Multi-Layer IBN in NFV ecosystem: functional architecture and practical experience

Barbara Martini
Molka Gharbaoui
Background:

- SDN/NFV enable flexible slicing of network, computing and storage infrastructures customized to meet specific needs of vertical applications (VAs)
- ETSI NFV MANagement and Orchestration (MANO) is the reference framework for the orchestration and lifecycle management of physical and/or software resources supporting network slicing
- SDN can provide flexibility features to allow for programmable interconnection among Virtual Network Functions (VNFs) composing the network slices

Challenge:

- SDN/NFV and MANO lead to increasingly network and service platforms flexibility and new business opportunities provided that an end-to-end vision is embraced and multi-layer orchestration functionalities are put in place.

Open question:

- Does multi-layer orchestration imply multi-layer intents?
- Does different involved stakeholders imply multi-layer intent?
- What is the role of SDN in this multi-layer vision?
- How SDN can be part of an intent specification?
Multi-layer Orchestration and Intents in NFV
Outline

• Intent-based service chain deployment
  • Scope: network slice
  • Perspective Tenant/Vertical

• Intent-based Virtual Link deployment
  • Scope: virtual infrastructure (WAN)
  • Perspective: Network service provider

• Intent-based SDN-capable slice deployment
  • Scope: slice as a service
  • Perspective: Tenant/Vertical
Intent-based service chain deployment

Fed4FIRE+ Open Call for Experimentation
«Latency-aware and self-Adaptive Service cHaining in reliable 5G/SDN/NFV infrastructures (LASH-5G)” (GA: 732638)

Contributors:
B. Martini, W. Cerroni, F. Paganelli, M. Gharbaoui, C. Contoli, G. Davoli, G. Cuffaro, P. Cappanera
Overview

• 5G NR provides higher throughput and lower latency thereby enabling a unique smart environments and a new breed of vertical applications (e.g., connected cars, industry 4.0, e-health)
• network slicing enable tailored softwarized infrastructures to run those application mostly demanding for stringent end-to-end latency requirements
• traditionally vertical applications are delivered as chains of composite services (security, firewall, optimization) according to specified composition rules also based on context information (dynamic service composition)
• SDN network control capabilities to address an effective data delivery across dynamic (e.g., context-aware) chains of services
Reference architecture

Service Chain Optimizer

Orch. Layer

NFVI

VIM 1

WIM

VIM 2

VIM 3

SDN Contr. 1

WAN SDN Controller

SDN Contr. 2

SDN Contr. 3

Edge Cloud SDN Domain 1

WAN SDN Infrastructure

Edge Cloud SDN Domain 2

Edge Cloud SDN Domain 3

Application

service chain intent

Network/cloud intents

ETSI NFV MANO

NFVO/VNFM
Service Chain Intent

{"vfChain": "VF-1 in DC1, VF-2 in DC2"}
Some results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>64.34</td>
<td>40.84</td>
<td>1.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>69.34</td>
<td>36.29</td>
<td>1.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>74.96</td>
<td>35.98</td>
<td>1.48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>82.75</td>
<td>40.5</td>
<td>1.52</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Deploy Service Chain: NODE-B@dc1 -VF4-VF5-VF9-VF10- NODE-D@dc2
Chain 4: NODE-B@dc1-VF4@dc1 -VF5@dc3-VF9@dc2-VF10@dc2- NODE-D@dc2
Intent-based Virtual Link deployment
Overview (ETSI NFV)

• A unified Virtual Link connecting VNF is composed of 3 components:
  • Virtualised network resource intra-PoP: from the VNF to the network port attached to the WAN
  • Virtualised network resource inter-PoP
  • Virtualised network resource intra-PoP: from the VNF to the network port attached to the WAN
• Virtualised network resource inter-PoP: network connectivity between two sites across WAN
• Coordination among the VNFs deployments (including connectivity) at each site and the network connectivity between the two sites to set-up a Network Service (NS)/VNG-Forwarding Graph (NFV-FG)
Sequence Diagram (ETSI IFA022/IFA005)

1. InstantiateNsRequest(IFA013)
2. AllocateNetworkRequest(IFA005)
3. Create network connectivity between PE#1 and PE#2
4. Response
5. AllocateNetworkResponse(IFA005)
6. AllocateNetworkRequest(IFA005)
7. AllocateNetworkResponse(IFA005)
8. AllocateNetworkRequest(IFA005)
9. AllocateNetworkResponse(IFA005)
10. InstantiateNsResponse(IFA013)
Reference architecture

**NFVI-PoP topology:**
- Logical Link
  - Id
  - NFVI-PoP Id source
  - NFVI-PoP Id dest
  - Gw IP Addr source
  - Gw IP Addr dest
- info...
- NFVI-PoP
  - info...

**NFVI-PoP Attributes:**
- Id
- Location
- Gw IP addr
- CPU capacity / availability
- RAM capacity / availability
- STORAGE capacity / availability
- Link capacity
- Cost
- Failure rate

**WAN Connectivity Attributes:**
- Id (resource identifier used internally)
- Gw IP Addr(s) (Gw-a, Gw-b)
- info for connecting to the WAN (IP address+VXLAN ID, MPLS+VPN Route Distinguisher)
- Cost
- Total and Available bw
- Total delay

**Mobile Transport and Computing Platform (MTP)**

**Service Orchestrator (SO)**
WIM abstraction (1/2)

NFVI-PoP1 (VIM1)
192.168.3.30

NFVI-PoP2 (VIM2)
192.168.4.40

WAN Topology
WIM abstraction (2/2)

5G-SO

5G-MTP

WIM REST client

WIM REST server

WIM

REST client

REST ONOS APIs

ONOS SDN Controller

SOInterface

GET /abstract-resources
Retrieves aggregated Cloud NFVI-PoP and Inter-NFVI-PoP connectivity

POST /abstract-network-resources
Creates inter-NFVI-PoP connectivity

DELETE /abstract-network-resources
Deletes inter-NFVI-PoP connectivity

GET /abstract-network
Retrieves aggregated WAN Connectivity

DELETE /network-resources/{networkId}

GET /network-resources

POST /network-resources

WAN Topology

Logical view
WAN Virtual Link Intent

- **GET Abstract view**
  - Request for the abstract view of the WAN
  - Response: json file containing list of gateways and list of virtual links

- **POST request to allocate a virtual link**
  - Request: json body containing info relative to the VL (id, bandwidth, delay, ingress/egress points)
CNIT SDN WIM

- Exposes an IFA-compliant Intent-based REST API for NFV deployments in inter-PoP scenarios (IFA005/IFA022)
- Advertises the abstract network topology
- Provisions end-to-end network paths (with and without explicit specification of the path hops)
- Exposes monitoring data related to the inter-PoP transport network:
  - topology characteristics: number of OF switches, number of virtual links, etc.
  - performance data: number of bytes per port/flow, throughput, etc.
Entertainment PoC: Set-up and Workflow

1. Request for abstract view of PoP1
2. Allocate resources in PoP1
3. Virtual Link Intent (IFA-005)
4. Get set of switches and links of the WAN
5. Current status of network topology
6. Allocate resources in the WAN
7. Allocate resources in PoP2
8. Send intent to the SDN controller to allocate the resources
9. Set up the flow entries in the OpenFlow switches
10. Virtual Compute Intent (IFA-005)
Intent-based SDN-capable slice deployment

5GINFIRE Open Call for additional functionalities and infrastructures for experimentation «Slice SDN-based Network Management in 5G (SLICENET-5G)” GA: 732497
Overview

Network slice intents express the high-level requirements and constraints for the network service set-up hosting the vertical components:

- Vertical components and relevant service graph
- Requirements (e.g., CPU/memory/storage, link bandwidth)
- Constraints (e.g., delay, packet loss, throughput, location)
- Network functions and communication services (e.g., security, DPI, firewall, VPN, ...)

---

**Network Slice Intent**

<table>
<thead>
<tr>
<th>Atomic functional components involved</th>
<th>sdnLinux (see Table 74)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service sequence</td>
<td>VNFG in textual notation (ETSI NFV IFA 014) of</td>
</tr>
<tr>
<td>Connectivity service</td>
<td>n/a</td>
</tr>
<tr>
<td>External Interconnection</td>
<td>sapMgmt</td>
</tr>
<tr>
<td>Internal Interconnection</td>
<td>n/a</td>
</tr>
<tr>
<td>SST</td>
<td>n/a (see the field SLA instead)</td>
</tr>
<tr>
<td>Service constraints</td>
<td>Geographical area: n/a</td>
</tr>
<tr>
<td></td>
<td>Security: low</td>
</tr>
<tr>
<td></td>
<td>Priority: medium</td>
</tr>
<tr>
<td></td>
<td>Cost: n/a</td>
</tr>
<tr>
<td></td>
<td>Synchronization: low</td>
</tr>
<tr>
<td></td>
<td>etc.</td>
</tr>
<tr>
<td>Management and control capabilities for the tenant</td>
<td>Provider managed</td>
</tr>
<tr>
<td>SLA</td>
<td>n/a</td>
</tr>
<tr>
<td>Monitoring</td>
<td>rtprobe cyclicavg</td>
</tr>
<tr>
<td></td>
<td>rtprobe cyclicmax</td>
</tr>
<tr>
<td>Lifetime</td>
<td>On-demand, 1h</td>
</tr>
<tr>
<td>Charging</td>
<td>n/a</td>
</tr>
</tbody>
</table>

---

**5G-TRANSFORMER (5G-T):** [http://5g-transformer.eu](http://5g-transformer.eu)

**MATILDA:** [https://www.matilda-5g.eu/](https://www.matilda-5g.eu/)
SDN-enabled Network Slice Intent

- SDN-based connectivity in the slice to programmatically support:
  - internal L3/L2 forwarding functions across other slice components (L4-L7 network functions, vertical applications)
  - dynamic activation of data delivery services to recover from failure, traffic engineering
  - exposition of a control interface for interworking with external networks
- vertical may want to specify if SDN-related network capabilities are desired:
  - “I want to connect VNF-1, VNF-2, VA-a, VA-b through SDN switches and to program the flows among them”
- network slice intent should allow to request SDN capabilities in terms of:
  - which vertical applications or VNF to attach (id, type, network port)
  - SDN network interfaces and interconnections among VNFs/Vapp
Deploying SDN capabilities in a slice

Slice composed of:
• SDN virtual switches and controller as VNFs
• Set of Vertical Applications (VAs) or VNFs

SDN components are configured to automatically start once the SDN-based VNFs are up:
• SDN controller
• SDN topology (OVS, emulated)

VAs configured to be attached to the SDN network
Blueprint example

```yaml
name:
description:
deployment-details:
  min_applicative_instances_number:
    type: integer
    default: 1
  max_applicative_instances_number:
    type: integer
    default: 50
instance_config: # for each instance
  id:
  vCPU:
  RAM:
  storage-gb:
  image:
  management-IP:
  data-plane-IP:
connectivity-details: # virtual links connecting VAs to the SDN slice
  virtual-link-1:
  virtual-link-2:
constraints:
  location
  instances_max_load
```
VNF/NS Descriptors

vnfd:vnfd:
  - id: sdn_vnfd
    name: sdn_vnfd
    short-name: sdn_vnfd
    description: SDN-based VNF
    vendor: 5GlnFIRE
    version: '1.2'
    connection-point:
      - id: eth0
        name: eth0
        short-name: eth0
        type: VPORT
    mgmt-interface:
      cp: eth0
    vdu:
      - id: custom
        name: custom VDU
        image: custom
        count: 1
        cloud-init-file: cloud-config.yml
        vm-flavor:
          vcpu-count: 2
          memory-mb: 3072
          storage-gb: 5
        interface:
          - name: eth0
            position: 1
            type: EXTERNAL
            virtual-interface:
              external-connection-point-ref: eth0
        vnf-configuration:
          initial-config-primitive:
            seq: '1'
            name: config
            parameter:
              - name: ssh-hostname
                value: <rw_mgmt_ip>
              - name: ssh-username
                value: ubuntu
              - name: ssh-password
                value: osm

nsd:nsd:
  - id: SFC_nsd
    name: SFC_ns
    short-name: SFC_ns
    description: Chain of 3 VNFs
    vendor: OSM
    version: '1.0'

    # Place the logo as png in icons directory and provide the name here
    logo: osm_2x.png

    # Specify the VNFs that are part of this NSD
    constituent-vnfd:
      - id: sdn_vnfd
        vnfd-id-ref: simple1_vnfd
        - id: simple1_vnfd
          vnfd-id-ref: sdn_vnfd
        - id: simple2_vnfd
          vnfd-id-ref: sdn_vnfd

    # Networks for the VNFs
    vld:
      - id: management_vld1
        name: management_vld1
        short-name: management_vld1
        type: PLAT
        mgmt-network: 'true'
        vtn-network-name: shared
        vnfd-connection-point-ref:
          - member-vnf-index-ref: 1
            vnfd-id-ref: sdn_vnfd
            vnfd-connection-point-ref: eth0
          - member-vnf-index-ref: 2
            vnfd-connection-point-ref: eth0
          - member-vnf-index-ref: 3
            vnfd-connection-point-ref: eth0

      - id: management_vld2
        name: management_vld2
        short-name: management_vld2
        type: PLAT
        mgmt-network: 'true'
        vtn-network-name: shared
        vnfd-connection-point-ref:
          - member-vnf-index-ref: 1
            vnfd-id-ref: sdn_vnfd
            vnfd-connection-point-ref: eth0
          - member-vnf-index-ref: 2
            vnfd-id-ref: sdn_vnfd
            vnfd-connection-point-ref: eth0
          - member-vnf-index-ref: 3
            vnfd-id-ref: sdn_vnfd
            vnfd-connection-point-ref: eth0
Video Demo
Thank you!

barbara.martini@cnit.it
molka.gharbaoui@cnit.it
Approaches for Network Slice Intents

- **Provider-oriented approach** (5G-TRANSFORMER project):
  - The provider supports the Vertical Service Descriptors (VSDs) filled up with high-level requirements and constraints by the vertical.
  - VSDs meant as abstractions of Network Service Descriptors (e.g., service sequence as VNF-FG, components as VNFD).
  - Additional components specified to support the service from the provider perspective (e.g., firewall, load balancer) transparent to the vertical.
  - Possibility to specify vertical service chains.

- **Tenant-oriented approach** (MATILDA project):
  - The tenant requests the slice with specified requirements and constraints to set-up Vertical Applications in a service graph.
  - A metamodel (i.e., Slice Intent) specified for the request.
  - Vertical service chain natively supported (i.e., service mesh).
  - Possibility to specify logical components (e.g., firewall, VPN, IDS_IPS) required by the vertical to the telco operator.
Operations

• GET Abstract view
  • http://localhost:53000/wimsdnplugin/abstract-network
  • Request for the abstract view of the WAN
  • Response: json file containing list of gateways and list of virtual links

• POST request to allocate a virtual link
  • http://localhost:53000/wimsdnplugin/network-resources
  • Request: json body containing info relative to the VL (id, bandwidth, delay, ingress/egress points)
  • Steps:
    • Check if the requested parameters are available
    • Prepare an intent that corresponds to the requested parameters
    • Send the intent to the SDN controller (ONOS) to setup the flowentries in the OpenFlow switches of the WAN (network connectivity)
    • If the intent is correctly installed, update the network resource with the new values of Free bw and Allocated bw
    • Update the intents DB

• DELETE request to delete a virtual link
  • http://localhost:53000/wimsdnplugin/network-resources/reqId
  • Steps:
    • Check if the network service already exists
    • Send a delete request to the SDN controller to delete the flowentries
    • Update the bandwidth status after the release of the resources
    • Update the intents DB
```
{
  "gateways": [
    {
      "gatewayAttributes": {
        "geographicalLocationInfo": "Pisa",
        "wimId": "3",
        "networkConnectivityEndpoint": [
          {
            "netGwIpAddress": "192.168.1.10",
            "netGwInterfaceId": null
          },
          {
            "netGwIpAddress": "192.168.2.20",
            "netGwInterfaceId": null
          }
        ],
        "gatewayId": "192.168.1.10"
      }
    },
    {
      "gatewayAttributes": {
        "geographicalLocationInfo": "Pisa",
        "wimId": "3",
        "networkConnectivityEndpoint": [
          {
            "netGwIpAddress": "192.168.2.20",
            "netGwInterfaceId": null
          }
        ],
        "gatewayId": "192.168.2.20"
      }
    }
  ],
  "virtualLinks": [
    {
      "virtualLink": {
        "virtualLinkId": "101",
        "totalBandwidth": 1000000,
        "availableBandwidth": 1000000,
        "networkQoS": {
          "linkCostValue": 1,
          "linkDelayValue": 50,
          "packetLossRate": 0
        },
        "srcGwId": "192.168.1.10",
        "srcLinkId": 10,
        "dstGwId": "192.168.2.20",
        "dstLinkId": 20,
        "networkLayer": "vxlan"
      }
    }
  ]
}``
```
name:
description:
deployment-details:
  min_applicative_instances_number:
    type: integer
    default: 1

max_applicative_instances_number:
  type: integer
  default: 50

instance_config:
  # for each instance
  id:
  vCPU:
  RAM:
  storage-gb:
  image:
  management-IP:
  data-plane-IP:

connectivity-details:
  # virtual links connecting VAs to the SDN slice
  virtual-link-1:
  virtual-link-2:

constraints:
  location
  instances_max_load
nsd:nsd-catalog:
  nsd:
    id: SFC_nsd
    name: SFC_ns
    short-name: SFC_ns
    description: Chain of a 3 VNFs
    vendor: OSM
    version: '1.0'

# Place the logo as png in icons directory and provide the name here
logo: osm_2x.png

# Specify the VNFDs that are part of this NSD
constituent-vnfd:
  - # The member-vnf-index needs to be unique, starting from 1
    # vnfd-id-ref is the id of the VNFD
    # Multiple constituent VNFDs can be specified
    member-vnf-index: 1
    vnfd-id-ref: simple1_vnf
    member-vnf-index: 2
    vnfd-id-ref: sdn_vnf
    member-vnf-index: 3
    vnfd-id-ref: simple2_vnf

vld:
  # Networks for the VNFs
  - id: management_vld1
    name: management_vld1
    short-name: management_vld1
    type: ELAN
    mgmt-network: 'true'
    vln-network-name: shared
    vnfd-connection-point-ref:
      - member-vnf-index-ref: 1
        vnfd-id-ref: simple1_vnf
        vnfd-connection-point-ref: eth0
      - member-vnf-index-ref: 2
        vnfd-id-ref: sdn_vnf
        vnfd-connection-point-ref: eth0

  - id: management_vld2
    name: management_vld2
    short-name: management_vld2
    type: ELAN
    mgmt-network: 'true'
    vln-network-name: shared
    vnfd-connection-point-ref:
      - member-vnf-index-ref: 2
        vnfd-id-ref: sdn_vnf
        vnfd-connection-point-ref: eth0
VNFD - SDN

```
vnfd:vnfd-catalog:
    vnfd:
        - id: sdn_vnfd
          name: sdn_vnfd
          short-name: sdn_vnfd
          description: SDN-based VNF
          logo: 5GinFIRE.png
          vendor: 5GinFIRE
          version: '1.2'
          connection-point:
            - id: eth0
              name: eth0
              short-name: eth0
              type: VPORT
          mgmt-interface:
            cp: eth0
          vdu:
            - id: custom
              name: custom VDU
              image: custom
              count: 1
              cloud-init-file: cloud-config.yml
              vm-flavor:
                vcpu-count: 2
                memory-mb: 3072
                storage-gb: 5
              interface:
                - name: eth0
                  position: 1
                  type: EXTERNAL
                virtual-connection:
                  type: VIRTIO
                  external-connection-point-ref: eth0
              vnf-configuration:
                initial-config-primitive:
                  seq: '1'
                  name: config
                  parameter:
                    - name: ssh-hostname
                      value: <rw_mgmt_ip>
                    - name: ssh-username
                      value: ubuntu
                    - name: ssh-password
                      value: osm
```