



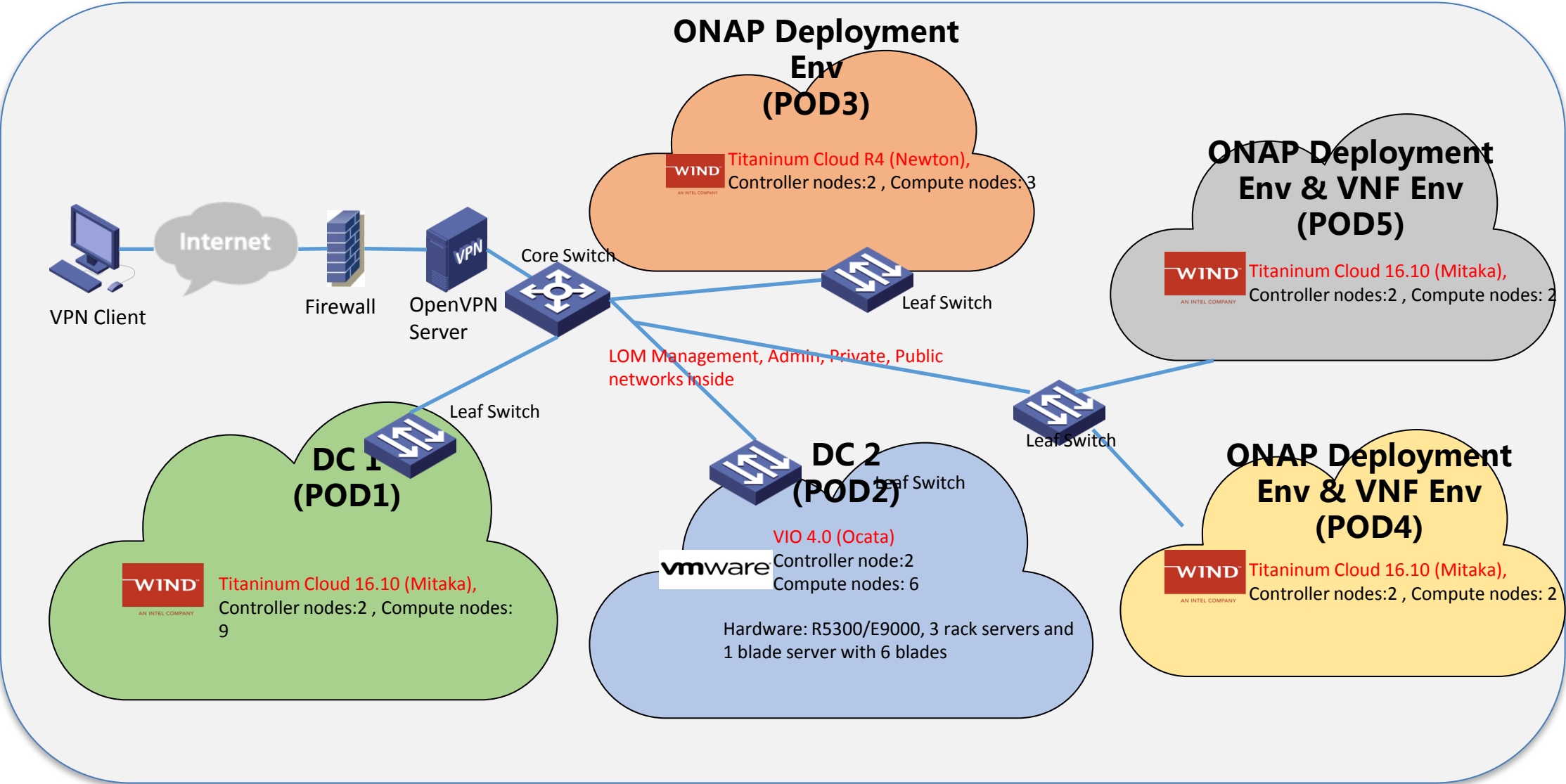
ONAP Casablanca Use Case in CMCC Lab

China Mobile: Yan Yang

Intel: Haibin Huang

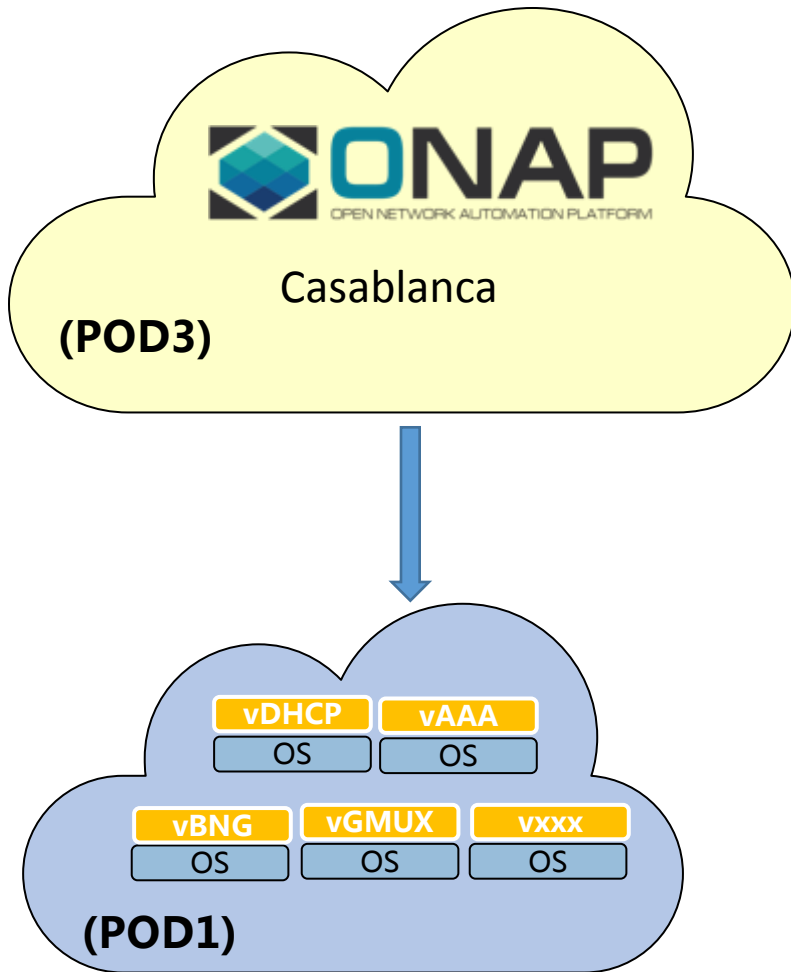
CMCC Lab Overview

Physical Lab Overview @CMCC

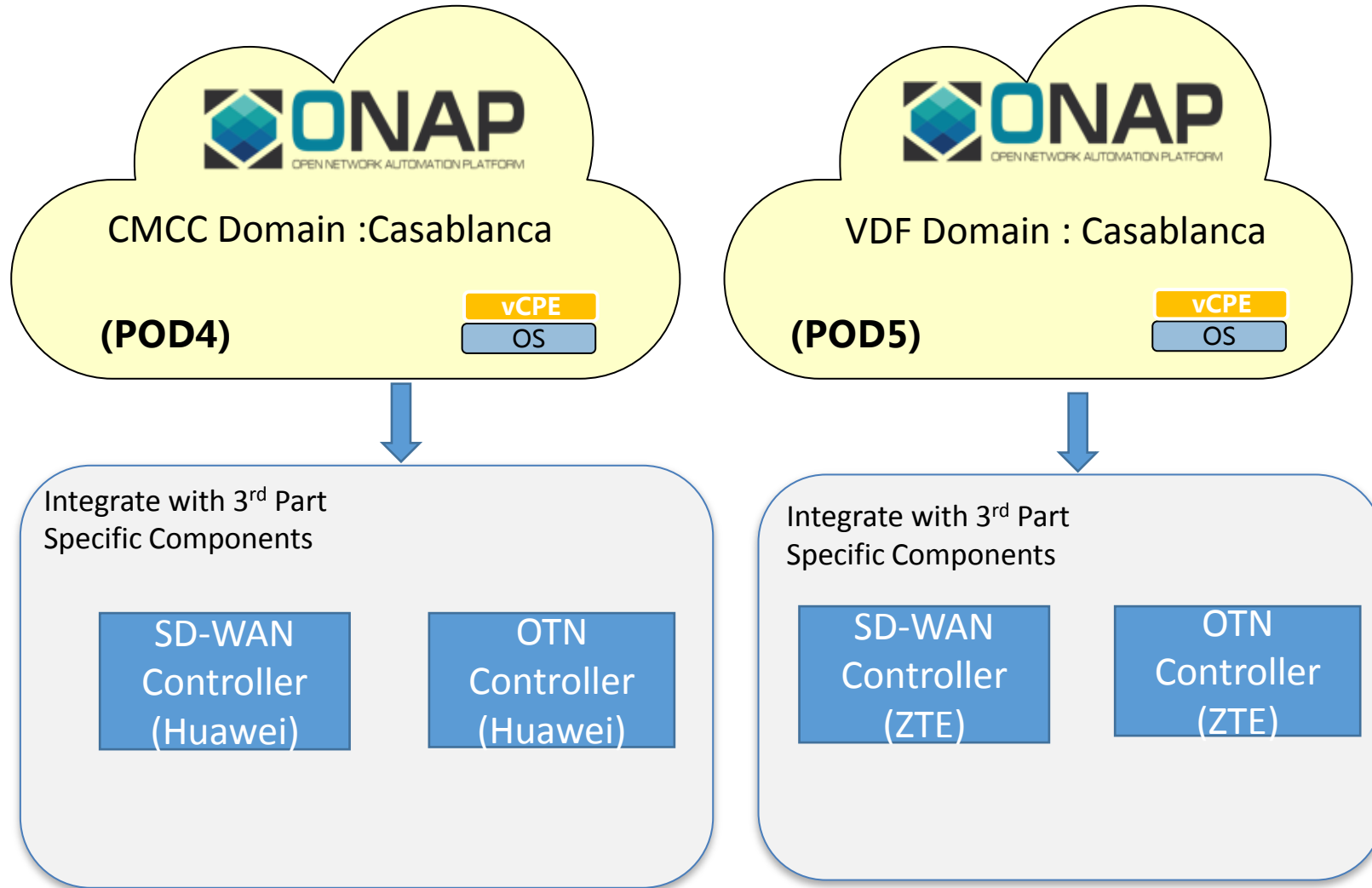


Use Case Deployment @CMCC

vCPE



CCVPN





Key Data @CMCC

CMCC Lab key data

➤ ONAPs : 3 

➤ Cloud : 5  (4) and  (1)

➤ Physical hardware : 73  

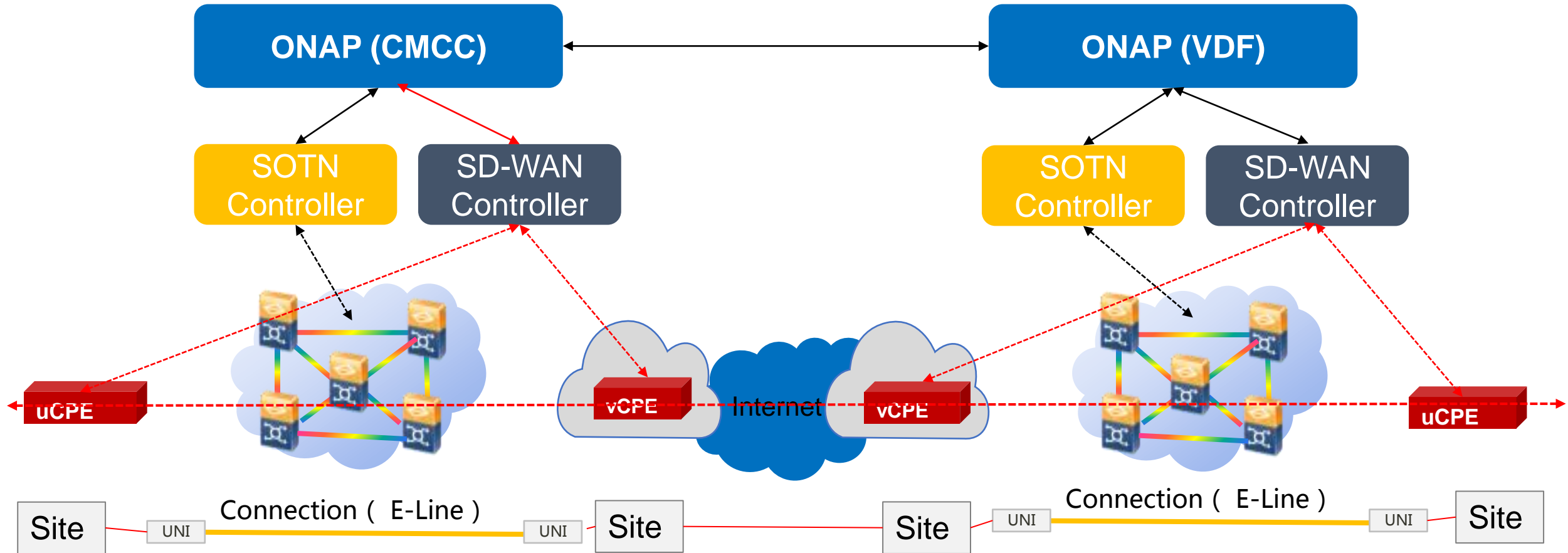
➤ Participating member : 8



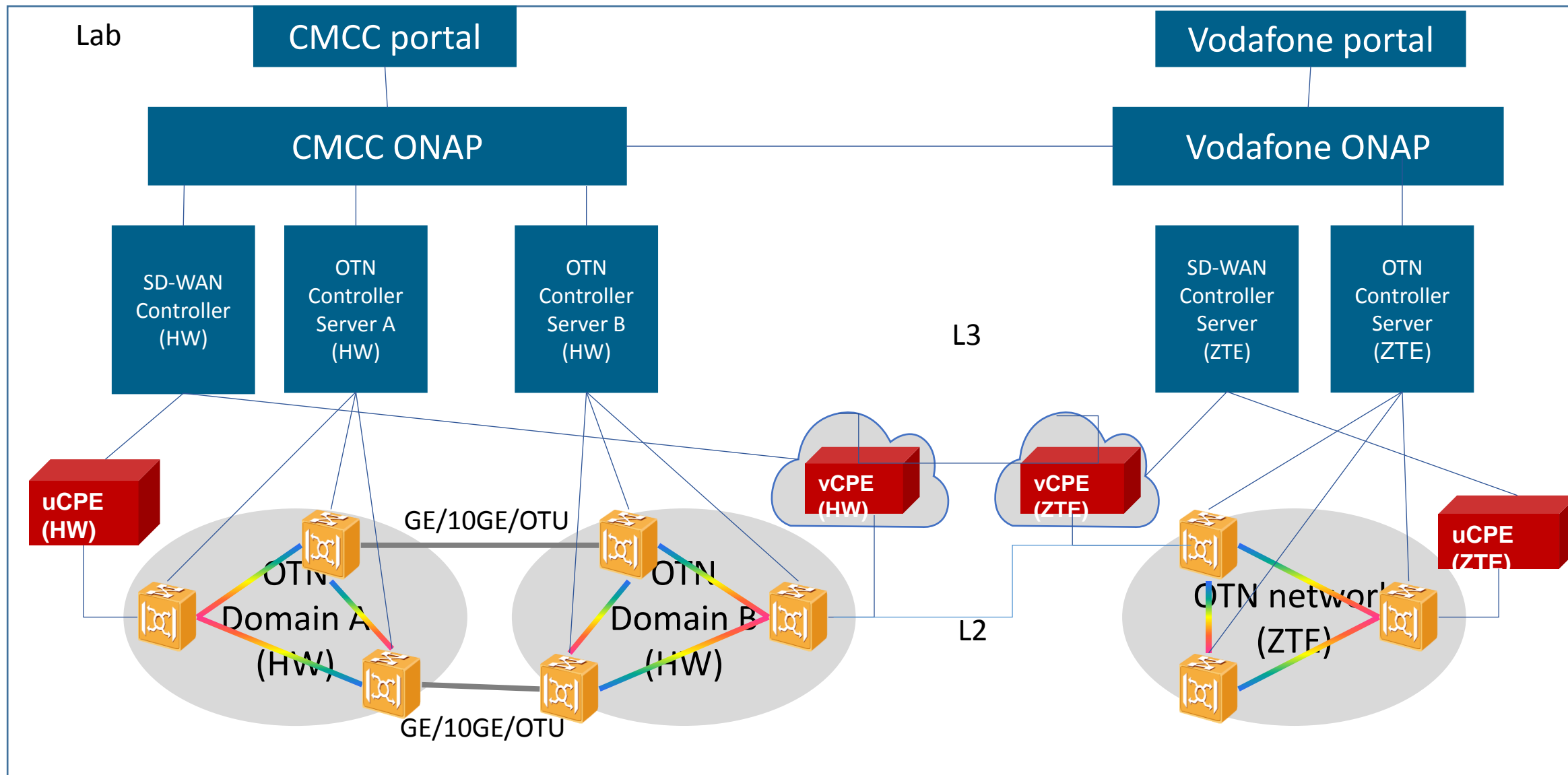
CCVNP Use Case

CCVPN Overview

CCVPN(Cross Domain and Cross Layer VPN) **by ONAP Peering Orchestration Between SPs**

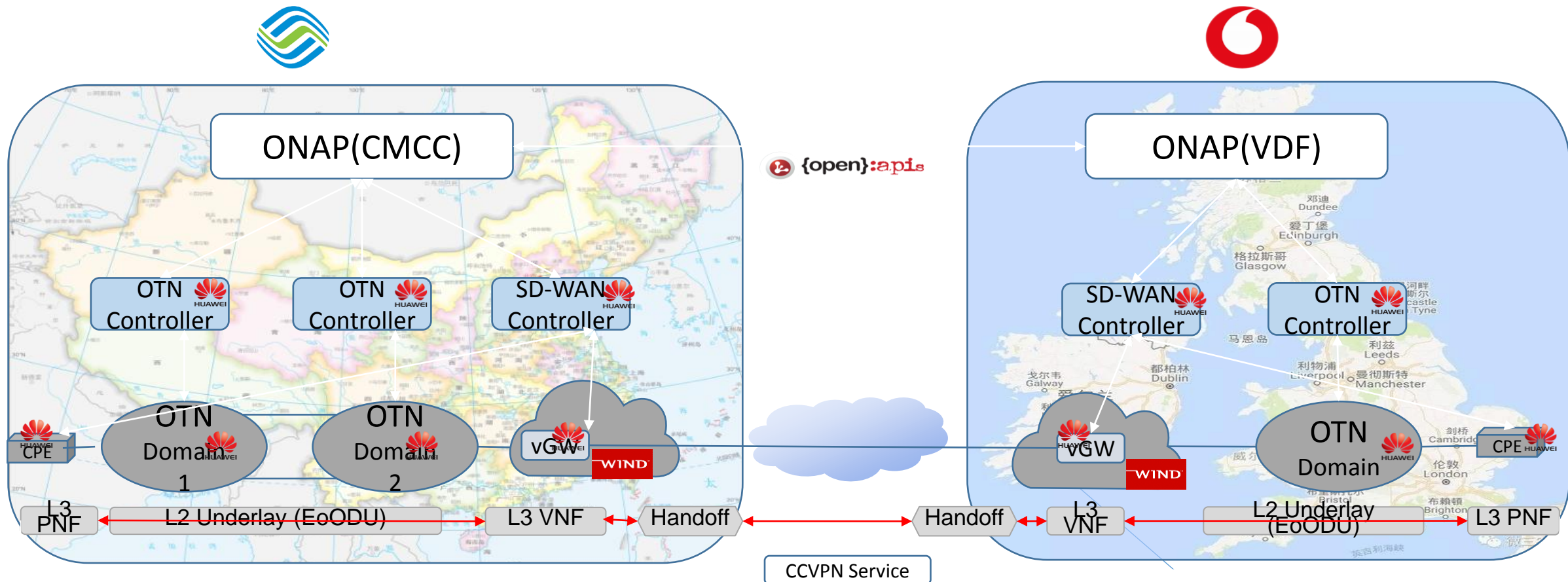


CCVPN Network Topology



CCVPN Test Case

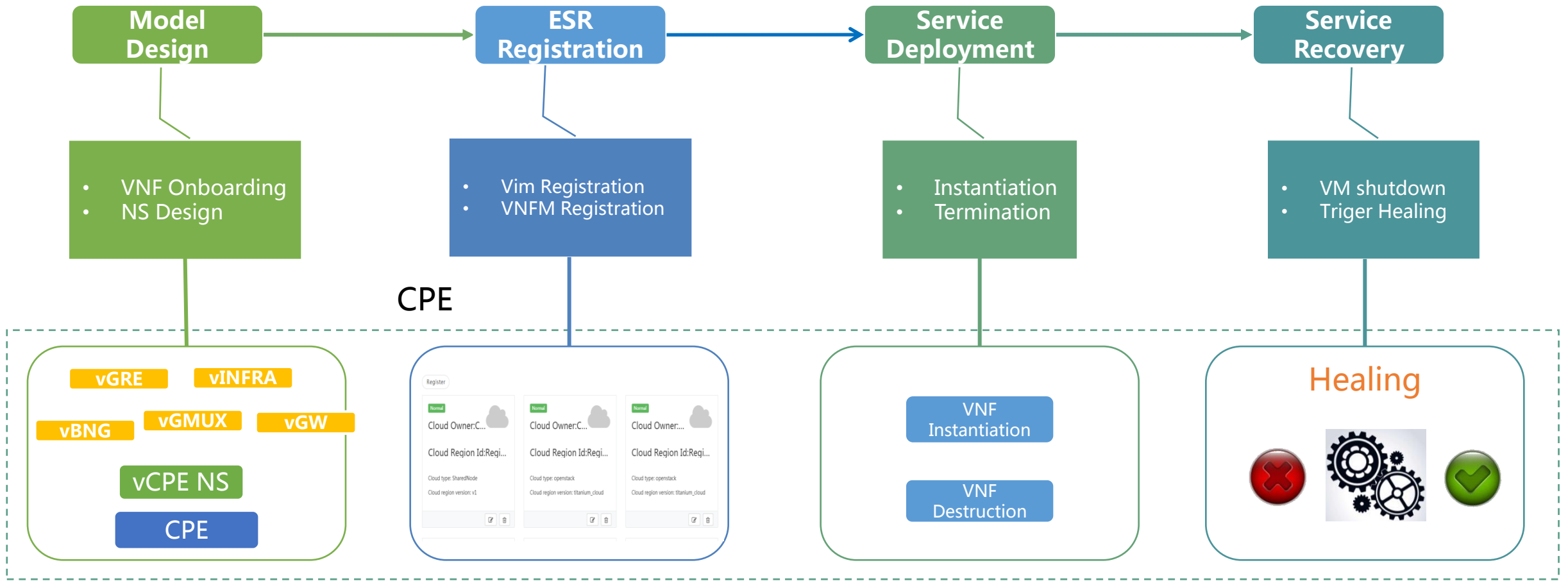
- Topology Discovery
- Service Provisioning
- Closed Loop

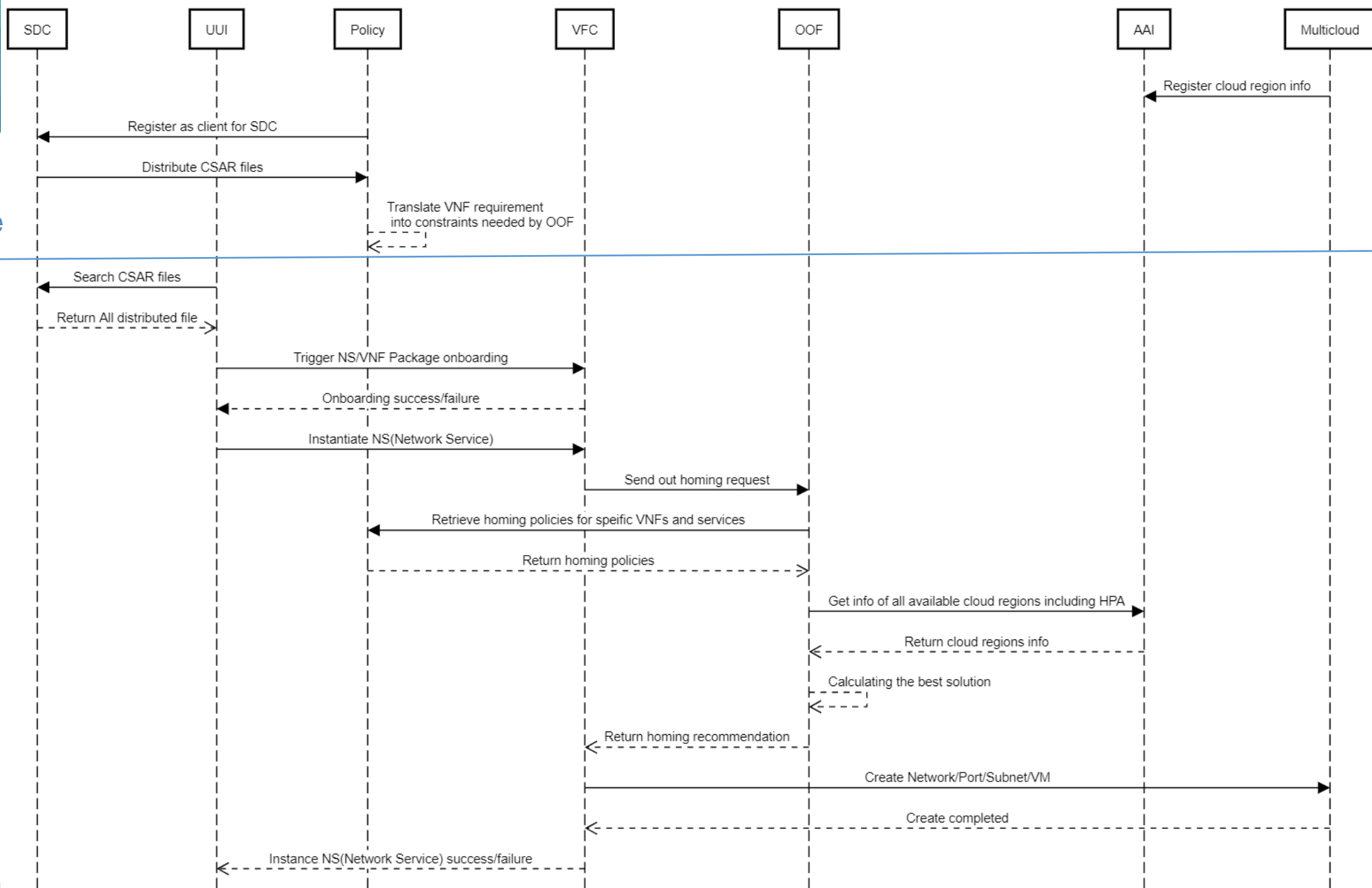


vCPE Use Case

vCPE Test Case

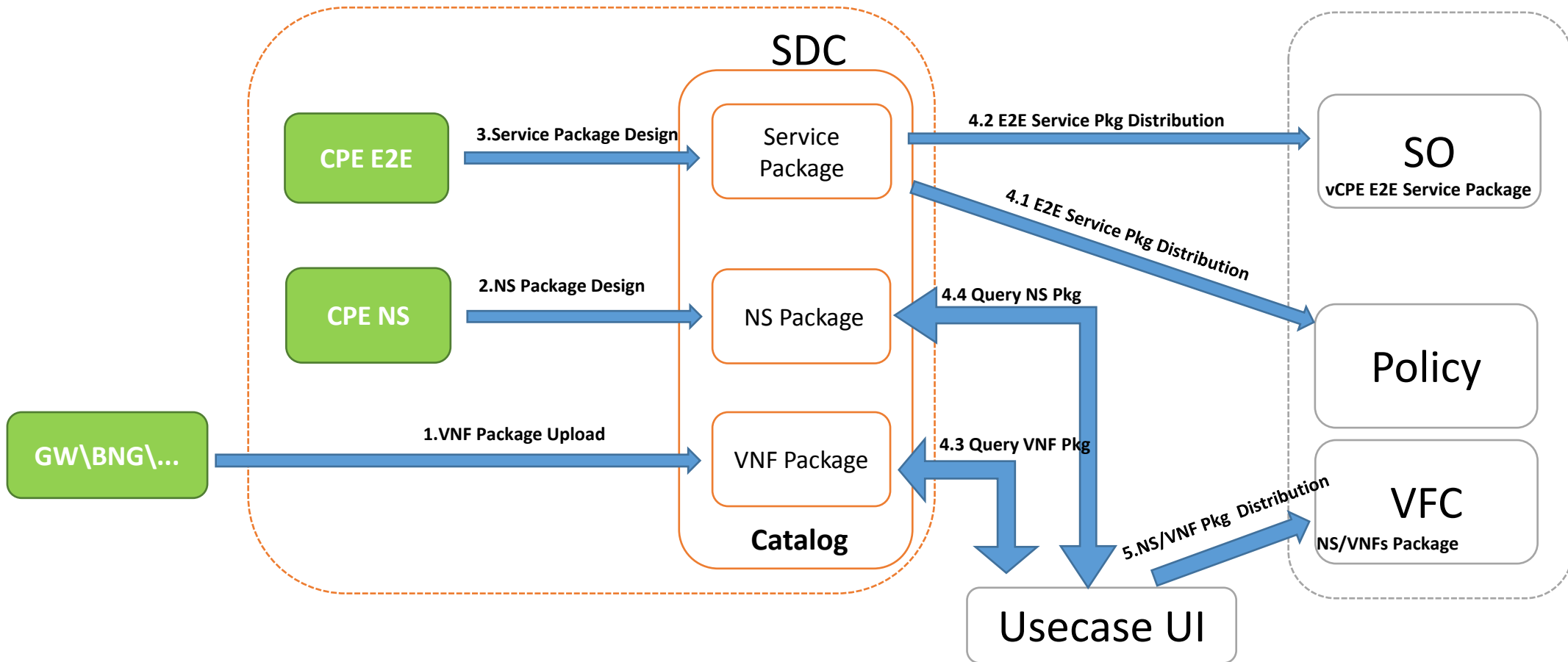
Key Test Case: ESR Registration, Model Design, Service Deployment, Service Recovery





Design Time

vCPE – Package Design & Distribution



vCPE – Package Design & Distribution



Home UI X What are you looking for?

NS VNF PNF

VNF Package Onboarding in UI

Onboard VNF

Click or drag file to this area to upload

Start Upload

NO	Name	Version	Onboarding State	Operational State	Usage State	Operation button
38037a12-a0d4-4aa4-ac50-cd6b05ce0b24			CREATED	NOT_IN_USE	DISABLED	

Home UI X What are you looking for?

NS VNF PNF

NS Package Onboarding in UI

Onboard NS

Click or drag file to this area to upload

Start Upload

NO	Name	Version	Onboarding State	Operational State	Usage State	Operation button
0b667470-e6b3-4ee8-8f08-186317a04dc2			CREATED	DISABLED	NOT_IN_USE	
4287208c-d125-450c-ab9e-5ea9b0315429			CREATED	DISABLED	NOT_IN_USE	

ESR Registration

VIM Configuration

- NIC configure

We can refer to [\[1\]](#) before “Launching instances with SR-IOV ports”

```
pci_passthrough_whitelist = { "devname": "eth3", "physical_network": "physnet2" }
```

- Create aggregate

```
$ openstack aggregate create --property sriov_nic=sriov-nic-intel-1234-5678-physnet1:1 aggr11
```

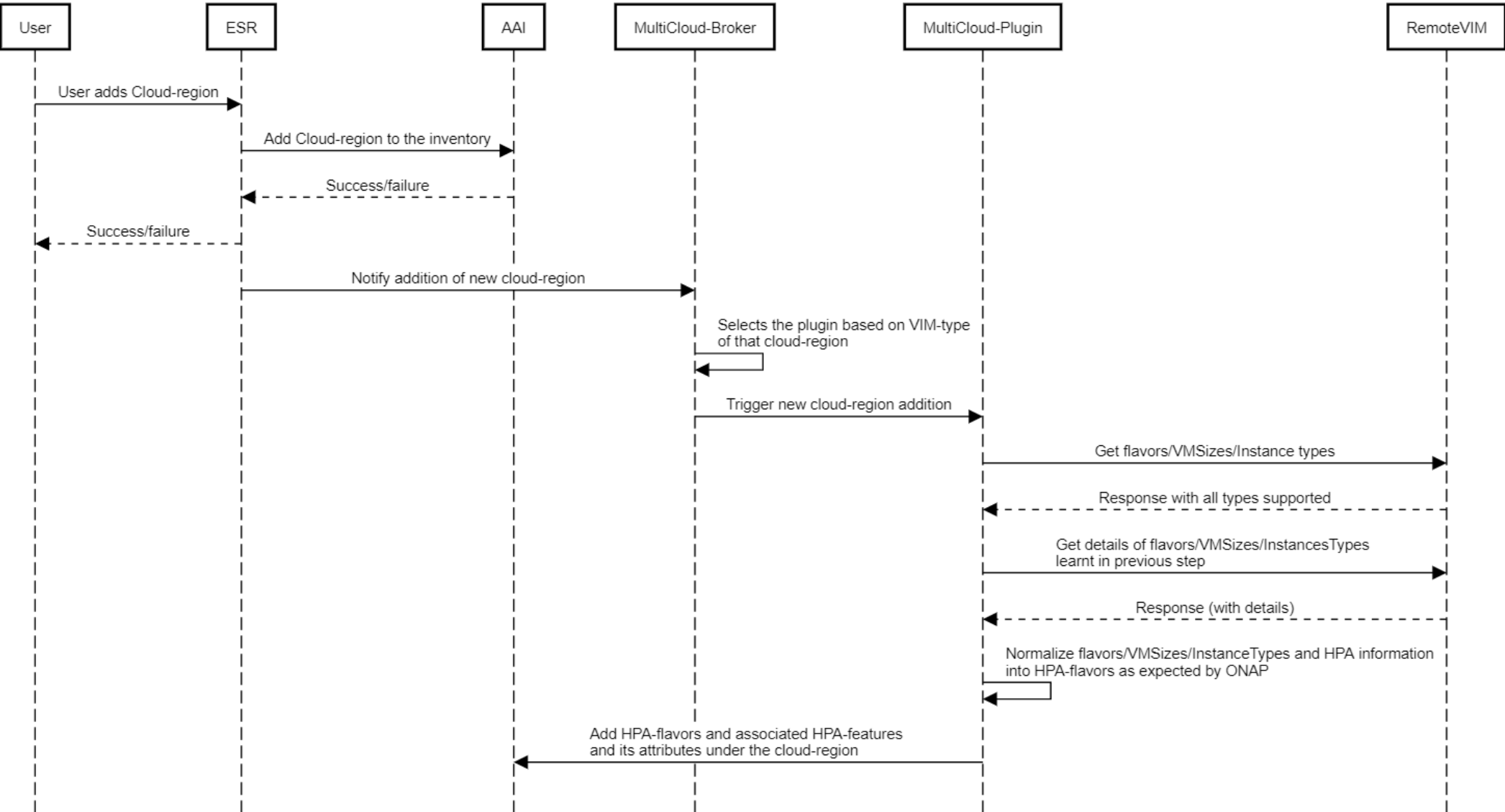
- Create flavor

```
$ openstack flavor create onap.flavor1 --id auto --ram 512 --disk 40 --vcpus 4
```

```
$ openstack flavor set onap.flavor1 --property sriov_nic=sriov-nic-intel-1234-5678-physnet1:1
```

- Create provider network

MultiCloud register and HPA discovery



VIM & GVNFM Registration via ESR GUI

The image displays two browser windows from the ESR GUI. The left window, titled 'vimView.html', shows the 'VIM Registration' form. The right window, titled 'vnfmView.html', shows the 'GVNFM Registration' form. Both forms contain various input fields for configuration.

VIM Registration Form Fields:

- Cloud Owner *: CPE-DC
- Cloud Region Id *: RegionOne
- Cloud Type *: openstack
- Cloud Region Version *: titanium_cloud
- Owner Defined Type *: t1
- Cloud Zone *: z1
- Physical Location Id *: cli1
- username *: vCPE
- password:
- Auth Url: http://172.30.1.2:5000/v3
- ssl Cacert: (empty)
- ssl Insecure: true
- Cloud Domain: Default
- Default Tenant: vEPC

GVNFM Registration Form Fields:

- Name *: GVNFM DRIVER
- type *: gvnfmdriver
- Vendor *: vfc
- version *: v1.0
- URL *: http://10.43.45.18:80/
- VIM: CPE-DC_RegionOne
- certificate URL: (empty)
- Username: (empty)
- Password: (empty)

Buttons: 'cancel' and 'save' are visible at the bottom right of the GVNFM form.

AAI information

```
hpa-capability-id="ty53fd3d-0b15-11w4-81b2-6210efc6dff9",
```

```
hpa-feature="sriovNICNetwork",
```

```
architecture="intel64",
```

```
hpa-version="v1",
```

hpa-attribute-key	hpa-attribute-value
pciCount	{value: 1}
pciVendorId	{value: "1234"}
pciDeviceId	{value: "5678"}
physicalNetwork	{value:"physnet1"}

sriov_nic=sriov-nic-<vendor>-<Vendor ID>-<Device ID>-<physicalNetwork:COUNT>

It is expected that Openstack administrator creates alias that starts with sriov and put the vendor ID, device ID.

Example:

Assume that there are two SRIOV-NIC cards supported by a region, Intel and Mellanox.

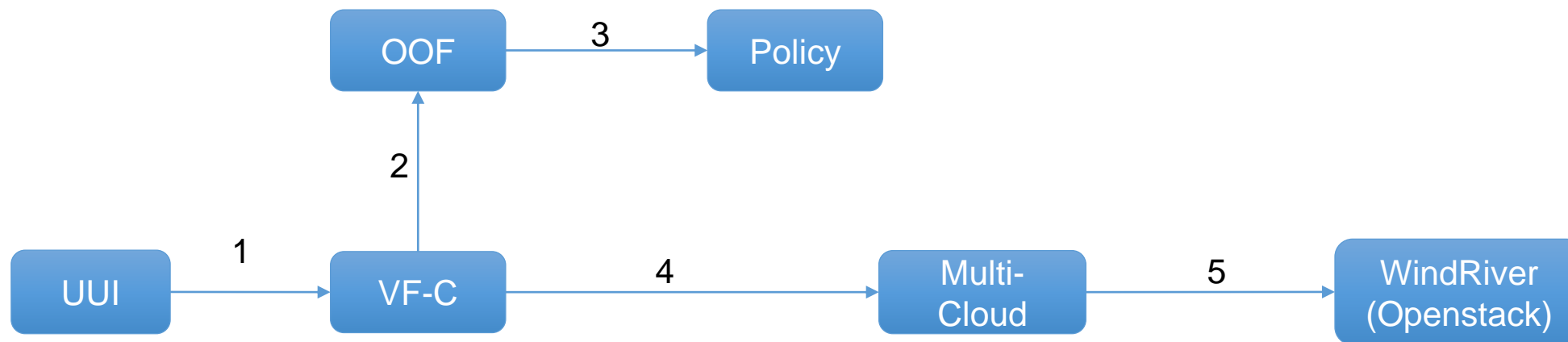
sriov-nic-intel-1234-5678-physnet1:1

sriov-nic-mellanox-2345-6543-physnet1:1

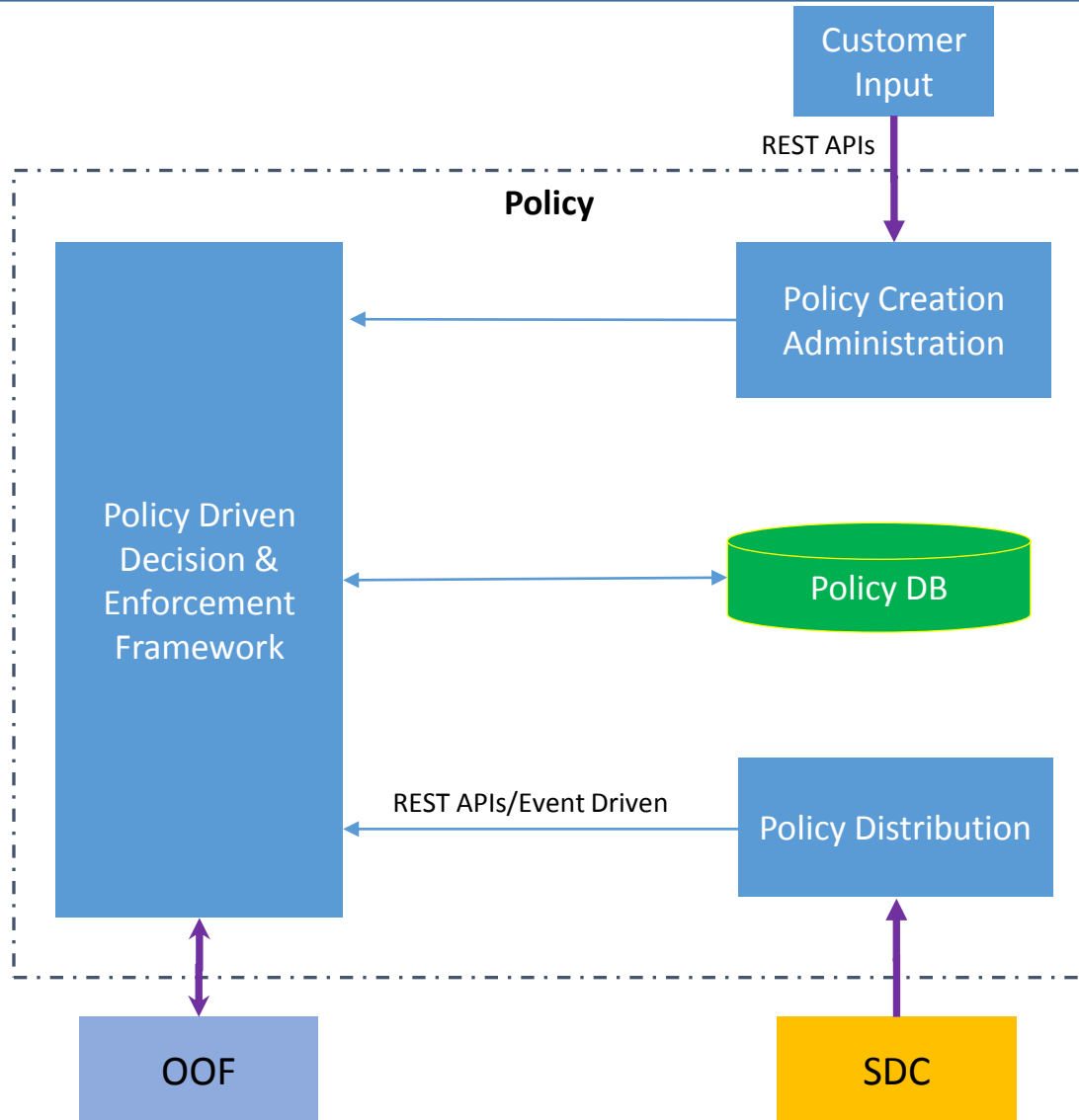
Run Time

ONAP – Run Time

Lifecycle: Instantiate/terminate/heal...



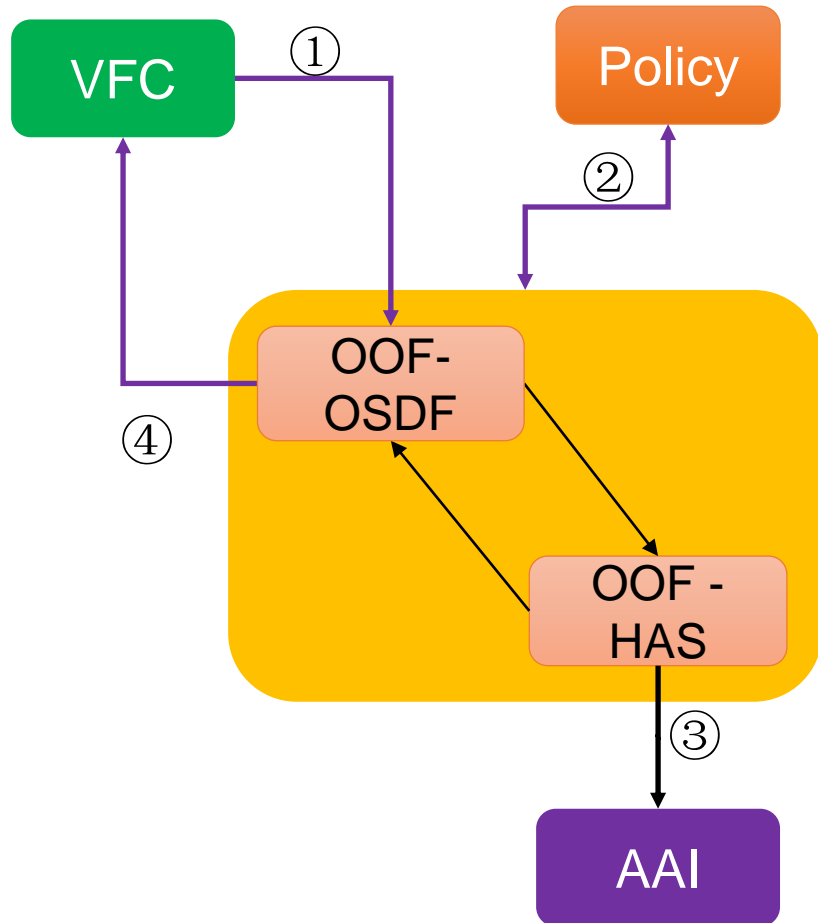
Policy



```
"flavorLabel": "vcpe.vgw",
"flavorProperties": [{
  "hpa-feature": "sriovNICNetwork",
  "mandatory": "True",
  "architecture": "generic",
  "hpa-version": "v1",
  "directives": [
    {
      "type": "sriovNICNetwork_directives",
      "attributes": [
        {"attribute_name": "vgw_nic_type", "attribute_value": "direct"},
        {"attribute_name": "vgw_provider_network", "attribute_value": "physnet1"}
      ]
    }
  ]
},
"hpa-feature-attributes": [
  {
    "hpa-attribute-key": "pciVendorId",
    "hpa-attribute-value": "1234", "operator": "=", "unit": ""
  },
  {
    "hpa-attribute-key": "pciDeviceId",
    "hpa-attribute-value": "5678", "operator": "=", "unit": ""
  },
  {
    "hpa-attribute-key": "pciCount",
    "hpa-attribute-value": "1", "operator": ">=", "unit": ""
  },
  {
    "hpa-attribute-key": "physicalNetwork",
    "hpa-attribute-value": "physnet1", "operator": "=", "unit": ""
  }
]
},
]
```

VF-C — OOF Homing

OOF: ONAP Optimization Framework
OSDF: Optimization Service Design Framework
HAS: Homing Allocation Service



1. VFC sends out homing request to OOF(OSDF) containing resource info
2. OOF(OSDF) pulls all the related homing constraints from Policy
3. OOF(HAS) check AAI database to pull region(flavor) information
4. OOF(OSDF) returns homing allocation solution to VFC

OOF Homing

```
hpa-capability-id="ty53fd3d-0b15-11w4-81b2-6210efc6dff9",
```

```
hpa-feature="sriovNICNetwork",
```

```
architecture="intel64",
```

```
hpa-version="v1",
```

hpa-attribute-key	hpa-attribute-value
pciCount	{value: 1}
pciVendorId	{value: "1234"}
pciDeviceId	{value: "5678"}
physicalNetwork	{value:"physnet1"}

[AAI](#)

```
"flavorLabel":"vcpe.vgw",  
"flavorProperties":[{"  
  "hpa-feature": "sriovNICNetwork",  
  "mandatory": "True",  
  "architecture": "generic",  
  "hpa-version": "v1",  
  "directives" : [  
    {  
      "type": "sriovNICNetwork_directives",  
      "attributes": [  
        {"attribute_name": "vgw_vnic_type", "attribute_value": "direct"},  
        {"attribute_name": "vgw_provider_network", "attribute_value": "physnet1"}  
      ]  
    }  
  ],  
  "hpa-feature-attributes": [  
    {  
      "hpa-attribute-key": "pciVendorId",  
      "hpa-attribute-value": "1234", "operator": "=", "unit": ""  
    },  
    {  
      "hpa-attribute-key": "pciDeviceId",  
      "hpa-attribute-value": "5678", "operator": "=", "unit": ""  
    },  
    {  
      "hpa-attribute-key": "pciCount",  
      "hpa-attribute-value": "1", "operator": ">=", "unit": ""  
    },  
    {  
      "hpa-attribute-key": "physicalNetwork",  
      "hpa-attribute-value": "physnet1", "operator": "=", "unit": ""  
    }  
  ]  
}],  
]
```

[Policy](#)

VFC Create network and port

```
"type": "sriovNICNetwork_directives",  
"attributes": [  
  {"attribute_name": "vgw_vnic_type", "attribute_value": "direct"},  
  {"attribute_name": "vgw_provider_network", "attribute_value": "physnet1"}]
```

a. Create network, ports

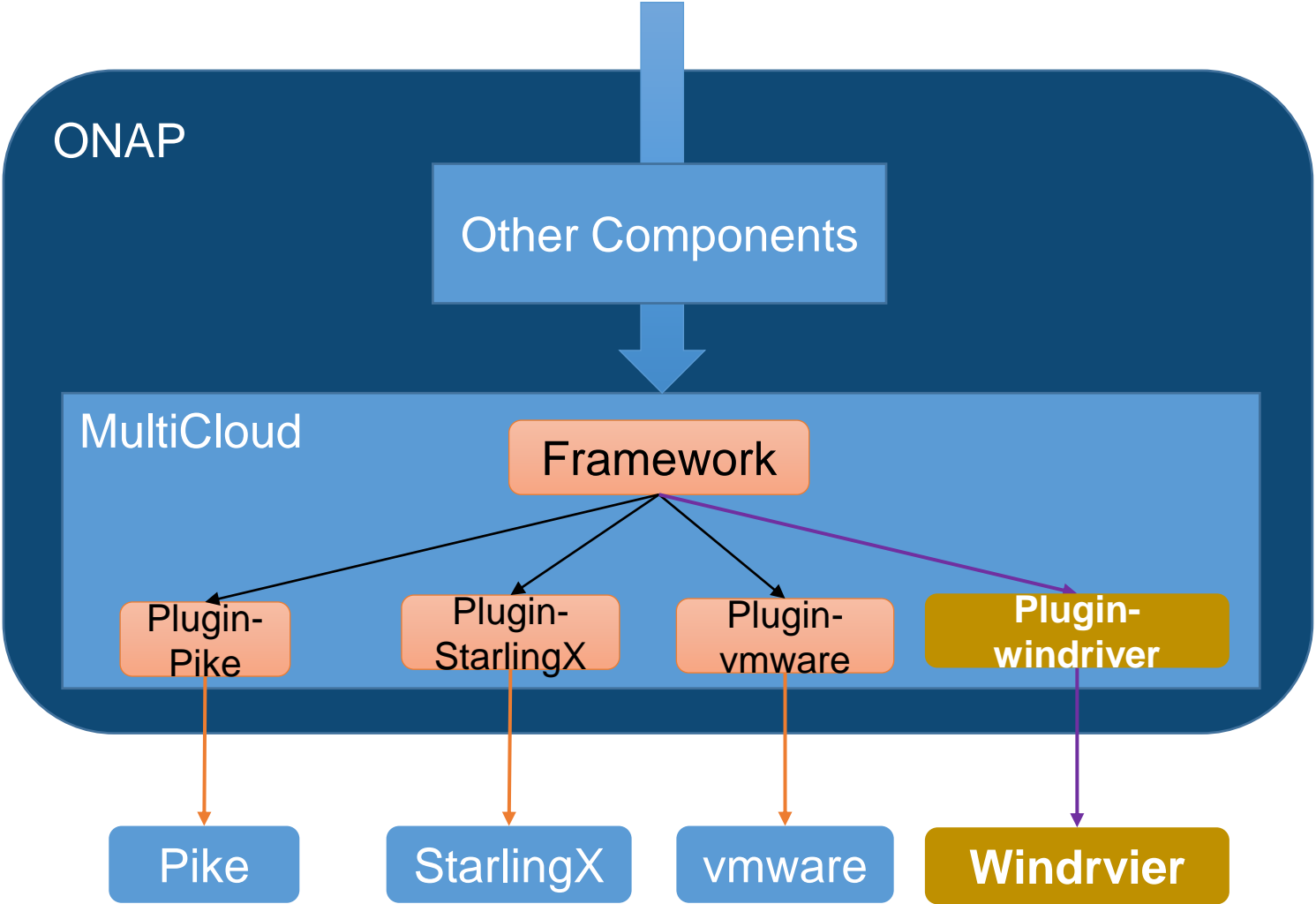
```
# openstack network create --provider-network-type vlan --provider-physical-network physnet1 tst-  
network
```

```
# openstack port create --name sriov-port --binding:vnic-type direct
```

b. Create server

```
# openstack server create --flavor onap.hpa.flavor11 --image Ubuntu_16.04 --nic port-id= test-sriov-nic
```

MultiCloud



SO and VFC will call Framework

Thank you!