Open Source MANO

Network Functions & Network Services Modeling
Gerardo García (Telefónica)
Network Function Virtualization provides a mean to make the network more flexible by minimizing dependence on HW constraints

Network functions are fully defined by SW, minimizing dependence on HW constraints

- **Traditional Network Model: APPLIANCE APPROACH**
  - DPI
  - BRAS
  - GGSN/SGSN
  - PE Router
  - Firewall
  - CG-NAT
  - Session Border Controller
  - STB

  - Network functionalities are **based on specific HW with specific SW linked to HW vendors**
  - One physical node per role

- **Virtualised Network Model: VIRTUAL APPLIANCE APPROACH**
  - DPI
  - BRAS
  - CG-NAT
  - GGSN/SGSN
  - Firewall
  - PE Router
  - Virtualized appliances

  - Network functionalities are **SW-based over COTS HW**
  - Multiple roles over same HW

Virtualised Network Model:
- ORCHESTRATED, AUTOMATIC & REMOTE INSTALL
- STANDARD HIGH VOLUME SERVERS & SWITCHES

© ETSI 2021
Network Functions

• One or many nodes in a Network Infrastructure that has well defined interfaces and functional networking capability.

• Examples: Firewall, Router, EPC, IMS, etc.

• Different types of Network Functions
  • Virtual Network Function
  • Cloud Native/Container Network Function
  • Physical Network Function
  • Hybrid Network Function
Network Functions can be composed of VMs, containers and/or physical elements.

- a) All VMs (VDU)
  - VM
  - VNF
  - VM

- b) All Containers (KDU)
  - K8s
  - Container Network Function (CNF)
  - VM

- c) All Physical (PDU)
  - Phy
  - PNF

- d) Hybrid cases (VDU, KDU and/or PDU)
  - K8s
  - Hybrid Network Function (HNF)
  - VM
Network Service

- The Network Service (NS) is a topology of interconnected NF
  - ABSTRACTION (NS definition): The topology is agnostic from the place where NF will be deployed
  - PARTICULARIZATION (NS instance): When instantiating it, parameters are provided specific for those NF instances

- It is deployed and operated as a whole
- Examples: LTE, VPN, LAN internet, etc.
Relation to ETSI NFV
Where does OSM fit in ETSI NFV architecture?

We are here!
Open Source MANO is an ETSI-hosted project to develop an Open Source NFV Management and Orchestration (MANO) software stack aligned with ETSI NFV.
OSM and NFV are different organizations in ETSI, and they complement each other

ETSI NFV: Industry Specification Group that elaborates specifications on Network Functions Virtualization

ETSI OSM: Open Source Group developing a Management and Orchestration (MANO) stack aligned with ETSI NFV Architectural Framework and Information Models
ETSI NFV architecture and specifications
All you need is a map

Source: ETSI. https://www.etsi.org/images/articles/NFV%20Architecture.svg

IFA (stage 2 specifications): development of architecture, interfaces and information model aspects
SOL (stage 3 specifications): specification of the implementable protocol and data model solutions

Os-Ma-Nfvo reference point (interface between OSS/BSS and NFVO)
Ve-Vnfm-em/vnf reference points (interface between VNF and EM/VNF)
VNF and NS descriptors and packages
OSM
Model-driven NaaS platform
OSM provides a platform to create Networks as a Service (NaaS) and to manage them conveniently.

- **Simplification**
- **Reusability**
- **Agility**

**Abstraction through Layering**
- Simplification
- Reusability
- Agility

**HIGHER-LEVEL PRIMITIVES**
- Add subscriber
- Add service profile
- Update subscriber profile
- Add service access to subscriber

**SERVICE ORCHESTRATION**
- NS LCM
- NSSlice LCM

**Virtual Domain**
- VNFs, PNFs, HNFs

**SDN & Transport Domain**
- Or-Vi (IaaS API)
- Or-Wi (TaaS API)

**OSS/BSS**
- SOL005 + NS LCM calls + Network Slice calls

**VIM Scope**
- Or-Vi (IaaS API)
- Or-Wi (TaaS API)

**OSM Scope**
- NS/NSI LCM + VNF LCM

© ETSI 2021
Packages embed resource description and operational procedures
Packages embed resource description and operational procedures
All in OSM is model-driven to make VNFs and NS as portable and reusable as possible

- **Provided by the vendor**, fully describe their own product:
  - Topology
  - Parametrized
  - Actions for Day-0, Day-1, and Day-2
- **Doesn’t** need to know any detail about:
  - The target infrastructure
  - Other components that will be part of the scenario
All in OSM is model-driven to make VNFs and NS as portable and reusable as possible.

(V)NF PACKAGES:

- Describes how to combine a set of NF packages to create a specific scenario.
- Parametrized.
- Have actions for Day-0, Day-1, and Day-2.

NS PACKAGES / SLICE PACKAGES:

(*) NS instances play the role of Slice Subnets of a given slice. Some of them may be shared by more than one slice instance. This is taken into account by OSM, so a slice is more sophisticated than just a “NS of NS”.

Slice Packages work similarly, but using NS as building blocks(*)
All in OSM is model-driven to make VNFs and NS as portable and reusable as possible

NS PACKAGES / SLICE PACKAGES:

Upon instantiation, you just need to decide:
- The target VIM (or VIMs)
- Values for the parameters (IP addresses, keys, etc.)
OSM Information Model
What is OSM IM?

• Information model (IM) to define the different descriptor templates:
  • Network Function (NF)
  • Network Service (NS)
  • Network Slice (NST)

• OSM IM is based on YANG*.

• OSM IM aligned with ETSI NFV, derived from SOL006 (which, in turn, derives from IFA011 and IFA014).
  • IFA011 describes the VNF descriptor specification whereas IFA014 on NS descriptor.

• SOL006 alignment since OSM Release NINE (now available!)

OSM IM and SOL006

- OSM IM augments ETSI NFV SOL006 by adding the support of PDU, the support of Kubernetes applications, Network Slices, day-1 and day-2 primitives at VNF and NS level, and enhancements for dataplane workloads

- OSM IM:
  - User guide: https://osm.etsi.org/docs/user-guide/11-osm-im.html
  - OSM repo (Gitlab mirror): https://osm.etsi.org/gitlab/osm/im

- SOL006:
  - ETSI Gitlab: https://forge.etsi.org/rep/nfv/SOL006

- More details in next session
We need to differentiate between different stages

**Design Time**
- Building NF and NS descriptors by referencing the OSM Information Model.
- Building Day-1/Day-2 logic to complete the packages
- Testing your packages

**Provisioning Time**
- Onboarding: package validation and uploading

**Run Time (see your VNF and NS in action)**
- Instantiation (Deployment + Day 0 + Day 1). Here is where the operator decides:
  - In which VIM to deploy
  - What instantiation parameters to provide (specific IP addresses, configuration params)
- Operation phase (Day 2)
Network Function and Network Service Modeling
xNF are modeled with a NF package, which consists of:

- **Descriptor:**
- **Other files:**
  - Charms
  - Helm-charts
  - KDU objects (juju bundle, helm chart)
  - Day-0 configuration files (i.e. cloudinit)
  - Checksums file

```
hackfest_virtual-pc_vnfd/
├── charms
│   └── virtual-pc
├── cloud_init
│   └── virtual-pc_init
├── checksums.txt
└── virtual-pc_vnfd.yaml```
Modeling NF
NF package vs NF descriptor

VNF package
- VNFD
- VNF artifacts
- Additional metadata?

Resource description aspects
- VNF resource orchestration info (EPA resources and internal connectivity)

Management procedures
- VNF primitives
  - Day-1
  - Day-2
- Charms/EE
- Additional info
  - Icon, README, etc.

Additional metadata?
- VDU: dataVM
  - Image name: US/1604
  - VM Flavor: 1 CPU, 1GB RAM, 10 GB disk
  - Interfaces:
    - dataVM-eth0: VIRTIO
    - dataVM-eth1: VIRTIO
- VDU: mgmtVM
  - Image name: US/1604
  - VM Flavor: 1 CPU, 1GB RAM, 10 GB disk
  - Interfaces:
    - mgmtVM-eth0: VIRTIO
    - mgmtVM-eth1: VIRTIO

External Connection point: vnf-mgmt
External Connection point: vnf-data
Descriptors are written in YAML and contain:

- Topology description (VDU, internal VLD, Connection Points)
- Scaling-groups
- Monitoring params
- Reference to day-0 configuration file
- Execution environment list (e.g. charms, monitoring environments)
- Day-1 primitives (sequence)
- Day-2 primitives

```yaml
vnfd:
  description: Virtual Desktop Computer
  ext-cpd:
    - id: virtual-pc-private-ext
      int-cpd:
        cpd: eth0-int
        vdu-id: virtual-pc
        id: hackfest_virtual-pc_vnf
        mgmt-cp: virtual-pc-mgmt-ext
        product-name: hackfest_virtual-pc_vnf
  sw-image-desc:
    - id: ubuntu20.04
      image: ubuntu20.04
  vdu:
    - cloud-init-file: virtual-pc_init
description: virtual-pc
  id: virtual-pc
  int-cpd:
    - id: eth0-int
      virtual-network-interface-requirement:
        - name: eth0
          virtual-interface:
            type: PARAVIRT
        - id: eth1-int
          version: '1.0'
  virtual-compute-desc:
    - id: virtual-pc-vdu-compute
      virtual-cpu:
        num-virtual-cpu: 8
        virtual-memory:
        size: 32.0
      virtual-storage-desc:
        - id: virtual-pc-vdu-storage
```
Modeling NF
CNF descriptor

CNF descriptors must contain:

- List of KDU (and their associated helm-chart or juju-bundle)
- K8s cluster requirements

vnfd:
  description: CNF with single KDU
df:
  - id: default-df
  ext-cpd:
    - id: mgmt-ext
    k8s-cluster-net: mgmtnet
  id: openldap_knf
k8s-cluster:
  nets:
    - id: mgmtnet
kdu:
  - name: ldap
    helm-chart: stable/openldap
    mgmt-cp: mgmt-ext
    product-name: openldap_knf
    provider: Telefonica
    version: '1.0'
Modeling NF

PNF descriptor

PNF descriptors must contain:

- List of PDU (VDU with a pdu-type)

Available PDU must be registered to OSM upfront, since they are physical elements already available in the network.

vnfd:
  description: PNF entry for a firewall router
df:
  - id: default-df
    instantiation-level:
      - id: default-instantiation-level
        vdu-level:
          - number-of-instances: 1
          - vdu-id: vyos-VM
          - vdu-profile:
            - id: vyos-VM
              min-number-of-instances: 1
    ext-cpd:
      - id: gateway_public
        int-cpd:
          cpd: gateway_public
          vdu-id: vyos-VM
    - id: vnf_internal
      int-cpd:
        cpd: vnf_internal
        vdu-id: vyos-VM
  id: hackfest_firewall_pnf
  mgmt-cp: gateway_public
  product-name: hackfest_firewall_pnf

vdu:
  - id: vyos-VM
    pdu-type: gateway
    int-cpd:
      - id: gateway_public
        virtual-network-interface-requirement:
          - name: gateway_public
    - id: vnf_internal
      virtual-network-interface-requirement:
        - name: vnf_internal
  name: vyos-VM
Modeling NS
NS package

Network Services are modeled with a NS package, which consists of:

- Descriptor:
- Other files:
  - Charms
  - Checksums file

hackfest_virtual-pc_ns
├── README.md
├── checksums.txt
└── hackfest_virtual-pc_nsd.yaml
Modeling NS
NS package vs NS descriptor

- Resource description aspects
  - NS topology
- Management procedures
  - NS primitives
  - Day-1
  - Day-2
- Charms
- Additional info
  - Icon, README, etc.
Descriptors are written in YAML and contain:

- Topology description (NF, VL)
- Execution environment list (e.g. charms)
- Day-1 primitives (sequence)
- Day-2 primitives
NST are written in YAML and contain:

- List of netslice subnets
  - References to Network Services
  - Whether they are shared or not
- List of netslice vlds (networks interconnecting the subnets)

```
nst:
  - id: slice_basic NST
    name: slice_basic NST
    netslice-subnet:
      - id: slice_basic_nsd_1
        is-shared-nss: false
description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
        nsd-ref: slice_basic_ns
      - id: slice_basic_nsd_2
        is-shared-nss: true
description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
        nsd-ref: slice_basic_middle_ns
    netslice-vld:
      - id: slice_vld_mgmt
        name: slice_vld_mgmt
type: ELAN
        mgmt-network: true
        nss-connection-point-ref:
          - nss-ref: slice_basic_nsd_1
          nsd-connection-point-ref: nsd_cp_mgmt
          - nss-ref: slice_basic_nsd_2
          nsd-connection-point-ref: nsd_cp_mgmt
      - id: slice_vld_data1
        name: slice_vld_data1
type: ELAN
        nss-connection-point-ref:
          - nss-ref: slice_basic_nsd_1
          nsd-connection-point-ref: nsd_cp_data
          - nss-ref: slice_basic_nsd_2
          nsd-connection-point-ref: nsd_cp_data
```
Thank you

Find us at: osm.etsi.org osm.etsi.org/docs/user-guide opensourcemanono.slack.com