Making the Open Orchestration happen

Session: Fostering Creativity in Network Space: Open Orchestration for SDN and NFV

Eduardo Jacob <Eduardo.Jacob@ehu.eus>

TNC17 - Linz
• Introduction
• OSM Concepts
• Setup
• Demo
• We had our own research infrastructure (EHU-OEF)
  – The University of the Basque Country OpenFlow Enabled Facility
  – It started as a tool for helping us to run our research experiments
    • Involving our own IP addressing
    • Involving many VLANs (already used by our IT department)
    • Involving several experiments at once.
    • Maintaining both isolation and punctual connections to external networks.
  – OpenFlow seemed the obvious (and trendy) technology.
  – We developed our own Layer 2 virtualization solution (Layer 2 Prefix Based Network Virtualization L2PNV) and a solution to enroll in a slice after a successful AAA process (FlowNAC)
• EHU-OEF slicing
The EHU-OEF: An OpenFlow-based Layer-2 experimental facility
LIST OF SLICES AVAILABLE

<table>
<thead>
<tr>
<th>Slice Prefix</th>
<th>Name</th>
<th>Controller IP</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>02:00:00</td>
<td>DemoFIA</td>
<td>pox001.ehuoei2t.ehu.es</td>
<td>6633</td>
</tr>
<tr>
<td>02:00:01</td>
<td>Traffic Steering</td>
<td>ryu001.ehuoei2t.ehu.es</td>
<td>6633</td>
</tr>
<tr>
<td>02:00:02</td>
<td>DOCSIS integration</td>
<td>10.98.100.123</td>
<td>6633</td>
</tr>
</tbody>
</table>

ADD NEW SLICE

Slice Name
- Auto [ ]
- User-defined [ ]

Controller IP
- Auto: POX [ ]
- User-defined: URL [ ]

Port [ ]

Create  Reset

Creation may take up to 1 minute
• We soon realized we needed a way to deploy computing resources:
  – We developed our own solution ELwUD.
  – EHU-OEF’s Lightweight Unified Domain. Unified referred to fact that we could use x86 boxes both as computing and networking resources with good performance.
  – Based on libvirt.
  – Used the concept of Service Graph (similar to Network Service, in ETSI terminology).
  – It was based on a recursive orchestration.
  – Used for research on SFC (Service Function Chaining) and Traffic Steering.
  – We later learnt that we had in fact developed a MANO and a VIM.
**Detail of Service Graph 795: Dynamic Internet Access / Tenant 131125**

<table>
<thead>
<tr>
<th>SG Pos.</th>
<th>NF Type</th>
<th>NF Description</th>
<th>NF KQI Desc.</th>
<th>NF KQI Value</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Container</td>
<td>SP Gateway</td>
<td>Number of Users</td>
<td>10000</td>
<td>Update</td>
</tr>
</tbody>
</table>

**Actions with Service Graph**
- **Started**
- **Stop**
- **Remove**

**Drawings of Service Graph 795: Dynamic Internet Access / Tenant 131125**

- Graph at SI/Ro
- Graph at Ca
- Graph at CPE
- Graph at EHU-OEF
- Graph at OPT
- Graph at UN

---

**Research Group**

TNC17 - Linz - Making the Open Orquestration happen
• We decided to stop working on ELwUD.
  – Too much effort.
• We needed a solution to manage as freely as possible the computing AND the networking.
  – Independence of VIM.
  – Independence from VNF format.
  – Ease the research:
    • Elevating the role of SDN in NFV.
    • Stateful processing in networking elements.
• To integrate the SDN BoD service DynPaC
  – This meant going ONOS…
• And to be as aligned as possible with Standards
  – With ETSI NFV ISG.
• **NFVO (NFV Orchestrator)**: Responsible of “on-boarding” new NS and VNF packages and NFVI resource orchestration through multiple VIMs. Instantiates, manages and deletes NSs.

• **VNFM (VNF Manager)**: Manages life cycle of VNFs that are under NFVO control, by sending adequate commands to the VIM

• **VIM (Virtualized Infrastructure Manager)**: Manages NFVI resources.
  – Maintains a record of virtual resources mapping to physical resources.
  – Manages the hardware and software resources of the NFVI

• **VNF (Virtualized Network Function)**: It can run over one or several VMs

• **NFVI (NFV Infrastructure)**: Hardware and Software resources over which VNFs are run.
Many choices

- Examples:
  - MANO
  - VIM
  - VNF
• We chose OSM...
OSM aims to deliver a production-quality MANO stack...

- Capable of consuming **openly published IM/DM**
- **Available for everyone**, to minimize uncertainties
- Suitable for all VNFs, capturing real production complexity
- **Operationally significant**: including Service Orchestration too!
- VIM-independent

**ALIGNED TO NFV ISG INFORMATION MODELS**
- ... but capable of providing **prompt and constructive feedback** whenever needed

**ENABLING AN ECO-SYSTEM OF IM-COMPLIANT VNF VENDORS**
- Ready to be offered to cloud and service providers
- No need of integration per- customer & MANO vendor basis
- **SO (Service Orchestrator):**
  ![RIFT.io](image)

- **RO (Resource Orchestrator)**
  ![openmano](image)

- **VCA (VNF Configuration & Abstraction)**
  ![JUJU](image)

VCA + RO = NFVO + VNFM (ETSI)
• We decided to add ONOS support
  – The intent model is very well adapted to our needs and we can leverage DynPaC results.
  – We had experience with ONOS (DynPaC OpenCall and Geant work)
  – We used OpenVIM in “OF Only” mode and later tried over “normal” mode (compute nodes and OF hardware switch)

<table>
<thead>
<tr>
<th>mode</th>
<th>Computes hosts</th>
<th>Openflow controller</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
<td>fake</td>
<td>X</td>
<td>No real deployment. Just for API test</td>
</tr>
<tr>
<td>normal</td>
<td>needed</td>
<td>needed</td>
<td>Normal behaviour</td>
</tr>
<tr>
<td>host only</td>
<td>needed</td>
<td>X</td>
<td>No PT/SRIOV connections</td>
</tr>
<tr>
<td>develop</td>
<td>needed</td>
<td>X</td>
<td>Force to cloud type deployment without EPA</td>
</tr>
<tr>
<td>OF only</td>
<td>fake</td>
<td>needed</td>
<td>To test openflow controller without needed of compute hosts</td>
</tr>
</tbody>
</table>
• We decided to add ONOS support
  – The intent model is very well adapted to our needs and we can leverage DynPaC results.
  – We had experience with ONOS (DynPaC OpenCall and Geant work).
  – We used OpenVIM in “OF Only” mode and later tried over “normal” mode (compute nodes and OF hardware switch).

<table>
<thead>
<tr>
<th>mode</th>
<th>Computes hosts</th>
<th>Openflow controller</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
<td>fake</td>
<td>X</td>
<td>No real deployment. Just for API test</td>
</tr>
<tr>
<td>normal</td>
<td>needed</td>
<td>needed</td>
<td>Normal behaviour</td>
</tr>
<tr>
<td>host only</td>
<td>needed</td>
<td>X</td>
<td>No PT/SRIOV connections</td>
</tr>
<tr>
<td>develop</td>
<td>needed</td>
<td>X</td>
<td>Force to cloud type deployment without EPA</td>
</tr>
<tr>
<td>OF only</td>
<td>fake</td>
<td>needed</td>
<td>To test openflow controller without needed of compute hosts</td>
</tr>
</tbody>
</table>
Adding ONOS to OSM

- MANO
- VIM
- SDN Controller
- NFVI (fake hosts)

Server MANO-VIM

SO  RO  VCA  VIM

xbr0

Server-1
Server-2

Compute
NFV requires replaceable components that can be safely & automatically assembled...
- Network Service Descriptor and Virtual Network Function Descriptor: yaml representation: Samples from OSM Wiki*

Reference NS#1: Testing an endpoint VNF

The following network service captures a simple test setup where a VNF is tested with a traffic generator VNF (or a simple VNF/VM with a basic client application). For simplicity, this network service assumes that the VNF under test is the endpoint of a given service (e.g. DNS, AAA, etc.) and does not require special conditions or resource allocation besides the usual in a standard cloud environments.

In this example, unless otherwise specified in the description, the following defaults apply:

- CPs are regular para-virtualized interfaces (VirtIO or equivalent).
- VLs provide E-LAN connectivity via regular (overlay) networks provided by the VIM.
- VLs provide IP addressing via DHCP if applicable.
- Mapping between internal and external CPs may be either direct (as aliases) or via an intermediate VL.
- VIM+NFVI can guarantee predictable ordering of guest interfaces’ virtual PCI addresses.

In the case of REF_NS_1:

- When deploying the NS, VL1 would be typically mapped to a pre-created VIM network intended to provide management IP address to VNFs via DHCP.
- DHCP in VL2 may be optional.

• Network Service Descriptor and Virtual Network Function Descriptor: yaml representation: Samples from OSM Wiki*

NSD and VNFD

• Network Service Descriptor and Virtual Network Function Descriptor: yaml representation: Samples from OSM Wiki*

Reference VNF#11: Endpoint VNF

Description in common language

- **Name**: Ref_VNF_11
  - **Component**: Ref_VM1
    - **Memory**: 2 GB
    - **CPU**: 2 vCPU
    - **Storage**: 8 GB
    - **Image**: ref_vm1.qcow2
  - **Component**: Ref_VM2
    - **Memory**: 4GB
    - **CPU**: 2 vCPU
    - **Storage**: 16GB
    - **Image**: ref_vm2.qcow2
  - **Internal Virtual Link**: VL12
    - No DHCP server is enabled.
    - Static addressing may be used at CP iface11 and CP iface21.

OSM VNF descriptor for VNF#11

VNF11.yaml

NSD and VNFD

- Network Service Descriptor and Virtual Network Function Descriptor: yaml representation: Samples from OSM Wiki*

NSD and VNFD

• Network Service Descriptor and Virtual Network Function Descriptor: yaml representation: Samples from OSM Wiki*

Reference VNF#21: Generator 1 port

Description in common language
- Name: Ref_VNF_21
  - Component: Ref_VM5
    - Memory: 1 GB
    - CPU: 1 vCPU
    - Storage: 16 GB
    - Image: ref_vm21.qcow2

OSM VNF descriptor for VNF#21
VNF21.yaml

SO

RO

Management Network

VIM

VCA

NS instantiation

Specific configuration of VNFD already deployed

* Charms instantiation

VCA is used only if VNFD does have charms.

Charms do describe operations that can be executed on a given VNF either automatically or on demand

setip

startupapache

stopapache
• MANO
• VIM
• SDN Controller
• NFVI
Demo NSD

![Diagram showing a network setup with NSD, Apache, Squid, and client nodes connected through arrows]

- Apache
- Squid
- Client

TNC17 - Linz - Making the Open Orquestration happen
Apache VNFD config

Server 1

Apache
VNF1
Iface SR-IOV: 10 Gbps

Server 2

Squid
VNF2
Iface SR-IOV: 1 Gbps

VNF3
Iface SR-IOV: 1 Gbps
- Interacting with VNF

UI

Action X

SO → VCA → Juju

- Juju
- Proxy VNF_X
- Proxy VNF_Y
- ...

VNF_X

Proxy VNF_X

1 container

1 proxy container for each VNF with charms
DEMO
Thank you!
Eduardo.Jacob@ehu.eus