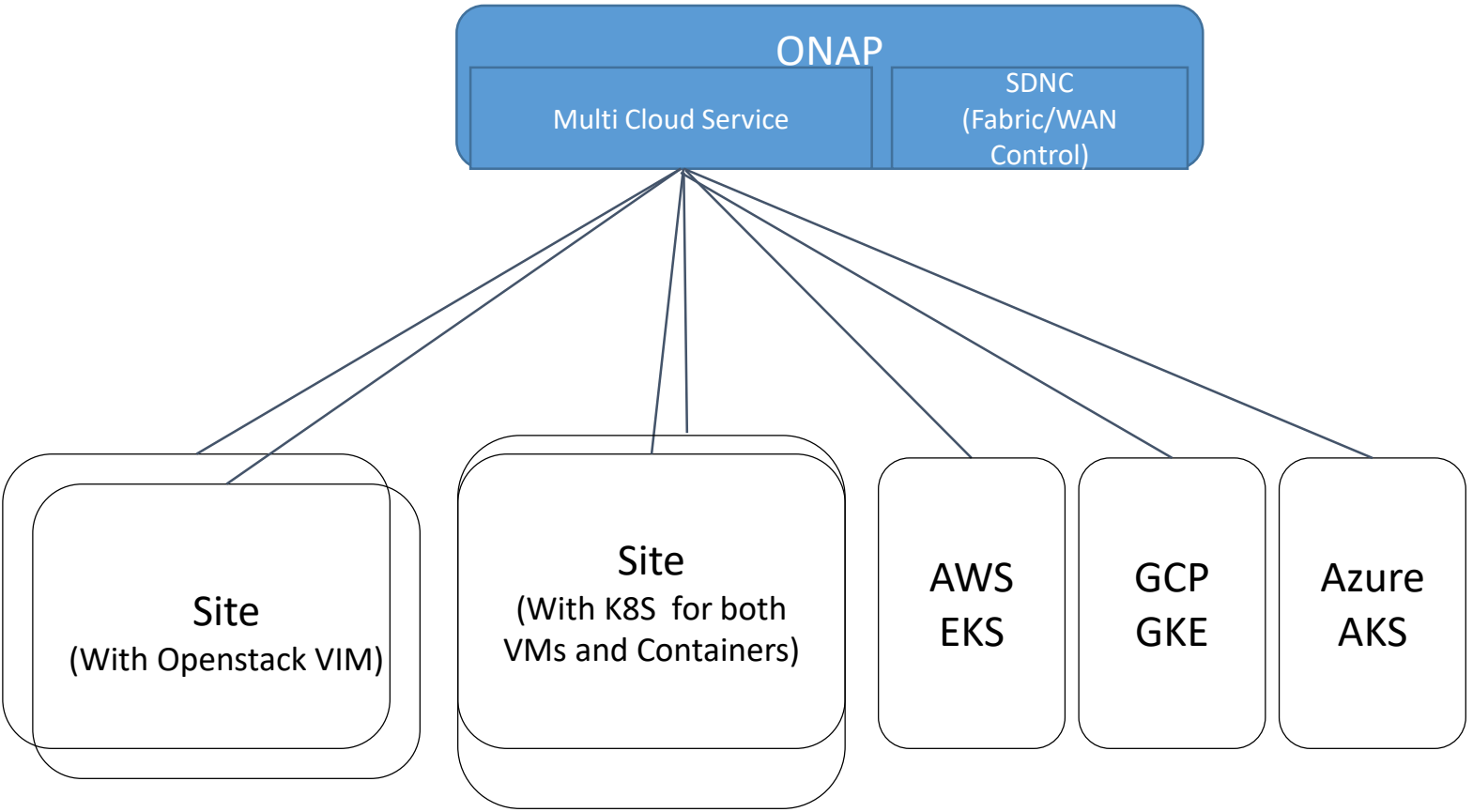
K8S for NFV

Leveraging ONAP, OPNFV, CNCF

Contacts: ONAP Edge automation working group
Presentation contacts: Srinivasa.r.addepalli@intel.com, Ritu.sood@intel.com

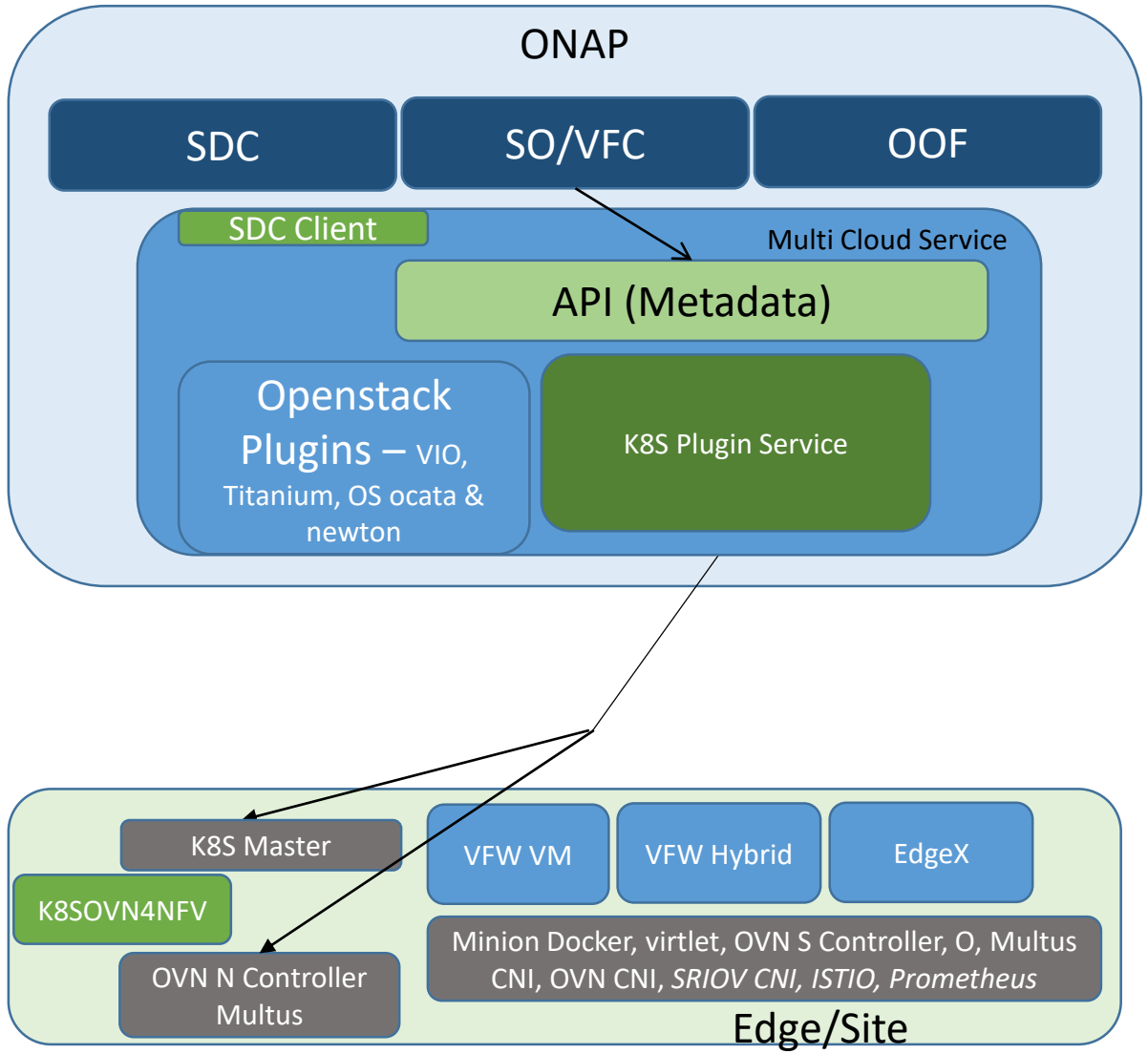
ONAP – Support for K8S based Sites



- Current support as in R2: Openstack based remote Clouds, Support multiple Openstack variations – Windriver Titanium, VMWare VIO, Native Newton, Ocata. Only VM based VNFs.

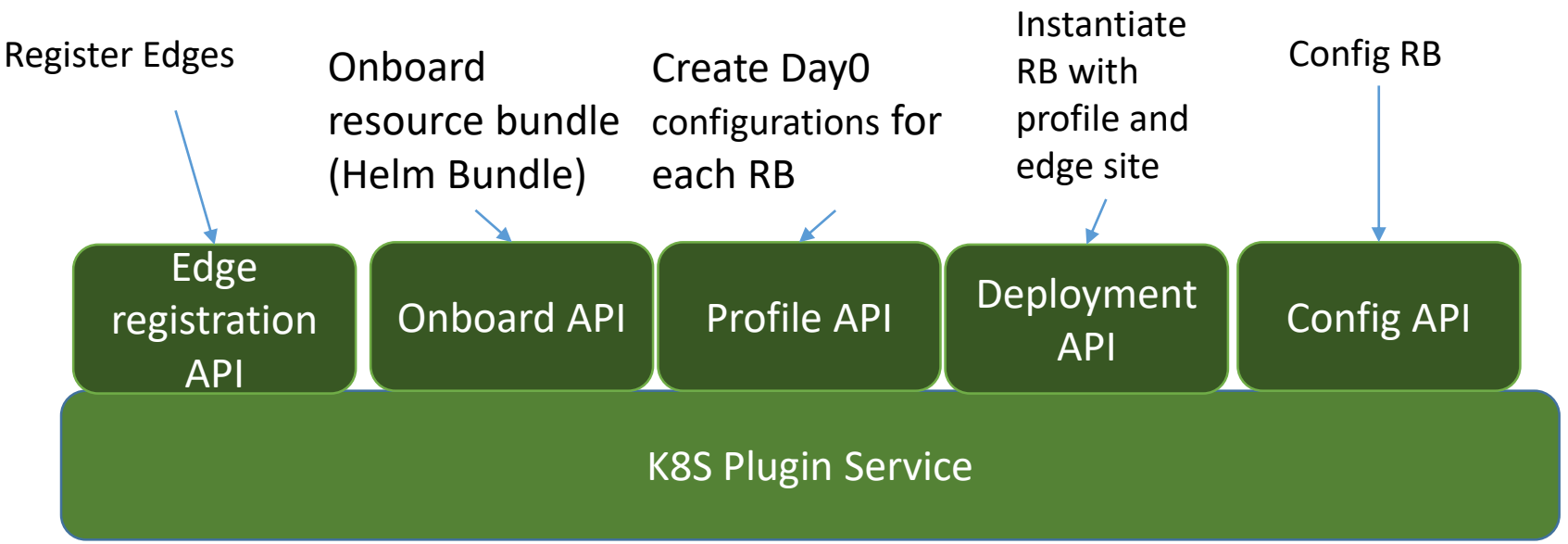
- Goals for R3 and R4
  - Support containerized workloads
  - Support containerized VNFs
  - Support both VMs and containers on same compute nodes. (Bare-metal deployment)
  - Support for multiple virtual networks
  - Support for dynamic creation of Virtual networks
  - Support public cloud CaaS such as AWS EKS, GCP GKE and Azure AKS (Only containers, not VMs)

ONAP – K8S Support

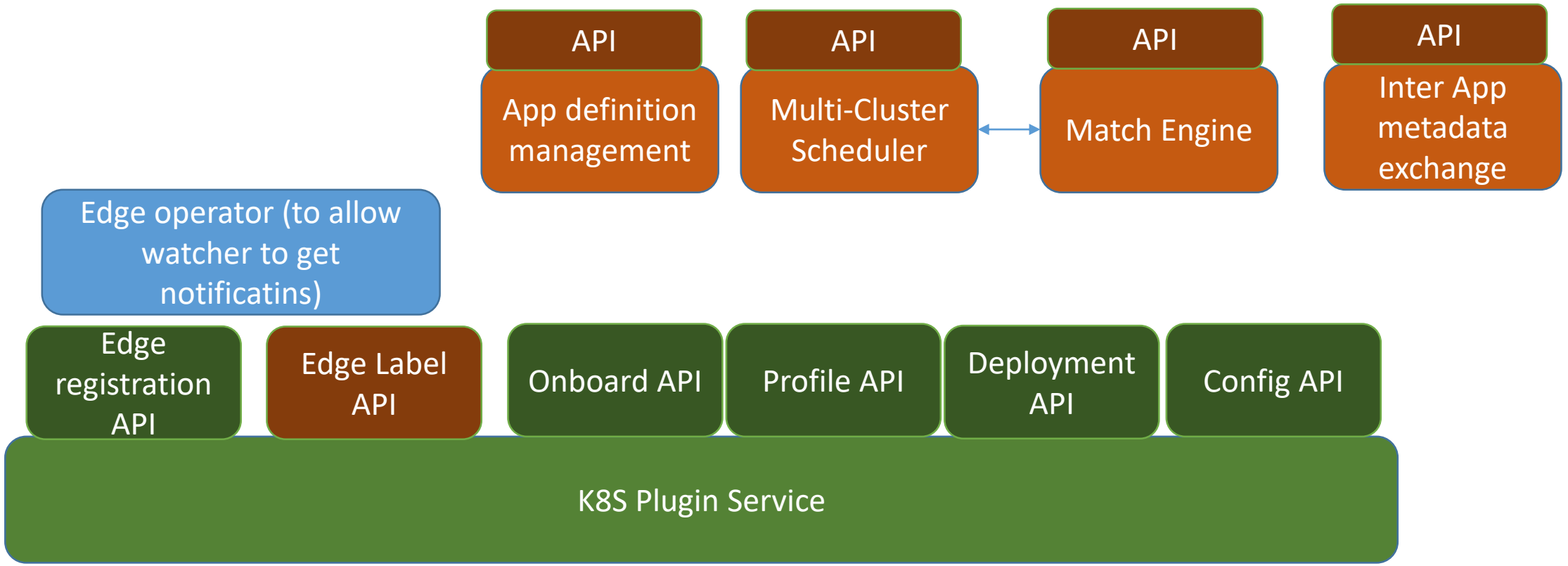


1. Uniform API across cloud technologies (HEAT, K8S, Azure etc..)
2. K8S Multi-Cloud Service plugin
   - Support for deployment and services.
   - K8S yaml artifacts
   - Networking – OVN, flannel and Multus (Create/Delete VNs, Distributed Router, Gateways, SNAT in Gateway)
3. Kubernetes Deployment
   - Installation of software & configuration to make K8S based sites.
   - Additional of virtlet, Multus, OVN and flannel.
4. K8S-OVN4NFV (OPNFV project, visualized as part of ONAP work)
   - Support for multiple virtual networks
   - Support for dynamic creation/deletion of virtual networks
5. Support for Control/Management/Data Plane workloads
   a. Flannel or any for default Kubernetes networking
   b. L7 Application connectivity : ISTIO with Envoy for E-W, Ambassador (as L7GW) with MetalLB with BGP for N-S
   c. OVN for data plane networking

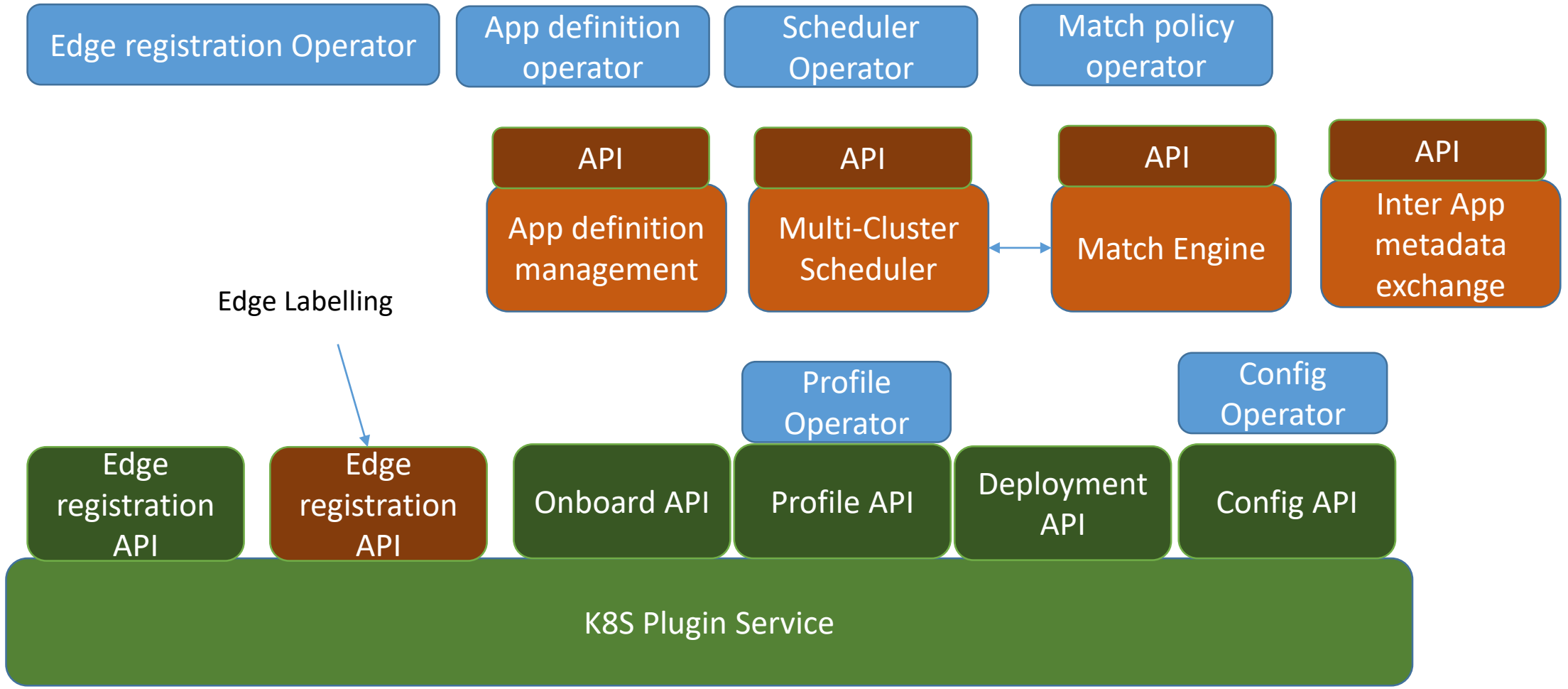
K8S Plugin Service as independent manager – Modular design



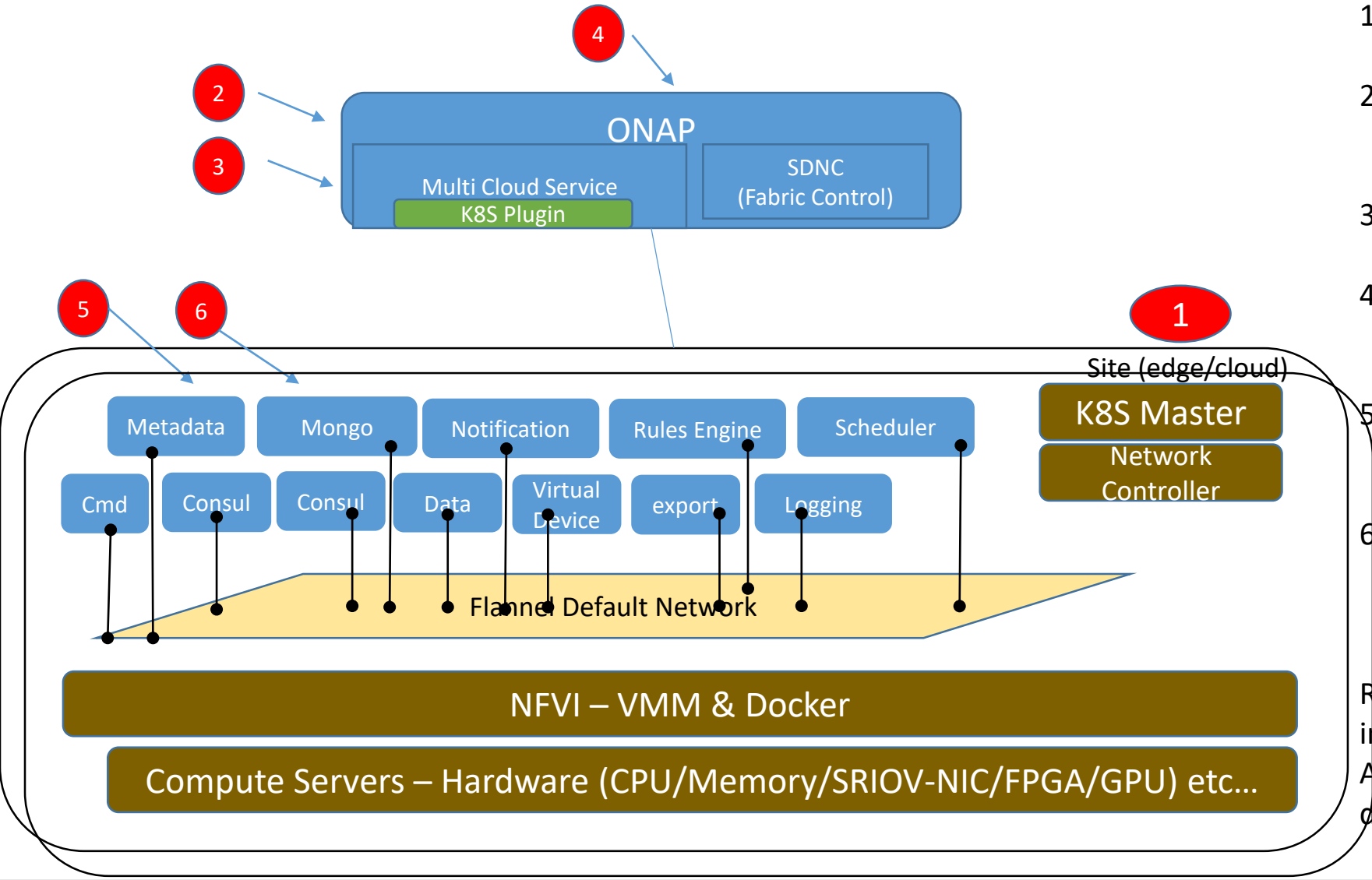
K8S based Cloud region support- Enhancements (R5+)



K8S based Cloud regions support – Enhance with Operators (R5++)



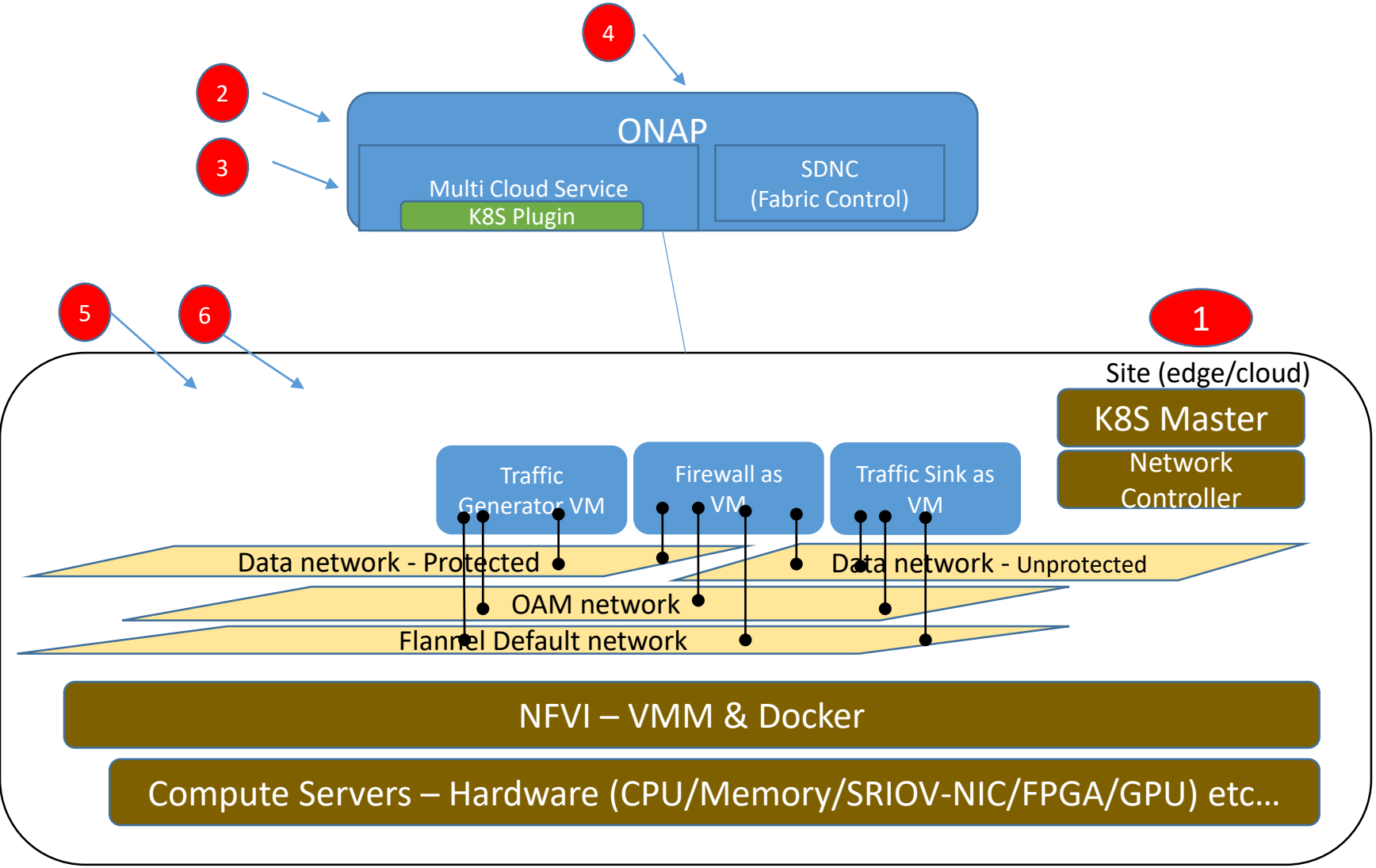
R4 Scenarios – EdgeX deployment



1. One time: Prepare K8S based site using KRD (if it does not exist)
2. One time: Register the K8S Site in ONAP by adding Kubeconfig file in ONAP (if the site is not added earilier)
3. EdgeX onboarding: EdgeX deployment and service helm charts in Multi-Cloud
4. Instantiate EdgeX (by calling Multi-Cloud Service API) via postman or via script (Multiple instances in various edge sites)
5. Check if all EdgeX containers are successful brought up on the site (using K8S utilities on the site)
6. Basic EdgeX testing to ensure that functionality also works
   - Use consul dashboard to check the services and their status

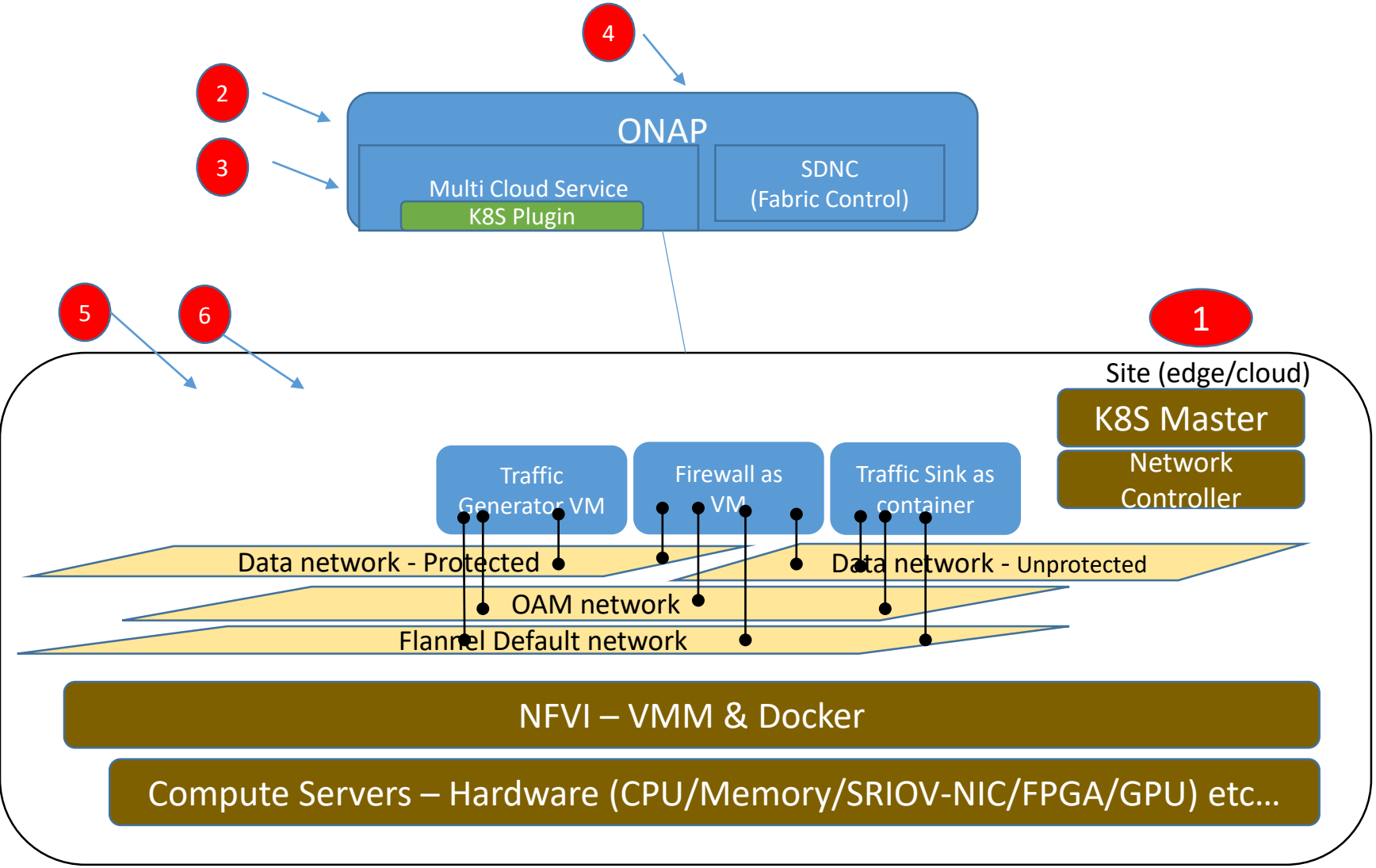
Repeat step 4 to 6 by bringing second instance of EdgeX on a different namespace. Also, work with Edgex team to automate deployment verification

vFirewall scenario (as VMs)



1. One time: Prepare K8S based site using KRD (if it does not exist)
2. One time: Register the K8S Site in ONAP by adding Kubeconfig file in ONAP (if the site is not added earilier)
3. vFirewall onboarding: Create deployment and service yaml files and put them in location expected by K8S plugin
4. Instantiate vFirewall (by calling Multi-Cloud Service API) via postman or via script
5. Check if all firewall containers are successful brought up on the site (using tools) and also ensure that three additional virtual networks are created. Also ensure that firewall belongs in all data networks. Ensure that generator and sink belong to different data networks.
6. Basic firewall testing to ensure that functionality also works
   - Check the sink dashboard to ensure that right packet streams are received by sink.

vFirewall scenario (as VMs and containers – Hybrid)

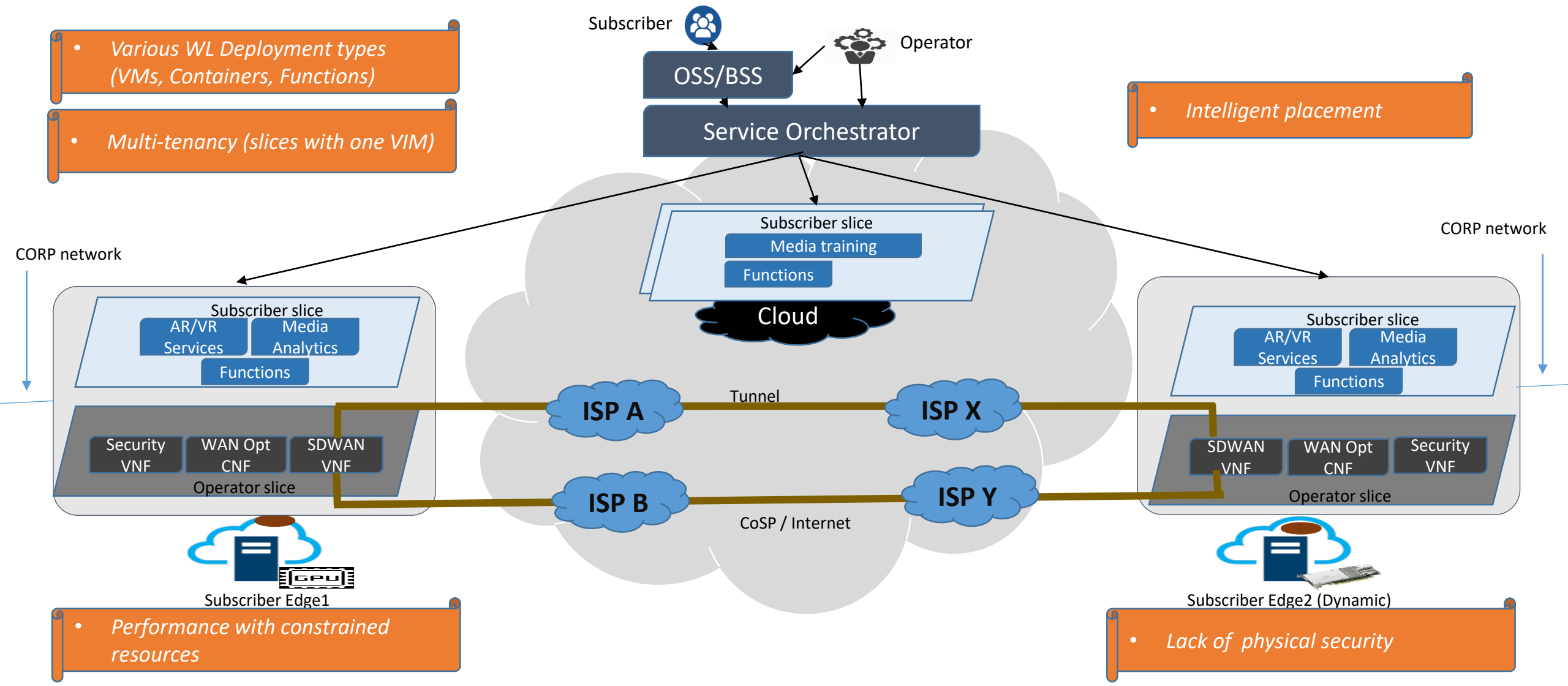


1. One time: Prepare K8S based site using KRD (if it does not exist)
2. One time: Register the K8S Site in ONAP by adding Kubeconfig file in ONAP (if the site is not added earilier)
3. vFirewall onboarding: Create deployment and service yaml file and put them in location expected by K8S plugin
4. Instantiate vFirewall (by calling Multi-Cloud Service API) via postman or via script
5. Check if firewall is successfully brought up on the site (using tools) and also ensure that three additional virtual networks are created. Also ensure that firewall belongs in all data networks. Ensure that generator and sink belong to different data networks.
6. Basic firewall testing to ensure that functionality also works
   - Check the sink dashboard to ensure that right packet streams are received by sink.

ONAP
OPEN NETWORK AUTOMATION PLATFORM

Provider & Multi-App Support

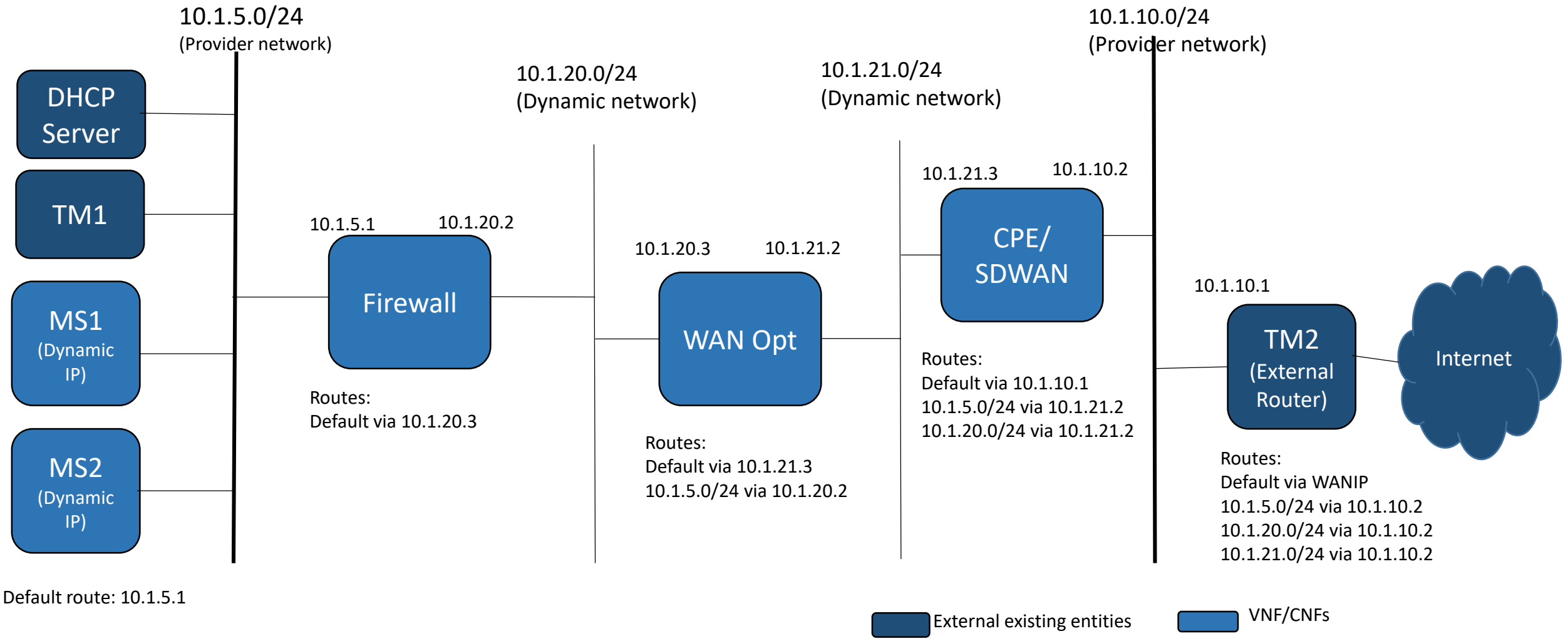
Next Generation Managed SDWAN



Context: Provider Networks and Kubernetes

- Provider networks are physical and/or VLAN networks that are part of data center. These are supposed to exist (or created) even before VIMs are used to create resources (such as VNFs, CNFs, dynamic networks etc…)
- In typical Kubernetes deployments, there is a never a need to place workloads on provider networks as many use cases and associated applications are endpoints.
- In case of Cloud native NFV:
  - Network functions are placed in the line of traffic – Security (firewall, IPS, WAF), Optimization (WAN Optimization, Transparent Caching), Probes & collectors and connectivity (WAN routers, SDWAN, UPF, etc…)
  - Network functions provide IP addresses and other network configuration (DHCP etc…)
- Provider network support is well taken care in Openstack, but not in Kubernetes
- Intel with guidance from few service providers is adding support for provider networks in Kubernetes based deployment (Starting with OVN as CNI)

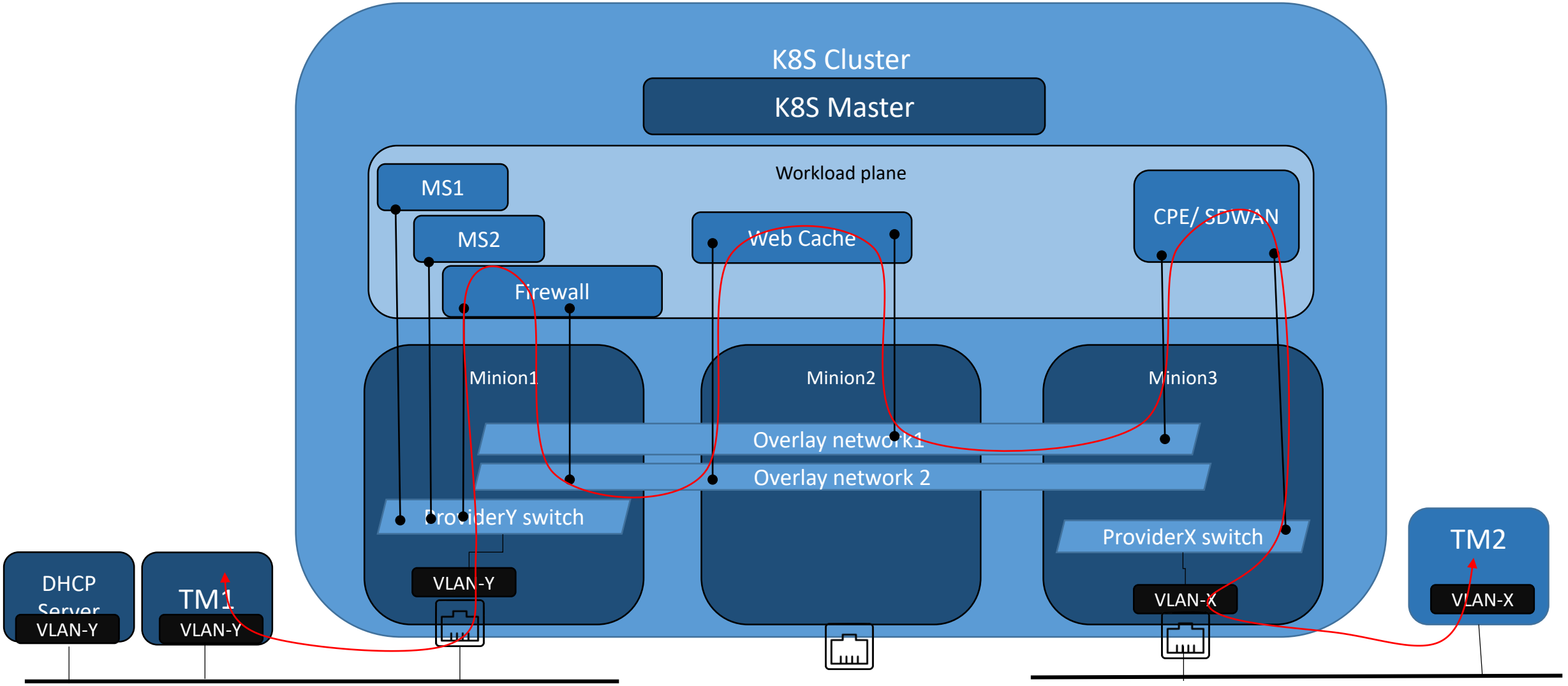
Test scenario – to comprehend multiple deployment variations



Deployment Variations related requirements

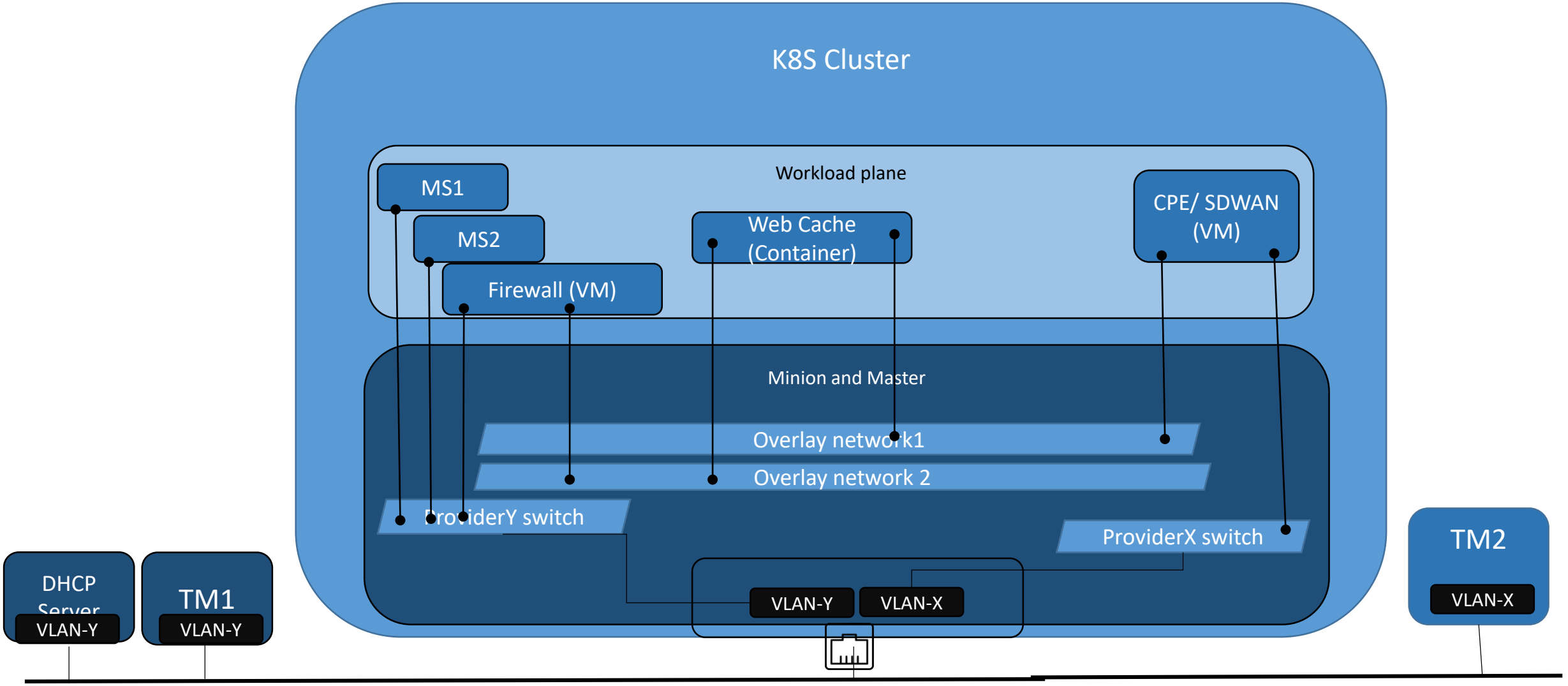
- Number of provider networks are not same across deployments
- Provider network IP addressing is different from deployment to deployment
- Some provider networks have DHCP Servers that provide IP addresses.
- Some customer micro services are expected to be part of one of provider networks
- Number of physical ports on the K8S cluster nodes can be less than the provider networks. Number of physical ports on K8S cluster nodes can vary across deployments.
- All K8S nodes might not be connected to all physical provider networks.
- Some deployments may require link aggregation across multiple ports for high availability and traffic scaling.
- VNFs/CNFs deployed are expected to provide security/network services.
- Each deployment might require multiple tiers (in previous example, there are three tiers – firewall, Cache and SDWAN. But different deployments might require more tiers or less tiers.
- Tiers can be static or even dynamic (adding of new tiers dynamically will be a requirement in future)

Use case scenario 1



All VNF/CNF are simulated using Ubuntu images as routers – They don't perform actual functionality of firewall, cache or SDWAN

Use case scenario 2



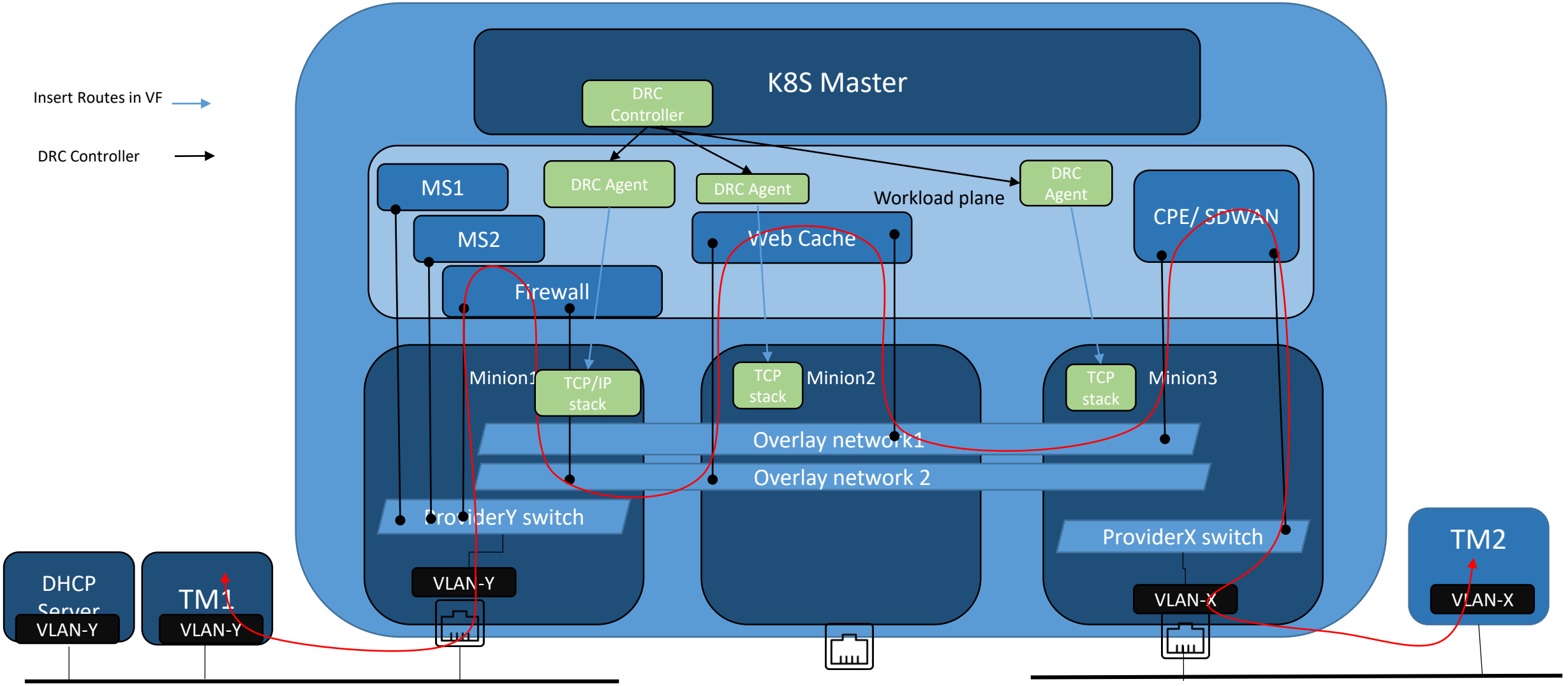
Provider networks – Dublin (R4) Scope

- All provider networks related configuration is done at the time of K8S deployment or right after K8S deployment before any VNF/CNFs are instantiated.
  - Ansible Framework used to run configuration script(s) on different cluster nodes based on the role of the node.
    - This script(s) is expected to be changed per deployment.
  - For Nodes with Provider network role the script will be responsible for
    - Creation of VLANs on various physical interfaces
    - Creation of link aggregation interfaces
    - Configuring OVN provider network support
- VNF/CNF tiers
  - All networks that constitute a chain to be created together.
  - All VNFs/CNFs that constitute a chain to be instantiated together with static IP addresses and static routes.
- Validation to ensure dynamic networks created don’t conflict with provider networks with respect to IP addresses

Provider networks – Roadmap (R5 and beyond)

- Provider network – Declarative configuration
- Dynamic Route operator
  - To allow dynamic insertion of tiers and services.
  - Implementation suggestions:
    - No expectation that there is NET_ADMIN privilege for VNFs.
    - Daemon set to create routes in various network namespaces on behalf of VNFs.
- Fix any issues related to learn routes by virtlet based VMs.

Use case scenario 1 - Dynamic Route configuration(DRC) support



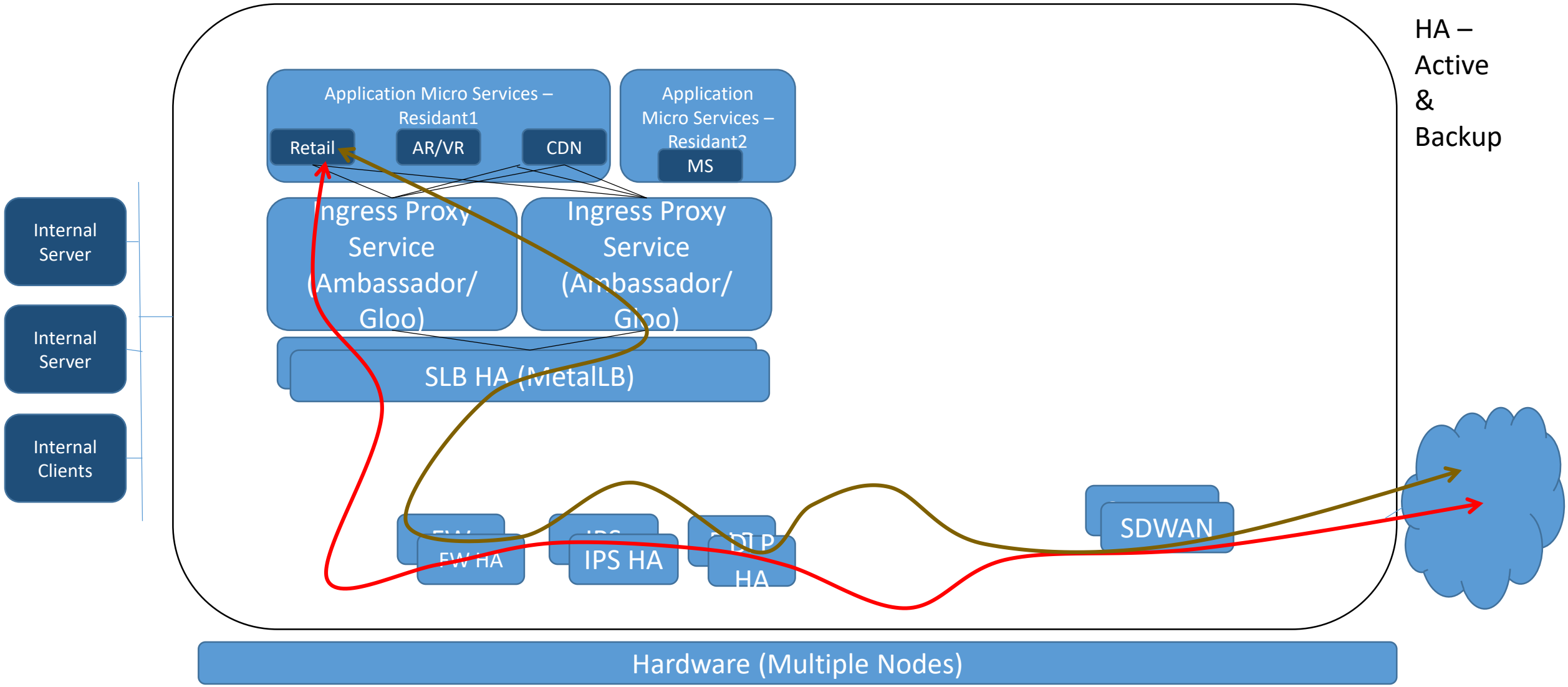
All VNF/CNF are simulated using Ubuntu images as routers – They don't perform actual functionality of firewall, cache or SDWAN

Other related roadmap items

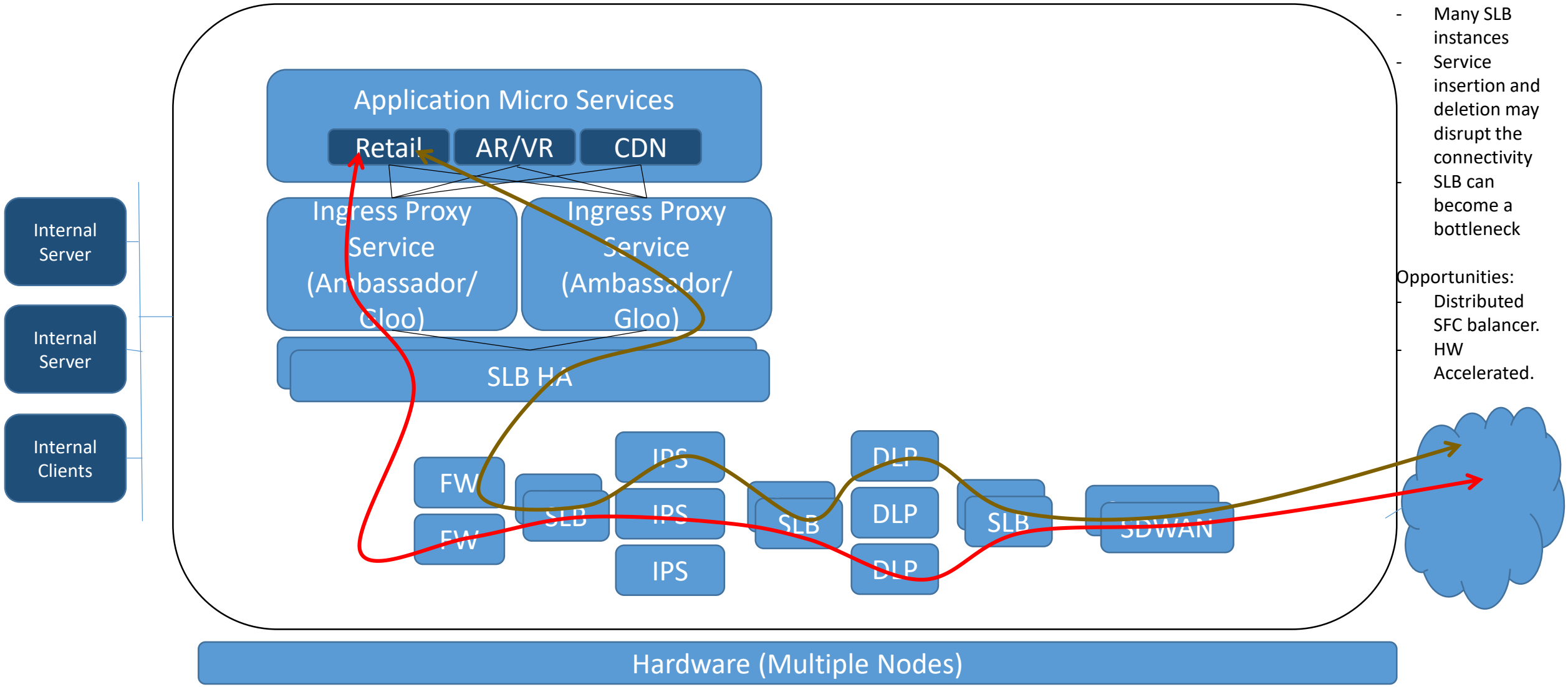
- OVN operator
  - Avoid OVN client library in K8S plugin of ONAP.
  - Ability to create OVN resources via K8S operator facilities
- Integration of digital rebar (or Ironic) for bare-metal provisioning.
- Improve the performance of KRD
- Separate image creation of various packages from deployment in KRD.

Cloud Native NFV Stack – Service Function Chaining (WIP in the community)

Service Functions – Scenario (Within a K8S cluster) – Immediate requirement



Service Functions – Scenario (Within a K8S cluster)



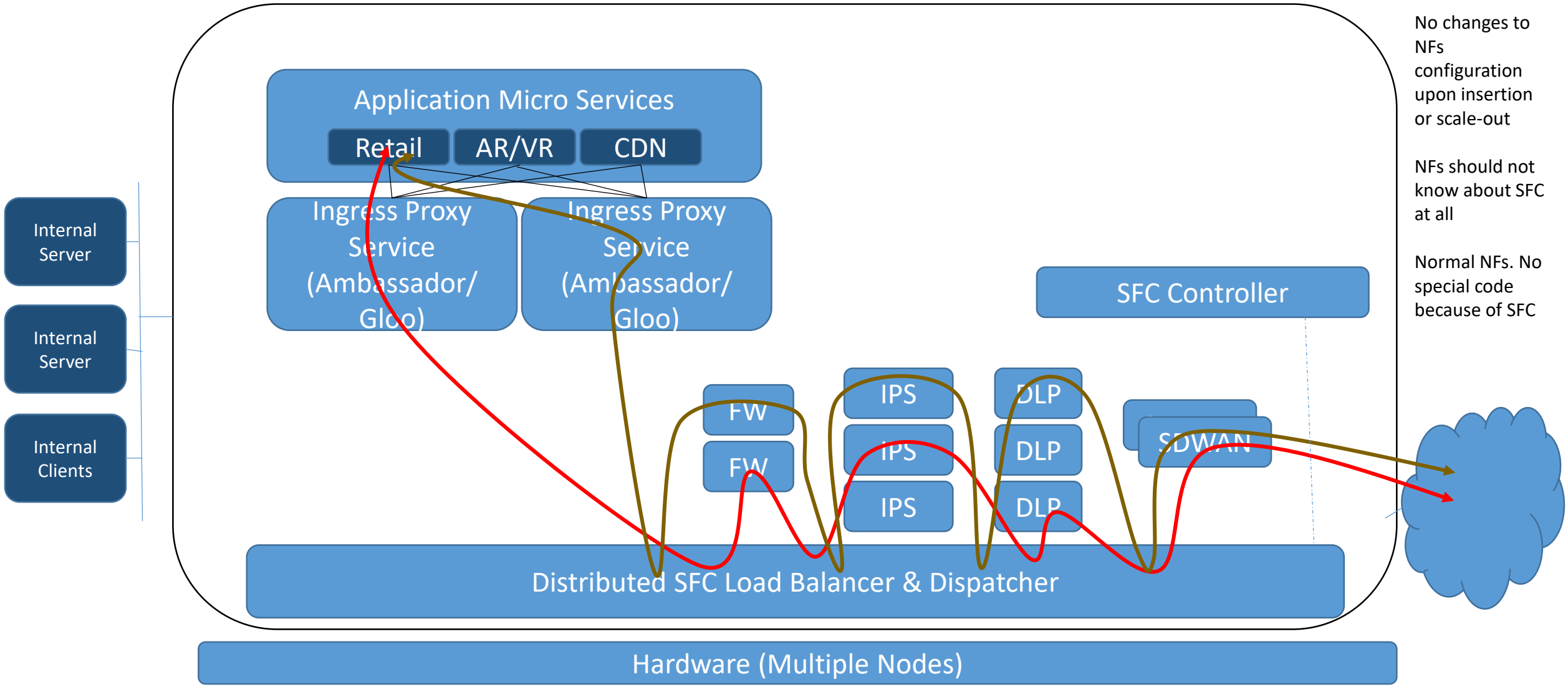
Challenges:

- Many SLB instances
- Service insertion and deletion may disrupt the connectivity
- SLB can become a bottleneck

Opportunities:

- Distributed SFC balancer.
- HW Accelerated.

Service Functions – Intended Scenario (Within a K8S cluster)

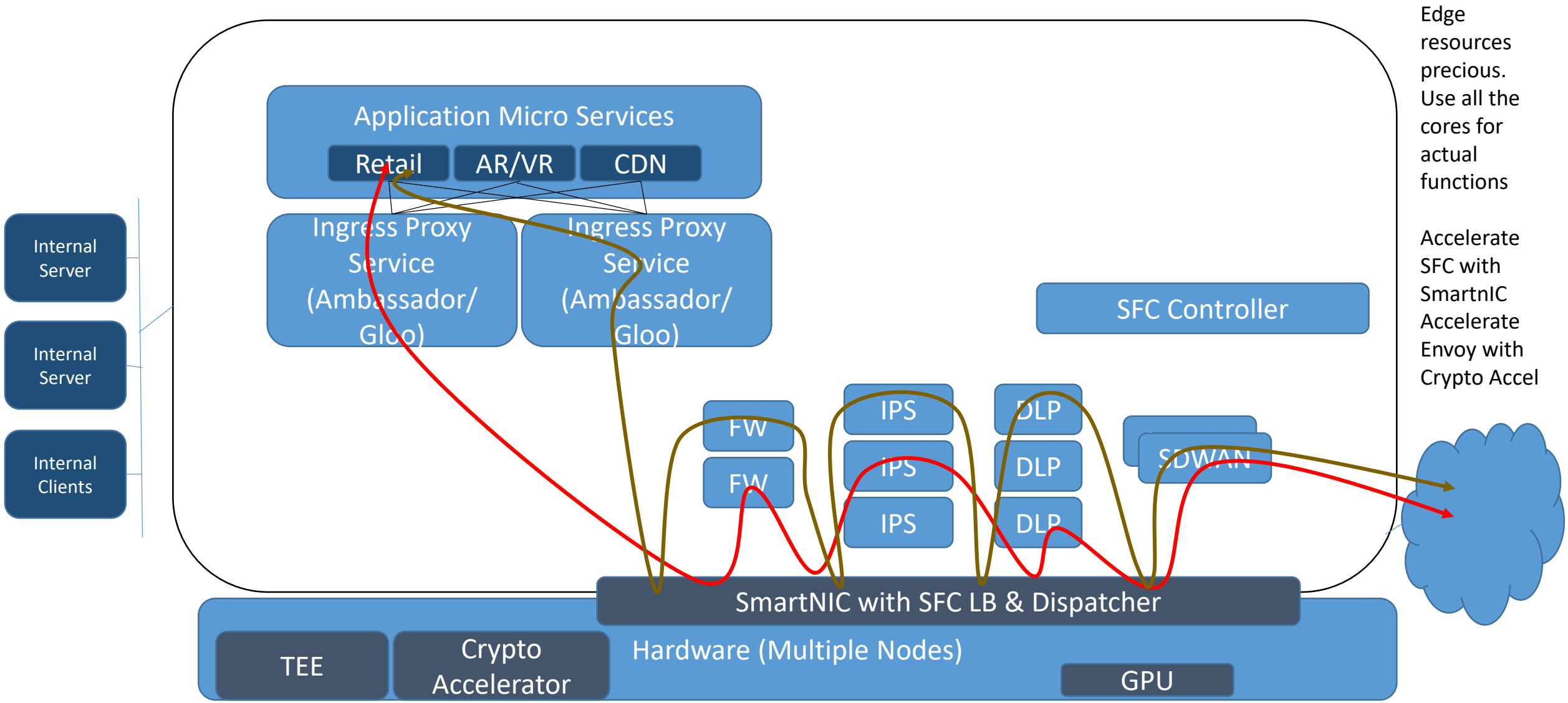


No changes to NFs configuration upon insertion or scale-out

NFs should not know about SFC at all

Normal NFs. No special code because of SFC

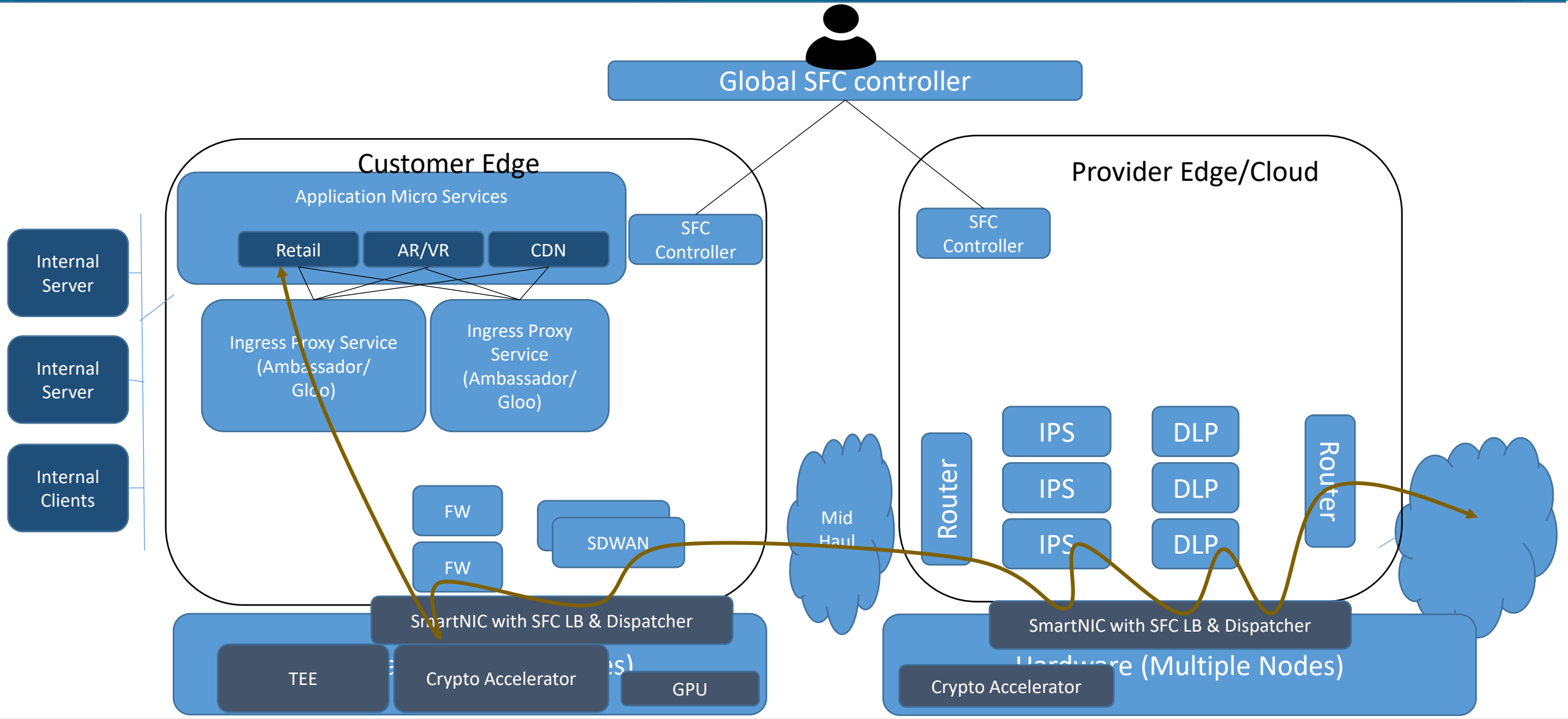
Service Functions – Hardware Accelerated(Within a K8S cluster)



Edge resources precious. Use all the cores for actual functions

Accelerate SFC with SmartnIC Accelerate Envoy with Crypto Accel

Service Functions – Multiple K8S Clusters



SFC requirements

- Multiple Service function chains
- Each service function to have multiple functions in sequence.
- SFs can be stateful or stateless
- Dynamic insertion of SFs without impacting existing flows.
- L3 SFs (L2SFs good to have)
- No changes to SF configuration when new SFs are introduced or when existing SFs scale-out or scale-in
- Chain to extend multiple K8S clusters
- Chain to be accelerated (No additional cores)
- Working with existing CNIs (OVN to start with)


ONAP
OPEN NETWORK AUTOMATION PLATFORM