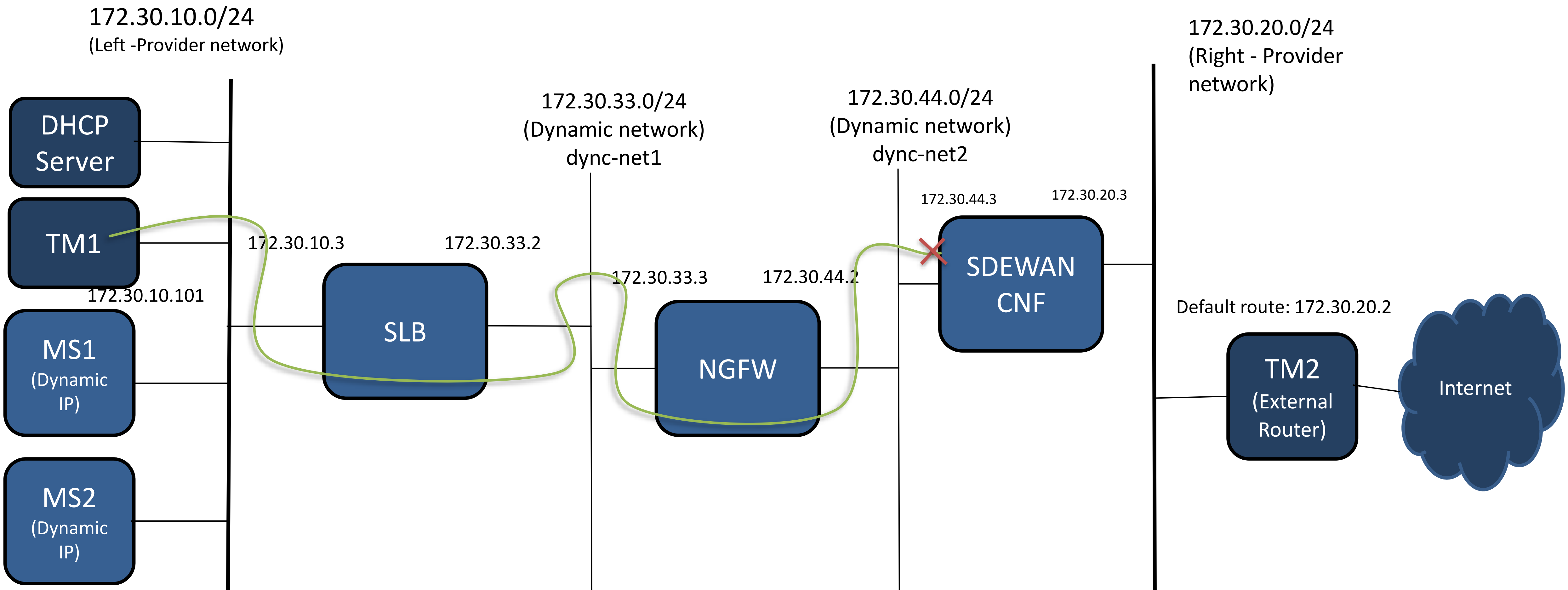
Internet

Try it yourself
Demo link
https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/nodus-primary-sfc-setup

■ External existing entities

■ VNF/CNFs

Nodus Demo - Traffic from external entities – Firewall icmp reject



Try it yourself

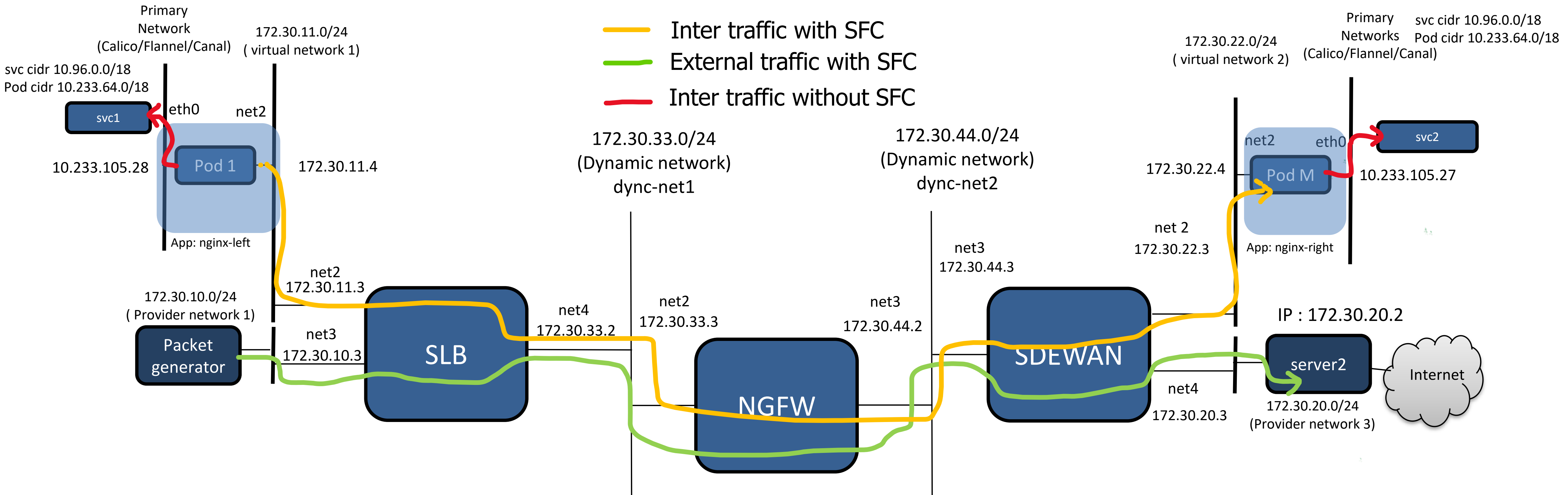
Demo link

https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/nodus-primary-sfc-setup

▬ External existing entities

▬ VNF/CNFs

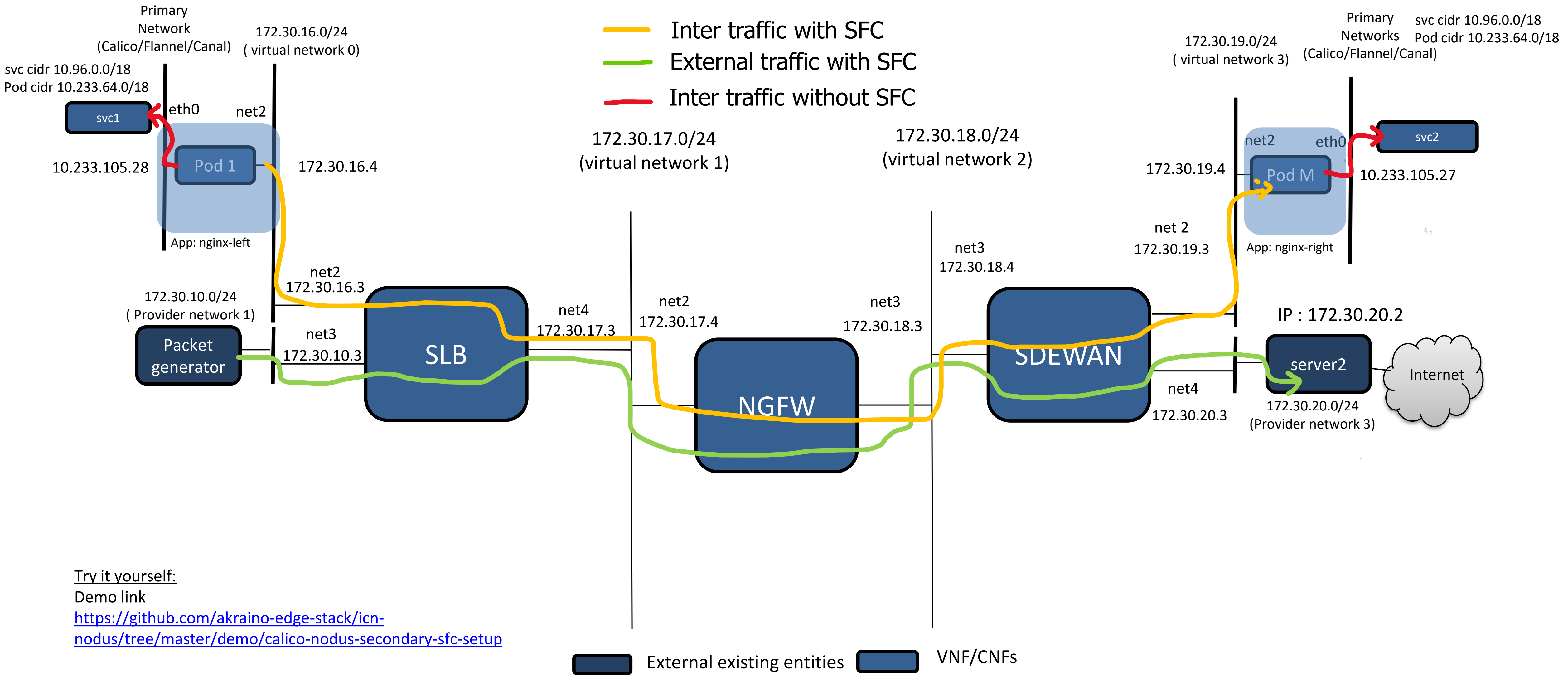
Nodus Advanced SFC - using provider networks & one Virtual networks with pod labels



Try it yourself:
Demo link
https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/calico-nodus-secondary-sfc-setup

▇ External existing entities ▇ VNF/CNFs

Nodus Advanced SFC - using Dynamic Network creation



Try it yourself:

Demo link

https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/calico-nodus-secondary-sfc-setup

Nodus Status

Current

- Dynamic Network Creation
- VLAN Provider Network Support – Controller and Agent
- Direct Provider Network Support – Controller and Agent
- SFC feature – Controller and Agent
- Kubespray default primary network plugin
- Tested with sdewan CNFs and SDEWAN Controller
- Multiple SFC Network chaining – Working on 4 SFC models
- EMCO network controller uses Nodus

Link to Repo:
https://github.com/akraino-edge-stack/icn-nodus
Demo:
https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/nodus-primary-sfc-setup
https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/calico-nodus-secondary-sfc-setup

Network Innovation continues….

Work In Progress

- VM SFC in Kubernetes
- SRIOV NIC as primary network interfaces
- Using OVN Load balancer for Kubernetes service(without kube-proxy)
- SFC support with OVN load balancer support for NF Elasticity
- Network policy with OVS
- Proxy less service mesh with OVN & Ipsec in network namespace
- IPv6 support
- Traffic interception method with 5G UPF
- Kubespray Centos CI/CD, SFC advance testing
- Standard Software Defined Network Defacto standard in Kubernetes – Google Docs