Service Function Chaining Using OVN in Kubernetes

Contacts:
kuralamudhan.ramakrishnan@intel.com
todd.malsbary@intel.com
Agenda

• What is Kubernetes Network?
• Nodus Deep Dive
• Service Function chaining model
What is Kubernetes Networking?

- Kubernetes Network doesn’t handle network at the first place, it calls 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

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

Centralized computing to Geo distributed computing
How does NFV based deployment with Cloud-native applications look like (Taking SDWAN with security NFs as an example)
Nodus Features

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)

Provider network 1 (OVN using L2 breakout, OVN LB on L2 Switch)

SLB

NGFW

SDWAN CNF

Provider Network 2 (OVN)

Internet

Hardware (Multiple Nodes)

Dynamic virtual Networks
Provider networks
Multiple interfaces
Network function chaining
Network function load balancing

Finite network SRIOV Overlay networking
Smart NIC friendly & AF_XDP for packet processing NFs

No changes to NFs
No changes to Apps
Configuration via operators

Feature Reqmts

NODUS

Finite network SRIOV Overlay networking

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, Network Policy)
- SmartNIC friendly
Nodus Architecture blocks

**NFC 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.

**NFC 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

```
Master

kube_pod_subnet: 10.244.64.0/18

nodus-default-nw

10.233.64.6  10.233.64.7  10.233.64.8

nodus-node0     nodus-node1     nodus-node2

eth0

192.168.121.18

Minion-01

pod11

eth0

10.233.64.2

veth-p11

br-int

nodus-node1

10.233.64.7

Geneve

pod12

eth0

10.233.64.4

veth-p12

genev_sys_6081

Minion-02

pod21

eth0

10.233.64.5

veth-p21

br-int

nodus-node2

10.233.64.8

pod22

eth0

10.233.64.3

veth-p22

genev_sys_6081

10.233.64.3

10.233.64.4

10.233.64.5

10.233.64.6

10.233.64.7

10.233.64.8

10.233.64.9

SNAT

192.168.121.2

192.168.121.18

192.168.121.28
```

- **External traffic**
- **Inter traffic**
- **Intra traffic**
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 Advanced SFC - using provider networks & one Virtual networks with pod labels

External existing entities  VNF/CNFs

Try it yourself:
Demo link
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
• Kubernetes Network policy based on OVN ACLs

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
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
• 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
SFC Model Demo in Kubernetes

Goal: Labels eliminates Pod annotations

Overview

Contacts: kuralamudhan.ramakrishnan@intel.com
todd.malsbary@intel.com
Nodus Demo - Traffic from pod within the cluster with sfc

172.30.10.0/24 (Left - Provider network)

172.30.10.3 172.30.33.2

SLB

172.30.33.0/24 (Dynamic network) dync-net1

172.30.33.3 172.30.44.2

NGFW

172.30.44.0/24 (Dynamic network) dync-net2

SDEWAN CNF

172.30.20.0/24 (Right - Provider network)

172.30.20.3

Default route: 172.30.20.2

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