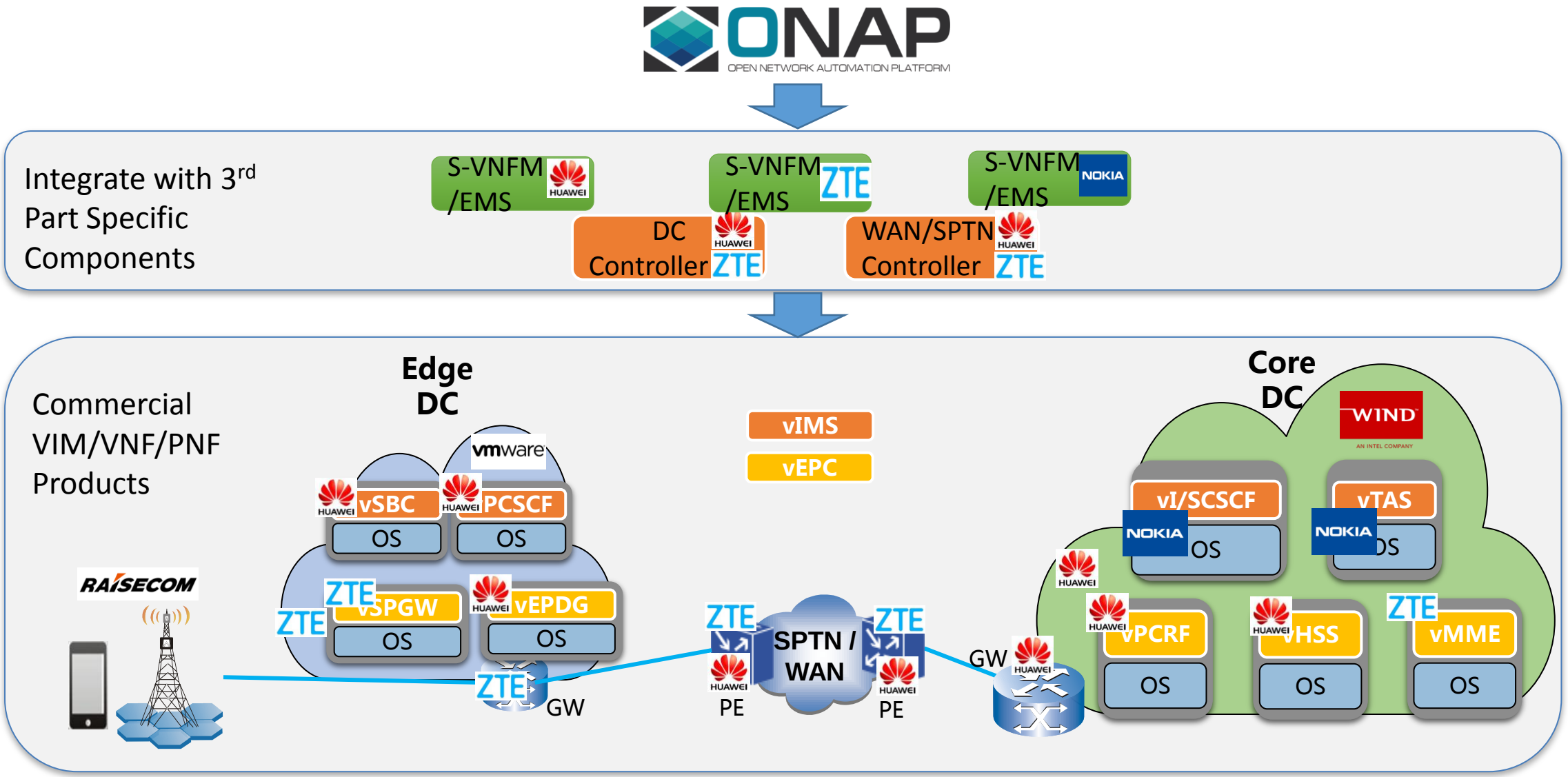
ONAP
OPEN NETWORK AUTOMATION PLATFORM

VoLTE E2E Service Review

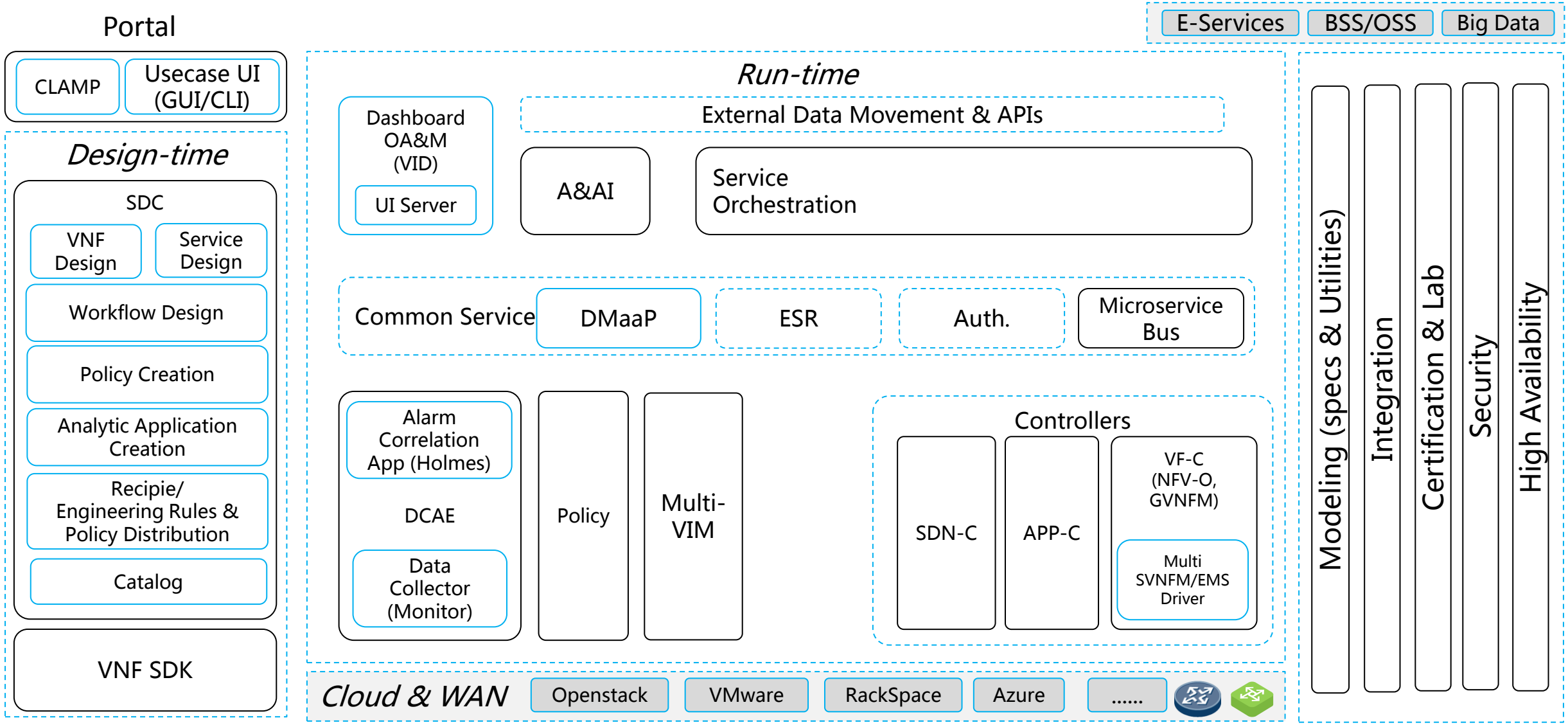
September 26, 2017

Chengli Wang/Yang Xu

VoLTE Use Case



ONAP Architecture & VoLTE Use Case



VNF Onboarding and Service Design & Creation Process

- ❑ VNF Onboarding
- ❑ vEPC Network Service Design
- ❑ vIMS Network Service Design
- ❑ Underlay Template Import
- ❑ Overlay Template Import
- ❑ VoLTE E2E Service Design with vEPC, vIMS, Underlay and Overlay
- ❑ Closed-loop Design, e.g. Holmes Rules, Operational Policy

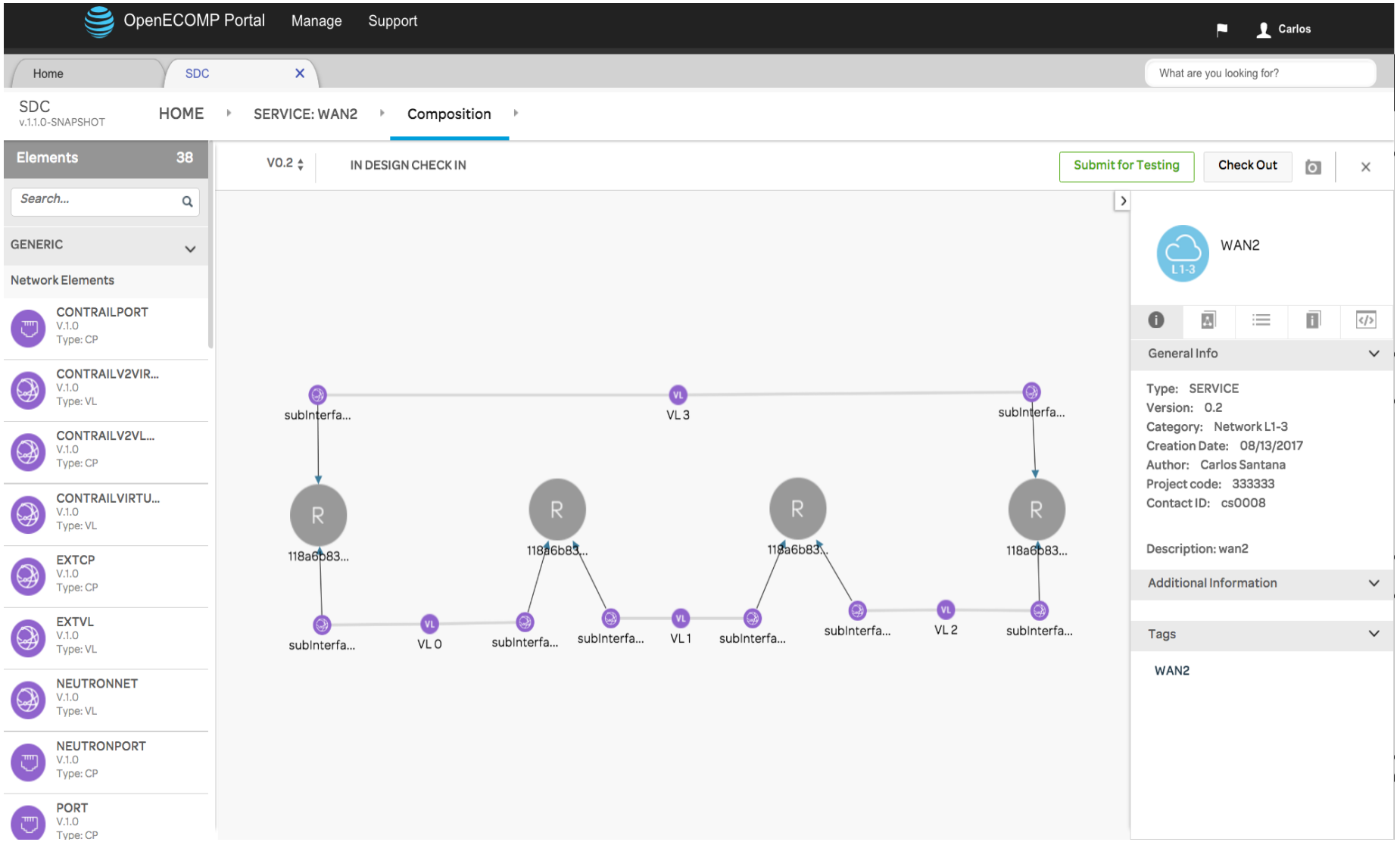
* VNF template is provided by vendor
* SO and VFC workflows are preinstalled
* SDNC DG is preinstalled

VNF CSAR File Imported by SDC

```
ROOT\
        MainServiceTemplate.mf                  //a map representing the different parts of the CSAR structure.
        MainServiceTemplate.yaml                //copy of the main service template located under Definitions
        \TOSCA-Metadate\
                TOSCA.meta                      //metadata regarding the CSAR structure
         \Artifacts\                            //includes all artifacts, but Images are not supported
                \Informational\                 //will hold all informative artifacts in the direct asset level
                \Deployment\                    //Deployment artifacts
         \<VFC TOSCA name>\
                \Informational\
                \Deployment\                    //will hold all deployment artifacts on VFC level
         \Definitions\                          //includes all TOSCA yaml files
```

* See more details on https://wiki.onap.org/display/DW/Csar+Structure
* Images of VNF should be imported to VIMs manually before provisioning, according to the location of VNFs

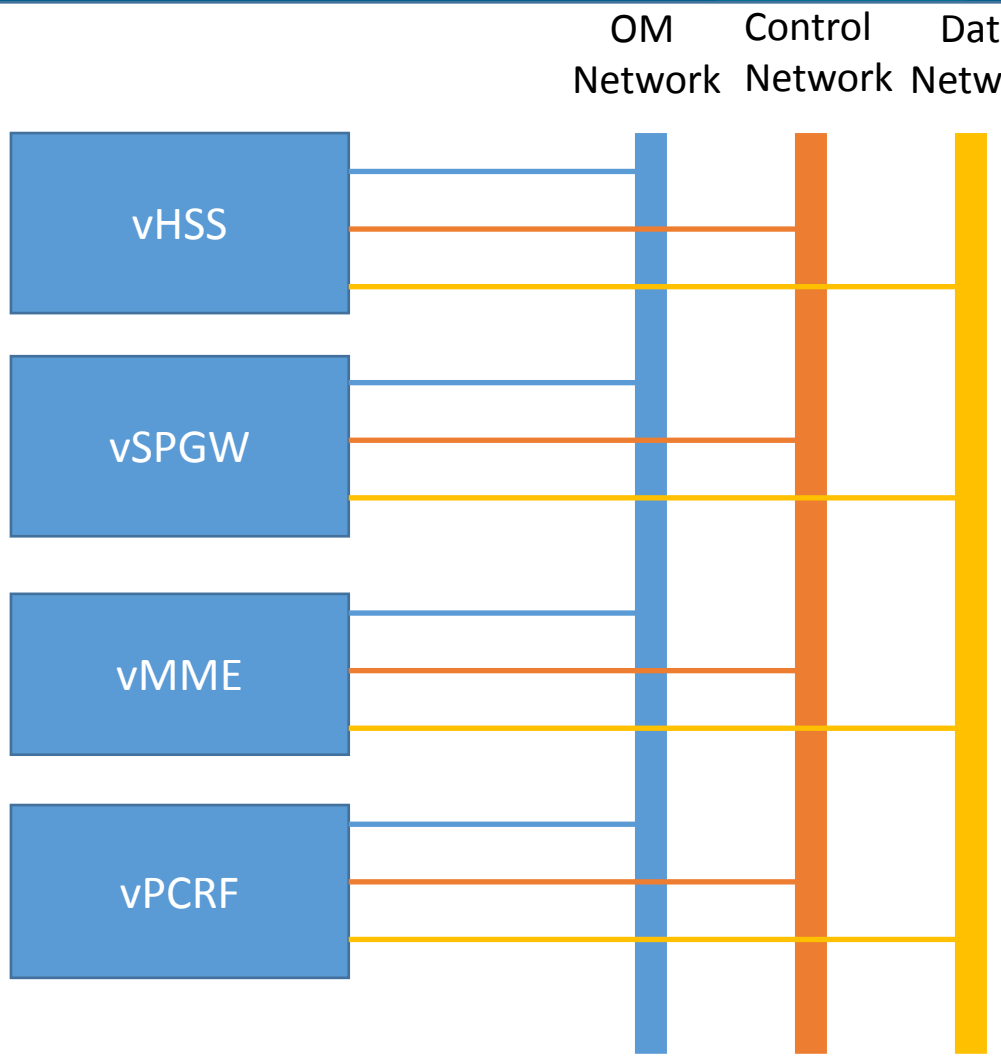
vEPC and vIMS Network Service Design



Network Service Design Steps

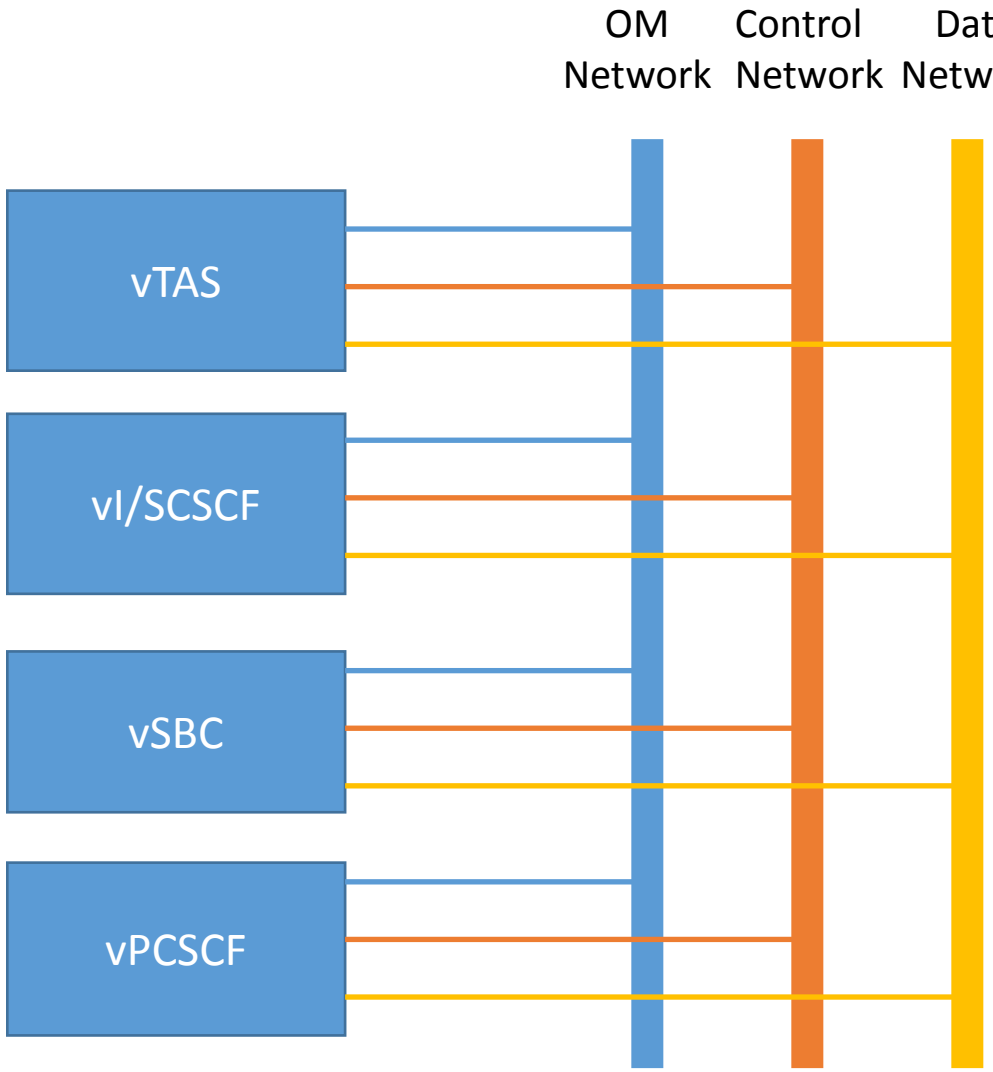
- Design network topology inside network service, connect VNFs by VLs via CPs
- Indicate values of parameters of NS

vEPC Network Service



vEPC

vIMS Network Service



vIMS

Underlay Network Template

```
tosca_definitions_version: tosca_simple_yaml_1_0
# ************************************************************
# underlay vpn type definitions
# ************************************************************
metadata:
  version: 0.1
  vendor: ONAP
  template_author: ONAP
node_types:
  org.openecomp.resource.vl.underlayvpn:
    derived_from: tosca.nodes.Root
    description: undelay vpn type definitions
    properties:
      id:
        type: string
        required: false
      template_name:
        type: string
        required: false
      version:
        type: string
        required: false
```

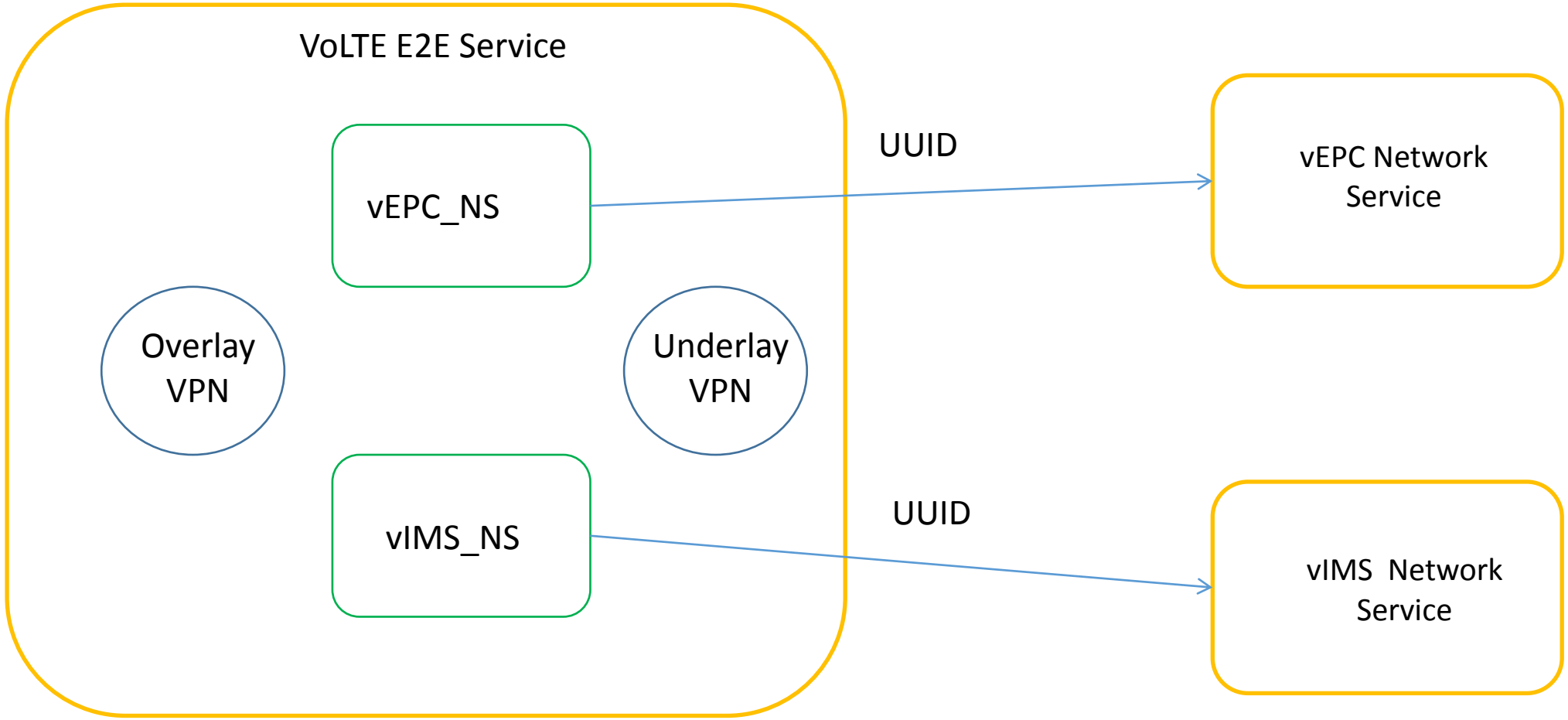
```
vendor:
        type: string
        required: false
      template_author:
        type: string
        required: false
      name:
        type: string
        required: false
      description:
        type: string
        required: false
.... ....
requirements:
    - virtualLink:
        capability: tosca.capabilities.network.Linkable
        relationship: tosca.relationships.network.LinksTo
    capabilities:
      virtual_linkable:
        type: tosca.capabilities.network.Linkable
```

Overlay Network Template

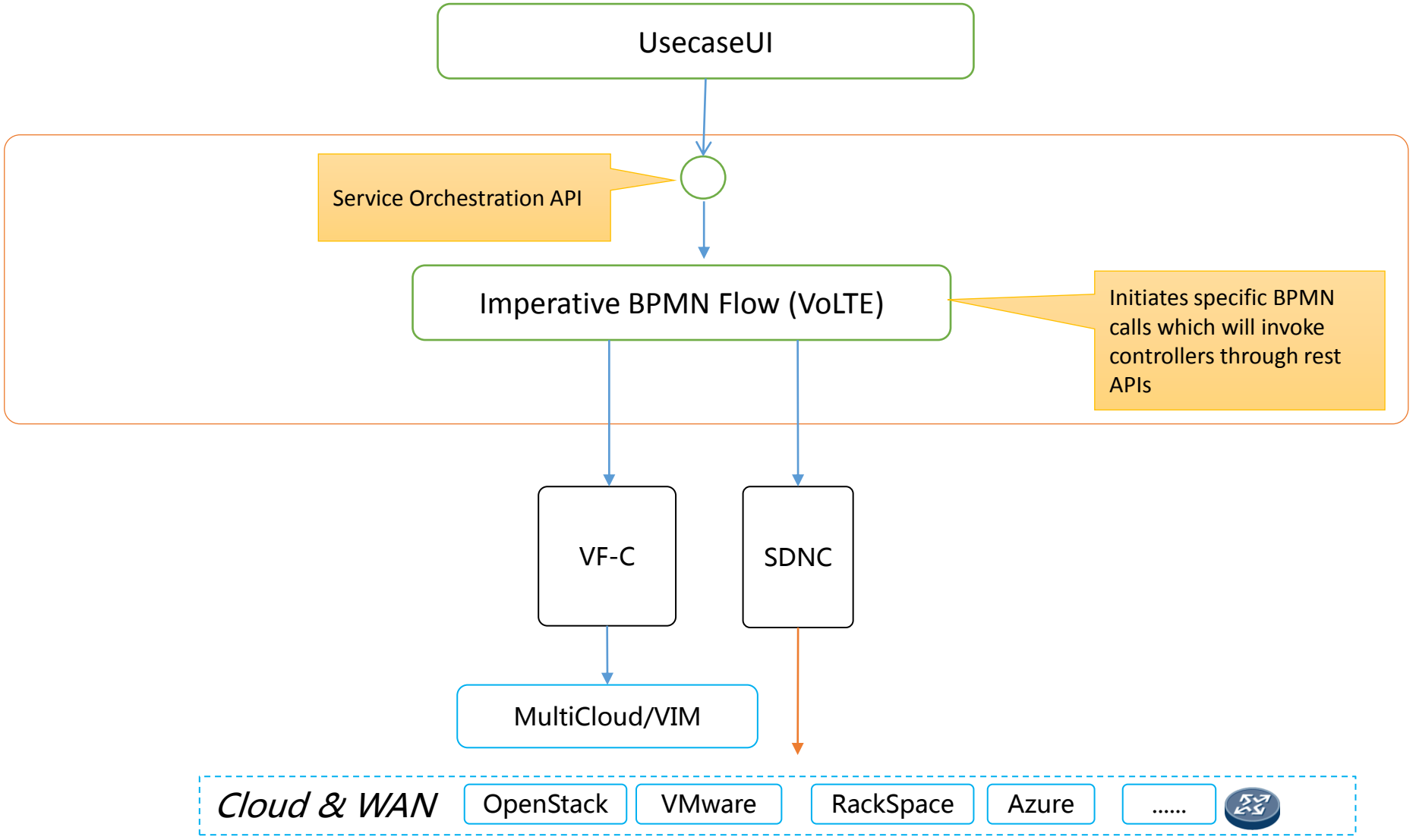
```
metadata:
  id: : overlayTunnelDefinition
  version: 0.1
  vendor: ONAP
  template_author: ONAP
node_types:
  org.openecomp.resource.vl.overlaytunnel:
    derived_from: tosca.nodes.Root
    description: >-
      This entity represents abstract overlay tunnel end point.
    properties:
      id:
        type: string
        description: Identifier of the Tunnel Endpoint node.
        required: false
      template_name:
        type: string
        required: false
      version:
        type: string
        required: false
      name:
        type: string
        description: Overlay tunnel name used by for reference by the administrator.
        required: false
```

```
    ... ...
    description:
      type: string
      description: Additional comments/information about overlay tunnel.
      required: false
    tunnelType:
      type: string
      constraints:
          - valid_values: ['L3-DCI','L2-DCI']
      description: type defines if the overlay tunnel is L3-DCI tunnel or L2-DCI tunnel.
      required: true
    site1_importRT1:
      type: string
      description: List of Route targets imported by the local router.
      required: false
    site1_networkIds:
      type: String
      required: true
      description:  site1 local subnet id
    site2_networkIds:
      type: String
      required: true
      description:  site2 local subnet id
  requirements:
    - virtualLink:
      capability: tosca.capabilities.network.Linkable
      relationship: tosca.relationships.network.LinksTo
  capabilities:
    virtual_linkable:
    type: tosca.capabilities.network.Linkable
```

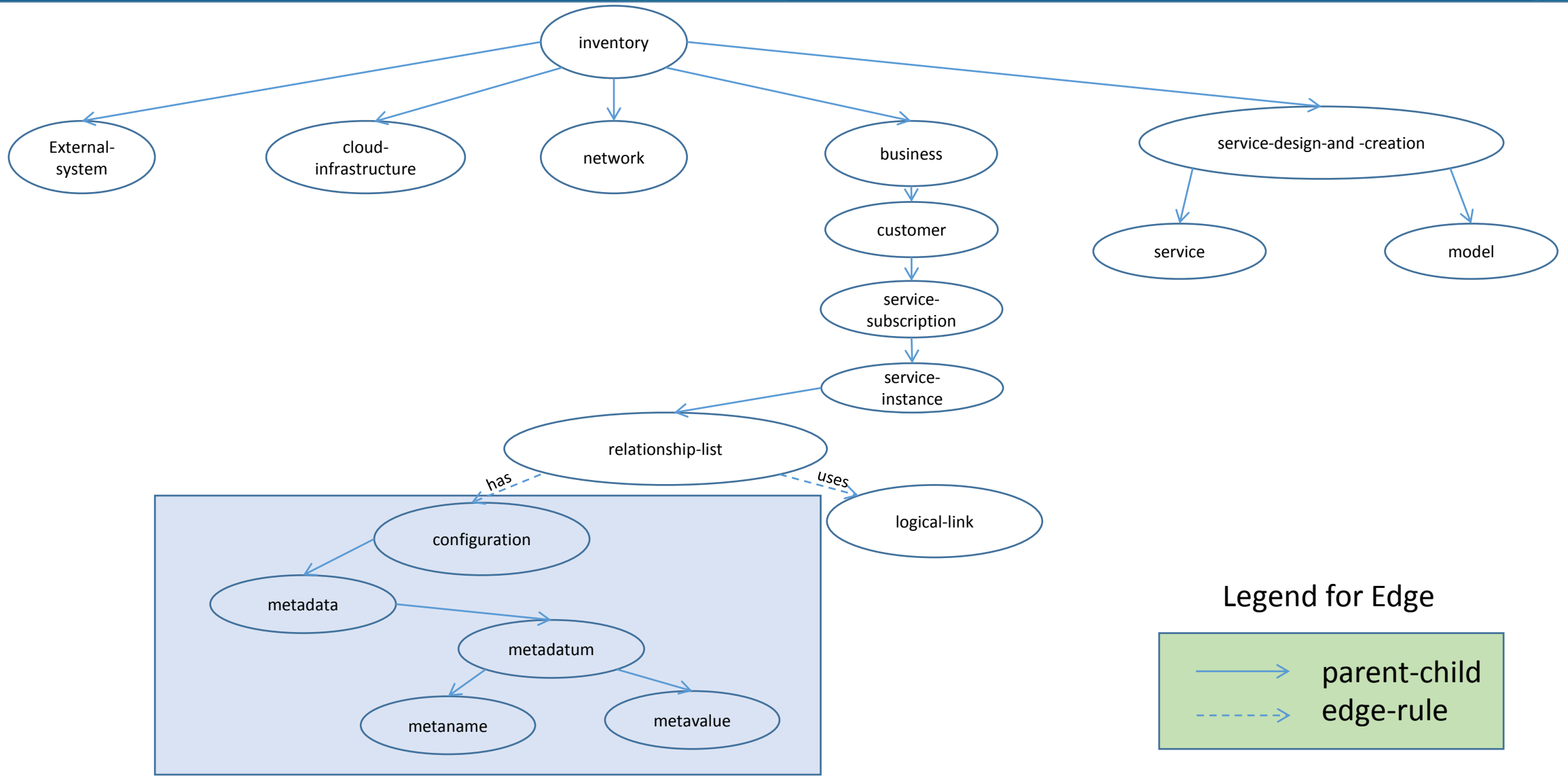
VoLTE E2E Service Design



VoLTE Service Instantiation

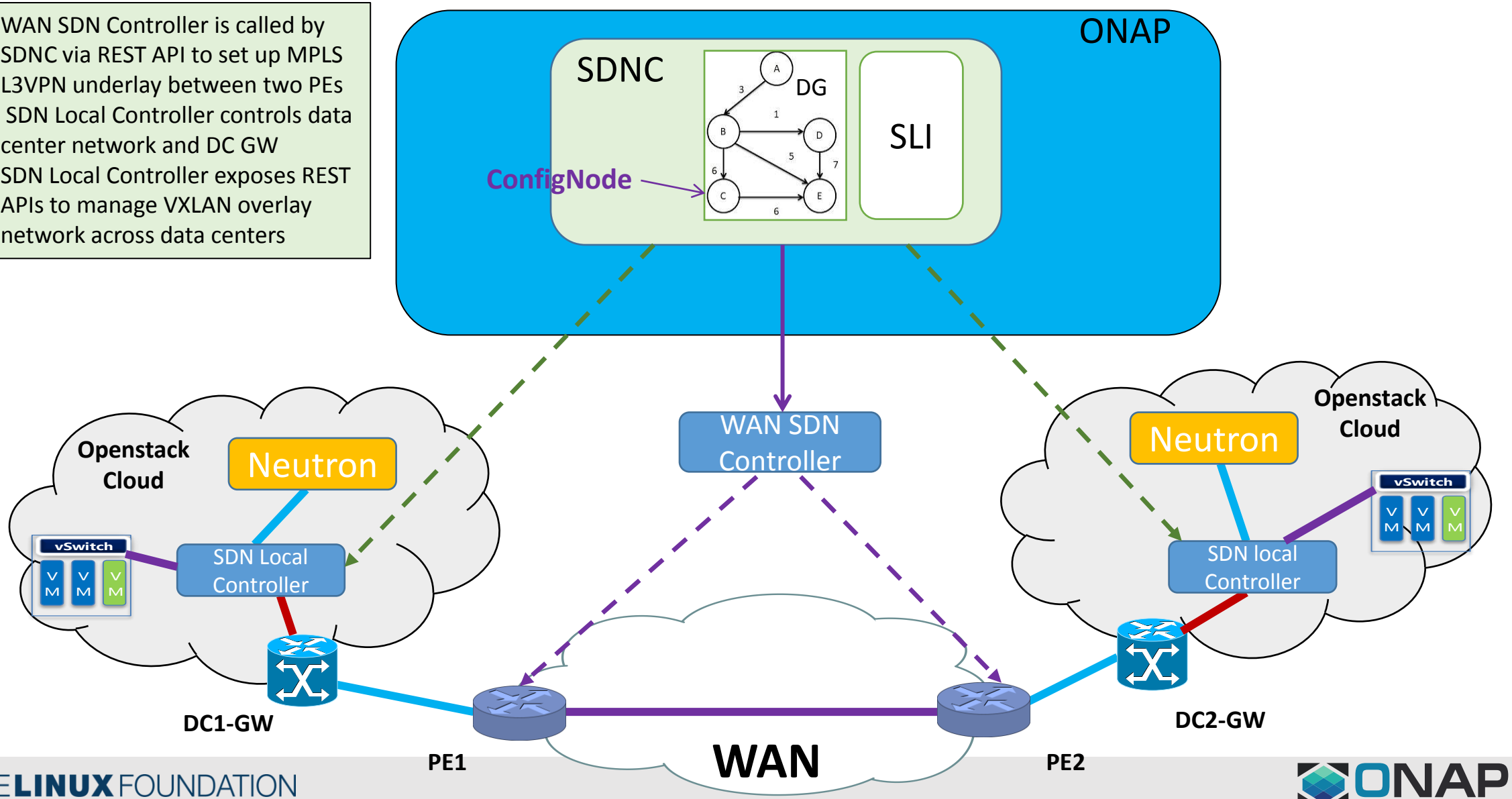


A&AI Nodes Used by SDNC to Store Underlay and Overlay Parameters



VoLTE Use Case DCI Overlay and Underlay Network Orchestration

- WAN SDN Controller is called by SDNC via REST API to set up MPLS L3VPN underlay between two PEs
- SDN Local Controller controls data center network and DC GW
- SDN Local Controller exposes REST APIs to manage VXLAN overlay network across data centers



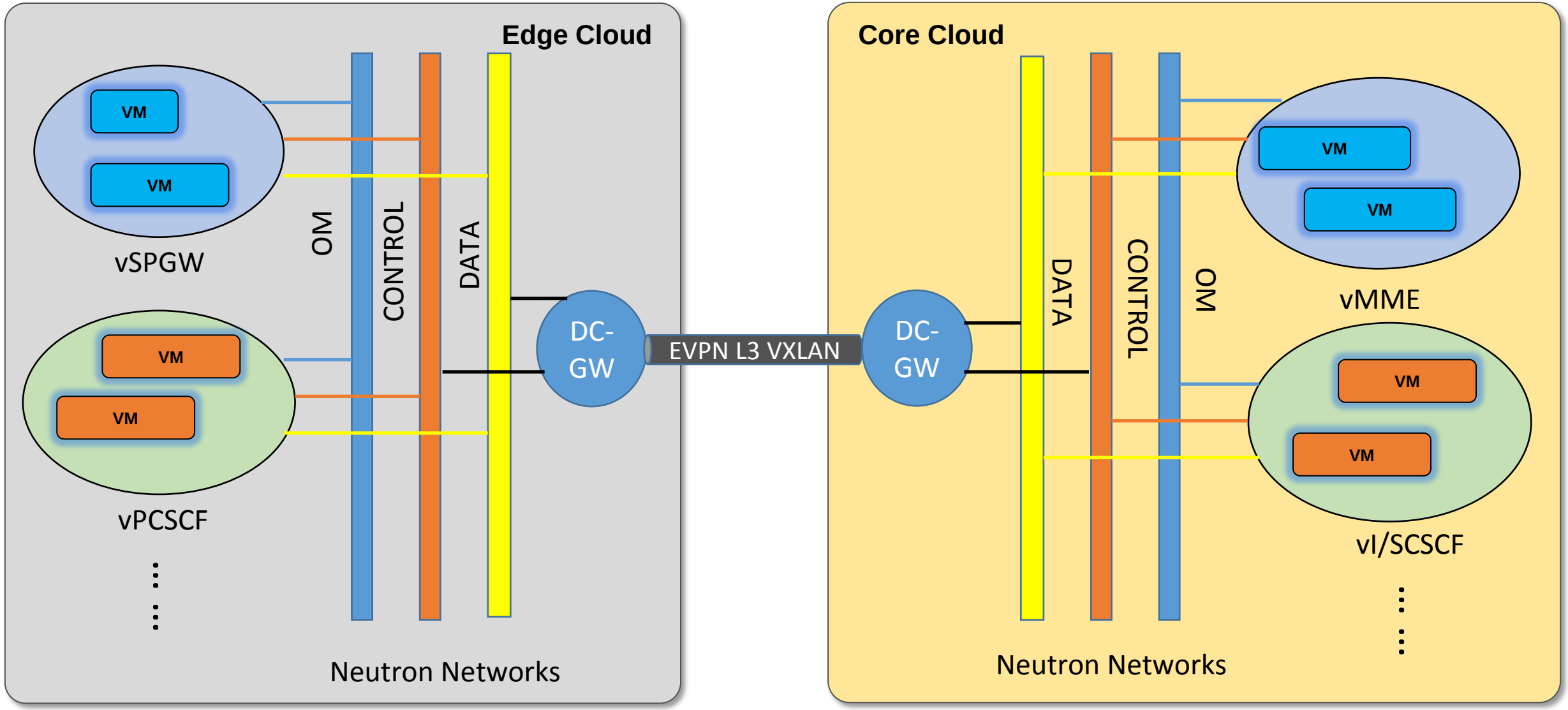
REST API to Create DCI L3 VXLAN Overlay

POST local-controller-ip:/v2.0/l3-dci-connects

```
{
    “l3-dci-connect": {
        “id": "CDD702C3-7719-4FE6-A5AD-3A9C9E265309",                  //UUI or user will generate
        "name": "PODX-routerY",                                        // UUI input by User
        "description": "VPC A connect VPC B",                          // UUI input by User
        "router_id": "CBB702C3-6789-1234-A5AD-3A9C9E265309",           //neutron router id from SO. TOR or vRouter?
        "firewall_enable": "false“                                     //false
        "local_networks": ["8a41319d-87cf-4cd6-8957-f4a1066c63a8"],    //neutron network ids
        "local_network_all ":false,                                    //false
        "evpn_irts": ["1:5000"],                                       //input from UUI
        "evpn_erts": ["1:5000" ],                                      //input from UUI
        "l3_vni": "5001",                                              //input from UUI
    }
}
```

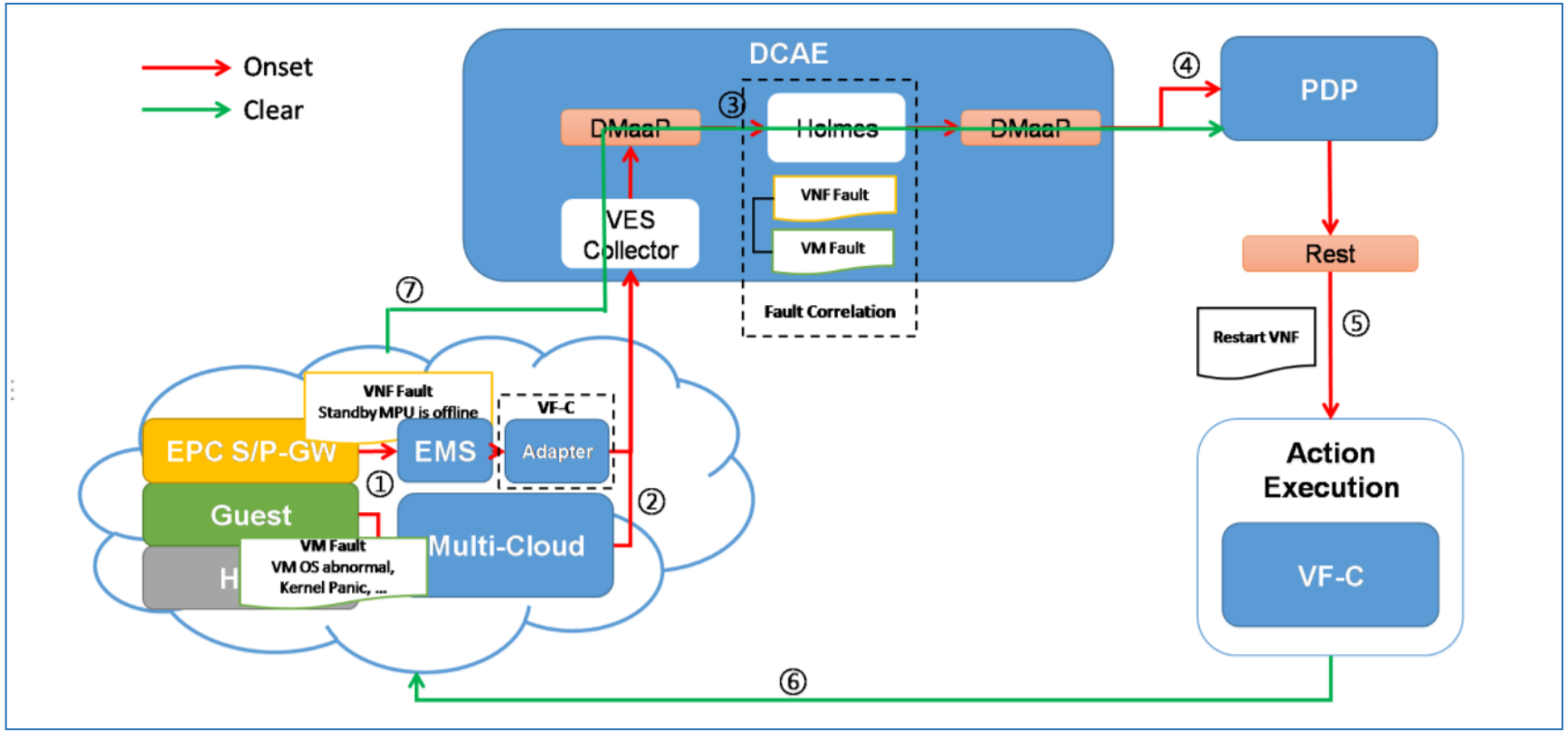
* VXLAN is actually set up manually. The API above is used to exchange routes after tunnel setup

Networks Interconnected by DCI EVPN L3 VXLAN Overlay



- VXLAN and EVPN between two DC-GWs are manually set up before VoLTE service is instantiated
- vRouter and TOR switches are not shown in the picture

Closed Loop Message Flow



Steps:

1) Multi-cloud sends alarm via VES client to DCAE in real time.
2) EMS collects and reports alarms of VNFs in real time
   - EMS collect alarms for VNF and report to EMS driver
   - EMS driver receive alarm and report/notify to DCAE via VES client
3) VES collector receive alarm data and publish to DMaaP BUS.
4) Holmes subscribes alarm topic from DMaaP and does the correlation according to the rules. Holmes publishes root cause event to DMaaP
5) Policy subscribes DMaaP event and matches to the operational rules, then invokes VF-C healing API
6) VF-C invokes S-VNFM to healing VNF via S-VNFM
7) EMS and Multi-cloud send CLEAR event

Holmes Rules for Closed Loop

```
"parameters": [
    {
        "name": "holmes.default.rule.volte.scenario1",
        "value": "package dcae.ves.test
            import org.onap.some.related.packages;
            rule"SameVNF_Relation_Rule"
            salience 120
            no-loop true
                when
                    $root : VesAlarm(
                        $sourceId: sourceId, sourceId != null && !sourceId.equals(""),
                        specificProblem in ( "LSS_cpiPCSCFFailReg(121297)", "LSS_cpiSIPRetransmitInvite(120267)" ),
                        $eventId: eventId)
                    $child : VesAlarm( eventId != $eventId,
                        CorrelationUtil.getInstance().isTopologicallyRelated(sourceId, $sourceId),
                        specificProblem in ("LSS_externalLinkDown(4271)","LSS_failedAttachReqsRateExceeded(4272)"),
                        this after [-60s, 60s] $root)
                then
                    DmaapService.publishResult(...);
            end",
        "description": "The default rule for the VoLTE usecase."
    },
```

* In the dedicated duration, only if Holmes receives both parent and child alarms will it publish root alarm based on the alarm correlation rules

DCAE Closed Loop Event Sent by Holmes


```
{
   "closedLoopEventClient": "DCAE.HolmesInstance",
   "policyVersion": "1.0.0.5",
   "policyName": "vVOLTE",
   "policyScope": "resource=volte,service=VolteSErvice,type=SampleType,closedLoopControlName=CL-VOLTE-SIG-d925ed73-8231-4d02-9545-db4e101f88f8",
   "target_type": "VM",
   "AAI": {
      "vserver.vserver-name": “TBD",
      "service-instance.service-instance-id" : "TBD",
      "generic-vnf.vnf-id" : "TBD",
      "generic-vnf.vnf-name" : "TBD"
   },
   "closedLoopAlarmStart": 1484677482204798,
   "closedLoopEventStatus": "ONSET",
   "closedLoopControlName": "CL-VOLTE-SIG-d925ed73-8231-4d02-9545-db4e101f88f8",
   "version": "1.0.2",
   "target": "vserver.vserver-name",
   "requestID": "97964e10-686e-4790-8c45-bdfa61df770f",
   "from": "DCAE"
}
```

Closed Loop Operational Policy Example

```
controlLoop:
  version: 2.0.0
  controlLoopName: CL-VOLTE-SIG-d925ed73-8231-4d02-9545-db4e101f88f8
  trigger_policy: unique-policy-id-1-restart
  timeout: 3600

policies:
  - id: unique-policy-id-1-restart
    name: Restart the VM
    description:
    actor: VFC
    recipe: Restart
    target:
    type: VM
    retry: 3
    timeout: 1200
    success: final_success
    failure: final_failure
    failure_timeout: final_failure_timeout
    failure_retries: final_failure_retries
    failure_exception: final_failure_exception
    failure_guard: final_failure_guard
```

Test Cases

Please review VoLTE test cases

https://wiki.onap.org/pages/viewpage.action?pageId=11928104

Thank You!