K-ONE 기술 문서 #43

Auto healing and scaling feature for VNFFG in OpenStack Tacker

Document No. K-ONE #43
Version 1.0
Date 2019-11-25
Author(s) 이재욱, 이호찬

K-ONE

ONK
OpenNetworkingKorea

■ 문서의 연혁

버전	날짜	작성자	내용
0.1	2019.10.11	이재욱	목차 작성
0.2	2019.10.16	이호찬	1,2 장 작성
0.4	2019.10.21	이호찬	3장 작성
0.6	2019.11.01	이호찬	4장 작성
0.8	2019.11.13	이호찬	초안 작성
1.0	2019.11.25	이재욱	검토 및 수정

본 문서는 2018년도 정부(미래창조과학부)의 재원으로 정보통신 기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-16-2012, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. B0190-16-2012, Global SDN/NFV OpenSource Software Core Module/Function Development)

기술문서 요약

본 고는 OpenStack Tacker의 구성요소들에 대한 설명과 VNF의 고가용성을 위한 모니터링 프레임워크, 그리고 VNFFG의 고가용성을 위한 기능들로 auto-healing 기능, scaling 기능 개발에 관한 기술문서이다. 본 고는 다음과 같이 구성된다. 1장은 OpenStack Tacker 프로젝트 구성요소들에 대해 알아보고, 2장에서는 Tacker에서 제공하는 VNF 고가용성을 위한 모니터링 프레임워크에 대해 알아본다. 3장에서는 VNFFG의 고가용성의 필요성 및 auto-healing, scaling 기능에 대해 서술한다.

Contents

K-ONE #34. Auto healing and scaling feature for VNFFG in OpenStack Tacker

그림 목차

K-ONE #43. Auto healing and scaling feature for VNFFG in OpenStack Tacker

1. OpenStack Tacker

1.1. OpenStack Tacker 개요

Tacker [1] 는 OpenStack [2] 의 Kilo 버전부터 공식적으로 OpenStack 의 프로젝트로 추가되었고 2019년 11월 현재 Train 버전이 최신 버전으로 릴리즈 된 상태이다. Tacker 프로젝트는 ETSI NFV ISG에서 제안한 아래 그림의 NFV Mangement and Orchestration (MANO) 구조에서 VNF Manager와 NFV Orchestrator의 역할을 담당하고 있다 [3].

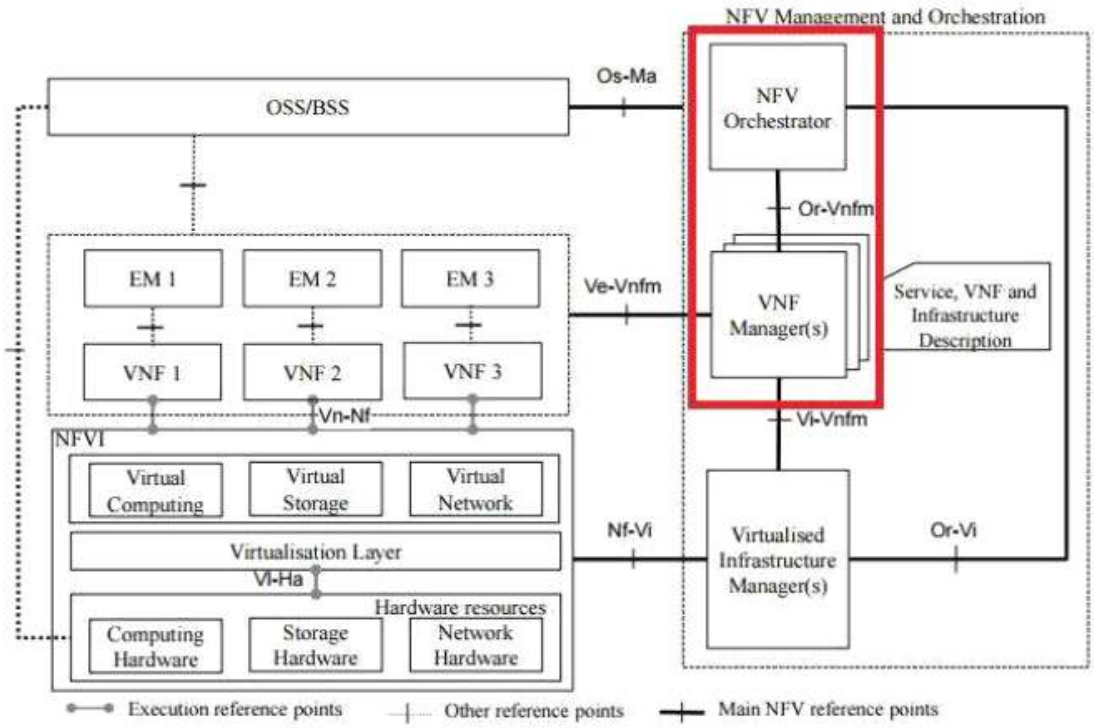

그림 1. NFV MANO에서의 Tacker 프로젝트의 역할

Tacker의 구조는 아래 그림과 같다. Tacker 사용자는 OpenStack에서 제공하는 웹 기반 GUI 인터페이스를 제공하는 Horizon을 이용하거나 또는 CLI를 통해 Tacker의 기능들을 관리할 수 있다. Tacker에서는 이러한 외부 접근을 위해 여러 API들을 제공하고 있다. NFV Catalog는 NFV 환경에서 서비스를 제공하기 위해 미리 필요한 템플릿들을 저장하고 있다. 여기에는 VNF 및 네트워크 서비스들의 배포를 위한 템플릿들이 포함된다. 이를 통해 Tacker 사용자는 템플릿 기반으로 편리하게 네트워크 서비스 배포가 가능하다. 해당 템플릿들은 추후 오케스트레이션 서비스를 제공하는 Heat 프로젝트에 Translation되어 전달된다. Tacker는 VNF Manager (VNFM) 모듈과 NFV Orchestrator (NFVO) 모듈로 구성되어 있다. VNFM은 가상화된 리소스를 바탕으로 VNF의 생성, 관리, 제거와 같은 기본적인 VNF 라이프사이클 관리를

담당하고 있고 NFVO는 VNF의 배치 및 end-to-end 네트워크 서비스 배포 및 관리를 담당한다. 각 모듈들은 인프라 드라이버를 통해 VIM과 연동될 수 있고 이를 통해 특정 VIM에 속한 VNF들의 라이프사이클의 관리, 모니터링, scaling, VNFFG 배포 등을 수행할 수 있다. Tacker의 Mikata 버전부터 Multi-site VIM 기능을 지원하여 현재 여러 VIM 사이트에 배포되어 있는 VNF들과 네트워크 서비스들의 관리가 가능하게 되었다. Tacker는 ETSI NFV MANO 구조를 기반으로 개발되고 있고 있기 때문에 특정 벤더에 종속되지 않고 멀티 벤더로부터 VNF를 배치하는 것이 가능하다는 특징을 가지고 있다.

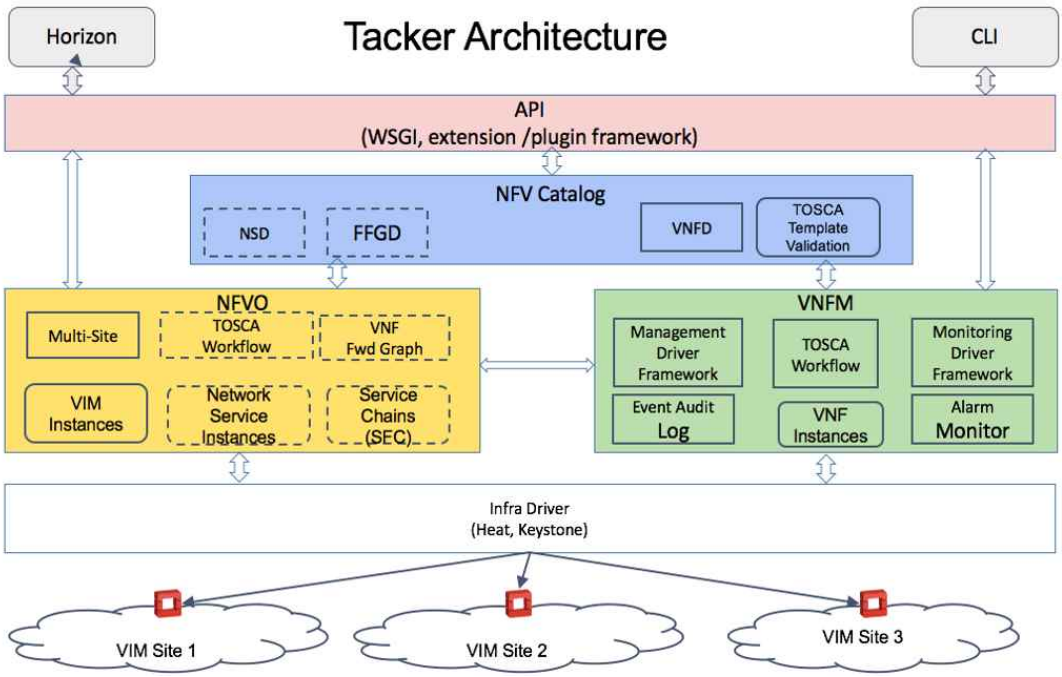

그림 2. Tacker Architecture

1.2. VNF Manager

1.2.1. VNFM의 역할

ETSI에서 제안한 NFV MANO 스펙에서 VNFM은 VNF들의 라이프사이클 관리를 책임지는 모듈이다. 각 VNF들은 특정 VNFM에 속해있다고 가정한다. VNFM은 하나의 VNF에 할당되어있을 수도 있고, 혹은 여러 타입의 많은 VNF들에 할당될 수 있다. 대부분의 VNFM 기능들은 어떤 타입의 VNF라도 일반적인 기능들을 적용시킬 수 있도록 정의된다. 스펙에서 명시된 VNFM의 필요한 기능들은 아래와 같다.

- VNF Descriptor를 이용한 VNF의 생성
- VNF 소프트웨어 업데이트 / 업그레이드
- VNF의 수정 및 제거
- VNF scaling out/in and up/down
- VNF와 관련한 성능 측정 결과, 결함/이벤트 정보 등의 수집
- VNF auto-healing
- VNF 라이프사이클 관리 변화 알림
- 라이프사이클을 통한 VNF의 무결성 관리

1.2.2. VNFM의 기능

Tacker 프로젝트의 VNFM의 기능은 다음과 같다. VNF의 생성, 업데이트, 제거와 같은 기본적인 VNF의 라이프사이클 관리와 배포된 VNF들이 정상적으로 동작하고 있는지에 대한 모니터링을 수행할 수 있다. 또한 사용자가 원하는 정책 기반으로 VNF들의 auto-healing이나 auto-scaling을 수행할 수 있는 기능이 존재한다.

VNFM에서 VFN를 관리하기 앞서, VNF들이 배포될 VIM 사이트를 등록해야 한다. 현재 Tacker는 Multi-site VIM 기능을 Mikata 버전부터 지원하고 있기 때문에 여러 VIM 사이트들에 대한 정보를 업데이트받고 각 VIM 사이트들마다 독립적인 VNF 관리가 가능하다. VNFM에서 VIM 정보를 등록하기 위한 명령어는 아래와 같다.

```
tacker vim-register --config-file <config_file_path> <name>
```

여러 VIM들을 등록한 이후에 VIM들의 리스트와 각 VIM들에 대한 세부 정보, VIM의 update 기능은 각각 아래와 같은 명령어로 가능하다.

```
tacker vim-list
tacker vim-show <vim_name>
tacker vim-update <vim_name>
```

```
stack@ubuntu:~$ tacker vim-list --fit-width
Deprecated: tacker command line is deprecated, will be deleted after Rocky is released.
+--------------------+--------------------+--------+-----------+------------+--------------------+-----------+
| id                 | tenant_id          | name   | type      | is_default | placement_attr     | status    |
+--------------------+--------------------+--------+-----------+------------+--------------------+-----------+
| 45a1e86a-fbbf-4fe8- | 6a98ecaf3a324255b9d | vim_nfv | openstack | True       | {u'regions':       | REACHABLE |
| 91f5-2e9d924fbfec  | 62a6cef3eff33      |        |           |            | [u'RegionOne']}    |           |
+--------------------+--------------------+--------+-----------+------------+--------------------+-----------+
```

그림 3. VIM 리스트의 확인

```
stack@ubuntu:~$ tacker vim-show vim_nfv --fit-width
Deprecated: tacker command line is deprecated, will be deleted after Rocky is released.
+---------------+-----------------------------------------------------------------------------------------------------+
| Field         | Value                                                                                               |
+---------------+-----------------------------------------------------------------------------------------------------+
| auth_cred     | {"username": "nfv_user", "password": "***", "project_name": "nfv", "cert_verify": "False",          |
|               | "user_domain_name": "Default", "key_type": "barbican_key", "secret_uuid": "***", "auth_url":        |
|               | "http://127.0.0.1/identity/v3", "project_id": null, "project_domain_name": "Default"}               |
| auth_url      | http://127.0.0.1/identity/v3                                                                        |
| created_at    | 2019-08-30 07:34:06                                                                                 |
| description   |                                                                                                     |
| id            | 45a1e86a-fbbf-4fe8-91f5-2e9d924fbfec                                                                |
| is_default    | True                                                                                                |
| name          | vim_nfv                                                                                             |
| placement_attr | {"regions": ["RegionOne"]}                                                                          |
| status        | REACHABLE                                                                                           |
| tenant_id     | 6a98ecaf3a324255b9d62a6cef3eff33                                                                    |
| type          | openstack                                                                                           |
| updated_at    | 2019-08-30 07:34:07                                                                                 |
| vim_project   | {"name": "nfv", "project_domain_name": "Default"}                                                   |
+---------------+-----------------------------------------------------------------------------------------------------+
```

그림 4. VIM의 세부 내용 확인

VNFM에서 VNF의 생성은 NFV 카탈로그의 VNF Descriptor (VNFD)를 이용해서 수행한다. VNFD에는 기본적으로 VNFD의 id, 기본정보, vdu 정보, virtual link 정보, 모니터링 및 오토 scaling 관련 파라미터 등이 포함되어 있다. 이를 통해, VNF의 생성뿐만 아니라, 생성이후에 필요한 관리 기능에 대한 정의 및 모니터링 이후의 액션에 대한 정의도 포함되어 있어 VNF 관리에 유용하다. VNFD의 샘플은 아래 그림과 같다.

VNFD template을 온보딩하는 명령어는 다음과 같다.

```
tacker vnfd-create --vnfd-file <yaml-file-path> <vnfd-name>
```

VNFD template을 온보딩 시킨 후 VNF를 deploy 시키는 방법은 두 가지로 다음과 같다.

1. Tacker에서 가장 일반적으로 쓰이는 방법으로 Tacker NFV Catalog에 온보딩되어 있는 VNFD template을 이용하여 VNF를 생성한다. 이 방법을 통해 VNF를 생성

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0

description: Demo example

metadata:
  template_name: sample-tosca-vnfd

topology_template:
  node_templates:
    VDU1:
      type: tosca.nodes.nfv.VDU.Tacker
      capabilities:
        nfv_compute:
          properties:
            num_cpus: 1
            mem_size: 512 MB
            disk_size: 1 GB
      properties:
        image: cirros-0.4.0-x86_64-disk
        availability_zone: nova
        mgmt_driver: noop
        config: |
          param0: key1
          param1: key2

    CP1:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        management: true
        order: 0
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL1
        - virtualBinding:
            node: VDU1

    CP2:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        order: 1
        anti_spoofing_protection: false
```

그림 5. Sample VNF Descriptor (1/2)

```
  CP3:
    type: tosca.nodes.nfv.CP.Tacker
    properties:
      order: 2
      anti_spoofing_protection: false
    requirements:
      - virtualLink:
          node: VL3
      - virtualBinding:
          node: VDU1

  VL1:
    type: tosca.nodes.nfv.VL
    properties:
      network_name: net_mgmt
      vendor: Tacker

  VL2:
    type: tosca.nodes.nfv.VL
    properties:
      network_name: net0
      vendor: Tacker

  VL3:
    type: tosca.nodes.nfv.VL
    properties:
      network_name: net1
      vendor: Tacker
```

그림 6. Sample VNF Descriptor (2/2)

시키는 명령어는 다음과 같다.

```
tacker vnf-create --vnfd-name <vnfd-name> <vnf-name>
```

즉, 미리 온보딩 되어있는 VNFD template을 이용하여 VNF를 생성한다.

2. TOSCA template으로 Tacker Catalog에 온보딩 시키지 않고 직접적으로 VNF를 생성한다. 이 방법은 NFV Catalog가 Tacker에 밖에 위치하여 Tacker가 단지 NFV workflow engine으로만 사용될 때 주로 쓰이는 방법이다. 이 방법을 통해 VNF를 생

성하는 명령어는 다음과 같다.

```
tacker vnf-create --vnfd-template <vnfd-file-name> <vnf-name>
```

또한 아래와 같은 명령어들로 현재 VNFM이 관리하고 있는 VNF들의 리스트와 온보딩된 VNFD의 리스트들을 확인할 수 있다.

```
tacker vnf-list
tacker vnfd-list
```

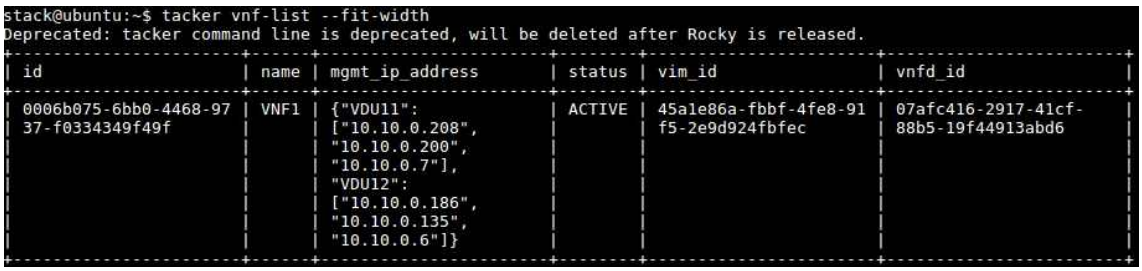

그림 7. VNF 리스트 확인

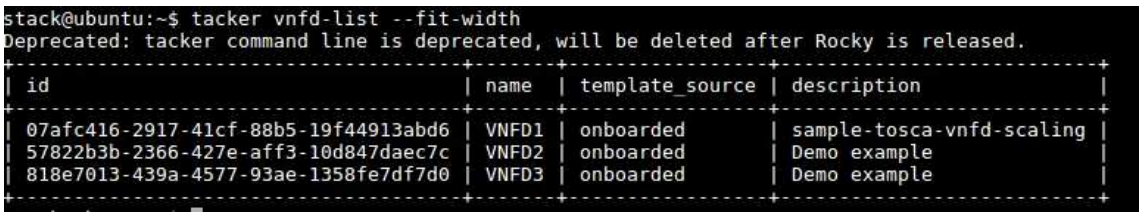

그림 8. VNF Descriptor 리스트 확인

각 리스트에 있는 VNF와 VNFD의 자세한 내용을 확인하려면 아래와 같은 명령어들로 확인이 가능하다.

```
tacker vnf-show <vnf_id/name>
tacker vnfd-show <vnfd_id/name>
```

VNFD의 경우 현재 온보딩 되어있는 VNFD의 템플릿을 확인하고 싶을때는 아래와 같은 명령어로 확인이 가능하다.

```
stack@ubuntu:~$ tacker vnf-show VNF1 --fit-width
Deprecated: tacker command line is deprecated, will be deleted after Rocky is released.
+-----------------+---------------------------------------------------------------------------------------------+
| Field           | Value                                                                                       |
+-----------------+---------------------------------------------------------------------------------------------+
| attributes      | {"scaling_group_names": "{\"SP1\": \"SP1_group\"}", "heat_template": "heat_template_version: |
|                 | 2013-05-23\ndescription: 'sample-tosca-vnfd-scaling\n\n  '\nparameters: {}\nresources:\n     |
|                 | SP1_scale_out:\n    type: OS::Heat::ScalingPolicy\n    properties:\n                        |
|                 | auto_scaling_group_id: {get_resource: SP1_group}\n      adjustment_type:                    |
|                 | change_in_capacity\n      scaling_adjustment: 1\n      cooldown: 120\n  SP1_group:\n        |
|                 | type: OS::Heat::AutoScalingGroup\n    properties:\n      min_size: 1\n                      |
|                 | desired_capacity: 2\n      cooldown: 120\n      resource: {type: SP1_res.yaml}\n            |
|                 | max_size: 3\n  SP1_scale_in:\n    type: OS::Heat::ScalingPolicy\n    properties:\n          |
|                 | auto_scaling_group_id: {get_resource: SP1_group}\n      adjustment_type:                    |
|                 | change_in_capacity\n      scaling_adjustment: -1\n      cooldown: 120\noutputs: {}\n",      |
|                 | "SP1_res.yaml": "heat_template_version: 2013-05-23\ndescription: Tacker Scaling              |
|                 | template\nresources:\n  CP12:\n    type: OS::Neutron::Port\n    properties:                 |
|                 | {port_security_enabled: false, network: net0}\n  VDU11:\n    type: OS::Nova::Server\n       |
|                 | properties:\n      user_data_format: SOFTWARE_CONFIG\n      availability_zone: nova\n       |
|                 | image: OpenWRT\n      flavor: m1.tiny\n      networks:\n      - port: {get_resource: CP11}\n |
|                 | config_drive: false\n  CP11:\n    type: OS::Neutron::Port\n    properties:                  |
|                 | {port_security_enabled: false, network: net0}\n  VL1: {type: 'OS::Neutron::Net'}\n  VDU12:\n |
|                 | type: OS::Nova::Server\n    properties:\n      user_data_format: SOFTWARE_CONFIG\n          |
|                 | availability_zone: nova\n      image: OpenWRT\n      flavor: m1.tiny\n      networks:\n     |
|                 | - port: {get_resource: CP12}\n      config_drive: false\noutputs:\n  mgmt_ip-VDU11:\n       |
|                 | value:\n      get_attr: [CP11, fixed_ips, 0, ip_address]\n  mgmt_ip-VDU12:\n    value:\n     |
|                 | get_attr: [CP12, fixed_ips, 0, ip_address]\n"}                                              |
| created_at      | 2019-08-30 13:10:57                                                                         |
| description     | sample-tosca-vnfd-scaling                                                                   |
| error_reason    |                                                                                             |
| id              | 0006b075-6bb0-4468-9737-f0334349f49f                                                        |
| instance_id     | a0e3c7f7-3d8b-4885-b3e7-d3bcf78ef484                                                        |
| mgmt_ip_address | {"VDU11": ["10.10.0.208", "10.10.0.200", "10.10.0.7"], "VDU12": ["10.10.0.186",             |
|                 | "10.10.0.135", "10.10.0.6"]}                                                                |
| name            | VNF1                                                                                        |
| placement_attr  | {"vim_name": "vim_nfv"}                                                                     |
| status          | ACTIVE                                                                                      |
| tenant_id       | 6a98ecaf3a324255b9d62a6cef3eff33                                                            |
| updated_at      |                                                                                             |
| vim_id          | 45a1e86a-fbbf-4fe8-91f5-2e9d924fbfec                                                        |
| vnfd_id         | 07afc416-2917-41cf-88b5-19f44913abd6                                                        |
+-----------------+---------------------------------------------------------------------------------------------+
```

그림 9. VNF 세부 내용 확인

```
stack@ubuntu:~$ tacker vnfd-show VNFD1 --fit-width
Deprecated: tacker command line is deprecated, will be deleted after Rocky is released.
+-----------------+---------------------------------------------------------------------------------------------+
| Field           | Value                                                                                       |
+-----------------+---------------------------------------------------------------------------------------------+
| attributes      | {"vnfd": "description: sample-tosca-vnfd-scaling\nmetadata: {template_name: sample-tosca-   |
|                 | vnfd-scaling1}\ntopology_template:\n  node_templates:\n    CP11:\n      properties:          |
|                 | {anti_spoofing_protection: false, management: true, order: 0}\n      requirements:\n      - |
|                 | virtualLink: {node: VL1}\n      - virtualBinding: {node: VDU11}\n      type:                 |
|                 | tosca.nodes.nfv.CP.Tacker\n    CP12:\n      properties: {anti_spoofing_protection: false,   |
|                 | management: true, order: 0}\n      requirements:\n      - virtualLink: {node: VL1}\n      - |
|                 | virtualBinding: {node: VDU12}\n      type: tosca.nodes.nfv.CP.Tacker\n    VDU11:\n          |
|                 | properties: {availability_zone: nova, flavor: m1.tiny, image: OpenWRT, mgmt_driver:         |
|                 | openwrt}\n      type: tosca.nodes.nfv.VDU.Tacker\n    VDU12:\n      properties:              |
|                 | {availability_zone: nova, flavor: m1.tiny, image: OpenWRT, mgmt_driver: openwrt}\n          |
|                 | type: tosca.nodes.nfv.VDU.Tacker\n    VL1:\n      properties: {network_name: net0, vendor:  |
|                 | Tacker}\n      type: tosca.nodes.nfv.VL\n  policies:\n  - SP1:\n      properties: {cooldown: |
|                 | 120, default_instances: 2, increment: 1, max_instances: 3,\n        min_instances: 1}\n     |
|                 | targets: [VDU11, VDU12]\n      type:                                                        |
|                 | tosca.policies.tacker.Scaling\ntosca_definitions_version:                                   |
|                 | tosca_simple_profile_for_nfv_1_0_0\n"}                                                      |
| created_at      | 2019-08-30 13:10:37                                                                         |
| description     | sample-tosca-vnfd-scaling                                                                   |
| id              | 07afc416-2917-41cf-88b5-19f44913abd6                                                        |
| name            | VNFD1                                                                                       |
| service_types   | vnfd                                                                                        |
| template_source | onboarded                                                                                   |
| tenant_id       | 6a98ecaf3a324255b9d62a6cef3eff33                                                            |
| updated_at      |                                                                                             |
+-----------------+---------------------------------------------------------------------------------------------+
```

그림 10. VNF Descriptor 세부 내용 확인

```
tacker vnfd-template-show <vnfd-name>
```

VNF들과 VNFD 의 삭제는 아래와 같은 명령어들로 제거가 가능하다.

```
tacker vnf-delete <vnf_id/name>
tacker vnfd-delete <vnfd_id/name>
```

또한 Tacker 서버가 특정 리소스에 이벤트가 발생했을 때의 기록을 갖고 있어서 아래와 같은 명령어로 VNF와 관련된 이벤트들의 리스트들을 확인할 수 있다.

```
tacker events-list
```

이를 통해 아래 그림과 같이 이벤트 ID와 리소스 타입, 리소스 ID, 리소스 상태, 이벤트 타입, 타임 스탬프, 이벤트 세부 사항들을 확인할 수 있다.

```
stack@ubuntu:~$ tacker events-list --fit-width
Deprecated: tacker command line is deprecated, will be deleted after Rocky is released.
+----+---------------+----------------------+------------------+------------+---------------------+----------------------+
| id | resource_type | resource_id          | resource_state   | event_type | timestamp           | event_details        |
+----+---------------+----------------------+------------------+------------+---------------------+----------------------+
|  1 | vnfd          | b5b7c9d2-a62a-4312-9 | OnBoarded        | CREATE     | 2019-08-29 15:52:47 |                      |
|    |               | 0f5-95da2b0ccad6     |                  |            |                     |                      |
|  2 | vnfd          | b5b7c9d2-a62a-4312-9 | Not Applicable   | DELETE     | 2019-08-29 15:53:20 |                      |
|    |               | 0f5-95da2b0ccad6     |                  |            |                     |                      |
|  3 | vnfd          | 16fe87fe-3b77-4583-9 | OnBoarded        | CREATE     | 2019-08-29 15:53:31 |                      |
|    |               | 028-d34e74543af4     |                  |            |                     |                      |
|  4 | vim           | ba5fae1d-84e1-4cef-  | PENDING          | CREATE     | 2019-08-29 16:01:37 |                      |
|    |               | a8bb-fc4aa473e7c7    |                  |            |                     |                      |
|  5 | vim           | ba5fae1d-84e1-4cef-  | REACHABLE        | MONITOR    | 2019-08-29 16:01:44 |                      |
|    |               | a8bb-fc4aa473e7c7    |                  |            |                     |                      |
|  6 | vnf           | 4d7b6902-f8ca-4ed6-9 | PENDING_CREATE   | CREATE     | 2019-08-29 16:03:04 | VNF UUID assigned.   |
|    |               | 795-0bd23366031a     |                  |            |                     |                      |
|  7 | vnf           | 4d7b6902-f8ca-4ed6-9 | PENDING_CREATE   | CREATE     | 2019-08-29 16:03:51 | Infra Instance ID    |
|    |               | 795-0bd23366031a     |                  |            |                     | created: e22c600b-   |
|    |               |                      |                  |            |                     | ee0d-48ef-           |
|    |               |                      |                  |            |                     | aac8-dc36cd0f6db7 and |
|    |               |                      |                  |            |                     | Mgmt IP address set: |
|    |               |                      |                  |            |                     | {"VDU1":             |
|    |               |                      |                  |            |                     | "192.168.120.29"}    |
|  8 | vnf           | 4d7b6902-f8ca-4ed6-9 | ACTIVE           | CREATE     | 2019-08-29 16:03:51 | VNF creation         |
|    |               | 795-0bd23366031a     |                  |            |                     | completed            |
|  9 | vnf           | 4d7b6902-f8ca-4ed6-9 | ACTIVE           | MONITOR    | 2019-08-29 16:03:51 | VNF added for        |
|    |               | 795-0bd23366031a     |                  |            |                     | monitoring.          |
|    |               |                      |                  |            |                     | mon_policy_dict =    |
|    |               |                      |                  |            |                     | {"vdus": {"VDU1":    |
|    |               |                      |                  |            |                     | {"ping": {"actions": |
|    |               |                      |                  |            |                     | {"failure":          |
|    |               |                      |                  |            |                     | "respawn"}, "name":  |
|    |               |                      |                  |            |                     | "ping", "parameters": |
|    |               |                      |                  |            |                     | {"count": 3,         |
|    |               |                      |                  |            |                     | "interval": 10},     |
|    |               |                      |                  |            |                     | "monitoring_params": |
|    |               |                      |                  |            |                     | {"count": 3,         |
|    |               |                      |                  |            |                     | "interval": 10}}}}}, |
| 10 | vnfd          | 9fbab830-f2b3-48b7-9 | OnBoarded        | CREATE     | 2019-08-29 16:09:29 |                      |
|    |               | 967-078290726a65     |                  |            |                     |                      |
| 11 | vnf           | 8736743a-3ae8-44a8-8 | PENDING_CREATE   | CREATE     | 2019-08-29 16:09:52 | VNF UUID assigned.   |
|    |               | 636-14a3cc14d087     |                  |            |                     |                      |
| 12 | vnf           | 8736743a-3ae8-44a8-8 | PENDING_CREATE   | MONITOR    | 2019-08-29 16:09:52 | Alarm URL set        |
|    |               | 636-14a3cc14d087     |                  |            |                     | successfully: {'vdu_h |
|    |               |                      |                  |            |                     | cpu_usage_scaling_out |
|    |               |                      |                  |            |                     | ': 'http://ubuntu:989 |
|    |               |                      |                  |            |                     | 0/v1.0/vnfs/8736743a- |
|    |               |                      |                  |            |                     | 3ae8-44a8-8636-14a3cc |
|    |               |                      |                  |            |                     | 14d087/vdu_hcpu_usage |
|    |               |                      |                  |            |                     | _scaling_out/SP1-out/ |
|    |               |                      |                  |            |                     | c9gtqhwn'}           |
```

그림 11. Tacker 이벤트 리스트

1.3. NFV Orchestrator

1.3.1. NFVO의 역할

NFVO는 여러 VIM들에 걸친 자원들의 오케스트레이션과 네트워크 서비스들의 오케스트레이션을 수행하는 역할을 한다. ETSI NFV MANO 스펙에 명시된 NFVO에서 필요한 기능들은 아래와 같다.

- 네트워크 서비스 배포 템플릿 및 VNF 패키지들의 관리
- 네트워크 서비스 생성 및 네트워크 서비스 라이프사이클 관리 (업데이트, scaling, 성능 측정 결과 수집, 이벤트 수집, 종료)
- VNFM과의 연동을 통한 VNF 생성 관리
- VNFM으로부터 요청된 NFVI 리소스들의 검증 및 권한 부여
- 네트워크 서비스와 VNF들간의 관계 및 라이프사이클을 통한 네트워크 서비스들의 무결성 및 시각성 관리
- 네트워크 서비스 토폴로지의 관리 (생성, 업데이트, 요청, VNFFG의 제거)
- 네트워크 서비스 자동화 관리
- 네트워크 서비스와 VNF들과 관련된 정책 관리 및 평가

1.3.2. NFVO의 기능

Tacker의 NFVO는 템플릿 기반의 end-to-end 네트워크 서비스 배포와 효율적인 VNF들의 배포를 보장하기 위한 VNF 배치 정책, VNF Forwarding Graph (VNFFG)를 이용한 VNF들간의 연결 기능, VIM 자원 체크 및 자원 할당, 여러 사이트틀 내에 존재하는 VNF들을 오케스트레이션하는 기능을 제공하고 있다.
NFVO에서 핵심적으로 사용되는 기능은 사용자 정책을 기반으로 VNF들을 논리적으로 연결하여 체인 형태로 서비스를 제공해주는 VNFFG 구성 및 관리기능이다. Tacker에서는 VNFFG를 형성하기 위한 드라이버로 networking-sfc 드라이버를 사용한다. 이는 OpenStack에서 네트워킹 기능을 담당하는 Neutron의 VNFFG를 생성하는 기본적인 방법은 미리 생성된 VNFFG Descriptor (VNFFGD)를 인스턴스화하여 수행할 수 있다. VNFFGD의 샘플은 아래 그림과 같다.
VNFFGD를 생성하기 위한 명령어는 다음과 같다.

```
tacker vnffgd-create --vnffgd-file <vnffgd file> <vnffgd name>
```

이후 생성된 VNFFGD를 이용하여 VNFFG의 생성을 아래와 같은 명령어로 수행할

수 있다. 특정 VNF를 매핑시키고 싶을 경우 --vnf-mapping 옵션을 추가하여 생성할 수 있다. 또한 VNFFG를 대칭적으로 배포하고 싶을 경우 --symmetrical 옵션을 사용하면 된다.

```
tacker vnffg-create --vnffgd-name <vnffgd name> \
        --vnf-mapping <vnf mapping> --symmetrical <boolean>
```

이후 생성된 VNFFGD와 VNFFG의 리스트를 보기 위해서는 아래와 같은 명령어를 수행한다.

```
tacker vnffgd-list
tacker vnffg-list
```

```
stack@ubuntu:~$ tacker vnffgd-list
Deprecated: tacker command line is deprecated, will be deleted after Rocky is released.
+--------------------------------------+--------+-----------------+-----------------------+
| id                                   | name   | template_source | description           |
+--------------------------------------+--------+-----------------+-----------------------+
| fb5d67ef-0046-4cdf-a15f-be25d6341238 | vnffgd | onboarded       | Sample VNFFG template |
+--------------------------------------+--------+-----------------+-----------------------+
```

그림 13. VNFFG Descriptor 리스트 확인

```
stack@ubuntu:~$ tacker vnffg-list --fit-width
Deprecated: tacker command line is deprecated, will be deleted after Rocky is released.
+--------------------------+-------+-------+-----------------------+--------+--------------------------+
| id                       | name  | ns_id | description           | status | vnffgd_id                |
+--------------------------+-------+-------+-----------------------+--------+--------------------------+
| 806cccb9-0127-43a6-b7f5-54 | vnffg |       | Sample VNFFG template | ACTIVE | fb5d67ef-0046-4cdf-a15f- |
| 519dcebc05               |       |       |                       |        | be25d6341238             |
+--------------------------+-------+-------+-----------------------+--------+--------------------------+
```

그림 14. VNFFG 리스트 확인

각 리스트에 있는 VNFFGD와 VNFFG의 자세한 내용을 확인하려면 아래와 같은 명령어들로 확인이 가능하다.

```
tacker vnffgd-show <vnffgd-name>
tacker vnffg-show <vnffg-name>
```

VNFFGD의 경우 VNFFGD의 템플릿을 확인하고 싶을 때는 아래와 같은 명령어로 확인이 가능하다.

```
tacker vnffgd-template-show <vnffgd-name>
```

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0

description: Sample VNFFG template

topology_template:

  node_templates:

    Forwarding_path1:
      type: tosca.nodes.nfv.FP.TackerV2
      description: creates path (CP12->CP22)
      properties:
        id: 51
        policy:
          type: ACL
          criteria:
            - name: block_tcp
              classifier:
                network_src_port_id: 14ad4f29-629f-4b97-8bc8-86e96cb49974
                destination_port_range: 80-1024
                ip_proto: 6
                ip_dst_prefix: 10.10.0.5/24
        path:
          - forwarder: VNFD1
            capability: CP12
          - forwarder: VNFD2
            capability: CP22

  groups:
    VNFFG1:
      type: tosca.groups.nfv.VNFFG
      description: HTTP to Corporate Net
      properties:
        vendor: tacker
        version: 1.0
        number_of_endpoints: 2
        dependent_virtual_link: [VL12,VL22]
        connection_point: [CP12,CP22]
        constituent_vnfs: [VNFD1,VNFD2]
      members: [Forwarding_path1]
```

그림 12. Sample VNFFG Descriptor

VNFFGD와 VNFFG의 제거는 아래와 같은 명령어로 가능하다.

```
tacker vnffgd-delete <vnffgd-name>
tacker vnffg-delete <vnffg-name>
```

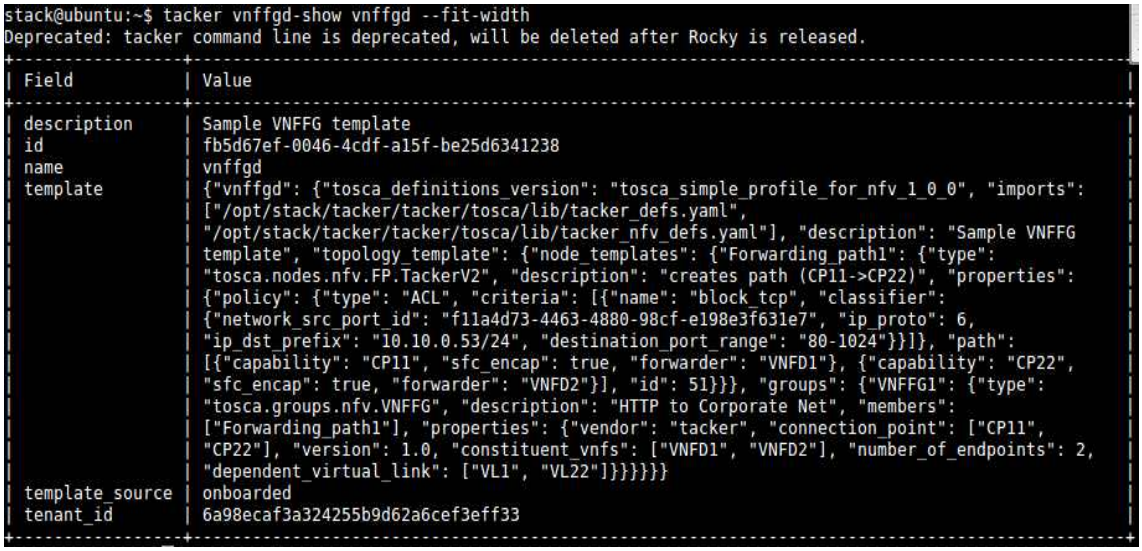

그림 15. VNFFG Descriptor 세부 내용 확인

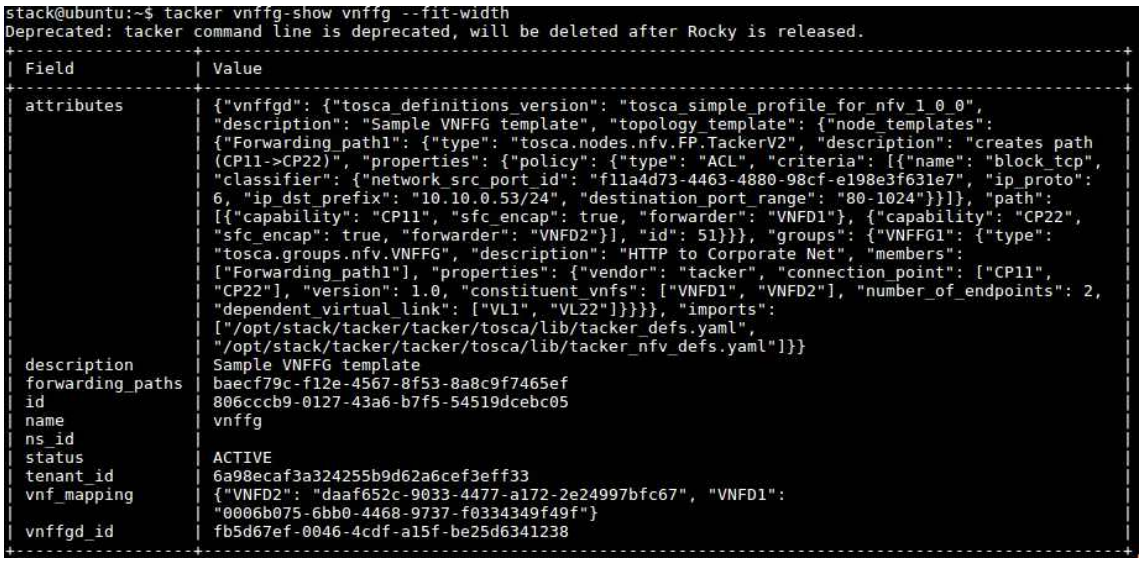

그림 16. VNFFG 세부 내용 확인

그림 17. VNFFG Descriptor 템플릿 내용 확인

또한 한번 배포된 VNFFG를 수정하기를 원하는 경우, 기존에는 VNFFG를 수정하기 위해서는 배포되어 있는 VNFFG를 제거한 후 새로운 VNFFGD의 생성 및

VNFFG의 생성을 해야 했다. 이는 VNFFG의 재배포동안 네트워크 서비스를 제공할 수 없다는 문제가 있었기 때문에 Queens 버전부터 배포되어 있는 VNFFG를 제거하지 않고 VNFFG 업데이트 기능을 수행할 수 있는 기능이 추가되었다. 해당 기능을 수행하기 위해서는 새로 작성된 VNFFGD 템플릿 파일이 필요하다. VNFFG 업데이트 기능의 명령어는 아래와 같다.

```
tacker vnffg-update --vnffgd-template [vnffgd-scale.yaml] [vnffg]
```

2. VNF의 고가용성을 위한 모니터링 프레임워크

2.1. Tacker에서의 모니터링 프레임워크

Tacker에서는 VNF의 자원상태나 생존 유무를 확인하기 위한 모니터링 프레임 워크를 제공하고 있다. 모니터링 프레임 워크는 크게 ping 기반의 모니터링과 알람 기반의 모니터링으로 구분된다.

2.1.1. ping 기반 모니터링

Monitoring Framework for VNF Manager는 liberty버전에 제안된 기능이다. 제안된 기능을 통해 Tacker에서는 VNF 관리를 위한 모니터링 기능을 제공한다. 또한, liberty이전에 제공되던 ping을 통해 단순히 VNF 인스턴스의 생존 유무확인 뿐만 아니라 더 복잡한 모니터링 기능 추가확정을 목표로 하고 있다.

Monitoring Framework을 구현하기 위해 변경사항은 다음과 같다. 기존의 관리 및 인프라 드라이버 (Management and Infrastructure Driver)와 유사한 드라이버 모델을 통해 간단한 모니터링과 외부 모니터링 기능 수행을 통해 모니터링 기능이 향상된다. 따라서 기존의 관리 드라이버를 복제하여 모니터링 기능을 모듈화 함으로써, 모니터링 방법을 쉽게 추가 시킬 수 있다. 해당 방식을 구현하기 위해 tacker/vm/drivers내에 mon_driver을 생성하고 기존의 ping 모니터링 기능을 새 모듈화된 드라이버로 이동시킨다. 비록 기존의 모니터링 프레임워크를 구조적 변경없이 모니터링기능을 추가함으로써 확장시킬 수 있지만 제안한 드라이버를 사용하면 OpenStack에 존재하는 Monasca나 Ceilometer와 같은 모니터링 프로젝트를 쉽게 추가 할 수 있다.

이러한 모니터링 기능은 VNF 별로 정의가능하다. Tacker 서버에서 VNF생성 할 때 참조하는 VNFD를 통해 정의 할 수 있다. 모니터링 기능이 정의된 TOSCA 양식은 아래와 같다.

```
vduN:
  monitoring_policy:
    <monitoring-driver-name>:
      monitoring_params:
        <param-name>: <param-value>
        ...
      actions:
        <event>: <action-name>
        ...
    ...
```

그림 18. 모니터링 TOSCA 양식

드라이버는 monitor_drivers 디렉토리 구조에 존재해야하며, 불러올 수 있는 클래스여야 한다. 또한, setup.cfg 파일이 존재해야하며, 해당 파일은 Tacker 서버가 초기화 될 시 읽어져야 한다. 뿐만 아니라, 모니터링 쓰레드는 기본값이 30초로 지정된 global boot_wait configured time이후에 VNF와 VDU를 모니터링 하며, 드라이버는 기본값이 10초로 설정된 check_intvl interval time 간격으로 호출되어 모니터링을 수행한다. 사용되어지는 boot_wait 지연시간과 check_intvl 주기는 향후에 template으로 설정 가능하게 변경 될 예정이다.

```
vdu1:
  monitoring_policy:
    ping:
      actions:
        failure: respawn

vdu2:
  monitoring_policy:
    http-ping:
      monitoring_params:
        port: 8080
        url: ping.cgi
      actions:
        failure: respawn

    acme_scaling_driver:
      monitoring_params:
        resource: cpu
        threshold: 10000
      actions:
        max_foo_reached: scale_up
        min_foo_reached: scale_down
```

그림 19. 모니터링 기능을 위한 sample VNFD

2.1.2. 알람 기반 모니터링

기존의 존재하는 모니터링 모듈을 사용하여, Tacker 서버에서 모니터링 기능을 지원하기 위한 알람기반의 모니터링 드라이버가 구현되었다. 아래의 그림은 알람기반의 모니터링 구조를 나타낸다. 기존의 OpenStack에서 모니터링 기능을 제공하는 Ceilometer나 Monasca 혹은 Custom 모니터링 기능을 통해 Tacker 서버에서 VNF를 모니터링 한다.

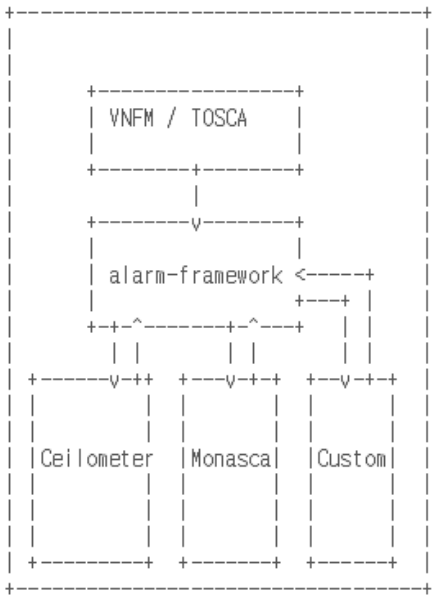

그림 20. 알람 기반 모니터링

아래의 그림은 알람 모니터링을 위한 VNF Descriptor 예제를 나타낸다. 해당 예제 탬플릿 통해 생성한 VNF의 경우 다음과 같은 특징을 갖는다. 우선 VNF를 위해 1GB의 디스크 사이즈와 512MB의 메모리 그리고 2개의 CPU가 제공된다. 또한, 기본 OS로는 cirros-0.3.4-x86_64가 설치된다. 해당 VNF의 모니터링을 위해서 Ceilometer를 이용하며, Ceilometer에서는 VNF의 자원 상태로 지속적으로 모니터링하면서 탬플릿에 명시된 이벤트가 발생되는 경우 VNFM에게 전송하게 된다. 해당 예제에서는 CPU사용량을 타겟 이벤트로 명시하였다. VNF의 CPU 사용량이 60초 동안 평균적으로 70%이상인 경우 Ceilometer에서 VNFM으로 이벤트를 알려주게 되며, 해당 이벤트를 받은 VNFM은 해당 VNF의 VDU를 scaling up하게 된다.

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: Demo example

metadata:
template_name: sample-tosca-vnfd

topology_template:
node_templates:
   vdu1:
     type: tosca.nodes.nfv.VDU.Tacker
     capabilities:
       nfv_compute:
         properties:
           disk_size: 1 GB
           mem_size: 512 MB
           num_cpus: 2
     properties:
       image: cirros-0.3.4-x86_64-uec
       mgmt_driver: noop
       availability_zone: nova

   vdu1_cpu_usage_monitoring_policy:
       type: tosca.policies.tacker.Monitoring
       targets: [vdu1]
       triggers:
           resize_compute:
               event_type:
                   type: tosca.events.resource.utilization
                   implementation: Ceilometer
               metrics: cpu_util
               condition: utilization greater_than 70%
                   threshold: 70
                   period: 60
                   evaluations: 1
                   method: average
                   comparison: gt
               action:
                   resize: vdu1_scaling_policy
```

그림 21. 알람 모니터링 기능을 위한 sample VNFD

2.2. 모니터링 기능을 이용한 VNF의 고가용성 제공 기능

Tacker에서는 상기에 기술한 모니터링 프레임워크를 기반으로 VNF의 고가용성을 위한 기능 또한 제공하고 있다. 모니터링 프레임워크에서 사용자가 정의한 이벤트가 발생하는 경우 대응 가능한 정책으로는 크게 3가지 (Loging, healing, scaling) 으로 나눌 수 있으며, 본 고에서는 healing과 scaling에 대해서 분석 한다.

2.2.1. VNF Healing 기능

VNF가 불능이 되는 경우 Ping 모니터링 드라이버를 통해 해당 이벤트를 VNFM이 인지하고, 해당 VNF를 지우고 다시 배포 할 수 있다. 해당 기능을 respawn기능으로 명명하고 있다.

```
vdu1:
  monitoring_policy:
    ping:
      actions:
        failure: respawn

vdu2:
  monitoring_policy:
    http-ping:
      monitoring_params:
        port: 8080
        url: ping.cgi
      actions:
        failure: respawn

    acme_scaling_driver:
      monitoring_params:
        resource: cpu
        threshold: 10000
      actions:
        max_foo_reached: scale_up
        min_foo_reached: scale_down
```

그림 22. respawn 기능을 위한 sample VNFD

위의 그림은 respawn 기능을 위한 VNFD 예제를 나타낸다. 해당 템플릿을 통해 발생된 VNF의 경우 vdu1에 대해서는 VNFM의 모니터링 드라이브와 VNF간의 주기적인 ping을 통해 vdu1의 불능상태를 인지하고, 미리 정해진 n번의 ping에 대해서 반응이 없는 경우 vdu1이 불능상태라고 정의된다. 이와 유사하게 vdu2의 경우 http-ping을 통해 미리정해진 n번의 http-ping에 대해서 반응이 없는 경우 vdu2가 불능 상태로 정의하게 된다. vdu1과 vdu2가 불능상태로 정의되면 VNFM은 vdu들을 삭제하고 다시 배포하는 respawn 기능을 수행하게 된다. respawn기능을 통해 VNF가 불능 상태가 되더라도 즉각적으로 다시 생성하기 때문에 VNF의 고가용성을 지원해 줄 수 있다.

하지만, respawn 기능의 경우 VNF의 고가용성을 위해 하나의 vdu가 불능상태가 되더라도, VNF를 삭제하고 다시 배포하기 때문에 문제가 없는 vdu들도 삭제되고 다시 배포되는 문제점이 존재한다. 이러한 문제점을 해결하기 위해서 vdu auto-healing 기능이 새로이 추가 되었다. vdu auto-healing 기능은 문제가 존재하는 vdu만 삭제하고 다시 배포하는 기능을 제공한다. 이러한 기능을 통해 문제가 없는 vdu는 지속적으로 서비스가 가능하면서 문제가 발생한 vdu만 대처 할 수 있는 장점이 존재한다.

```
:caption: Example VNFD
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: Monitoring policy action : vdu_autohealing
topology_templete:
  node_templates:
    VDU:
      type: tosca.nodes.nfv.VDU.Tacker
 # ...snip...
      properties:
          monitoring_policy:
              name: ping
              parameters:
                  monitoring_delay: 45
                  count: 3
                  interval: 1
                  timeout: 2
              actions:
                  failure: vdu_autohealing
```

그림 23 vdu auto-healing 기능을 위한 sample VNF Descriptor

위의 그림은 vdu auto-healing을 위한 샘플 VNF Descriptor 를 나타낸다. vdu를 모니터링을 하기 위한 프레임워크는 ping / 알람 모니터링 프레임워크 혹은 다른 프레임 워크를 사용해도 무방하다. 해당 예제에서는 vdu를 ping을 통해서 불능 유무를 확인하며, 불능이 되는 경우 해당 vdu만 복구시키는 기능을 제공함으로써 고 가용성을 지원 한다.

2.2.2. VNF Scaling 기능

VNFM에서 VNF를 배포할 때, VNF의 CPU와 메모리 자원은 VNFD에 작성된 값들로 설정되어 배포된다. 처음 배포할 때부터 VNF의 CPU 및 메모리 자원이 정해지므로 생성된 VNF의 자원을 충분히 활용하지 못할 경우 자원을 낭비할 수 있는 문제가 생기고, 반대로 VNF 자원 용량 이상의 작업들이 VNF에 요청된다면, 해당 VNF의 과부하가 발생할 수 있는 문제가 있다. 이를 방지하기 위해 VNF 자원들의 설정을 변경하고 재 배포 하는 방안이 있으나 이는 서비스의 중단을 유발할 수 있다는 문제점이 있다. 따라서 이를 해결하기 위해 태커의 Newton 버전부터 VNF scaling 기능이 구현되었다.

VNF scaling기능은 VNFD에 초기 인스턴스 수, 최소 인스턴스 수, 최대 인스턴스 수, 추가 및 제거되는 인스턴스들의 증가량을 정의할 수 있게 한다. 또한 연속적인 scaling이 일어날 수 있는 최소한의 시간 간격인 cooldown 시간도 정의할 수 있다. scaling의 종류로는 같은 VNFD를 기반으로 새로운 VNF 인스턴스를 생성하는 스케일-아웃, 중복된 기능의 VNF 인스턴스를 제거하는 스케일-인 동작을 지원한다. 아래 그림은 Scaling 정책이 포함된 샘플 VNFD를 나타낸다.

현재 VNF scaling을 실행하는 방법으로는 두 가지가 있다. 첫 번째 방법은 manual-scaling으로, 이는 존재하는 VNF들의 REST API를 통해 scaling을 수행하는 방법이다. scaling action의 URL로 POST 메시지를 보내면 VNF scaling이 수행된다. 해당 메시지는 scale 타입과 scaling 정책의 이름이 포함되어 있다. scale 타입은 사용자가 선택 가능하며 scaling 정책의 이름은 VNFD에서 정의할 때 사용한 이름을 사용한다. VNF의 mannual scaling 기능은 아래와 같이 수행할 수 있다.

```
tacker vnf-scale --vnf-name <vnf-name> --scaling-policy \
 <scaling-policy-name> --scaling-type <out/in>
```

VNF scaling을 수행하는 두 번째 방법으로는 모니터링 드라이버를 이용한 auto-scaling 방법이 있다. 모니터링 드라이버는 성능 보틀넥과 같은 사전에 정의된 이벤트들을 기반으로 현재 동작하고 있는 VNF들의 scaling-out/in을 자동적으로 수행할 수 있게 한다. 성능 보틀넥은 VNF의 CPU, 메모리 사용량과 같은 성능 지표들로 나타낼 수 있다. 해당 수치들의 경계값과 모니터링 빈도, 모니터링 이벤트 타입, 어떤 모니터링 드라이버를 사용할 것인지는 VNFD에서 정의할 수 있다. auto-scaling 기능은 알람을 HOT template에 등록해서 모니터링 구성요소들이 Heat 자원 그룹들에서 scaling을 실행시킬 수 있게 한다. 아래 그림은 Ceilometer를 모니터링 구성요소로 사용했을 때의 동작과정을 나타낸다.

VNFD 내에 모니터링 정책을 포함한 VNF가 배포될 때, 알람 URL이 Tacker의 webhook 요소에 의해 생성된다. 모니터링 성능 지표가 정해진 경계값보다 높거나

```yaml
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0

description: sample-tosca-vnfd-scaling

metadata:
  template_name: sample-tosca-vnfd-scaling

topology_template:
  node_templates:
    VDU1:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        image: cirros-0.4.0-x86_64-disk
        mgmt_driver: noop
        availability_zone: nova
        flavor: m1.tiny

    CP1:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        management: true
        order: 0
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL1
        - virtualBinding:
            node: VDU1

    VDU2:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        image: cirros-0.4.0-x86_64-disk
        mgmt_driver: noop
        availability_zone: nova
        flavor: m1.tiny

    CP2:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        management: true
        order: 0
```

그림 24. Scaling 정책이 포함된 샘플 VNF Descriptor (1/2)

```
      anti_spoofing_protection: false
    requirements:
      - virtualLink:
          node: VL1
      - virtualBinding:
          node: VDU2

  VL1:
    type: tosca.nodes.nfv.VL
    properties:
      network_name: net_mgmt
      vendor: Tacker

policies:
  - SP1:
      type: tosca.policies.tacker.Scaling
      targets: [VDU1, VDU2]
      properties:
        increment: 1
        cooldown: 120
        min_instances: 1
        max_instances: 3
        default_instances: 2
```

그림 25. Scaling 정책이 포함된 샘플 VNF Descriptor (2/2)

낮아지면 모니터링 드라이버는 Ceilometer로부터 알람을 받게 되고, 이는 Heat를 통해 자동적으로 scaling 기능을 수행하도록 한다.

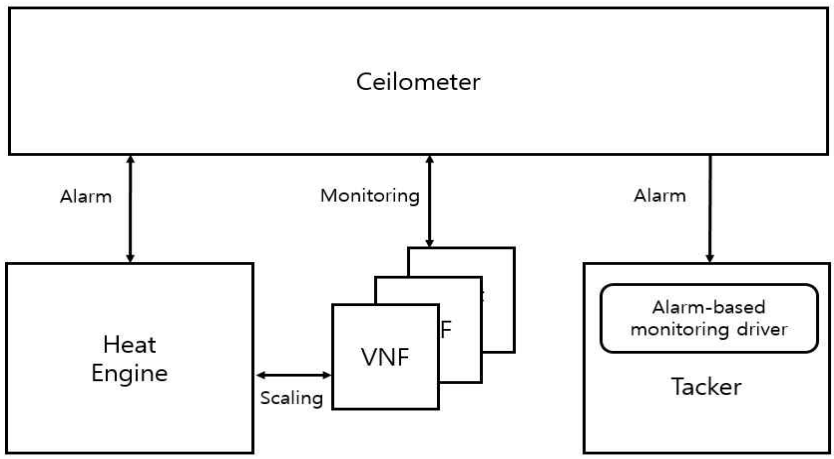

그림 26. 알람 기반의 모니터링을 이용한 VNF auto-scaling

아래 그림은 manual scaling-out 이후에 오픈스택의 서버 리스트들을 확인한 내용이다. 기존 VNF에 속하던 VDU11, VDU12들이 각각 하나씩 더 추가됨을 확인할 수 있다.

```
stack@ubuntu:~$ openstack server list --fit-width
+----------------------+----------------------+--------+-----------------+-----------------------+---------+
| ID                   | Name                 | Status | Networks        | Image                 | Flavor  |
+----------------------+----------------------+--------+-----------------+-----------------------+---------+
| 54894896-f27c-4376-8 | VN-dadge-uj3u45yqb54 | ACTIVE | net0=10.10.0.135 | OpenWRT               | m1.tiny |
| ba8-c591d671e547     | 5-iremslrmbny2-VDU12 |        |                 |                       |         |
|                      | -rs6xfbu5r3ox        |        |                 |                       |         |
| 97561ff8-6caf-40d4-9 | VN-dadge-uj3u45yqb54 | ACTIVE | net0=10.10.0.200 | OpenWRT               | m1.tiny |
| 889-7b0ce34d752d     | 5-iremslrmbny2-VDU11 |        |                 |                       |         |
|                      | -be7j32lu52eo        |        |                 |                       |         |
| 6eb03412-0c3d-468a-b | VN-dadge-            | ACTIVE | net0=10.10.0.6   | OpenWRT               | m1.tiny |
| 975-b02f4170b4f0     | frv2u62yurjh-        |        |                 |                       |         |
|                      | gvd4ov6bpeqt-        |        |                 |                       |         |
|                      | VDU12-eh6g7kupknmt   |        |                 |                       |         |
| 7679cf35-b5d7-4a88-b | VN-dadge-            | ACTIVE | net0=10.10.0.7   | OpenWRT               | m1.tiny |
| 618-39a047fbe4d3     | frv2u62yurjh-        |        |                 |                       |         |
|                      | gvd4ov6bpeqt-        |        |                 |                       |         |
|                      | VDU11-5sq4evqu6hsm   |        |                 |                       |         |
| bff06347-e497-44ed-  | http_server          | ACTIVE | net0=10.10.0.53  | cirros-0.4.0-x86_64-di | m1.tiny |
| 94d8-37fe528d92e0    |                      |        |                 | sk                    |         |
| be6f5d70-7cd3-4697-8 | http_client          | ACTIVE | net0=10.10.0.17  | cirros-0.4.0-x86_64-di | m1.tiny |
| e04-5a6fac6ec98e     |                      |        |                 | sk                    |         |
+----------------------+----------------------+--------+-----------------+-----------------------+---------+
```

그림 27. VNF Scaling-out 이후의 서버 리스트

그러나 현재 태커에서는 VNF의 scaling 기능만 지원 하고 있고 VNFFG의 scaling 기능은 구현되지 않았다. VNF뿐만 아니라 VNFFG의 scaling 또한 지원된다면 사용자에게 VNFFG 수준에서도 고가용성을 제공해줄 수 있을 것이다.

3. VNFFG의 고가용성을 위한 기능

3.1. VNFFG 고가용성의 필요성

2장에서 Tacker는 VNF의 고가용성을 위해 VNF healing 기능과 scaling 기능을 제공한다는 점을 알 수 있었다. 따라서 VNF에 장애가 발생하거나 발생할 가능성이 높아질 경우 VNF 차원에서 복구와 예방하는 기능이 실행될 수 있다. 그러나 만약 VNF가 특정 VNFFG에 속해있을 경우 해당 VNF에 장애가 발생하여 auto-healing 기능 및 respawn 액션이 실행되어도 기존의 VNFFG는 복구하는 기능까지는 Tacker에서 지원하지 않고 있다. 또한, VNF scaling-out 기능으로 추가로 생성된 VNF 인스턴스들은 기존의 VNF 인스턴스가 속한 VNFFG에 속하지 않는다는 문제가 있다. 따라서 특정 VNF 인스턴스에 많은 부하가 발생하여 scaling-out을 수행해도 VNFFG에 따르는 트래픽들이 같은 VNF 인스턴스를 계속 거쳐갈 것이므로 scaling-out의 이점을 잃게 된다는 문제가 있다. 결과적으로 VNF의 healing 및 scaling은 VNFFG의 가용성을 유지하기 어렵다. 3.2절에서는 VNFFG의 고가용성을 위한 기능들의 개발을 위해 현재 Tacker에서 제안된 스펙 분석 및 개발 진행 사항에 관해 서술한다.

3.2. VNFFG 고가용성을 위한 기능

3.2.1. VNFFG Auto-healing

Tacker의 NFVO 는 VNFFG 구성을 위해 Neutron 프로젝트의 포트-체이닝을 이용한다. 포트-체이닝을 위해 우선 하나의 VNF 인스턴스 내의 포트들을 묶어 포트-페어를 형성하고 같은 VNF 인스턴스들의 포트-페어들을 묶어 포트-페어-그룹을 형성한다. 포트-페어-그룹은 같은 VNF 기능을 수행하는 인스턴스들의 집합을 의미한다고 볼 수 있다. 그리고 생성된 포트-페어-그룹들을 순차적으로 묶어 포트-체인을 형성하면 VNFFG 가 구성된다.

2장에서 언급했듯이 Tacker는 ping과 http-ping, 알람 모니터링 드라이버를 이용하여 개별적인 VNF들에 대해 auto-healing 기능을 지원한다. 그러나 현재 Tacker에서는 auto-healing으로 respawan된 VNF들의 ID는 그대로 유지되지만 Connection Point(CP)는 유지되지 않고 변한다는 문제가 있다. CP ID는 Neutorn SFC에서 포트-페어로서 사용되고 있기 때문에 resapwn된 CP의 ID도 그대로 유지하는 기능이 필요하다. 따라서 이를 위한 스펙이 현재 등록된 상태이다. 해당 스펙은 단일 VIM 사이트 내에서 VNFFG의 auto-healing에 관한 기능들을 제안했다. 제안된 스펙의 구조는 아래 그림과 같다. 해당 구조에서는 vnffg-ha engine 모듈이 VNFFG의 auto-healing 기능, scaling 기능을 제공해줄 수 있다.

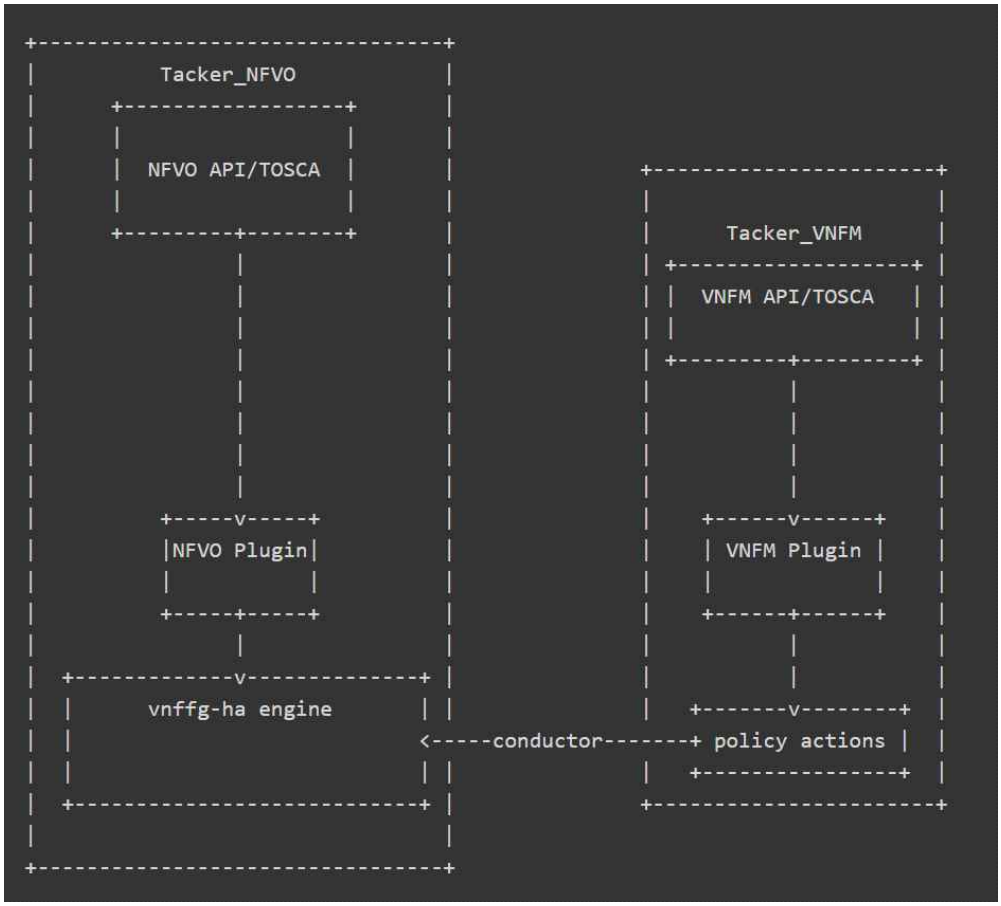

그림 28. VNFFG auto-healing 및 scaling 을 위한 구성 요소들

VNF의 healing 기능 자체는 VNFM이 담당하고 있기 때문에 VNFFG 관리를 담당하는 NFVO에서는 관련 이벤트의 발생 여부를 알 수 없다. 따라서 VNFM에서 VNF와 관련된 이벤트가 발생했을 때 이를 NFVO 모듈 쪽에 알려줄 수 있는 새로운 policy action이 필요하다. 또한 NFVO 모듈에서는 VNF respawn 이벤트 정보를 받았을 때 해당 VNF가 VNFFG에 포함되어 있었는지를 확인하고, 해당 VNFFG 정보를 vnffg_db로부터 수집해야 한다. VNFFG ID를 기반으로 Neutron SFC에서 사용되고 있는 체인의 ID를 얻어오고 이를 통해 체인의 포트-페어-그룹에서 기존의 장애가 발생한 VNF의 CP를 제거하고 새로 respawn된 VNF의 CP ID를 추가함으로써 VNFFG Auto-healing 기능을 제공할 수 있다.

3.2.2. VNFFG Scaling

VNFFG scaling 기능 또한 VNFFG healing과 같이 vnffg-ha engine 모듈 및 다음과 같은 기능들을 구현함으로써 제공될 수 있다. VNFM에서는 scaling을 수행한 VNF 인스턴스들의 정보를 DB 에 저장하는 기능이 필요하다 [5]. 또한 VNF scaling으로

VNF 인스턴스들이 생성 및 제거되었을 때 해당 정보를 NFVO 에게 전달해줄 수 있는 정책 행동이 필요하다. 해당 정책 행동은 VNFD에서 설정되어 실행될 수 있다. NFVO 에서는 VNFM 으로부터 받은 정보들을 기반으로 VNF가 현재 배포된 VNFFG 중 어떤 VNFFG 에 속해있는 지를 확인하는 기능이 필요하다. 위의 기능들은 VNFFG healing을 제공하기 위해 함께 필요한 기능들이고 VNFFG scaling의 경우 Neutron의 포트-체인 API 를 통해 해당 VNFFG 에 대응하는 포트-체인의 업데이트를 수행하는 기능이 필요하다. 또한 VNFFG 를 구성할 때 scaling 기능 정책의 적용 유무를 VNFFGD 에서 설정할 수 있는 기능이 필요하다.

VNFFG scaling은 다음과 같은 상황에서 고가용성을 제공해줄 수 있을 것으로 예상된다. 배포된 VNFFG 내에서 특정 VNF 인스턴스로 트래픽이 집중될 경우 VNFFG 스케일-아웃 기능을 이용하여 새로운 VNF 인스턴스들로 트래픽을 분산시키는 로드 밸런싱 기능을 제공해줄 수 있을 것이다. 또한 전체 트래픽이 감소하여 다수의 VNF 인스턴스들을 이용한 VNFFG 구성이 필요하지 않은 경우에는 VNFFG 스케일-인 기능을 이용하여 VNF 의 자원 활용을 최적화할 수 있을 것이다.

4. 결론

본 문서에서는 OpenStack Tacker의 구성요소들을 설명하고 각 요소들의 기본적인 사용법을 및 VNF의 고가용성을 위한 모니터링 프레임워크와 VNF healing, VNF scaling 기능에 대해 분석하였다. 그리고 현재 지원되지 않고 있는 VNFFG의 고가용성을 위한 기능들의 필요성 및 이를 위한 auto-healing 기능, scaling 기능 개발 사항에 대해 기술하였다. 향후 해당 기능들을 구현하고 VNFFG scaling을 트래픽 양에 따라 유연하게 수행할 수 있는 알고리즘을 제안할 계획이다.

References

[1] Tacker, [Online] Available : https://wiki.openstack.org/wiki/Tacker
[2] OpenStack, [Online] Available : https://www.openstack.org/
[3] ETSI GS NFV-MAN 001 V1.1.1 (2014-12) [Online] Available : https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
[4] VNFFG Healing Specification, [Online] Available : https://specs.openstack.org/openstack/tacker-specs/specs/pike/vnffg-autohealing.html
[5] VNFFG Scaling Specification, [Online] Available : https://specs.openstack.org/openstack/tacker-specs/specs/pike/vnffg-scaling.html

K-ONE 기술 문서

- 이 문서의 기술적인 내용은 프로젝트의 진행과 함께 별도의 예고 없이 변경될 수 있습니다.

- 본 문서와 관련된 문의 사항은 아래의 정보를 참조하시길 바랍니다.
  (Homepage: http://opennetworking.kr/projects/k-one-collaboration-project/wiki,
  E-mail: k1@opennetworking.kr)

작성기관: K-ONE Consortium
작성년월: 2019/11