On the use of OSM to allow for automated network slice scaling in multi-site environments

Jose Ordonez-Lucena (Telefónica)
PoC overview

• Presentation of ZSM PoC#2
  • **PoC objective:** Demonstrate the capacity to automatically scale out a deployed network slice instance across multiple administrative domains
  • **PoC timeline:** 01.01.2021– 31.03-2021 (3 months)

• Alignment with ZSM
  • ZSM001 (Use Cases) and ZSM003 (Network slicing features)
Agenda

- Background information
- PoC: Automated Network Slice Scaling In Multi-Site Environments
- PoC: OSM domain (steps 1-6)
- Concluding remarks
5G-VINNI project

- **5G-VINNI project’s vision**: build an open large-scale 5G End-to-End (E2E) facility that can
  - demonstrate that their 5G network KPIs can be met
  - be validated, accessed and used by vertical industries to test use cases and validate KPIs.

- 5G-VINNI built out of a set of **interworking** facility sites

- **Main facility sites**: offer E2E 5G network capabilities to real-world verticals, with well-defined SLAs
  - Oslo-Kongsberg (Norway); Martlesham (UK); Madrid (Spain); Patras (Greece)

- **5G-VINNI experimentation facility sites**: provide environment for advanced experimentation and testing possibilities on (combination of) elements of the E2E model
  - Aveiro (Portugal); Berlin (Germany); Munich (Germany)

- **Moving experimentation Facility site**: satellite connected vehicle
5G-VINNI facility architecture

From a generic 5G-VINNI facility blueprint...

... to a specific 5G-VINNI facility realization

The PoC makes use of the assets from these two 5G-VINNI facility sites
OSM – Release EIGHT

- Alignment with ETSI ISG ZSM architecture (cf. ZSM002)
  - OSM can be mapped to a ZSM management domain
  - OSM stack aligned with ZSM architecture principles (SBMA, modularity, scalability, zero-touch,...)
- From OSM perspective, OpenSlice behaves as upper OSS/BSS layer
  - OpenSlice (Service Orchestrator) consumes OSM NBI capabilities.

On the mapping of OSM modules with ZSM services
OpenSlice

- **Open-source** operations support system (OSS) solution providing **Service Orchestration** functionality
  - Including both service fulfillment and assurance lifecycle phases

- **CFS viewpoint**: user-friendly web portal for the interaction with the vertical customers:
  - Browse slice templates (VINNI-SB’s) in the Service Catalog.
  - Issue and capture service orders
  - Retrieve PM/FM data on deployed slices.

- **RFS viewpoint**: consumption of OSM NBI capabilities to deploy and operate the virtualized components of the slices.

Link: [https://openslice.io](https://openslice.io)
Agenda

- Background information
- PoC: Automated Network Slice Scaling In Multi-Site Environments
- PoC: OSM domain (steps 1-6)
- Concluding remarks
Scenario description

- Vertical industry (e.g., PPDR, e-Health) NetApps hackathon
  - Involving developers from Spain and Greece
  - NetApp submission service hosted in Madrid facility site
  - Re: GDPR policy, NetApps binaries and data must be hosted in the home country.

- Network slice scaling out operation:
  - Automatically triggered in Madrid, because of unexpected load surges -> **reactive correction action**
  - Propagated towards Patras, due to forecasting reasons -> **pro-active corrective action**
Multi-site NW slice – design & deployment

NSS-A

- NSD_F
  - LB-1
  - Web Server
  - LB-2
  - Backend API brokering service

NSS-B

- NSD_SRV
  - LB-3
  - Repository Catalogue Service
  - Catalogue DB

NSS-C

- NSD_SRV
  - LB-3
  - Repository Catalogue Service
  - Catalogue DB

VNFs

LB-1, LB-2, LB-3: LB_VNF (by OSM)

Web Server: Wserver_VNF:
- Scalable – VNF metrics, granularity 5m

Backend API brokering service: Wserver2_VNF
- Scalable – VNF metrics, granularity 5m

Catalogue DB: Generic_VNF (Vanilla Ubuntu 20.04)
Multi-site NW slice – operation

NSS-A

1. LB-1
2. Web Server
3. LB-2
4. Backend API brokering service

NSS-B

1. LB-3
2. Repository Catalogue Service
3. Catalogue DB
4. Backend API brokering service

NSS-C

1. LB-3
2. Repository Catalogue Service
3. Catalogue DB
4. Backend API brokering service

Traffic Generator

Madrid

Patras

* OSM-triggered scaling out (auto-scaling)
** OpenSlice-triggered scaling out
There is a sudden high demand of portal interaction at Madrid facility site (HTTP requests represents a traffic load surge with 3:1 ratio)

NSS-A's backend API brokering service collapses, being not able to forward traffic to either NSS-B or NSS-C

Based on day-2 activities, “Madrid-OSM” triggers NSS-A auto-scaling -> Web server (2 x scale out), LB-1 (reconfiguration), LB-2 (reconfiguration)

NSS-A's backend API brokering service back on normal operation, and starts sending traffic to NSS-B through LB-3. NSS-B's VNFs collapse.

Based on day-2 activities, “Madrid-OSM” triggers NSS-B auto-scaling -> Repository catalogue (2x scale out), DB (1 x scale out), LB-3 (reconfiguration)

“Madrid-OSM” notifies “Madrid-OpenSlice” of successful steps 3 and 5

“Madrid-OpenSlice” decides that NSS-C needs to be scaled out as NSS-B did, in order to avoid collapse as in Madrid -> OpenSlice is aware of UC semantics

“Madrid-OpenSlice” issues NSS-C scaling request to “Patras-OpenSlice”, using TMF’s APIs. “Patras-OpenSlice” checks this request.

“Patras-OpenSlice” forwards the request to the “Patras-OSM” for enforcement.

Unlike step 3 and 5, here there is no NSS-C auto-scaling

“Patras-OSM” notifies “Madrid-OpenSlice” of successful NSS-C scaling out.

“Patras-OpenSlice” notifies “Madrid-OpenSlice” of successful NSS-C scaling out.
Agenda

- Background information
- PoC: Automated Network Slice Scaling In Multi-Site Environments
- PoC: OSM domain (steps 1-6)
- Concluding remarks
**NSS-A Topology**

**LB_VNF:**
- CPU: 4
- RAM: 4 GB
- Disk: 10 GB
- Name in NSD-F: "haproxy_ubuntu" [image]

**Wserver_VNF (Scalable – VNF metrics, granularity 5min):**
- CPU: 1
- RAM: 1 GB
- Disk: 10 GB
- Name in NSD-F: "ubuntu-20.04-server-cloudimg-amd64" [image]
Wserver_VNF: VNFD

- Collecting VNF metrics
  - A simple charm containing a metrics.yaml file at its root folder specifies the metrics to be collected and associated comments

```yaml
metrics:
  users:
    type: gauge
    description: "# of users"
    command: who | wc -l
  load:
    type: gauge
    description: "5 minute load average"
    command: cat /proc/loadavg | awk '{print $1}'
  load_pct:
    type: gauge
    description: "1 minute load average percent"
    command: cat /proc/loadavg | awk '{load_pct=$1*100.00} END {print load_pct}'
```

```yaml
monitoring-param:
  - id: "Wserver_users"
    name: "Wserver_users"
    aggregation-type: AVERAGE
    vnf-metric:
      vnf-metric-name-ref: "users"
  - id: "Wserver_load"
    name: "Wserver_load"
    aggregation-type: AVERAGE
    vnf-metric:
      vnf-metric-name-ref: "load"
  - id: "Wserver_load_pct"
    name: "Wserver_load_pct"
    aggregation-type: AVERAGE
    vnf-metric:
      vnf-metric-name-ref: "load_pct"
```
### Wserver_VNF: VNFD

#### Autoscaling

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum instance count</td>
<td>0</td>
</tr>
<tr>
<td>Maximum instance count</td>
<td>3</td>
</tr>
<tr>
<td>CPU-scale-in-threshold (%)</td>
<td>20</td>
</tr>
<tr>
<td>CPU-scale-out-threshold (%)</td>
<td>80</td>
</tr>
<tr>
<td>threshold-time (s)</td>
<td>10</td>
</tr>
<tr>
<td>cooldown-time (s)</td>
<td>30</td>
</tr>
</tbody>
</table>

```yaml
scaling-group-descriptor:
  - name: "Wserver_autoscale"
    min-instance-count: 0
    max-instance-count: 3
  scaling-policy:
    - name: "Wserver_load_pct_above_threshold"
      scaling-type: "automatic"
      threshold-time: 10
      cooldown-time: 30
      scaling-criteria:
        - name: "Wserver_load_pct_above_threshold"
          scale-in-threshold: 20
          scale-in-relational-operation: "LT"
          scale-out-threshold: 80
          scale-out-relational-operation: "GT"
          vnf-monitoring-param-ref: "Wserver_load_pct"
  vdu:
    - vdu-id-ref: Wserver
      count: 1
```

© ETSI
• Background information
• PoC: Automated Network Slice Scaling In Multi-Site Environments
• PoC: OSM domain (steps 1-6)
• Concluding remarks
Golden nuggets

• ZSM poC#2 pipeline, with all ingredients
  • Research + Experimentation + Standardization
  • Open-source communities (OSM and OpenSlice) along the entire path

• ZSM PoC#2 showcasing by early April
  • Current status: automated scaling at individual facility sites (steps 1-6)
  • Next steps: Cross-domain orchestration @OpenSlice (steps 7-11)

• PoC#2 wikipage: https://zsmwiki.etsi.org/index.php?title=PoC_2_Automated_Network_Slice_Scaling_in_Multi-Site_Environments
Annex A
Scaling out in action
Scaling in action (pre-conditions)
Scaling in action (post-conditions)

OpenStack instances view
- ZSM-4-Wserver2-2
- ZSM-2-Wserver-3
- ZSM-2-Wserver-2

OpenStack topology view
Some scaling actions are completed successfully, and more are happening
The dashboard shows the new nodes (1 initial plus 2 scaled out)
Annex B

The role of OpenSlice in the PoC
OpenSlice – service fulfillment

Service specification tree

Service Frontend Spec (RFS): NSS-A
Service Frontend Spec (RFS): NSS-B
Service Frontend Spec (RFS): NSS-C

NSD_F
NSD_SVR
NSD_SVR

Aligned with GSMA GST/NEST

Service ordering

ServiceFrontend
ServiceBackend
ServiceBackendGR

Service PoC Spec bundle (CFS)

OSM (NPVO)

64x Order API

© ETSI
PoC#2 service prototypes running in OpenSlice
OpenSlice – service assurance

PoC#2: alarm management

PoC#2: policy design

PoC#2: policy management
PoC#2: cross-domain orchestration (steps 7-11)