Build ONOS VNF for FNCP Platform

발표자: 최진국
고려대학교

KOREA UNIVERSITY
Global KU - Frontier Spirit
KU is committed to becoming a leading world-class university

Contents

- FNCP Platform Introduction
- ONOS VNF Feature List
- ONOS VNF Design & Implementation
- Result & Demo

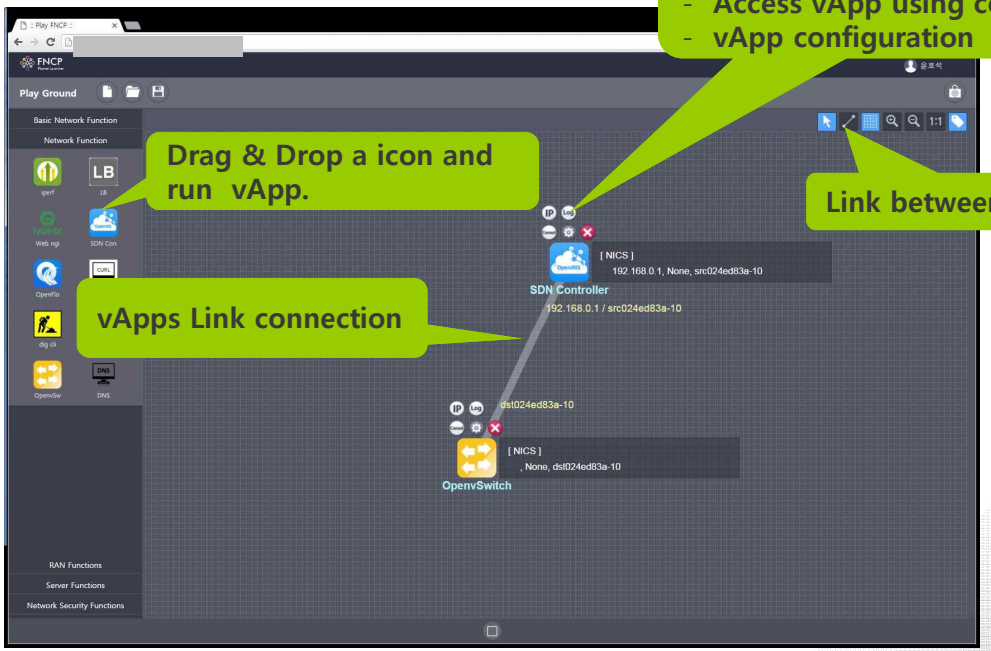
1. FNCP Platform Introduction

- NFV MANO Architecture
- FNCP Architecture
- FNCP Process

FNCP Platform Introduction – ETRI Project

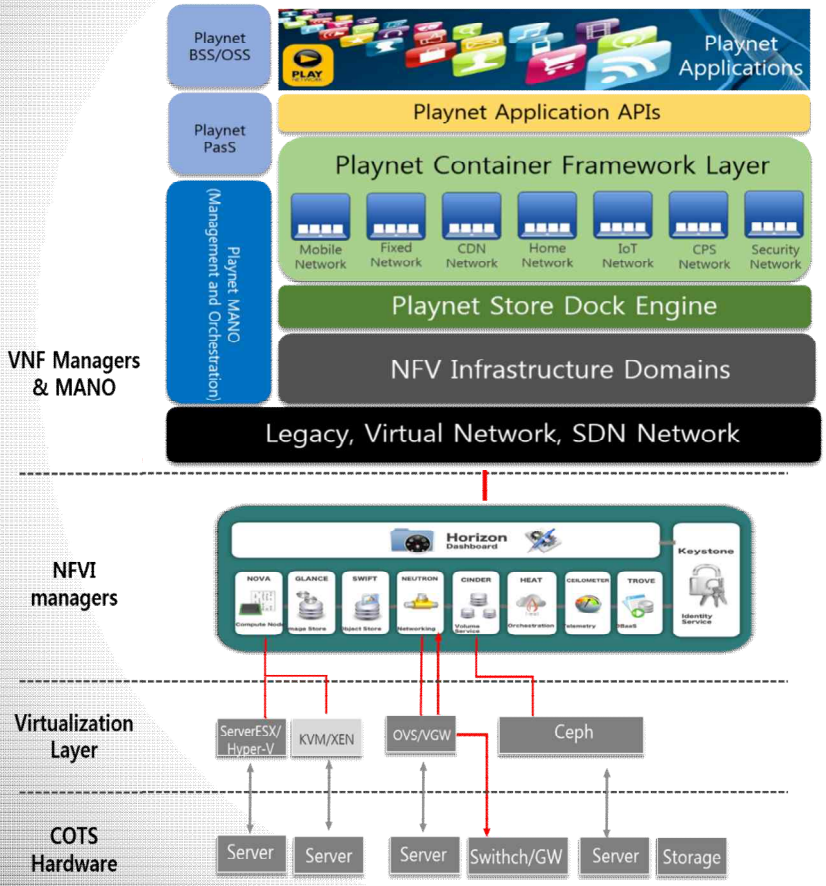
- FNCP(Future Network Computing Platform) is a cloud based network function test & development environment. It allows company or campus students to test/develop/verify network functions on the Platform.

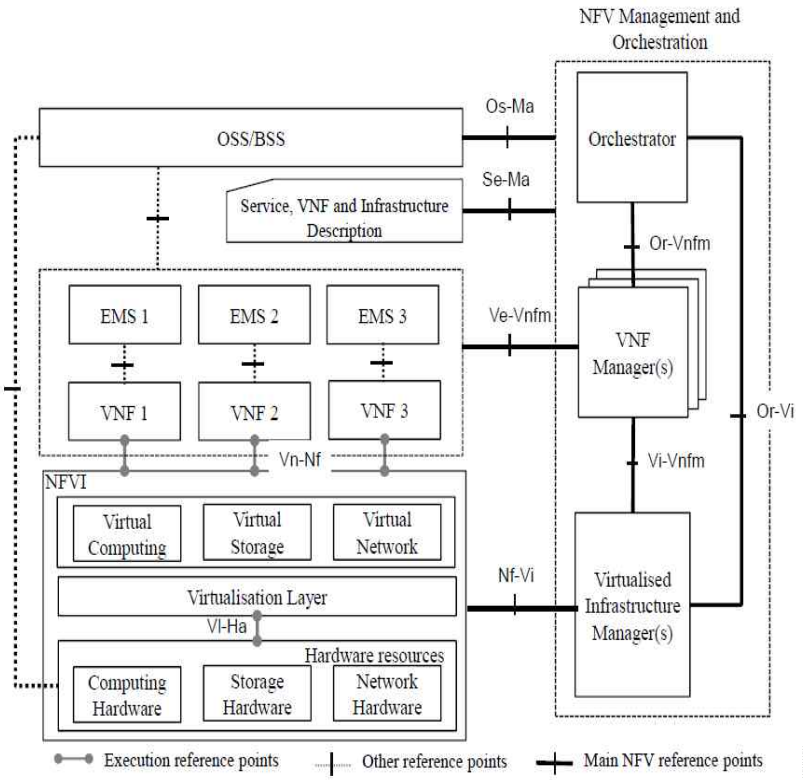
FNCP Playnet VNF & Project UI



FNCP Playnet UI

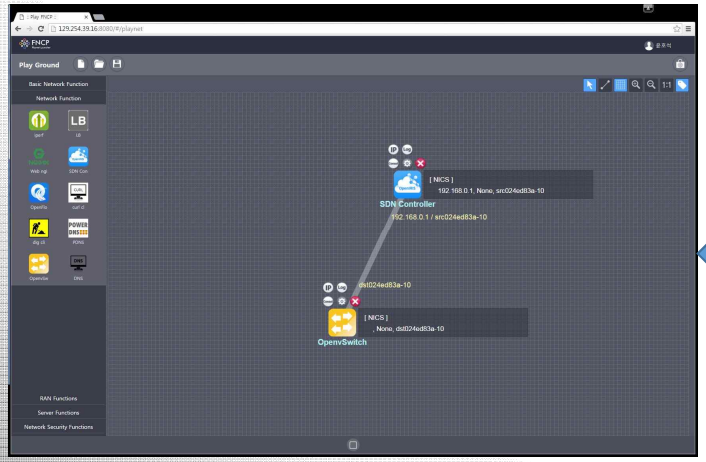
FNCP is based on NFV Architecture

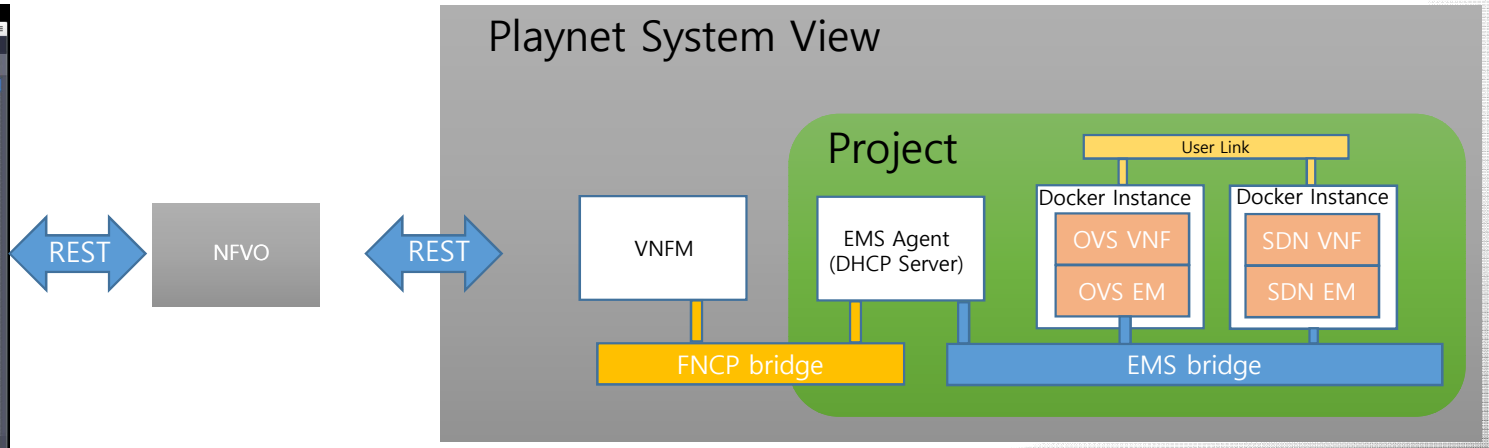




FNCP Features and Restrictions

- NFVO, VNFM, EMS communicate with each other via REST
- One EMS manages one VNF
- VNF is started with VNF image and saved configuration file
- When VNF is Initiated on the Playnet canvas, it will call *startVNF* function automatically.





FNCP Features and Restrictions cont'

Two type of functions

- **FNCP Playnet Function:** VNF Initiation, Link Create/Delete, Floating IP Management.
- **VNF Functions:** VNF Start, Stop, Save, VNF configuration.

Four files are required to deploy a VNF on the Playnet Store.

- **VNF Image**: (docker or vm) which contains EM and VNF
- **VNF Hardware Spec**: a XML file which defines VNF hardware spec and conf repository info.
- **VNF UI**: a JSON file which defines Input UI for VNF Functions
- **Logo**: a image file.

2. ONOS VNF Feature List

- ONOS VNF Definition
- ONOS VNF Features

ONOS VNF Definition

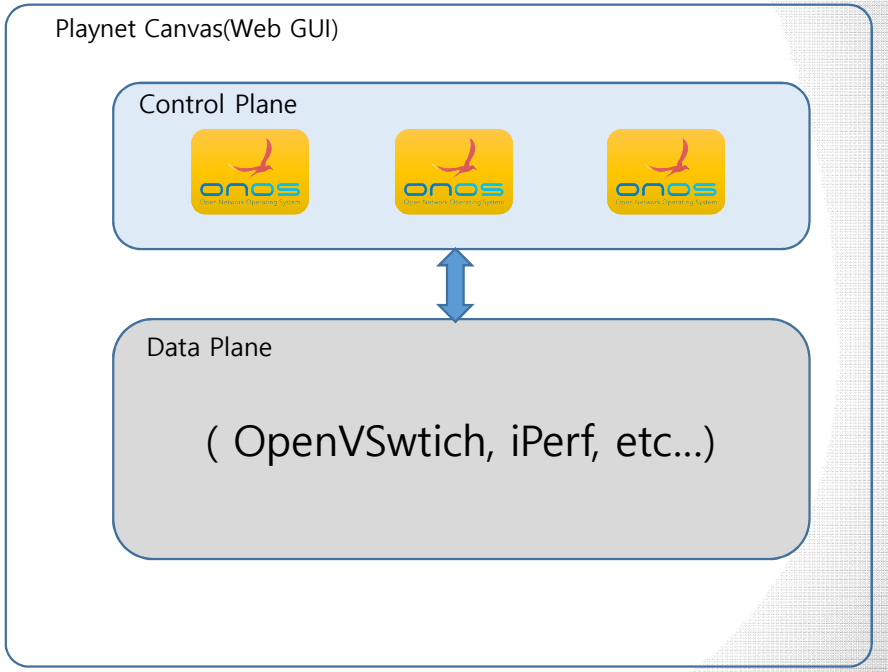
- ONOS VNF(for FNCP) is a set of process that is able to handle the message from VNFM and operates related ONOS Features.

General Features

- VNF Start, Stop, Status.

ONOS VNF Features

- ONOS restart
- ONOS app management(activate/de-activate/check-status)
- ONOS clustering



ONOS VNF Features

ONOS Restart

- Stop ONOS Process and Start again without delete & initiate VN
- Use case1: when ONOS VNF hardware resource is changed(ex: IP address).
- Use case2: when ONOS VNF is not running properly(ex: ONOS Web GUI not respond by socket connection issue).

ONOS App Management

- Customize ONOS app from the Playnet GUI
- Use case1: activate FWD(org.onosproject.acl) app to install flowrules to connected switches reactively.

ONOS Clustering

- Create ONOS cluster with several ONOS VNF
- Use case1: create ONOS cluster with 3 ONOS VNF

ONOS VNF Save & Load

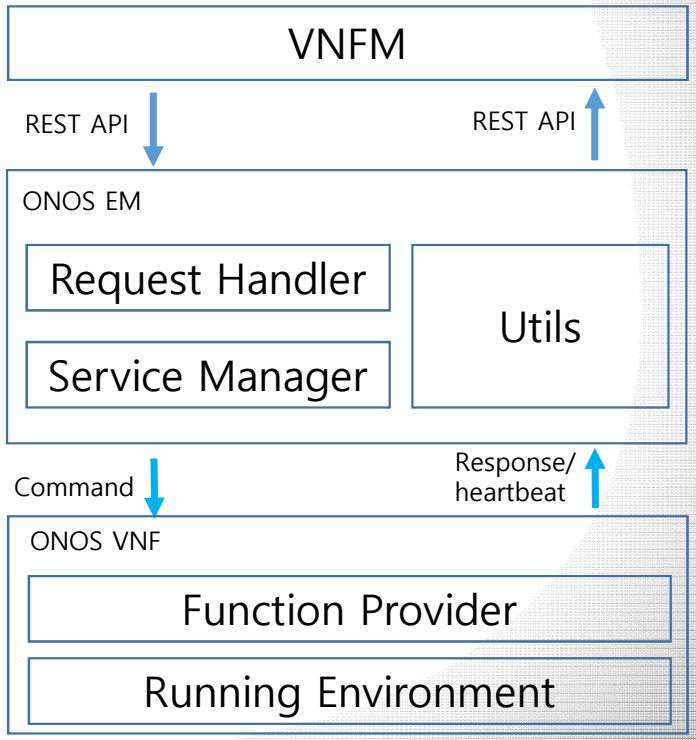
- In FNCP, user can save current Playnet project(which means save VNF configuration and links).
- There are two information need to store in FNCP DB.
  - App information: look for "active" file in each app folder.
  - Cluster information: "cluster.json"

3. ONOS VNF Design & Implementation

- ONOS VNF Design
- Sample Code

ONOS VNF Architecture

- Major Components:
  - VNFM
  - ONOS EM
    - Manages ONOS VNF life cycle
    - Call ONOS service
  - ONOS VNF
    - Provides ONOS Functions(ex: app activate, clustering)



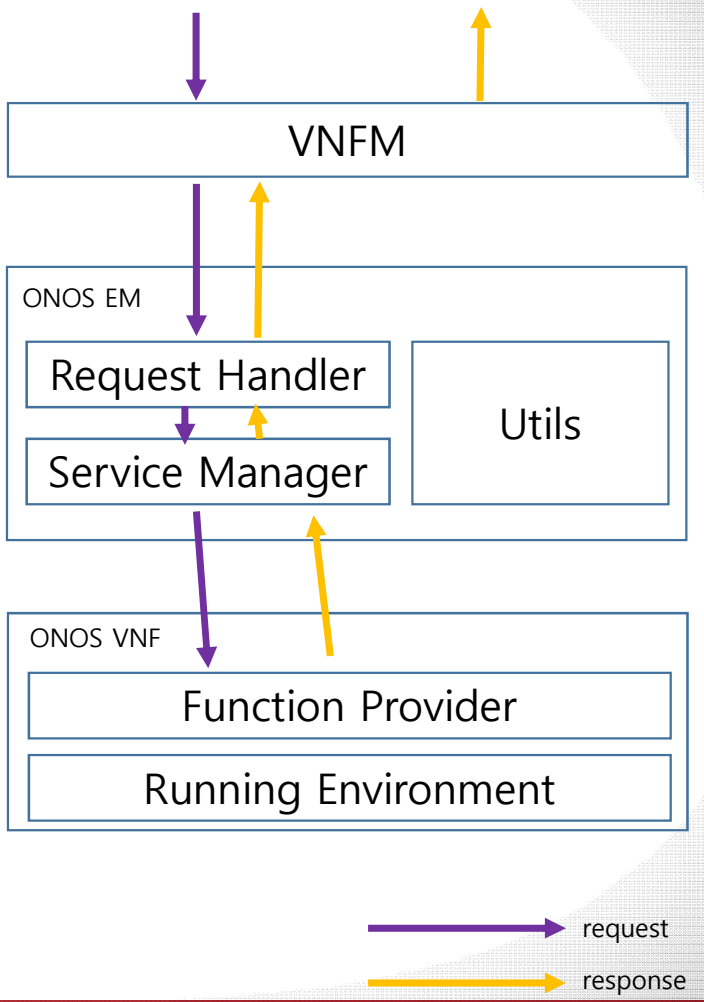
Process

(VNFM) Send request to *Request Handler*

(Request Handler) Parse request and call appropriate service

(Service Manager) Call one or many function of provider

(Function Provider) Execute functions and return result to *Service Manager*



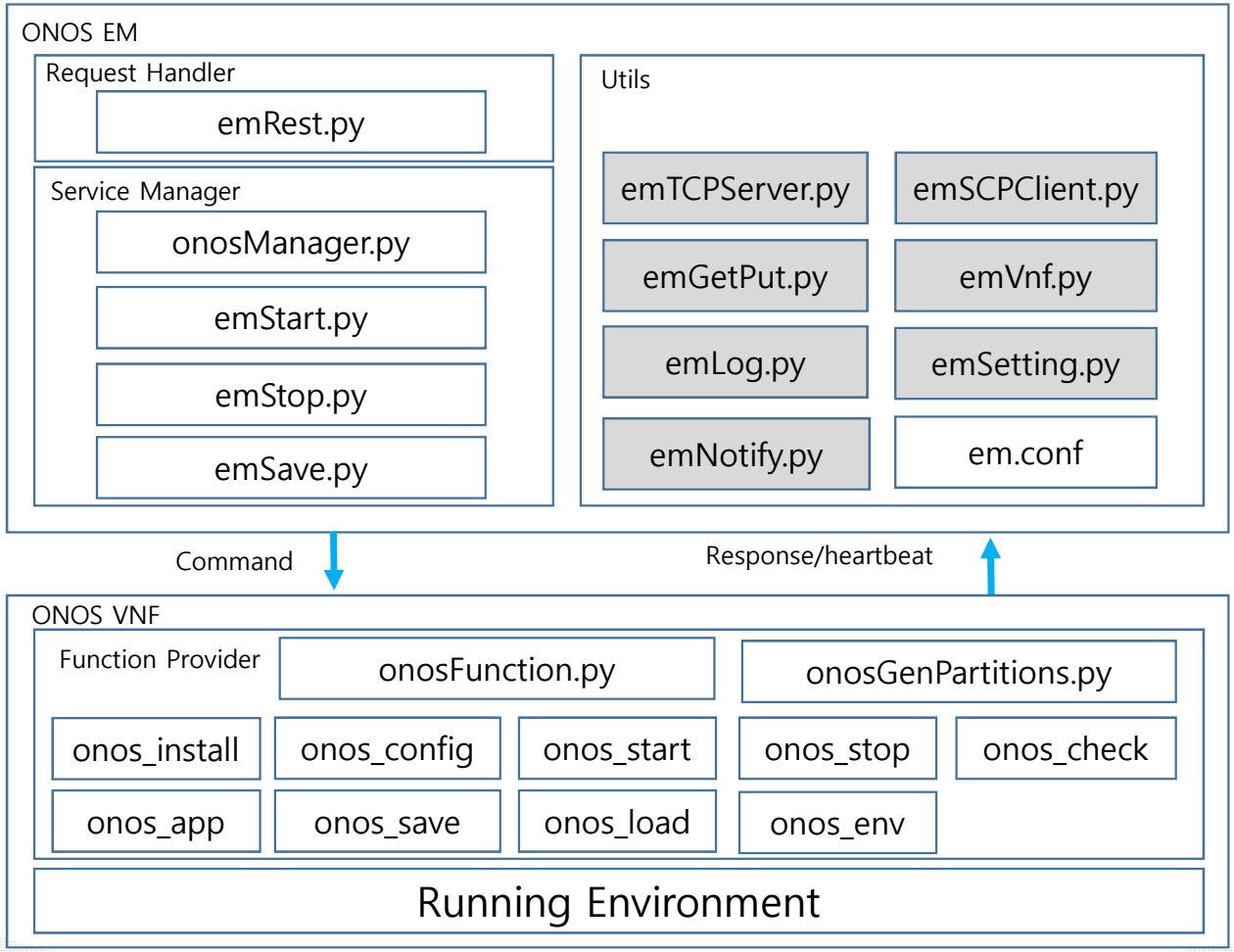
ONOS VNF Implementation for FNCP

- Framework and version
  - Docker: 1.9.1
  - Java: 1.8
  - Maven: 3.0.5
  - ONOS: 1.7.0

- ONOS EM: a python program, auto-started after VNF initialization
  - Request Handler: - emRest.py
  - Service Manager: emStart.py*, emStop.py*, emSave.py*, onosManager.py
- ONOS VNF:
  - Function Provider – a python file which is called by onosManager.py
  - Also, each functions(without clustering) implemented by a single bash file.

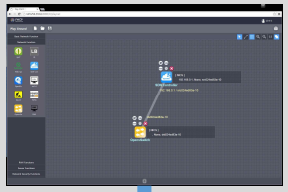
* Files are the original files from ETRI EM Template

ONOS VNFs Structure

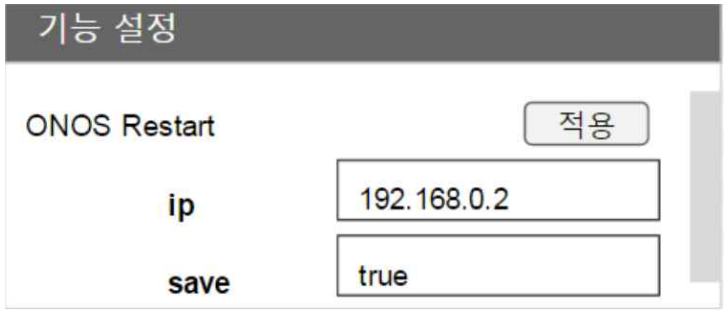


ONOS VNF Scenario – ONOS restart

- from request to reply



VNFM



Input ONOS VNF IP

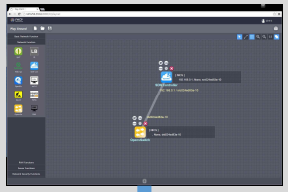
Save current status?
"true" : the ONOS will start with current conf files.
"false": the ONOS will start with latest conf files.

REST Data

```
data = {"id": "restart", "sync": True, "parameters":
[ {"index": 0,"varargs": [ {"name": "ip","value": 192.168.0.2},
                           {"name": "save","value": "true"}
]} ]}
```

ONOS VNF Scenario – ONOS restart Cont'

- from request to reply



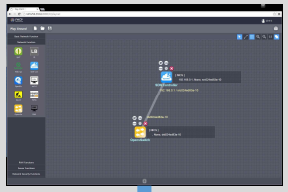
VNFM

Rest.py

```
@bottle.route('/listen', method='PUT')
def emRestRead():
    try:

        data = json.load(bottle.request.body)
        req_time = str(datetime.datetime.utcnow())
            ⋮
elif 'id' in data:
    if data['id'] == 'restart':
        result, reason = onosManager.onosRestartWithID(data)
        res_time = str(datetime.datetime.utcnow())
        if result == 'Success':
            if data['sync'] is True:
                result = 'completed'
            else:
                result = 'executed'
            ack = {'status': result, 'message': 'ok', 'req_time': req_time, 'res_time': res_time}
        elif result == 'Fail':
            result = 'error'
            ack = {'status': result, 'message': reason, 'req_time': req_time, 'res_time': res_time}

        log_send = "#####################################\n"
        log_send += "VNF Restart - ACK to VNFM\n"
        log_send += json.dumps(ack, indent=2)
        log_send += "\n#####################################"
        emSetting.emlog.saveLog(getLog(log_send, True))

        return json.dumps(ack)
```

ONOS VNF Scenario – ONOS restart Cont'

- from request to reply



VNFM

Rest.py

onosManager.py

```
def onosRestartWithID(data):

    """
    Restart ONOS
    process:
    1. stop onos
    2. if "save" flag is True, then save conf file to Local
    3. install onos
    4. update ip_nic
    5. config onos
    6. start onos

    @param data: json body from vnfm
    @return "Success" for True, "Fail" for False
    """

    log_rev = "**************************************\n"
    log_rev += "ONOS Restart - Msg rev from VNFM\n"
    log_rev += json.dumps(data, indent=2)
    log_rev += "\n**************************************"
    emSetting.emlog.saveLog(getLog(log_rev, True))

    data_rev = {}
    data_rev['save'] = "False"
    for parameter in data['parameters']:
        index = parameter['index']
        check_file = False
        for item in parameter['varargs']:
            if item['name'] == 'ip':
                data_rev['ip'] = item['value']
            elif item['name'] == 'save':
                data_rev['save'] = item['value']

    try:

        emStopVnfNone()
        if data_rev['save'] == "True" or data_rev['save'] == "true":
            onosFunction.save_local()
        onosFunction.install_onos()
        onosFunction.update_nic(data_rev['ip'])
        onosFunction.config_onos()
        emStartVnfNone()
        return 'Success', ''
    except Exception as e:
        onosFunction.log_message("onosRestart", getLog(e))
        emSetting.emlog.saveLog(getLog(e))
        return 'Fail', e
```

ONOS VNF Sample Code – App Management

onosManager.py

```
def onosAppManageWithID(data):

    """!
    Manage ONOS App: Activate or Deactivate

    @param data: json body from vnfm
    @return "Success" for True, "Fail" for False
    """

    log_rev = "****************************************\n"
    log_rev += "ONOS App Manager - Msg rev from VNFM\n"
    log_rev += json.dumps(data, indent=2)
    log_rev += "\n****************************************"

    data_rev = {}
    data_rev['id'] = data['id']
    data_rev['sync'] = data['sync']

    for parameter in data['parameters']:
        index = parameter['index']
        check_file = False
        for item in parameter['varargs']:
            if item['name'] == 'appName':
                data_rev['appName'] = item['value']
            elif item['name'] == 'activate':
                data_rev['activate'] = item['value']

    emSetting.emlog.saveLog(getLog(log_rev, True))
    onosFunction.log_message("onosAppManage: log receive", log_rev)
    onosFunction.log_message("onosAppManage: activate data", str(data_rev['activate']))
    if data_rev['activate'] == "True" or data_rev['activate'] == "true":
        activate = True
        onosFunction.log_message("onosAppManage", "True")
    else:
        activate = False
        onosFunction.log_message("onosAppManage", "False")

    try:
        result = onosFunction.manage_onos_app(data_rev['appName'], activate)
        return 'Success', ''
    except Exception as e:
        onosFunction.log_message("onosAppManage", getLog(e))
        emSetting.emlog.saveLog(getLog(e))
        return 'Fail', e
```

onosFunction.py

```
def manage_onos_app(app_name, activate = True, host= "127.0.0.1"):
    """! Function for activate/ deactivate onos application
    Process:
    1 read input value, which contains app name and activate
    2-1 if activate is "true", activate app
    2-2 else activate is "false", deactivate app
    3 call restapi for activting, & deactivating app

    @param app_name String: name for onos App
    @param activate Boolean: True for activate, False for deactivate
    @param host String: ONOS IP (default is "127.0.0.1")
    @return True for Success, False for fail
    """
    cmd = "/opt/onos_project/onos-app " + host
    cmd += " activate" if activate else " deactivate"
    cmd += " " + app_name

    p = Popen(cmd, shell=True)
    result = p.wait()

    if result == 0:
        log_message("manage_onos_app", "stop is done")
        emSetting.emlog.saveLog(getLog('manage_onos_app is Success.........'))
        return True
    else:
        log_message("manage_onos_app", "stop is failed")
        emSetting.emlog.saveLog(getLog('manage_onos_app is failed.........'))
        return False
```

ONOS VNF Sample Code - Clustering

onosManager.py

```
def onosClusterWithID(data):
    """!
    Set ONOS Cluster

    @param data: json body from vnfm
    @return "Success" for True, "Fail" for False
    """
    log_rev = "****************************************\n"
    log_rev += "ONOS Cluster - Msg rev from VNFM\n"
    log_rev += json.dumps(data, indent=2)
    log_rev += "\n****************************************"
    emSetting.emlog.saveLog(getLog(log_rev, True))

    data_rev = {}
    data_rev['save'] = "False"
    for parameter in data['parameters']:
        index = parameter['index']
        check_file = False
        for item in parameter['varargs']:
            if item['name'] == 'ip':
                data_rev['ip'] = item['value']
            elif item['name'] == 'ipList':
                data_rev['ipList'] = item['value']
            elif item['save'] == 'save':
                data_rev['save'] = item['value']

    try:
        emStopVnfNone()
        if data_rev['save'] == "True":
            onosFunction.save_local()
        onosFunction.install_onos()
        onosFunction.set_cluster_info(data_rev['ip'], data_rev['ipList'], True)
        onosFunction.config_onos()
        emStartVnfNone()
        return 'Success', ''
    except Exception as e:
        onosFunction.log_message("onosCluster", getLog(e))
        emSetting.emlog.saveLog(getLog(e))
        return 'Fail', e
```

onosFunction.py

```
def set_cluster_info(ip, ip_list):
    """! Function for set Cluster
    Process:
    1 read input value, which contains IP list
    2 create cluster.json file
    3 move to directory

    @param ip String: ip for current ONOS VNF
    @param ip_list String: ip List for ONOS Cluster. "," separated
    @param restart Boolean: True for restart, Default is True.
    @return True for Success, False for fail
    """

    try:
        # set ip
        update_nic(ip)

        # read input value
        ips = ip_list.split(",")

        onosGenPartitions.generate_cluster_file('cluster.json', ips)
        emSetting.emlog.saveLog(getLog('create cluster json'))
        log_message("set_cluster_info", "create cluster json")
        # move to directory

        shutil.copy("cluster.json", "/opt/onos_project/onos/config/cluster.json")
        log_message("Cluster", "move file is done")
        emSetting.emlog.saveLog(getLog('move file is done'))
        return True

    except Exception as e:
        emSetting.emlog.saveLog(getLog(e))
        log_message("set_cluster_info Exception", getLog(e))
        return False
```

ONOS VNFD

- A XML file to define hardware spec.

```
<?xml version="1.0" encoding="utf-8"?>

<VNFDescriptor xmlns:vnfd="http://fncp.etri.re.kr">

<vnfdInfo>
        <vnfdVendor>ONOS</vnfdVendor>
        <vnfdImageName>onosdemo</vnfdImageName>
        <vnfdVersion>0305</vnfdVersion>
        <vnfdType>VNF_DOCKER</vnfdType>
        <vnfdFlavorId>20</vnfdFlavorId>
        <emVersion>0.2</emVersion>
        <vnfdDesc>demo 0125</vnfdDesc>
</vnfdInfo>

<vnfConfiguration>
        <configType>BOTH</configType>
        <configRepository>scp://116.89.184.90</configRepository>
        <configNumFile>1</configNumFile>
        <configPathFile configPath="/usr/local/onosConf" configFileName="*" />
</vnfConfiguration>

</VNFDescriptor>
```

ONOS VNFD

- JSON file to define UI for Playnet.


```
{
  "id" : "cluster",
  "name" : "ONOS Cluster",
  "desc" : "Set ONOS Cluster",
  "sync" : true,
  "parameters" : [{
    "index" : 0,
    "mandatory" : true,
    "argDefs" : [{"name" : "ipList","required" : true, "type" : "string"},
                 {"name" : "ip","required" : true,"type" : "string"}],
    "varargs" : [{"name" : "ipList", "desc" : "Input ONOS Cluster Node IP List.
ex:(192.168.0.2,192.168.0.3,192.168.0.4)},
                 {"name" : "ip","desc" : "Input Current ONOS IP.
ex:(192.168.0.2)"}]
   }]
 }
```

4. Result & Demo

Deploy & Test

- We deployed ONOS VNF on Playnet Dev Server.
- Also, tested ONOS VNF with these scenarios:
  1) Single ONOS Scenario:
     a. ONOS start/stop
     b. ONOS app management
     c. Project save & load

  2) Multi ONOS Scenario:
     a. ONOS Clustering
     b. Control Plane Failure recovery
     c. Project save & load

Future Work

- modify clustering mechanism for ONOS VNF.
- support custom ONOS apps on ONOS VNF.

Thankyou