High-level VNF Descriptors using NEMO

draft-aranda-nfvrg-recursive-vnf-00
Pedro A. Aranda pedroa.aranda@telefonica.com
Diego López diego.r.lopez@telefonica.com
Rationale

• No one in a clean state of mind can read VNFDs easily

• There is no simple way of reusing tested VNFs to build more elaborate VNFs

• This goes against one of the goodies of software design/production
  – RE-USABILITY

• Why?
  – It is easier to reuse things you understand
Easy vs. difficult

Easy to understand

More difficult

```python
vnf:
  name: TEMPLATE
  description: This is a template to help in the creation of
  # class: parent  # Optional. Used to organize VNFs
  external_connections:
    - name: mgmt0
      type: mgmt
      VNFC: TEMPLATE-VM
      local_iface_name: mgmt0
      description: Management interface
    - name: xe0
      type: data
      VNFC: TEMPLATE-VM
      local_iface_name: xe0
      description: Data interface 1
    - name: xe1
      type: data
      VNFC: TEMPLATE-VM
      local_iface_name: xe1
      description: Data interface 2
    - name: ge0
      type: bridge
      VNFC: TEMPLATE-VM
      local_iface_name: ge0
      description: Bridge interface
```
Alternative we propose

• Since VNFDs are not easy to understand
  – Why not use the network modelling language NEMO?
• BoF last summer in Prague
• Human readable AND human *understandable*
• Structured like high-level programming languages
How would this work?

- VNFDs like those in OpenMANO are used as low level blocks
- NEMO allows us to describe VNFs
  - Service graphs (the relationships between the VNFCs) become more obvious using the Connection concept
- NodeModels can be reused:
  - Opening the door to recursiveness
This is what we want

- Find a way to describe the VNF as close as possible to this graph

see http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
So let’s go step by step

- Import VNFD into NEMO
  - Most VNF producers will anyhow have a VNFD (for OSM, OpenMANO, etc.)
  - Requirement on NEMO: ConnectionPoint

```
CREATE NodeModel NAME SampleVNF
  IMPORT VNFD from https://github.com/nfvlabs/openmano.git/openmanano/vnfs/examples/daplaneVNF1.yaml
  DEFINE ConnectionPoint data_inside as VNFD:ge0
  DEFINE ConnectionPoint data_outside as VNFD:ge1
```
Step by step (2)

• Use the imported NodeModels to build more complex functionality:
  – Requirement on NEMO: Connection to define the service graph

```
CREATE NodeModel NAME ComplexNode
Node InputVNF TYPE SampleVNF
Node OutputVNF TYPE ShaperVNF
DEFINE ConnectionPoint input
DEFINE ConnectionPoint output
CONNECTION input_connection FROM input TO InputVNF:data_inside
  TYPE p2p
CONNECTION output_connection FROM output TO ShaperVNF:wan
  TYPE p2p
CONNECTION internal FROM InputVNF:data_outside TO ShaperVNF:lan
  TYPE p2p
```
And from here...

- Use NodeModels to create even more complex models once these are tested and prove to fulfil your requirements.
- Made easy when you understand what you read.
Acknowledgement

This work has been partially performed in the scope of the SUPERFLUIDITY project, which has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No.671566 (Research and Innovation Action).