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

Kubelet Network

CNI ARGS

CNI plugins

IP

Pod Network Namespace

eth0

IP

Pod Network Namespace

eth0

Node A
Pod Network in Kubernetes – example flannel

Node A

- Pod Network Namespace
- eth0
- veth-XXX
- cni0
- veth-YYYY
- eth0
- Pod Network Namespace
Pod Network in Kubernetes
• 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

- services
- Node A
- Nginx

Nginx

veth

eth0

cni0

veth-XXX

veth-YYYY

veth-YYYY

eth0

Nginx

Node A

Service IP

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
```
Pod Network in Kubernetes

Node A

Nginx

veth - XXX

eth0

veth - YYYY

eth0

veth - YYYY

eth0

Node A

Service IP

curl my-nginx.com

Kubernetes dns resolve it to Service IP

Load Balanced using Iptables/IPVS
Pod Network in Kubernetes

Node A

Nginx

veth-XXX

eth0

veth-YYYY

eth0

Nginx

Service IP

Kubernetes dns resolve it to Service IP

Load Balanced using Iptables/IPVS

curl my-nginx.com
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)
What is Nodus?

Feature Reqmts
- Dynamic virtual Networks
- Provider networks
- Multiple interfaces
- Network function chaining

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

Master

- **kube_pod_subnet:** 10.244.64.0/18
- **nodus-default-nw**
  - **nodus-node0:** 10.233.64.6
  - **nodus-node1:** 10.233.64.7
  - **nodus-node2:** 10.233.64.8

- eth0
  - 192.168.121.18

Minion-01

- pod11
  - eth0: 10.233.64.2
- pod12
  - eth0: 10.233.64.4
- nodus-node1
  - br-int
- veth-p11
- veth-p12

Minion-02

- pod22
  - eth0: 10.233.64.3
- pod21
  - eth0: 10.233.64.5
- nodus-node2
  - br-int
- veth-p22
- genev_sys_6081

Geneve

- 10.233.64.7
- 10.233.64.8
- 10.233.64.3
- 10.233.64.5

Geneve

- pod22
  - veth-p22
  - genev_sys_6081

- pod21
  - veth-p21
  - genev_sys_6081

- eth0: 192.168.121.2
- 192.168.121.18

External traffic

Inter traffic

Intra traffic
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

Revisited standards -
Draft_Kubernetes_Software_defined_Network_Custom_Resource_Definition_De-facto_Standard_out_of_CNI_scope
Pod Annotation

```yaml
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](https://example.com)
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 -

Network Chaining CR

apiVersion: k8s.plugin.opnfv.org/v1alpha1
kind: NetworkChaining
metadata:
  name: example-networkchaining
spec:
  chainType: "Routing"
  routingSpec:
    namespace: "default"
    networkChain: "net=virtual-net1,app=slb,net=dyneq-net1,app=ngfw,net=dyneq-net2,app=sdewan,net=virtual-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
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

Try it yourself
Demo link
https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/nodus-primary-sfc-setup
Nodus Demo - Traffic from external entities – Firewall icmp reject

172.30.10/24  
(Left - Provider network)

172.30.10.3  
DHCP Server

172.30.33.2  
SLB

172.30.30.20/24  
(Right - Provider network)

172.30.30.44/24  
(Dynamic network)

dync-net1

172.30.33.0/24  
(Dynamic network)

dync-net2

172.30.33.3  
NGFW

172.30.44.3  
SDEWAN CNF

172.30.20.2  
Default route

172.30.10.101  
Try it yourself

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

172.30.33.2  
172.30.33.3  
172.30.44.2  
172.30.44.3  
172.30.20.3

External existing entities

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

Try it yourself:
Demo link

External existing entities
VNF/CNFs

Inter traffic with SFC
External traffic with SFC
Inter traffic without SFC
Nodus Advanced SFC - using Dynamic Network creation

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
• 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
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