OVN4NFV - SFC

Goal: To have Cloud native SFC through K8s Custom resources

Overview

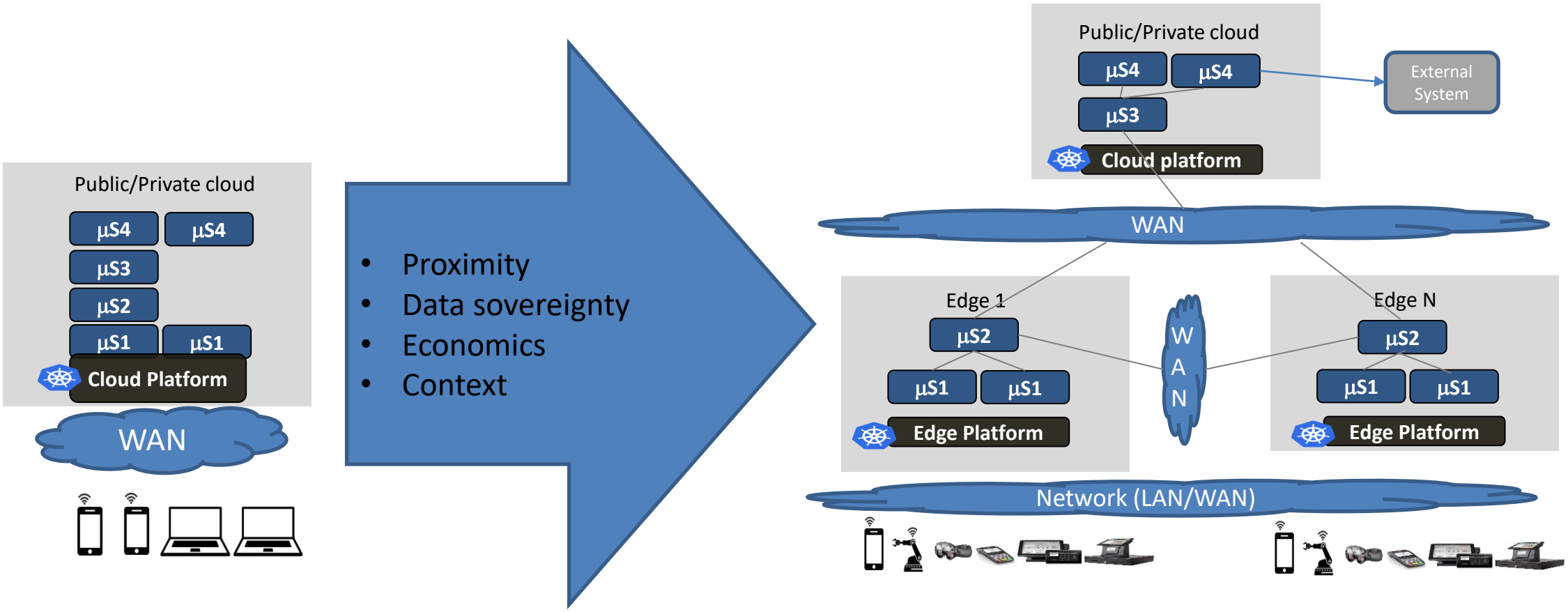
Contacts: Srinivasa.r.addepalli@intel.com; kuralamudhan.ramakrishnan@intel.com

Agenda

- Why network functions in Edge & K8s clusters?
- Edge-computing scenario to describe the K8s networking requirements
- Networking requirements
- OVN4NFV –SFC
- SDEWAN Use case

Application Transformation

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

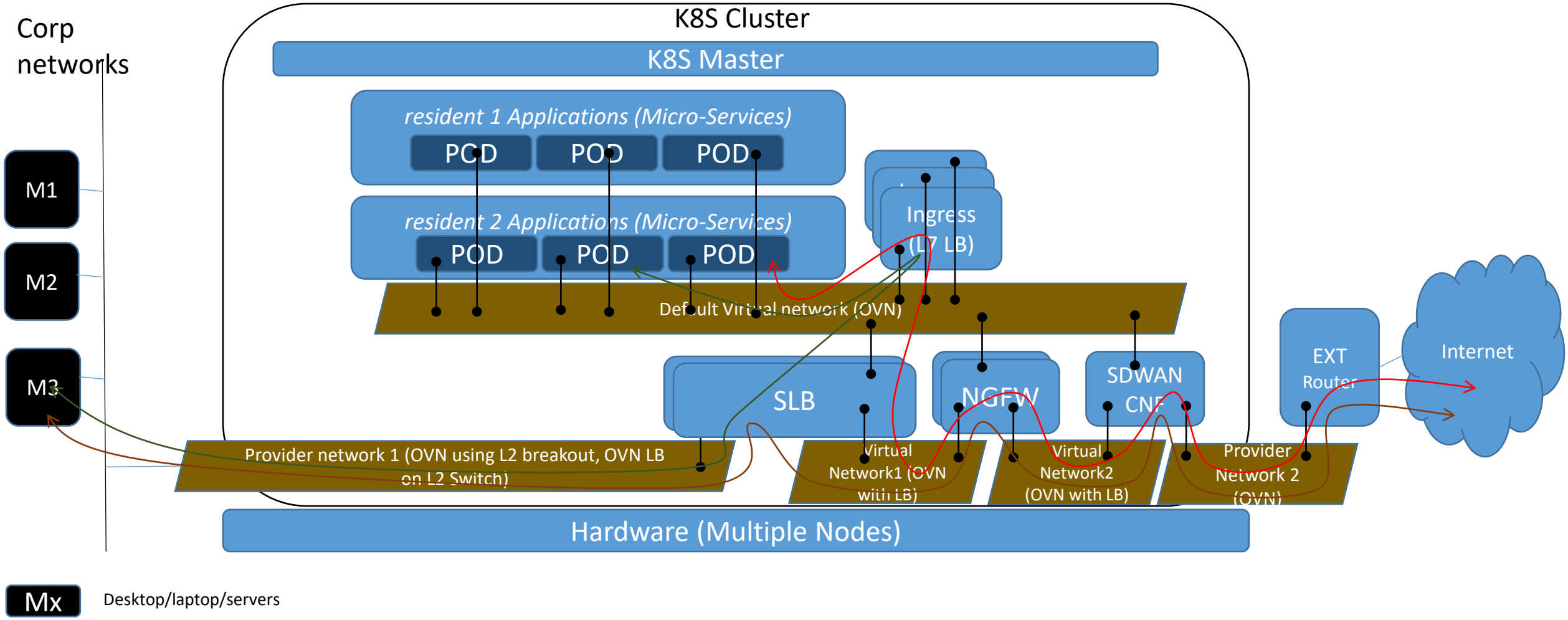


Centralized computing to Geo distributed computing

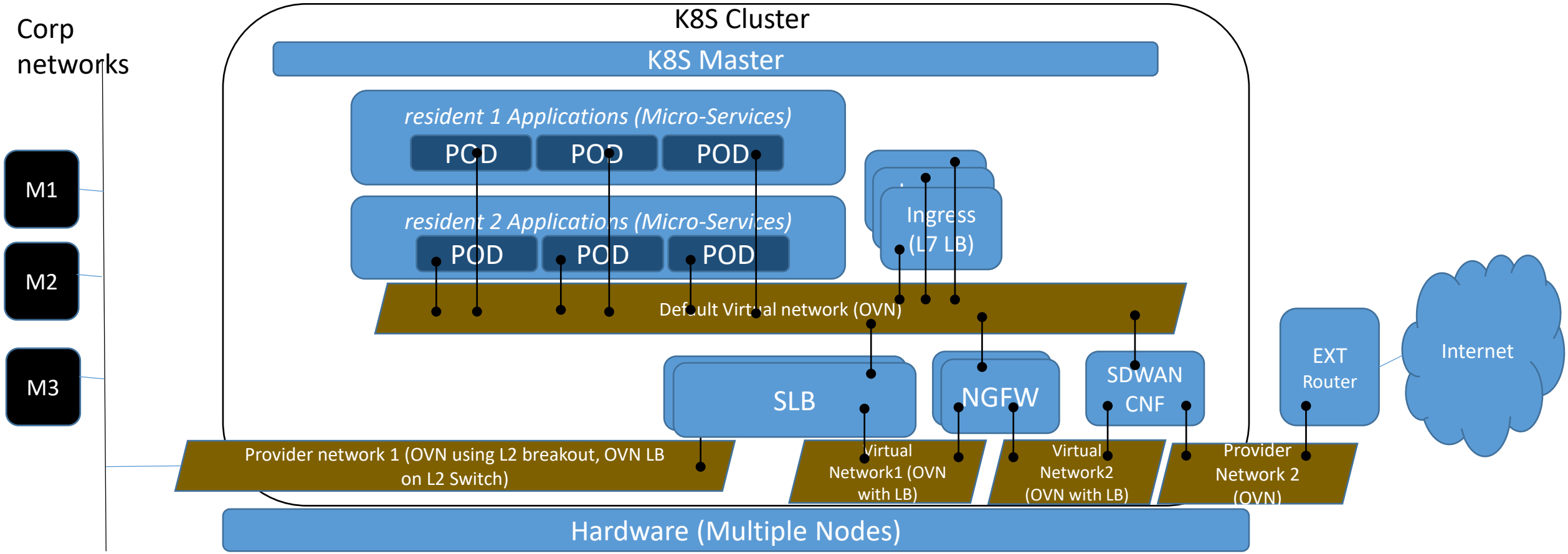
An App consisting of four Micro-services
ms1 talks to ms2, ms2 to ms3 and ms3 to ms4
ms1" is user facing service
"ms1", "ms2" are expected to be there together
"ms2" is stateful and hence need to talk to each other

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

View in Slide show



Networking Requirements

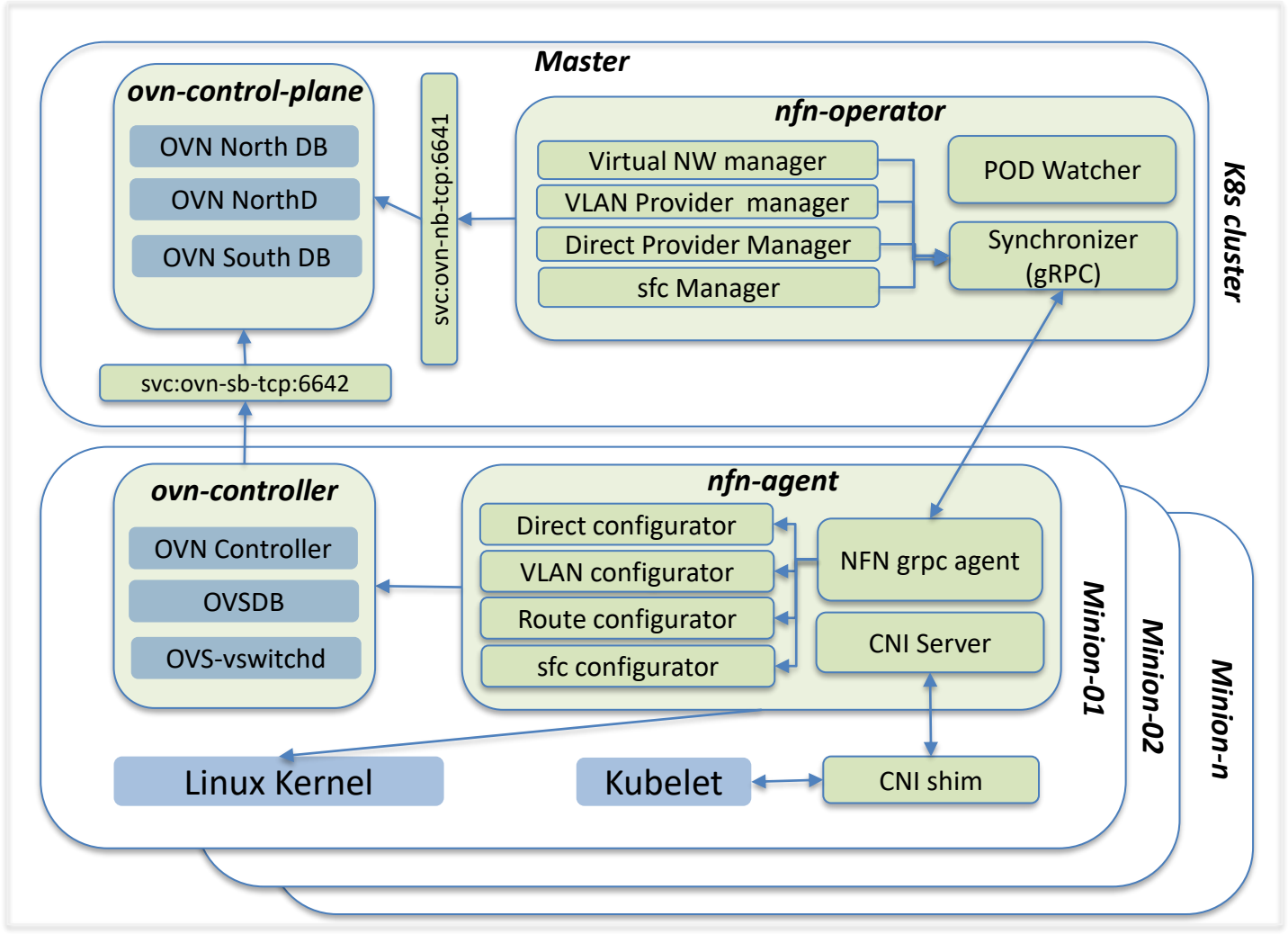


Feature Reqmts	Dynamic virtual Networks	Provider networks	Multiple interfaces	Network function chaining	Network function load balancing
Considerations	No changes to NFs	No changes to Apps	Configuration via operators	Finite network SRIOV Overlay networking	Smart NIC friendly & AF_XDP for packet processing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

Why did we choose OVN in for Edge Networking?

- 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

OVN for K8S and NFV Architecture blocks



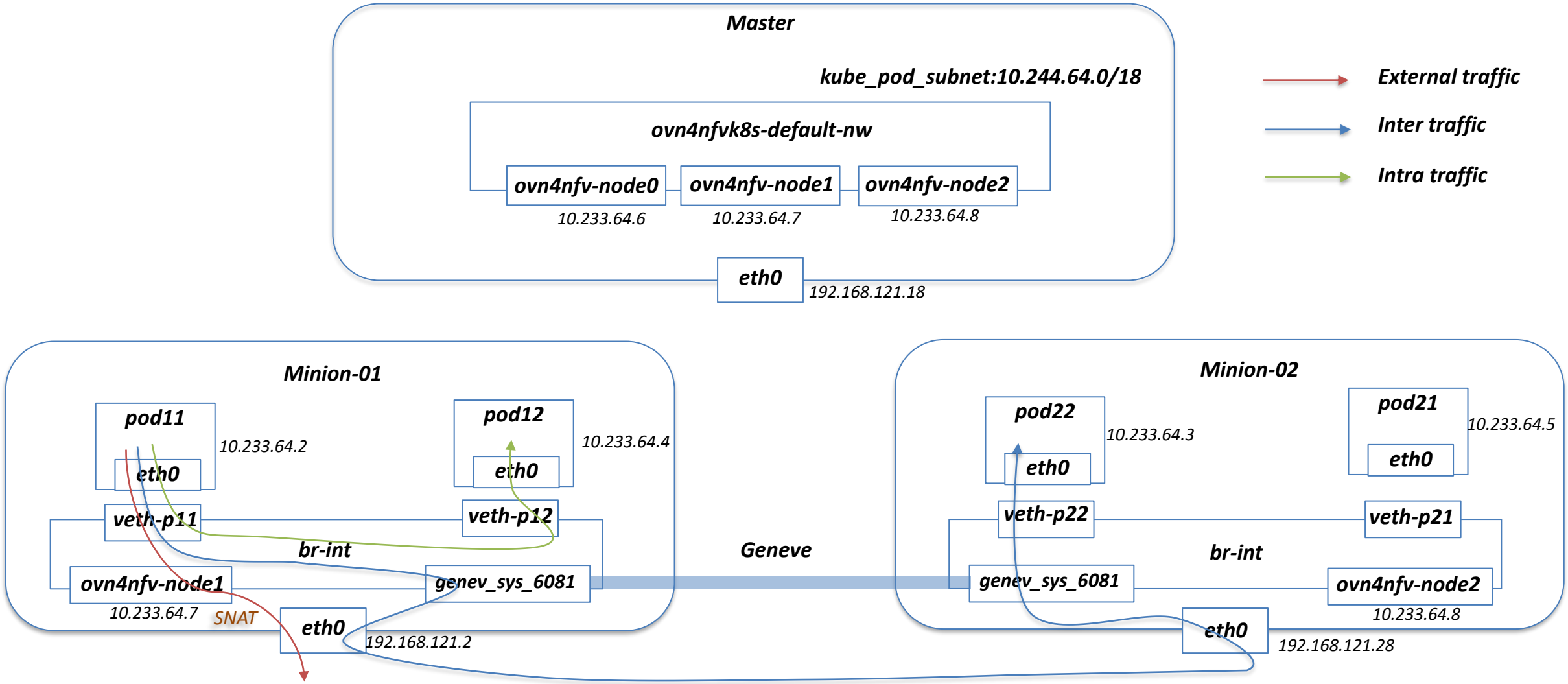
NFN Operator:

- 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)

Network traffic between pods



Virtual Network CR

```
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

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

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

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

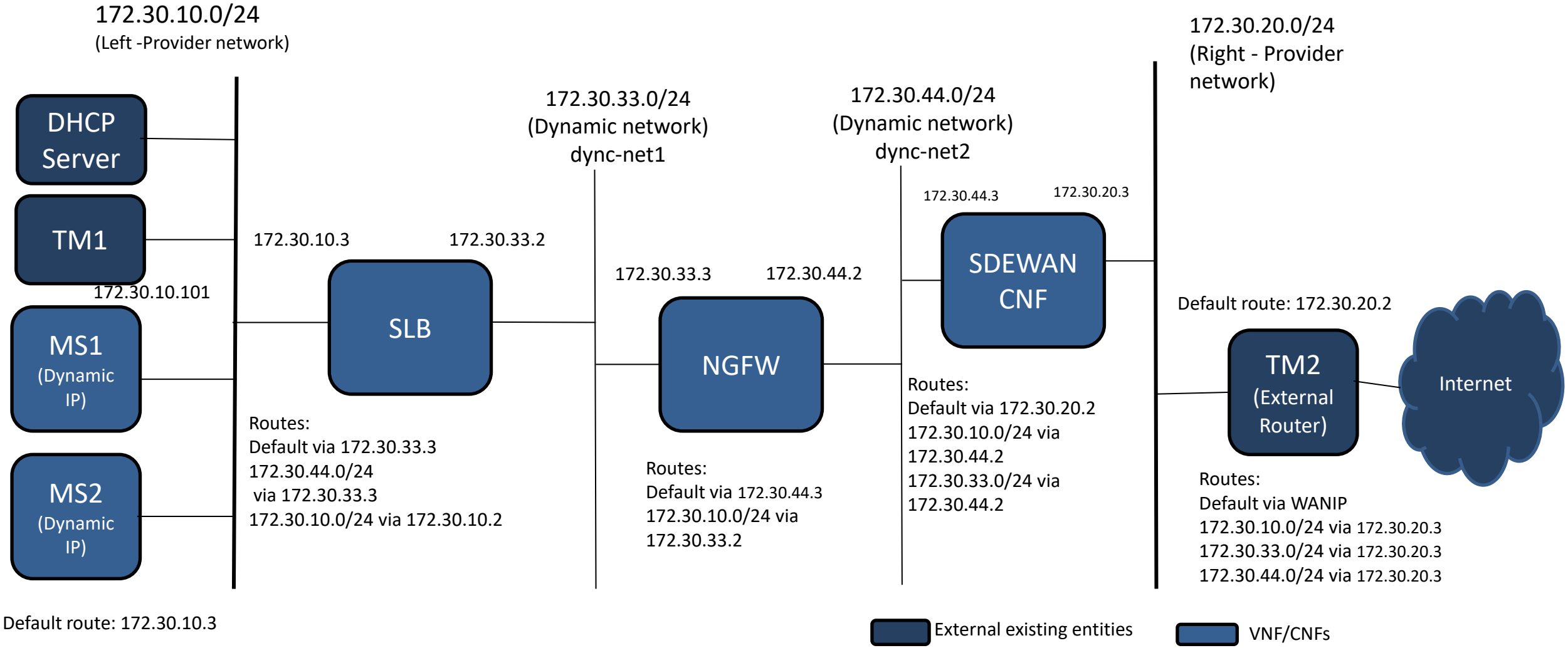
```
k8splugin.opnfv.org/nfn-network: '{ "type": "ovn4nfv", "interface": [
            { "name": “ovn-provider-net”, "interfaceRequest": “net0" }
     ]}’
```

Network Chaining CR

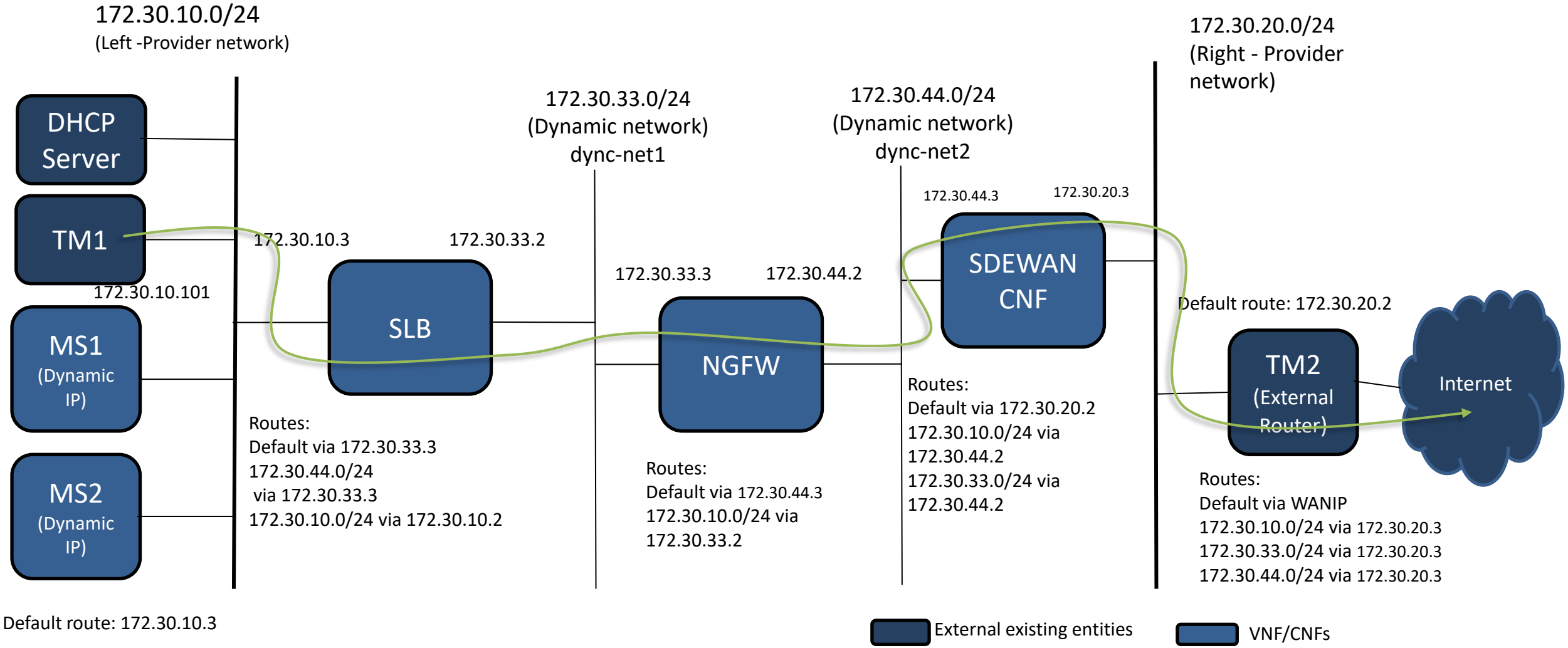
```
apiVersion:  k8splugin.opnfv.org/v1alpha1
kind: NetworkChaining
metadata:
  name: chain1
  namespace: vFW
spec:
  type: Routing
  routingSpec:
    leftNetwork:
      - networkName: ovn-provider1
        gatewayIP: 10.1.5.1
        subnet: 10.1.5.0/24
    rightNetwork:
      - networkName: ovn-provider1
        gatewayIP: 10.1.10.1
        subnet: default
    networkChain: app=slb, ovn-net1, app=ngfw, ovn-net2, app=sdwancnf
```

Inserts routes in Container Namespaces

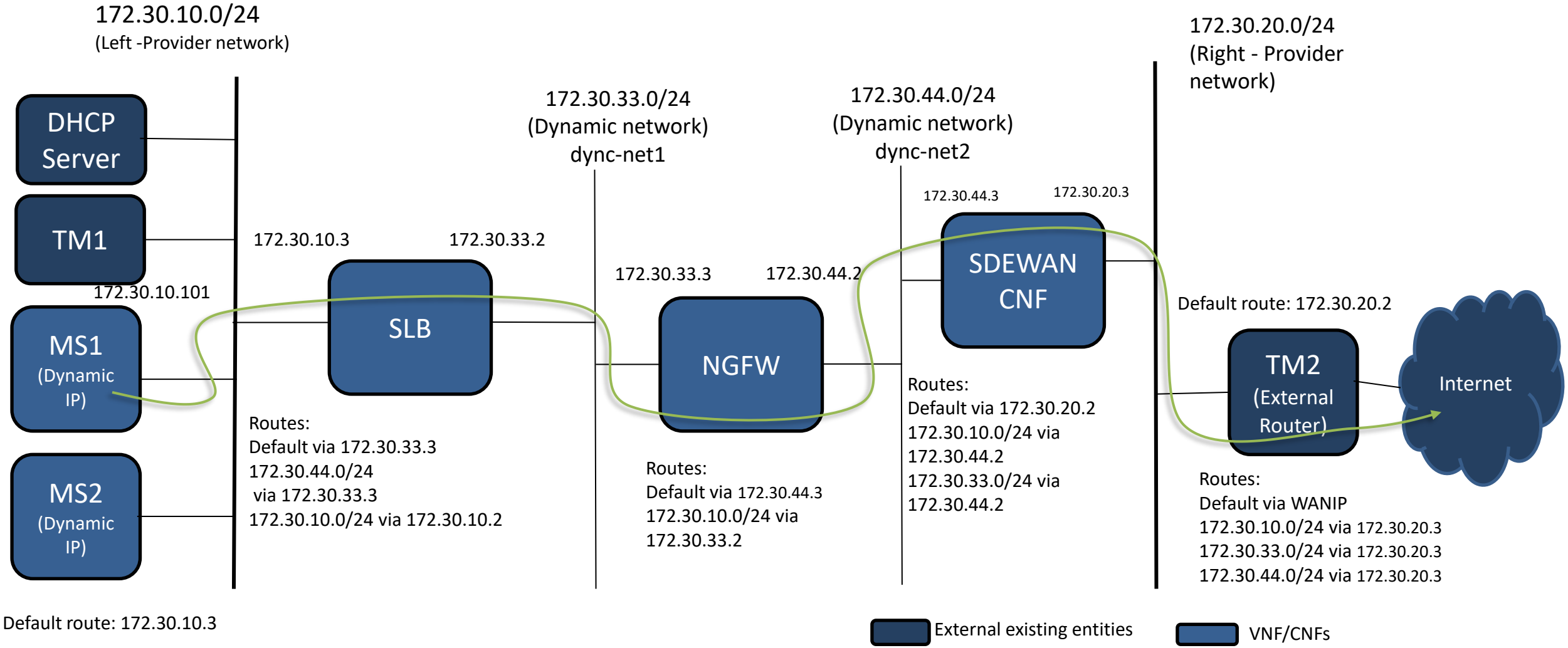
Test scenario – to comprehend multiple deployment variations



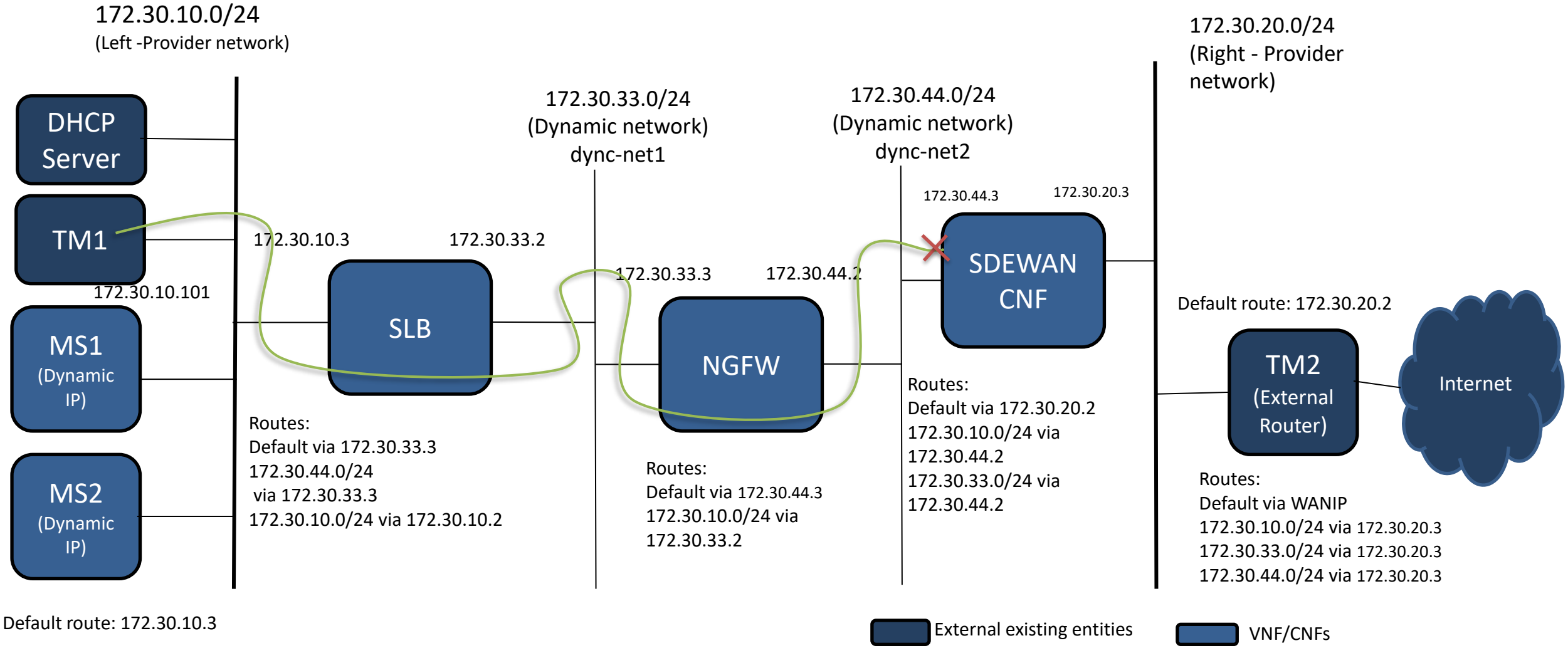
Traffic from external entities with sfc



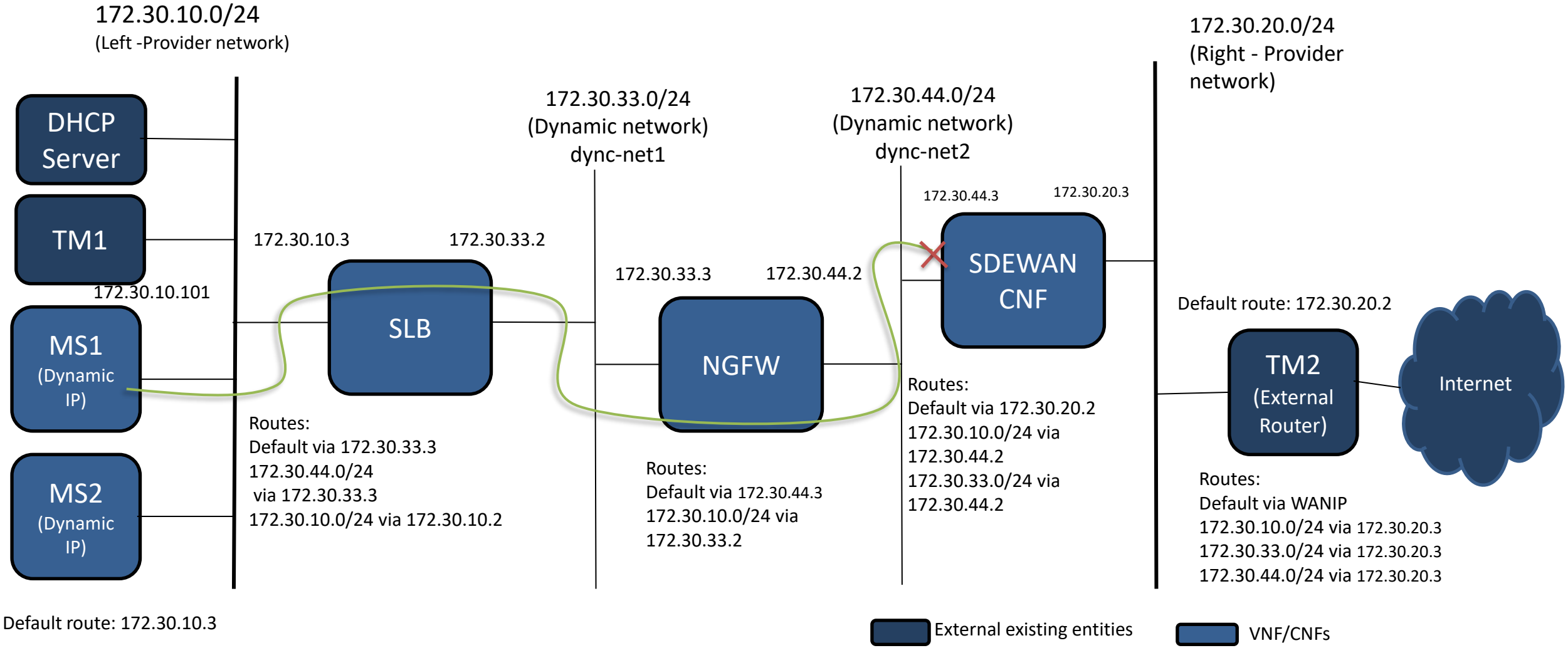
Traffic from pod within the cluster with sfc



Traffic from external entities – Firewall icmp reject



Traffic from pod within the cluster – Firewall icmp reject



OVN4NFV 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

Link to Repo:
https://github.com/akraino-edge-stack/icn-ovn4nfv-k8s-network-controller
https://github.com/kubernetes-sigs/kubespray/blob/master/docs/ovn4nfv.md

Upcoming features in OVN4NFV

Work In Progress

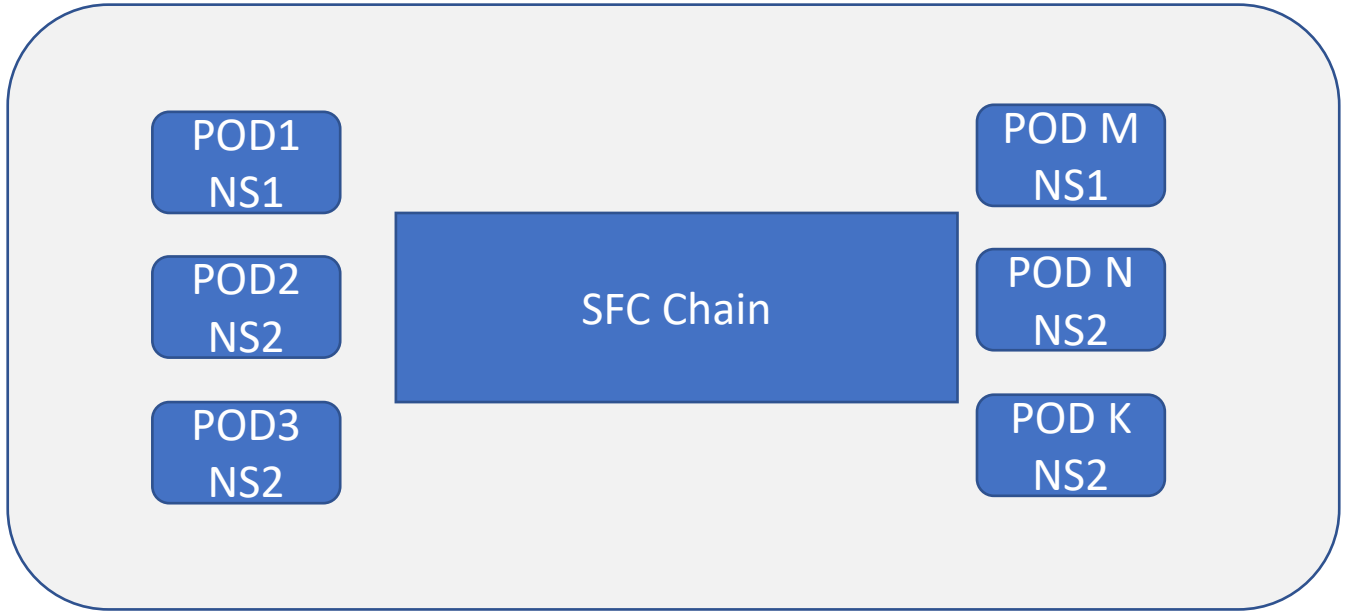
- Multiple SFC Network chaining – Working on 4 SFC models
- SRIOV NIC as primary network interfaces
- Using OVN Load balancer for Kubernetes service(without kube-proxy)
- SFC support with OVN load balancer support for NF Elasticity
- Network policy with OVS
- Proxy less service mesh with OVN & Ipsec in network namespace
- IPv6 support
- Traffic interception method with 5G UPF
- Kubespray Centos CI/CD, SFC advance testing

SFC Model in Kubernetes

Goal: Labels eliminates Pod annotations

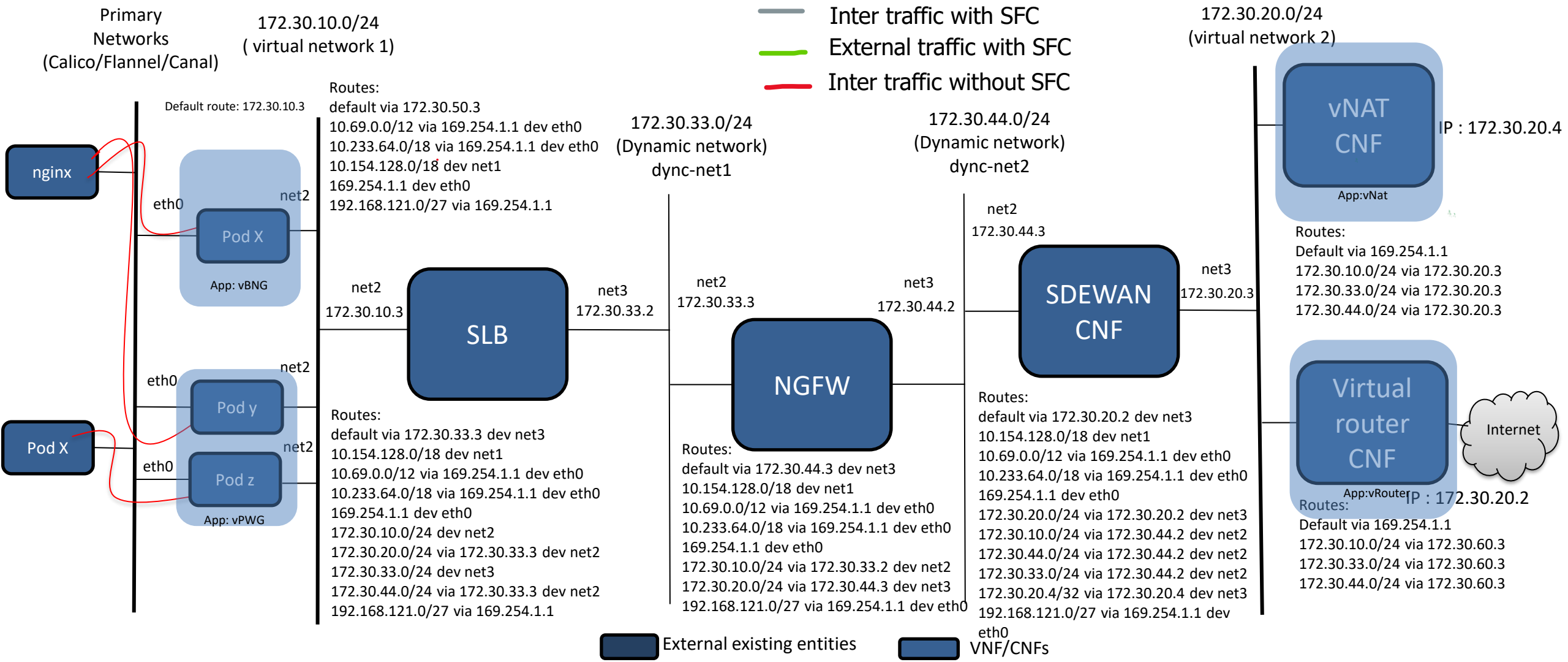
Overview

Contacts: Srinivasa.r.addepalli@intel.com; kuralamudhan.ramakrishnan@intel.com

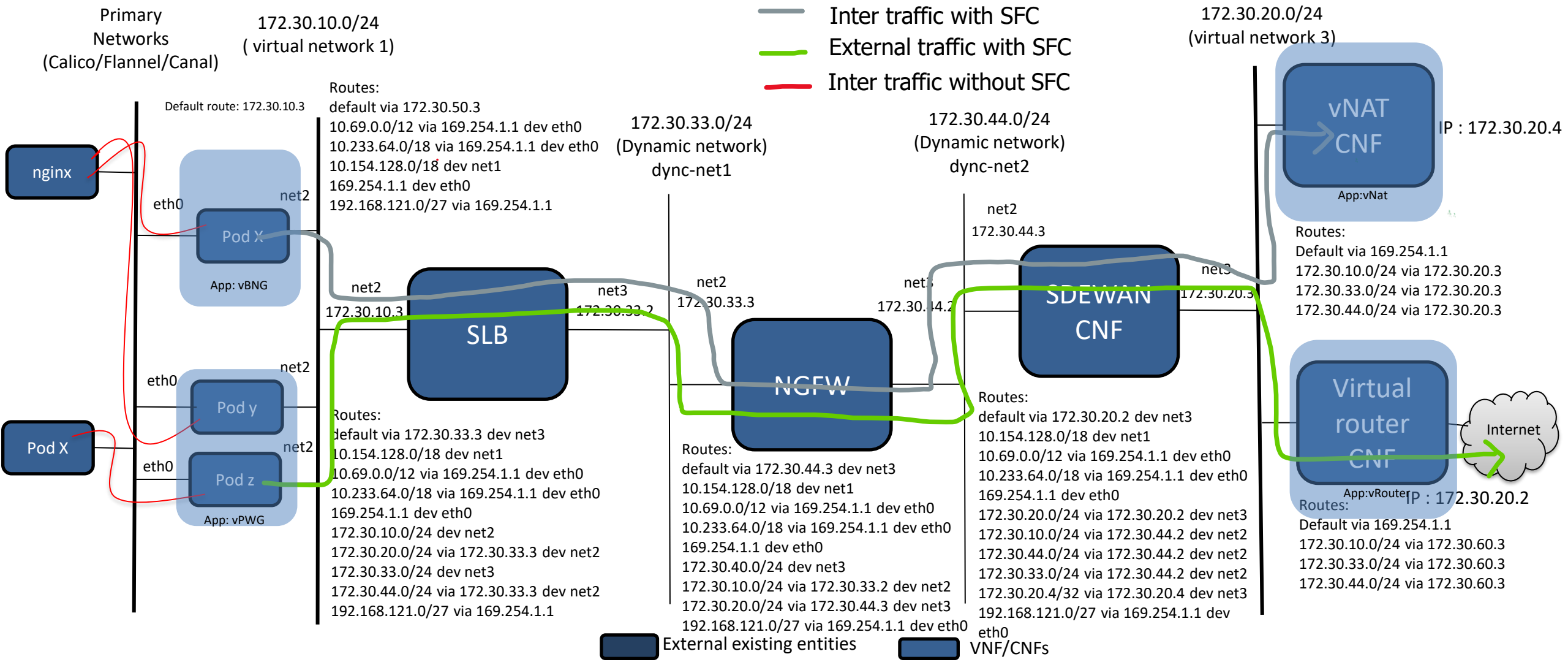


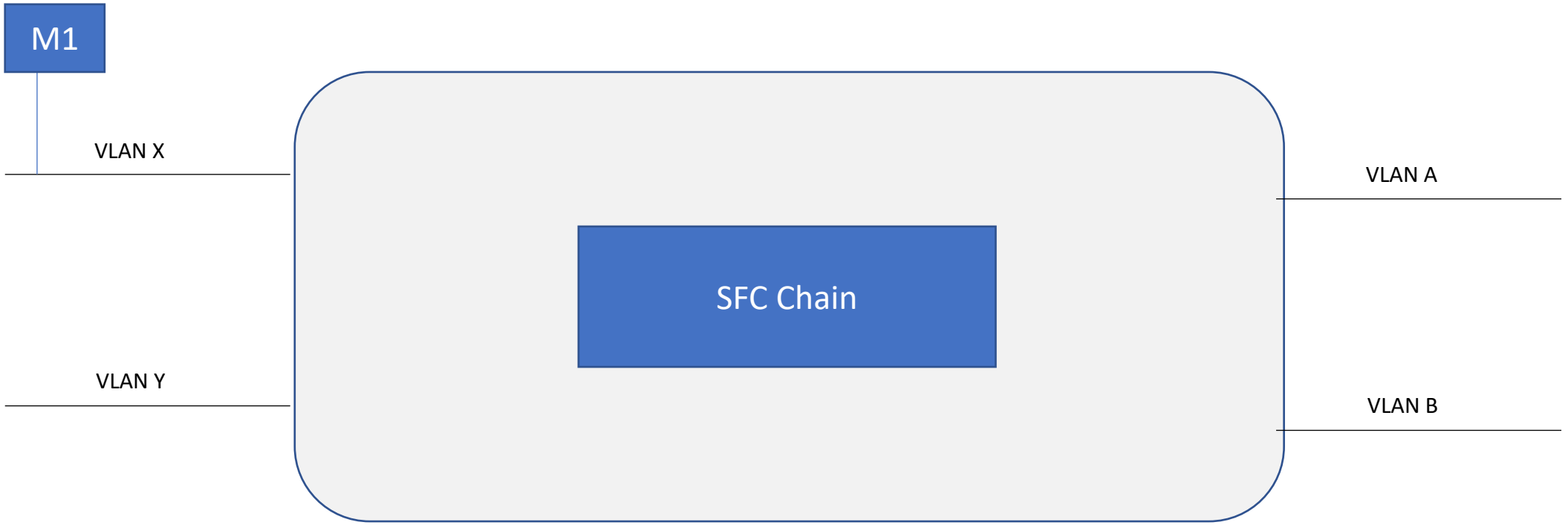
Model 1
Only Labels

Using only one virtual networks at head – tail of SFC & pod labels



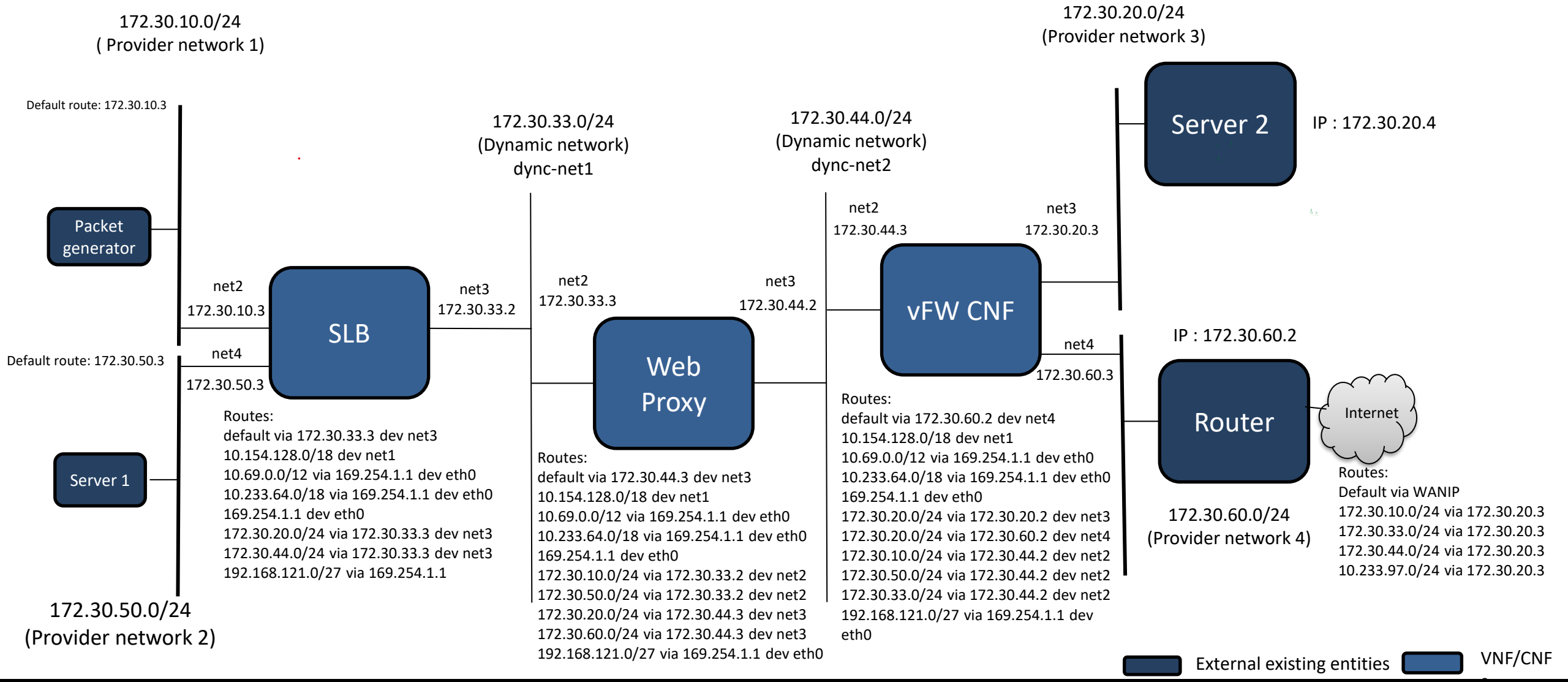
Using only one virtual networks at head – tail of SFC & pod labels



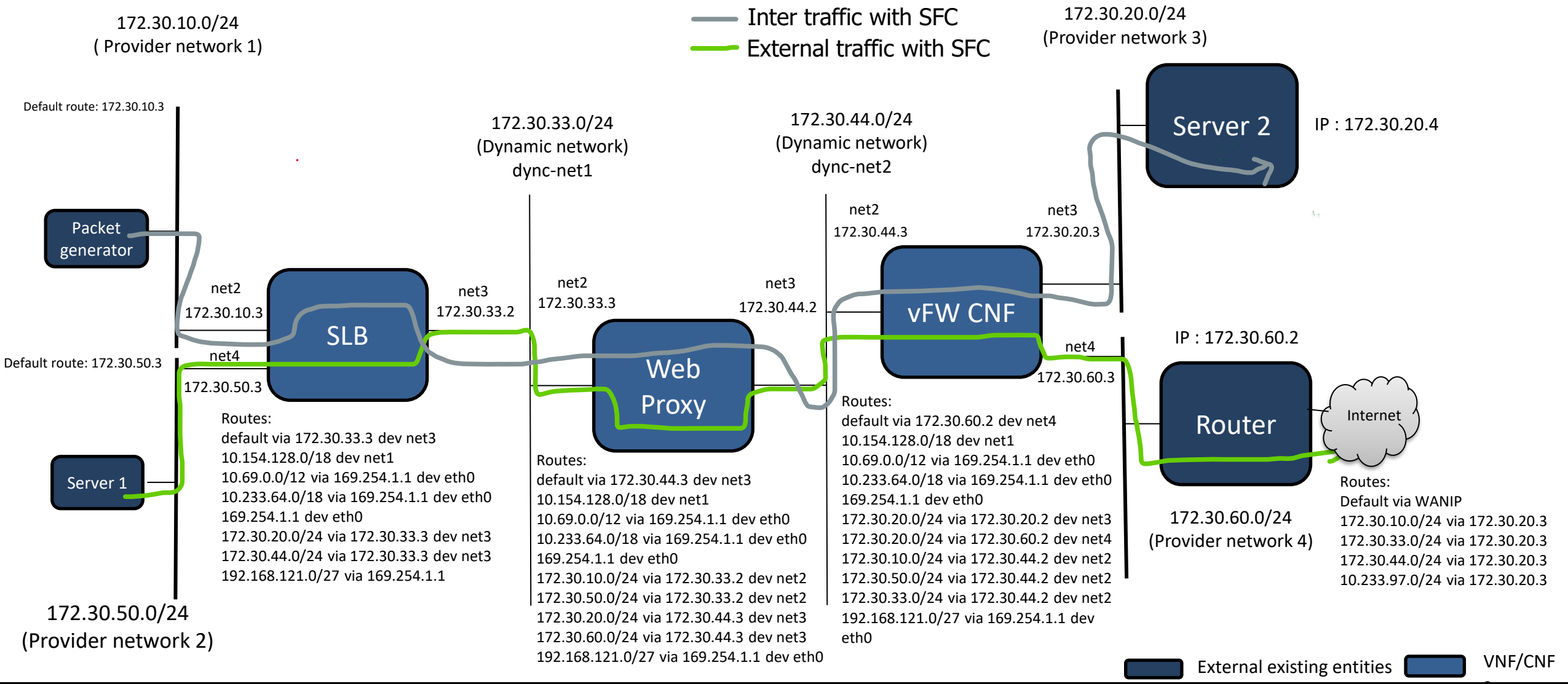


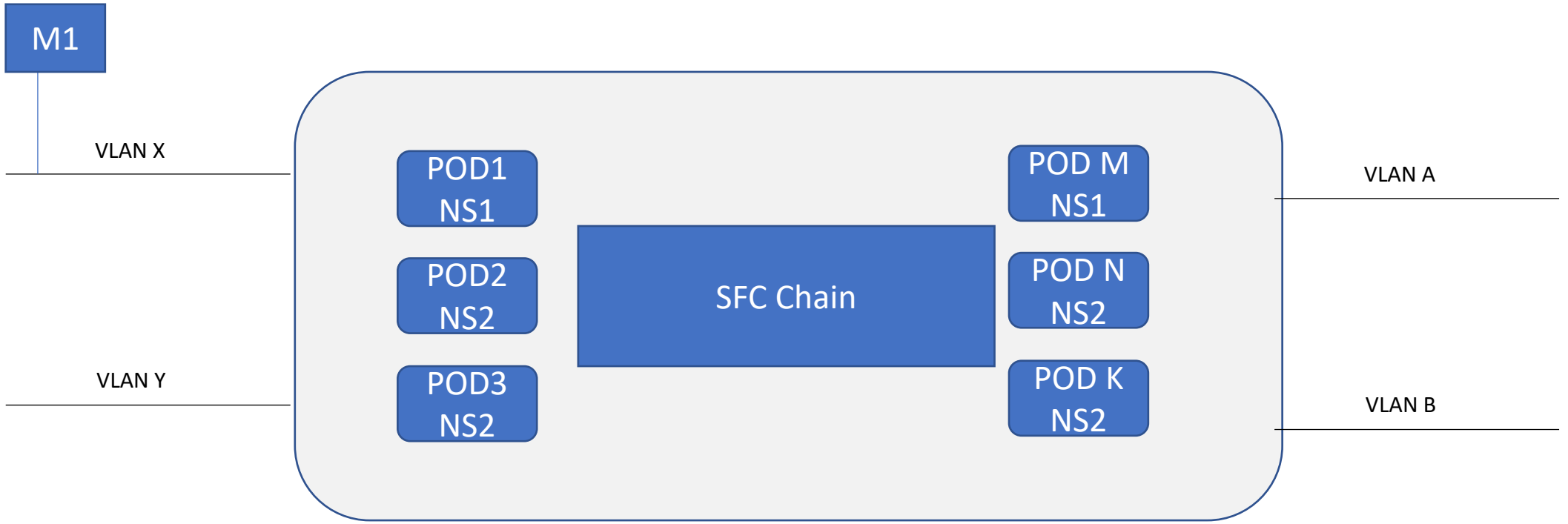
Model 2
Provider
network only

Using provider networks only(only servers connected)



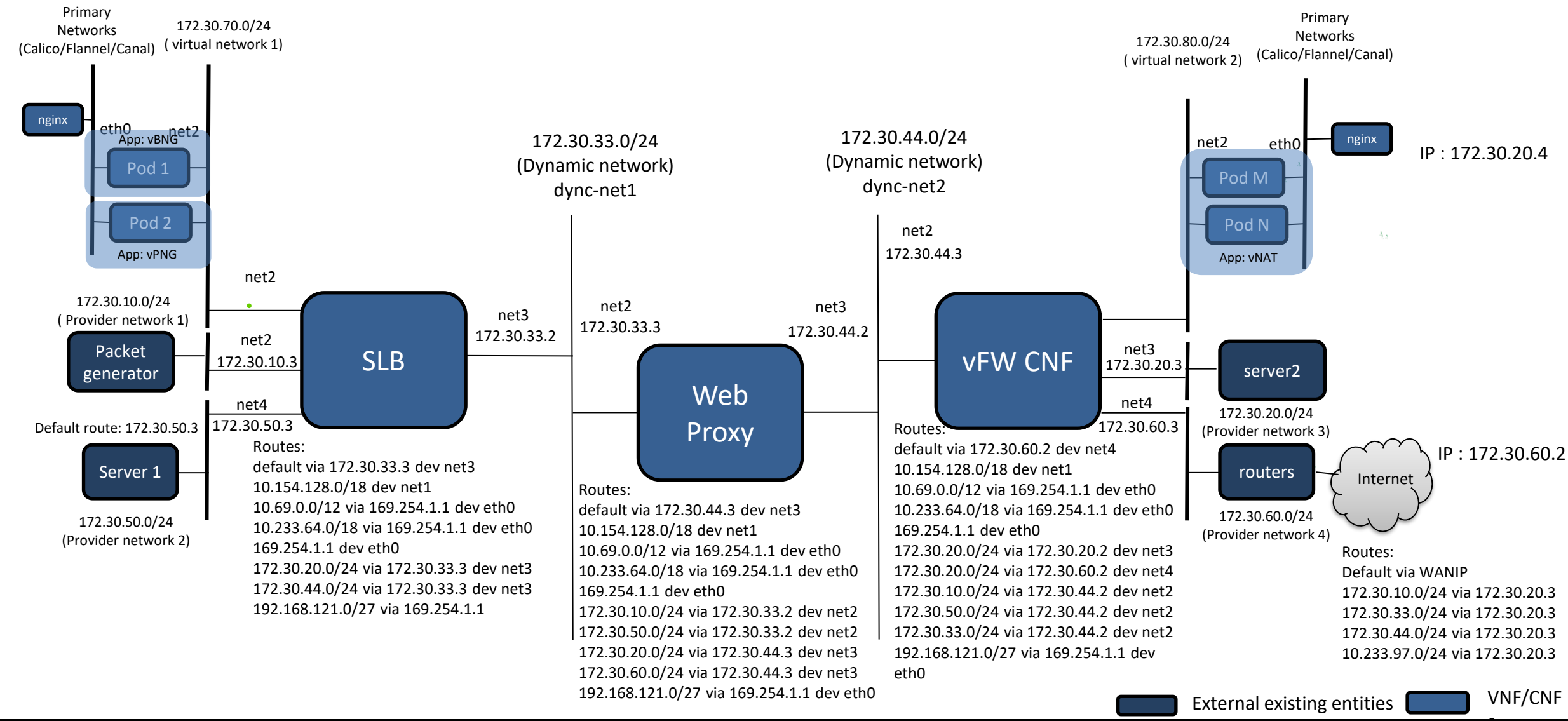
Using provider networks only(only servers connected)



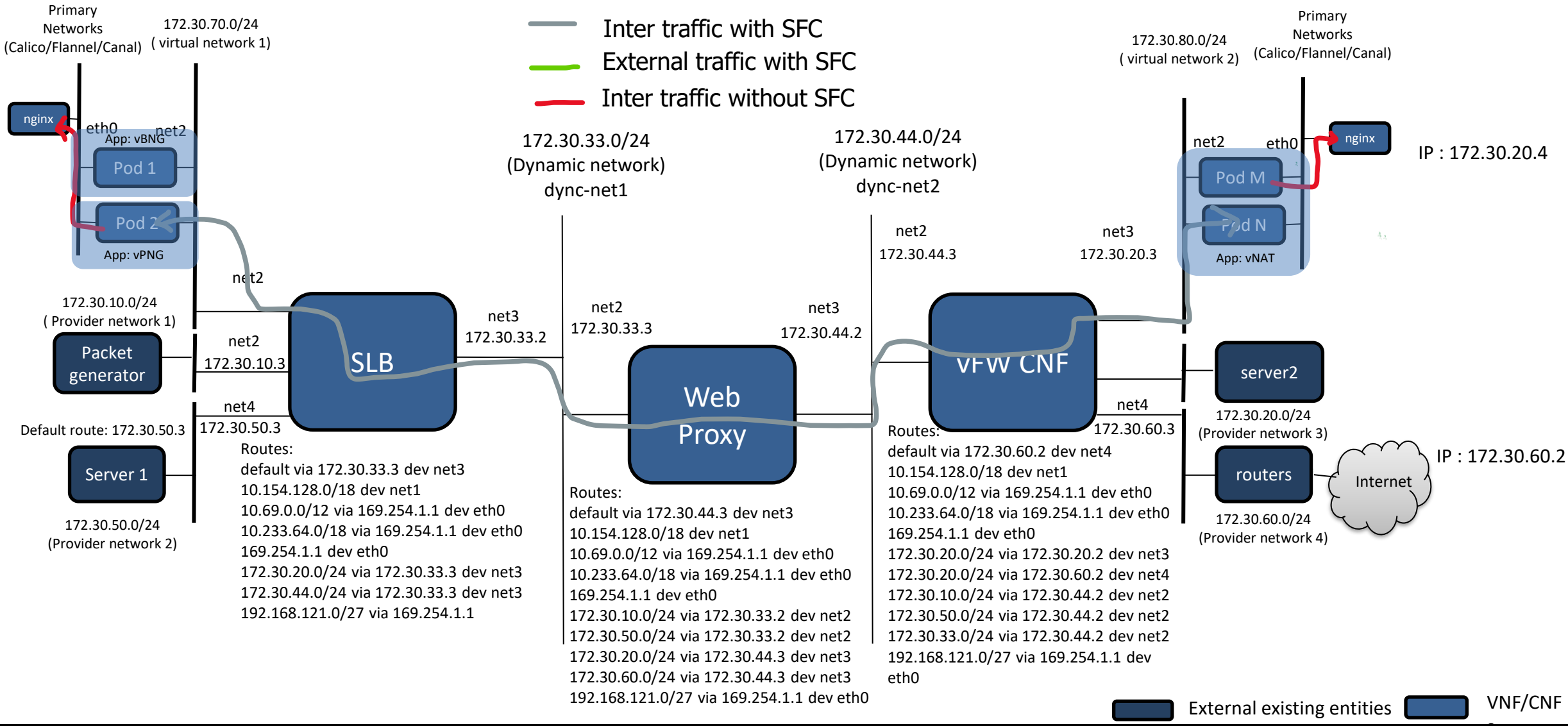


Model 3 & 4
Hybrid model
Labels + provider network

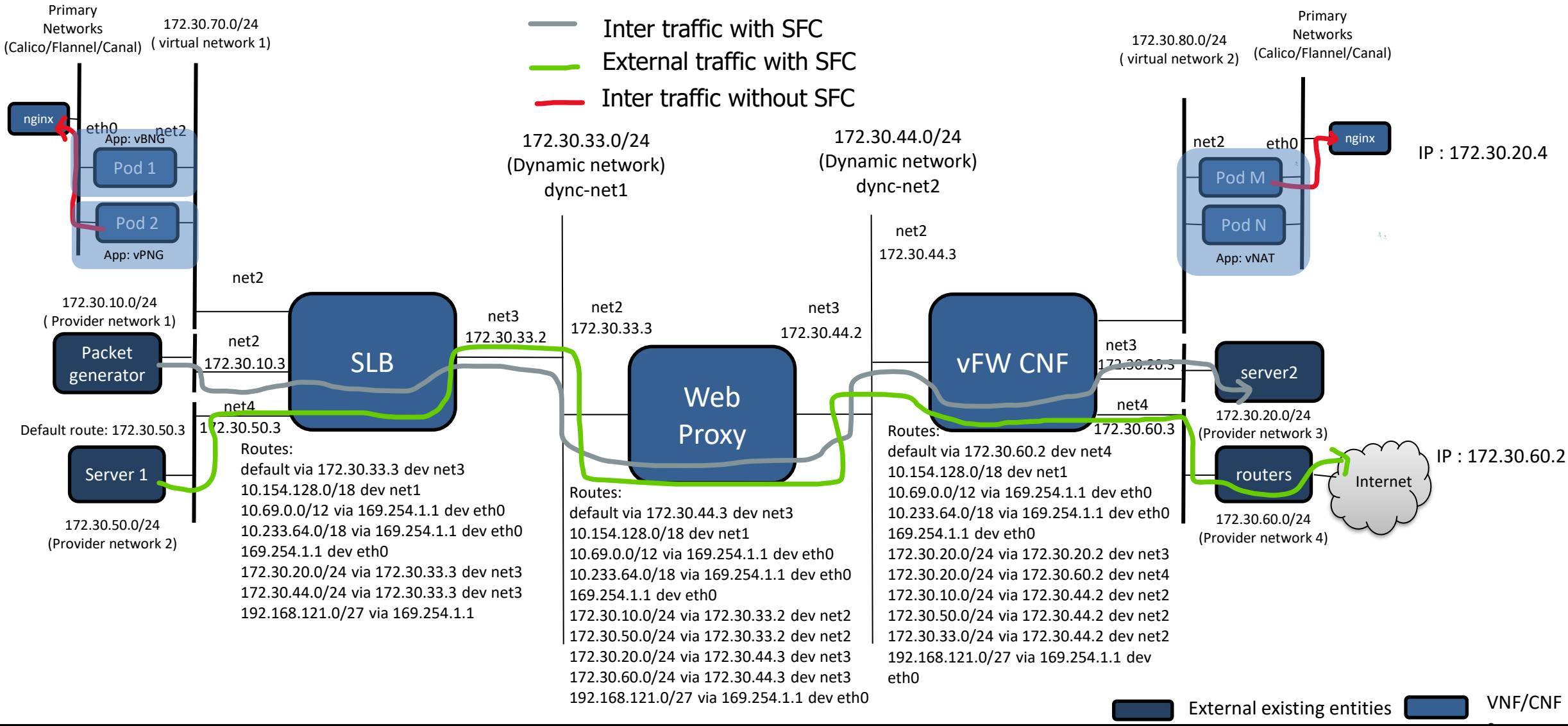
Using 2 provider networks & one Virtual networks with pod labels



Using 2 provider networks & one Virtual networks with pod labels – Model 3



Using 2 provider networks & one Virtual networks with pod labels – Model 3



Using 2 provider networks & one Virtual networks with pod labels – Model 4