OVN for Network functions with K8s

Srinivasa Addepalli, Ritu Sood - Intel
• Why network functions in Edge & K8s clusters?
• Edge-computing scenario to describe the K8s networking requirements
• Networking requirements
• OVN-for-K8s-NFV architecture blocks
• OVN-for-K8s-NFV details
• Current Status and roadmap
• Q&A
Application Transformation
(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
Current/In-progress Edge computing deployments: Multi-Cloud and Multi-Edge

Appx Manager
App Orchestrator – Simple (git based)
VNF Orchestrator
ADC Controller
WAF Controller
SDWAN Controller

Compute cluster (Edge computing)
SW Platform
HW

SDWAN & Security
SLB
WAF
SDWAN

WP
POD
POD
App1
App2

SDWAN Controller
SDWAN
WAF
SLB

ingress
ingress
ingress

SDWAN Controller
SDWAN
WAF
SLB

ingress
ingress
ingress

Compute Cluster (Edge computing)
SW Platform
HW

Internal machines
Edge/private-cloud/VPC

Internal machines
Edge/private-cloud/VPC
Challenge: Under utilization of resources

Compute nodes are divided for VNFs and Applications - Challenges in allocations and under utilization of resources.
Challenge: Multiple Site level orchestrators leading to wasting of resources

Multiple site level orchestrators (Openstack for VNFs and Kubernetes for applications)

Wasting resources and higher maintenance
Ask: All in One

Compute cluster (Edge computing)

Internal machines

Edge/private-cloud/VPC

Secure WAN controller

App1 Manager

ONAP4K8S

App1

POD
SC
POD
SC
App2

POD
SC
POD
SC

SW Platform
HW

SLB
NGFW
SDWAN

Ingress

Internal machines

Edge/private-cloud/VPC
How does NFV based deployment with Cloud-native applications look like (Taking SDWAN with security NFs as an example)
Networking Requirements

- **Corp networks**
  - M1
  - M2
  - M3

- **Networks**
  - **K8S Cluster**
    - **K8S Master**
    - **resident 1 Applications (Micro-Services)**
      - PCD
      - 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)**
    - **Provider network 2 (OVN)**
    - **SLB**
    - **NGFW**
    - **SDWAN CNF**

- **Internet**

- **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
  - Cloud Native (No SRIOV requirement)
  - Smart NIC friendly & AF_XDP for packet processing NFs
Why did we choose OVN in Akraino Edge platform?

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

Participants: Intel, Verizon, VMware, F5
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

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

- CNI (ovn4k8s-cni) Works with Multus
- Assumes primary/first interface provided by another CNI
- Supports Static IP addresses
Use case using OVN dynamic networks
Test scenario – to comprehend multiple deployment variations

Routing details:

**Provider network**:
- 10.1.5.0/24
- 10.1.10.0/24

**Dynamic network**:
- 10.1.20.0/24
- 10.1.21.0/24

**Internal Networks**:
- **DHCP Server**: 10.1.5.1
- **SLB**: 10.1.20.2
- **NGFW**: 10.1.21.2
- **SDWAN CNF**: 10.1.20.3

**Routes**:
- Default via 10.1.5.1
- Default via 10.1.20.3
- Default via 10.1.21.3
- Default via 10.1.10.1
- Default via WANIP

**External Routers**:
- **TM1 (External Router)**
  - 10.1.20.0/24
- **TM2 (External Router)**
  - 10.1.21.0/24
  - 10.1.10.0/24

**Internal Services**:
- **TM1**: 10.1.5.1
- **MS1 (Dynamic IP)**: 10.1.20.2
- **MS2 (Dynamic IP)**: 10.1.21.2

**Dynamic IPs**:
- MS1: 10.1.1.0/24
- MS2: 10.1.2.0/24
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
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" }
  ]}'
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
Status

Current
- Dynamic Network Creation
- Attaching multiple interfaces to Pods
- Multus integration
- Support for Virtlet with Multus
- Provider Network Support – Controller and Agent

Link to Repo: https://github.com/opnfv/ovn4nfv-k8s-plugin

WIP
- Chaining Controller
• Enabling SDWAN use case
  • Support for primary interface with NFN Operator and CNI
  • One switch for the whole deployment
• Network Policy Support
• LB support for NF elasticity
• Integrate with Kubevirt
• Proxy less service mesh with OVN & IPsec in network namespace