논문 19-44-04-15 The Journal of Korean Institute of Communications and Information Sciences '19-04 Vol.44 No.04
https://doi.org/10.7840/kics.2019.44.4.728

NFV 기반환경에서의 정책기반 네트워크 서비스 배치 설계 및 구현

황 공 푸 억•, 김 영 한°

Design and Implementation of Policy based Network Service Placement on NFV Infrastructure

Cong-Phuoc Hoang•, Young-han Kim°

요 약

NFV(network function virtualization) 기반 하에서 NS(network service) 배치에 관한 세부 요구사항은 네트워크 서비스 디스크립터를 사용하여 지정하게 된다. 그러나 이러한 방법은 네트워크 합수의 서비스별 요구와 NFV 인프라 능력의 다양성 등으로 최적으로 배치가 안 될 수 있다. 본 논문에서는 이러한 문제를 해결하는 방법으로 TOSCA 기반언어를 사용한 최적화된 배치방법으로서 정책기반의 프레임워크를 제안한다. 제안된 방법은 네트워크 인프라 구조, 플랫폼의 용량, 및 서비스별 요구조건들을 모두 고려하여 최적화된 배치를 가능하게 할 수 있다. 오픈소스 MANO 소프트웨어를 사용하여 제안된 정책기반 프레임워크를 구현하였고 VNF의 요구된 세부 조건을 만족하면서 자동으로 NS를 배치할 수 있음을 보였다.

Key Words : NFV, Orchestration, TOSCA, Network Service, Policy Management

ABSTRACT

In the Network Function Virtualization (NFV) environment, the placement requirements for network services can be specified manually by using network service descriptors. But its not usually optimal because of constituent virtual network functions with service-specific requirements and variant capabilities of NFV infrastructures (NFVIs). In order to solve the above issue, in this paper, a policy framework is proposed for automating network service deployment by using TOSCA specification language. The proposed policy framework could support the optimized placement by considering all of the network infrastructure, the platform capabilities, and the service specific constraints. Using an open source MANO (management and orchestration) software, the proposed policy framework is implemented and shows that it could place the NS automatically while meets the required specific constraints of VNFs (virtual network functions).

Ⅰ. Introduction

With the network function virtualization (NFV)[1] technology, the virtual network functions (VNFs) can be easily deployed on virtual machines or containers by decoupling the network functions from

※ This research was supported by the MSIT(Ministry of Science, ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2019-2017-0-01633) supervised by the IITP(Institute for Information & communications Technology Promotion)
• First Author : Soongsil Univ. Dep. of Electronic Engineering, phuoc.hc@dcn.ssu.ac.kr, 학생회원
° Corresponding Author : Soongsil Univ. Dep. of Electronic Engineering, younghak@ssu.ac.kr, 종신회원
논문번호 : 201812-389-D-RN, Received December 17, 2018; Revised February 14, 2019; Accepted February 15, 2019

the dedicated hardware appliances. Instead of implementing network functions, such as router, load balancer, and firewall etc., on some special hardware systems, they can be located on any general-purpose compute hardware or on the shared resource environment, i.e. the cloud platform, so that the operational expenditure (OPEX) and the capital expenditure (CAPEX) could be reduced. The VNFs can be interconnected to make network services(NSs) and the automated deployment and provision systems for the NS across data centers are becoming important to overcome the operational complexity and overhead increased by replacing the legacy physical network devices.

The NFV management and orchestration (MANO) system automates the deployment of network services over NFVIs (NFV infrastructures). Nowadays, most of the MANO systems support this deployment of VNFs and NSs using descriptors such as VNF descriptor (VNFD), VNF forwarding graph descriptor (VNFFGD) and NS descriptor (NSD) for specifying the requirements of each component. These descriptors are usually composed by the TOSCA (topology and orchestration specification for cloud applications)[2] which is a standard language for specifying the topology of cloud based services. By using service descriptors, network operators and consumers can define their own services and requirements (i.e., placement attributes like affinity or anti-affinity constraints of VNFs, availability zone, region constraints, and quality of service for VNFs and NSs.). The MANO systems extract the service-specific attributes from the service descriptor and then manage and orchestrate the network service using these attributes.

Although the MANO system can manage the network functions with these descriptors, it is also expected to support the advanced features such as the quality of service (QoS) and SLAs (e.g. bandwidth, latency, high availability, etc.) that can be realized with predefined semantics of the descriptors such as the hardware capability and the vendor specific VNF features which affect the performance of network service. As described in [3], network operators can reuse the same VNFDs to create their own network services. However, with different network service's requirements, VNFD's properties are also required to be updated. Therefore the solution for resolving the conflicts between VNFD and NSD is required. In the telecom networks, these capabilities are usually come from the policy management function with which the VNFs of a NS is properly placed on the infra system to meet the QoS and SLAs depending on runtime conditions of environment.

The policy management in MANO has been addressed by ETSI GS NFV-MAN 001[3] and GR NFV-IFA 023[4]. But these reports only provide the high-level use cases and the operation way in the NFV based networks, and the detailed information about the policy integration into NFV MANO system are not included. In the previous researches about the policy management, [5] provides several use cases of policy in MANO architecture, but no real work flows were studied. Bari et al. [6] also proposed an autonomic QoS policy enforcement framework for software defined networks by specifying service level agreements, and Alexander et al.[7] extended the TOSCA for policy description. However, these works only cover the framework but not mention the ways to implement the policies with MANO. In [8], the authors proposed the security policies to manage the virtual security functions to support the policy-based actions in 5G network, but the detail policies were not presented with real experiments.

In this paper, we propose a policy framework in NFV MANO system to automate the NS placement over multiple virtual infrastructures while satisfying the required service requirements. The proposed framework has the following features:

- use the standard TOSCA specification language to describe the placement policy rules
- automatically process the policy in run time environment
- resolve the conflicts between the NS and the VNF level policies.

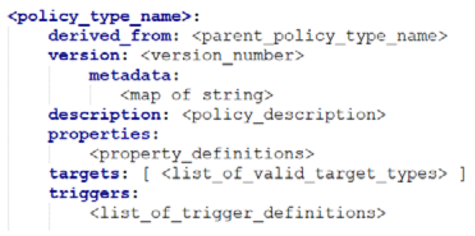
After the introduction, in section II, we describe the extension of the TOSCA template to support the placement policies. The detailed design and implementation of the policy framework are described in section III and IV. In section V, we conclude our work.

Ⅱ. Design of the Policy Template

In TOSCA, the standard definition of policy is shown as in Figure 1 which includes the properties of a policy, it's relation to other policies, and the applicable target node types. In this work, we design the new policy types for VNF and NS, which include placement and QoS-aware placement, according to this standard format.

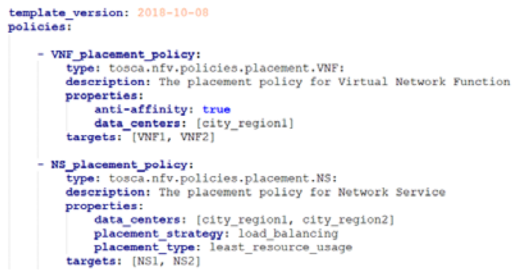
In Figure 2, the placement policy template is shown, it tends to determine the placement of VNFs of a network service with specific values, in which values are designed to enable the deployment of VNF's resources in specified locations. Placement policy allows network operators to specify host, availability zone, region with special rules, such as affinity or anti-affinity directly in policy template, or service's requirements. For instance, in "NS_placement_policy" policy, users can specify which data centers VNFs can be located on with values of "data_centers" field. "NS_placement_policy" policy also let users define how VNFs are distributed on these data centers by setting "placement_strategy" value ("load_balancing", "round_robin", "random", etc.). Depend on values of properties in placement policy and monitoring metrics, the placement of VNF will be automatically determined in real time environment.

Figure 3 gives an example of QoS-aware placement policy, that assure performance of a



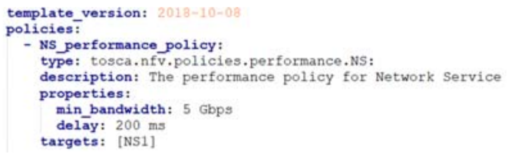
```
<policy_type_name>:
    derived_from: <parent_policy_type_name>
    version: <version_number>
        metadata:
            <map of string>
    description: <policy_description>
    properties:
        <property_definitions>
    targets: [ <list_of_valid_target_types> ]
    triggers:
        <list_of_trigger_definitions>
```

Fig. 1. TOSCA definition of policy



```
template_version: 2018-10-08
policies:

    - VNF_placement_policy:
        type: tosca.nfv.policies.placement.VNF:
        description: The placement policy for Virtual Network Function
        properties:
            anti-affinity: true
            data_centers: [city_region1]
        targets: [VNF1, VNF2]

    - NS_placement_policy:
        type: tosca.nfv.policies.placement.NS:
        description: The placement policy for Network Service
        properties:
            data_centers: [city_region1, city_region2]
            placement_strategy: load_balancing
            placement_type: least_resource_usage
        targets: [NS1, NS2]
```

Fig. 2. TOSCA policy in YAML for placement policy



```
template_version: 2018-10-08
policies:
  - NS_performance_policy:
    type: tosca.nfv.policies.performance.NS:
    description: The performance policy for Network Service
    properties:
      min_bandwidth: 5 Gbps
      delay: 200 ms
    targets: [NS1]
```

Fig. 3. TOSCA policy in YAML for QoS-aware placement policy

network services to satisfy user requirements. Because network services is not real object that can be deployed on infrastructure as VNFs, so QoS requirements must be translated into VNF placement decisions. To achieve QoS requirements, policy framework have to determine sufficient resources VNFs need to be provisioned and then figure out proper data center, availability zone or even host to place VNFs. In our work, we use a key performance indicator (KPI) functional block to determine sufficient resources for VNFs and then VNFD templates are tailored accordingly to NS-specific requirements. QoS-aware placement policy is useful when VNFD templates can be reused for each NS deployment. Rather than modifying VNFD templates manually to support each NS requirement, network operators only need to use different QoS-aware placement policy and original VNFD templates are automatically modified to meet requirements. To solve the conflicts between the NS and the VNF level policies, the framework uses global and local policies architecture to apply managing policies into MANO system. The network service level policies are global policies which have broader scope and higher priority than the local policies, so that they will affect the subsystem policies applying on VNFs. As shown in the Fig. 4, the global policy defines the requirements for the network service NS1 with the

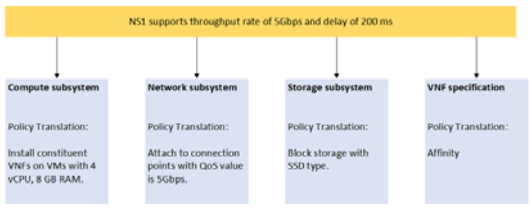

NS1 supports throughput rate of 5Gbps and delay of 200 ms
Compute subsystem
Policy Translation:
Install constituent VNFs on VMs with 4 vCPU, 8 GB RAM.
Network subsystem
Policy Translation:
Attach to connection points with QoS value is 5Gbps.
Storage subsystem
Policy Translation:
Block storage with SSD type.
VNF specification
Policy Translation:
Affinity

Fig. 4. Example for global and local policies

through put rate of 5Gbps and the end-to-end delay of 200ms. Then the policy framework processes this requirement to calculate the proper properties for VNFs. If the pre-defined values of VNF's properties could not meet the requirements for network service, they should be replaced by new values to solve the conflicts.

Ⅲ. Design of the Placement Framework

High-level architecture of policy framework is shown in Fig. 5. Inputs of this framework are network service templates with placement policies enabled and resource's usage of infrastructures. The output of policy framework is the placement decisions for VNFs. The placement decisions are translated into specific values of VNF configuration in output templates, which are used to deploy VNFs on destination infrastructure properly, to satisfy the network service requirements. To support making placement decisions, which depends on run time conditions of infrastructures, framework continuously gathers monitoring metrics of consumed resources such as CPU utilization, memory usage, network bandwidth, etc. using monitoring tools.

Policy framework is based on six key components and VIM's APIs. In our case, we used OpenStack APIs such as Nova and Ceilometer APIs to know about hypervisor information of underlying infrastructures. The detailed functional components of policy framework are shown in Figure 6, which includes:

1) **Monitoring module**: it is used to monitor pre-defined metrics of allocated resources and NFV infrastructures. It then regularly pushes monitoring metrics to policy engine with necessary information to support making efficient VNF placement decisions.

2) **Policy handler**: Policies can be created from OSS/BSS or directly from users. Policy handler is a hub for retrieving policies. It is in charge of parsering policies, collecting monitoring information and then it pushes them to policy engine.

3) **Policy engine**: Policy engine is in charge of evaluating of polices and it uses metrics and monitoring statistics to determine placement decisions of VNFs based on KPI to achieve network service's QoS requirements.

4) **Policy validator**: This module bases on the native TOSCA parser[9] to interprets the user request (with TOSCA template) and verifies the request correctness.

5) **Enforcer**: It is responsible for translating VNF's placement decisions, which are results of Policy engine, into appropriate values in VNF template. Enforcer tends to support various cloud orchestrator such as OpenStack Heat, AWS CloudFormation, etc.

6) **KPI**: Key performance indicator (KPI) defines a set of key-value, that match requirement to a specific value for each resource in infrastructure. For

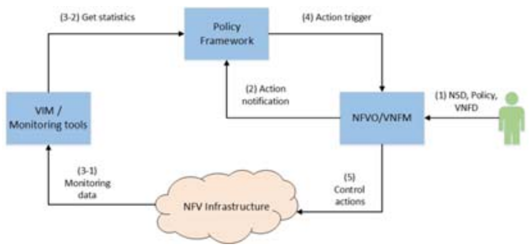


Fig. 5. Framework high level architecture

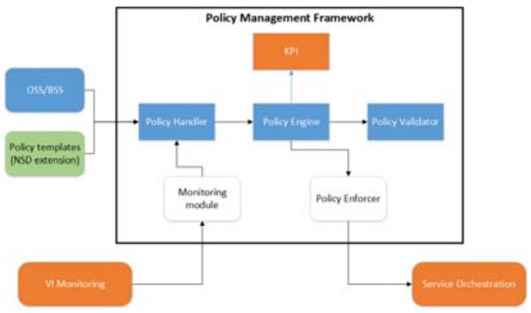


Fig. 6. Policy framework components

instance, requirement for low latency, high packet throughput, low jitter constraints of a VNF can be fulfilled by a kind of flavor in OpenStack environment. The appropriate resource types and resource configuration of VNF descriptor must be evaluated according to KPI in order to reach network service's QoS requirements.

The Figure 7 illustrates the sequence diagram of creating network service in MANO system with policy framework extension. Before NFVO send requests to VNFM for launching VNFs using VNFD templates, firstly configuration values of VNFD template will be processed in policy framework to determine optimal placement to satisfy requirements from NSD and policies templates. Policy framework collects consumed resources from virtual infrastructures by interacting with virtual infrastructure managements (VIMs) and monitoring tool such as Ceilometer and Nova compute in OpenStack cloud platform.

After collecting infrastructure data, policy framework uses predefined data in KPI component to automatically figure out proper configurations for VNF's placement. For example, to achieve throughput rate of 5Gbps, compute resource of VNFs must be 4 vCPU, 8 GB RAM and 5 Gbps throughput bandwidth for network card interface. Policy framework then updates configuration values to VNFD templates and send them back to NFVO. To the end, NFVO does normal procedures by passing new updated VNFD templates to VNFM to launch new VNFs.

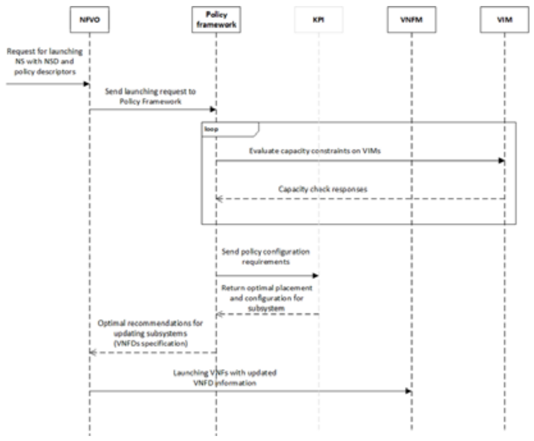

NFVO
Policy framework
KPI
VNFM
VIM
Request for launching NS with NSD and policy descriptors
Send launching request to Policy Framework
loop
Evaluate capacity constraints on VIMs
Capacity check responses
Send policy configuration requirements
Return optimal placement and configuration for subsystem
Optimal recommendations for updating subsystems (VNFDs specification)
Launching VNFs with updated VNFD information

Fig. 7. The procedure for creating network service

Ⅳ. Implementation and Experiments

In this paper, we develop a prototype of policy framework using open source projects (Tacker [10] and Tricircle [11]) and we also evaluate proposed framework with VNF placement scenarios. We consider Tacker project for NFVO and VNFM functional blocks to manage and orchestrate network services. OpenStack is used as VIMs for managing virtual infrastructures. Figure 8 describes applying policy framework into Tacker MANO system. New functional blocks of policy framework are policy handler, policy validator, policy engine and policy enforcer, while monitoring module just is invoked existing infrastructure APIs. Managing multiple virtual infrastructure is also important, we also use the Tricircle project to provide unified control plane view for multi-region OpenStack deployments in our experiment environment. The framework utilize VNFDs, NSDs, policies template from NFV catalogs to determine placement of VNFs on OpenStack VIMs.

Table 1 shows the details environment's specification. We customized the Tacker source code

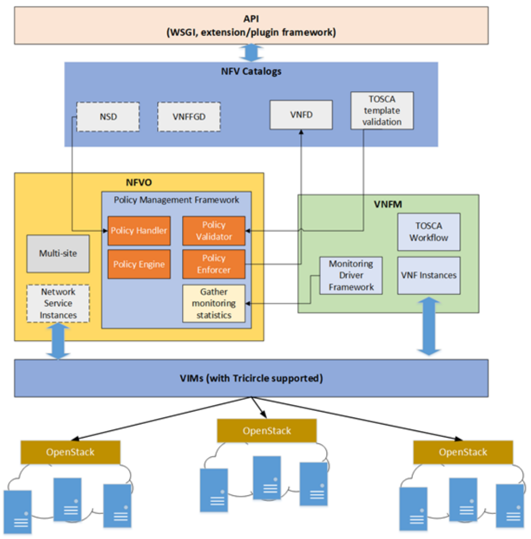

API
(WSGI, extension/plugin framework)
NFV Catalogs
NSD
VNFFGD
VNFD
TOSCA template validation
NFVO
Policy Management Framework
Multi-site
Policy Handler
Policy Validator
Policy Engine
Policy Enforcer
Network Service Instances
Gather monitoring statistics
VNFM
TOSCA Workflow
Monitoring Driver Framework
VNF Instances
VIMs (with Tricircle supported)
OpenStack
OpenStack
OpenStack

Fig. 8. Implementation of Policy Framework in MANO system

Table 1. Specifications of the experiment.

Entity	Details
Server hardware	Intel(R) Core(TM) i5-7600 CPU @ 3.50GHz (4 Cores), 16GB RAM, 120GB SSD
Operation System	Ubuntu server 16.04 LTS, 64-bit
Tacker	Version: Master
Tricircle	Version: Master
Devstack	Version: Master

to support policy framework. For simulating three data centers, we deployed DevStack tool [12] to create OpenStack environments on three hardware servers. In Fig. 9, the dash board of the implementation shows the applied policy framework deployed automatically the VNFs of a network service on multiple VIMs as user's requirements. With the policy framework, we could calculate the real-time resource usage and user's requirement to assign the VNFs on proper VIM from multiple VIMs. In the experiment, the overall NS provision time is measured as shown in Fig. 10. The overall NS provision time is composed of the mapping time and the instantiation time. The mapping time is the sum of the policies's mapping time and the time required to collect the monitoring metrics in real time conditions. And, the VNF instantiation time is for the parsing information from the VNFD, the generation of the HEAT template, and to send the request from Tacker VNFM functional block to the OpenStack Heat service to instantiation the VMs in the NFVI. The figure shows that the mapping time is minor compared to the instantiation time.

In the second experiment, the internal work flow of the proposed policy framework is investigated. There are two kind of templates, the TOSCA network service template for network operators to describe their service with requirements and the other template is the output of Tacker with policy framework enabled, which is used to launch the necessary resources of network services. Figure 11 shows a sample of input network service template with policy using TOSCA standard with the QoS requirement. The VNFDs can be reused in many NSDs, so that the values of hardware resources are not defined in VNFD template and usually be specified later depends on the virtual infrastructure used. In some cases, the network service requires to have a specific performance, so we improved the VNF's description by adding the QoS resource's definition based on each NS' requirements that are not supported in the Heat-translator. As an example, if a NS requires a lower end-to-end delay time, the

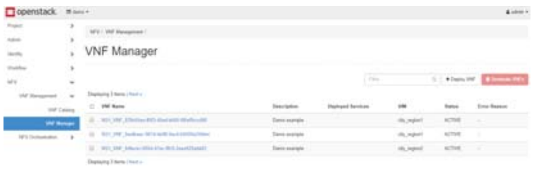


Fig. 9. Multiple VNFs of a NS are deployed automatically on VIMs by policy framework executions

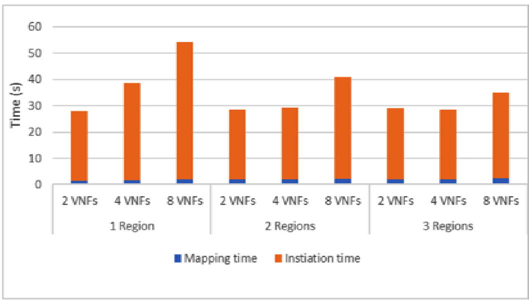


Fig. 10. Mapping and instantiation time for multiple NS instantiation

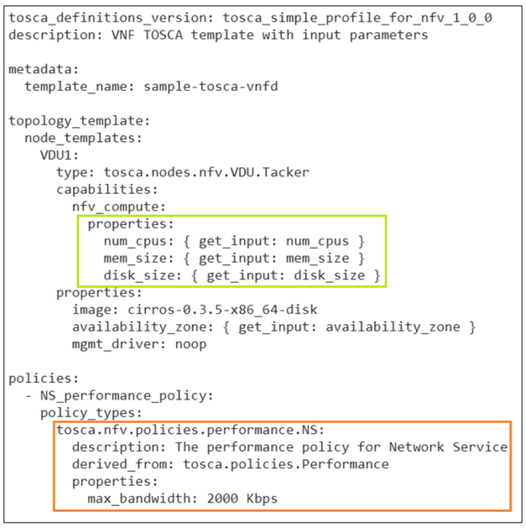
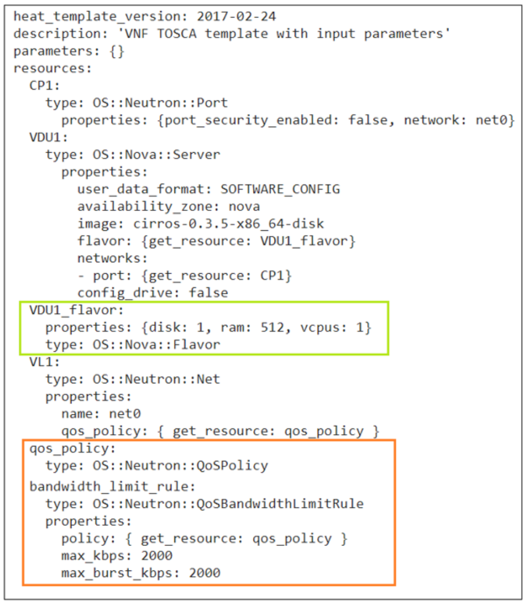

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: VNF TOSCA template with input parameters

metadata:
  template_name: sample-tosca-vnfd

topology_template:
  node_templates:
    VDU1:
      type: tosca.nodes.nfv.VDU.Tacker
      capabilities:
        nfv_compute:
          properties:
            num_cpus: { get_input: num_cpus }
            mem_size: { get_input: mem_size }
            disk_size: { get_input: disk_size }
      properties:
        image: cirros-0.3.5-x86_64-disk
        availability_zone: { get_input: availability_zone }
        mgmt_driver: noop

policies:
  - NS_performance_policy:
    policy_types:
      tosca.nfv.policies.performance.NS:
        description: The performance policy for Network Service
        derived_from: tosca.policies.Performance
        properties:
          max_bandwidth: 2000 Kbps
```

Fig. 11. TOSCA network service template with performance policy

VNFs could be placed on the same VIM, and affinity rules need be added to the HOT(Heat Orchestration Template) template to place the VNFs on same hardware. We use the OpenStack as a VIM for managing virtual infrastructure, so the HOT template, which is described in Figure 12, will be generated as the output of Tacker. The HOT template will be used by Heat [13] to create the necessary resources for network service. In this work, a new "qos_policy" for Neutron port is added in the HOT template to support the throughput requirement. The flavor's configuration parameters of VDU1 are also updated to support the minimum requirements of QoS as shown in the figure 12.



```
heat_template_version: 2017-02-24
description: 'VNF TOSCA template with input parameters'
parameters: {}
resources:
  CP1:
    type: OS::Neutron::Port
      properties: {port_security_enabled: false, network: net0}
  VDU1:
    type: OS::Nova::Server
      properties:
        user_data_format: SOFTWARE_CONFIG
        availability_zone: nova
        image: cirros-0.3.5-x86_64-disk
        flavor: {get_resource: VDU1_flavor}
        networks:
        - port: {get_resource: CP1}
        config_drive: false
  VDU1_flavor:
    properties: {disk: 1, ram: 512, vcpus: 1}
    type: OS::Nova::Flavor
  VL1:
    type: OS::Neutron::Net
    properties:
      name: net0
      qos_policy: { get_resource: qos_policy }
  qos_policy:
    type: OS::Neutron::QoSPolicy
  bandwidth_limit_rule:
    type: OS::Neutron::QoSBandwidthLimitRule
    properties:
      policy: { get_resource: qos_policy }
      max_kbps: 2000
      max_burst_kbps: 2000
```

Fig. 12. Resulting HOT template

V. Conclusion

In this work, we presented the components of policy framework to determine network service placement decisions over multi-clouds, in which VNF's specification can be determined automatically rather than network operators specify them manually. We also depict how to use TOSCA language to declare non-functional requirements by using policies (placement and QoS-aware placement) for NFV and cloud services. To the end, the implementation is presented by extending current Tacker MANO system and experiment results show use cases of applying policies that affect performance of network services. The proposed framework could be used to place the NS automatically while meets the required specific constraints of VNFs.

References

[1] ETSI NFV, *Network function virtualisation: An introduction, benefits, enablers, challenges & call for action*, Introductory White Paper, Issue 1, SDN & OpenFlow World Congr., Darmstadt, Germany, Oct. 2012.

[2] *Topology and Orchestration Specification for Cloud Applications*, Version 1.0, OASIS Standard TOSCA-v1.0, 2013.

[3] ETSI GS NFV-MAN 001, *NFV Management & Orchestration.* [Online], Available: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[4] ETSI GR NFV-IFA 023, *Report on Policy Management in MANO,* [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/023/03.01.01_60/gr_NFV-IFA023v030101p.pdf

[5] R. Szabo, S. Lee, and N. Figueira, *Policy-Based Resource Management*, RFC Editor, draft-irtf-nfvrg-policy-based-resource-management-02

[6] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "PolicyCop: An autonomic QoS policy enforcement framework for software defined networks," in *Proc. IEEE SDN Future Network Services* (SDN4FNS), pp. 1-7, Trento, Italy, Nov. 2013.

[7] K, Alexander, C. Lee, E. Kim, and S. Helal, "Enabling end-to-end orchestration of multi-cloud applications," *IEEE Access,* vol. 5, pp. 18862-18875, Aug. 2017.

[8] M. S. Siddiqui, et al., "Policy based

virtualised security architecture for SDN/NFV enabled 5G access networks," 2016 *IEEE Conf. Network Function Virtualization and Software Defined Networks* (NFV-SDN), pp. 44-49, Palo Alto, CA, USA, 2016.
[9] *Tosca Parser*, [Online]. Available: https://wiki.openstack.org/wiki/TOSCA-Parser
[10] *Tacker*, [Online]. Available: https://wiki.open-stack.org/wiki/Tacker
[11] *Tricircle*, [Online]. Available: https://wiki.open - stack.org/wiki/Tricircle
[12] *DevStack*, [Online]. Available: https://docs.openstack.org/devstack
[13] *Heat*, [Online]. Available: https://wiki.openstack.org/wiki/Heat
[14] *Kubernetes*, [Online]. Available: https://kubernetes.io/

황 공 푸 억 (Cong-Phuoc Hoang)

2015년 9월 : 하노이과학기술대학교 전자통신학과 학사
2017년 3월~현재 : 숭실대학교 정보통신공학과 석사과정
<관심분야> SDN/NFV, 클라우드 컴퓨팅, 오케스트레이션
[ORCID:0000-0002-3507-5702]

김 영 한 (Young-han Kim)

1986년 2월 : 한국과학기술원 전기 및 전자 공학과 석사
1990년 2월 : 한국과학기술원 전기 및 전자 공학과 박사
1994년~현재 : 숭실대학교 전자정보공학부 교수

<관심분야> 모바일 네트워크, 엣지 클라우드 시스템, ICN 및 센서 네트워킹, 클라우드 컴퓨팅
[ORCID:0000-0002-1066-4818]