논문 18-43-08-09 The Journal of Korean Institute of Communications and Information Sciences '18-08 Vol.43 No.08
https://doi.org/10.7840/kics.2018.43.8.1297

TOSCA 기반의 NFV 서비스 모델 및 구현

최 수 길◆, 최 성 윤*, 최 미 정°

Modeling and Implementation of TOSCA Based NFV Services

Soo-Gil Choi◆, Sung-Yun Choi*, Mi-Jung Choi°

요 약

최근 트래픽 사용량의 증가로 인하여 네트워크 인프라 확장이 필요하게 되었지만 기존의 네트워크 환경에는 이를 유연하게 확장하는데 있어서 어려움이 발생한다. 이를 해결하기 위해 NFV는 모든 유형의 네트워크 자원을 추상화하고, 소프트웨어적으로 자동 관리와 제어를 수행한다. NFV의 네트워크 서비스는 하나 이상의 VNF(Virtual Network Functions)들로 구성되어 특정 네트워크 서비스를 지원할 수 있게 한다. 네트워크 서비스는 일반 정보를 비롯해 해당되는 모든 정보들을 NSD(Network Service Descriptor)에 기술하여 서비스를 제공한다. 본 논문에서는 먼저 NFV에 대한 정보 모델을 클라우드 서비스와 어플리케이션을 위한 정보 모델링 언어인 TOSCA를 사용하고 정의하고, 이를 기반으로 NSD에 필요한 정보들도 모델링한다. 정의한 NSD를 오픈 소스 기반의 클라우드 오케스트레이션 프레임워크인 Cloudify에서 제공하는 Composer를 이용하여 설계 및 구현한다. 구현된 NSD를 Cloudify Manager를 이용하여 오픈스택 위에 배치하고 정상적으로 실행되는지 검증함으로 정의한 정보 모델링의 실현 가능성을 보인다.

Key Words : Cloud, NFV, Information Modeling, Network Management, TOSCA, NSD, Cloudify

ABSTRACT

Recently, the network infrastructure has been required to be expanded due to the increase of the traffic usage, but there are many difficulties in the existing network environment. In order to solve the difficulties, NFV abstracts all the types of network resources and enables software-based automatic management and control. Network Service (NS) of Network Function Virtualization (NFV) is composed of one or more Virtual Network Functions (VNFs) to support a specific network service. The NS describes all the relevant information as well as general information in Network Service Descriptor (NSD). In this paper, we first analyze the OASIS standard document of TOSCA-based NFV information modeling, and model the information necessary for NSD based on TOSCA, an information modeling language for cloud services and applications. We design and implement the defined NSD using Composer provided by Cloudify which is open source based cloud orchestration framework. Finally, the implemented NSD is deployed and validated on the OpenStack using Cloudify Manager.

※ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.NRF-2017R1A2B4010205, 표준 정보 모델 기반 NFV 통합 관리 시스템 설계 및 구현), 2015년도 강원대학교 대학회계 학술연구조성비로 연구하였음(관리번호-520150465)

◆ First Author : Kangwon National University Department of Computer Science, sgchoi89@kangwon.ac.kr 학생회원
° Corresponding Author :Kangwon National University Department of Computer Science, mjchoi@kangwon.ac.kr, 종신회원
* Kangwon National University Department of Computer Science, seongyun@kangwon.ac.kr, 학생회원
논문번호 : 201806-D-149-RE, Received March 22, 2018; Revised July 22, 2018; Accepted July 23, 2018

Ⅰ. 서 론

4차 산업혁명이 도래하면서 네트워크는 ICT 산업뿐만 아니라 미래 국가 경쟁력을 좌우하는 국가 핵심 인프라로의 중요성이 점차 커지고 있다. 하지만 기존의 ICT 산업환경이 변화하면서 네트워크에서는 몇 가지 문제점이 발생하고 있다. 첫째는 점차 증가하고 있는 트래픽 규모이다. 2016년 연간 트래픽 규모는 1.1ZB에서 2019년에는 연간 2.0ZB에 이르러 3년만에 트래픽이 2배 가량 증가할 것으로 전망[1]하고 있다. 둘째는 기존 네트워크 구조의 한계다. 현재 인터넷에서 사용되고 있는 트리형 구조는 서버에서 서버, 이스트 웨스트와 같은 수평적인 트래픽이 증가하는 현 상황에는 적합하지 않다. 셋째는 네트워크 유연성과 확장성에 대한 요구의 증가이다. 네트워크 토폴로지-변경과 네트워크 정책 및 설정 변경이 잦아지면서 네트워크 운용에도 혼란이 가중되고 있다. 따라서 다양한 서비스 대응이 가능하면서 네트워크 확장에도 유연하게 대응할 수 있는 네트워크의 필요성이 제기되고 있다. 마지막으로 기존 유무선 네트워크는 데이터 전송, 제어, 관리 소프트웨어 기능이 모두 하나의 통합 하드웨어 장비 안에 구성된다. 이에 따라 특정 업체의 제품에 대한 의존도 증가로 인프라 확장에 어려움이 발생할 수 있다.

현재 유무선 네트워크는 기능별로 다양한 업체들로부터 제공되는 전용(Dedicated) 하드웨어를 동시에 사용하기 때문에 네트워크 운영 자동화 및 연동 복잡성, 비용 대비 낮은 효율, 네트워크 복잡성 증가 등 네트워크 구조와 보안에 연관된 문제점이 나타나고 있다. 현재, 같은 기능과 성능을 지닌 네트워크 장비가 데이터센터, 캠퍼스 네트워크, 통신 사업자 서비스 네트워크 등 여러 영역에서 특성을 반영하지 못한 채 동일하게 설치 및 운영되고 있다. 이는 비싼 장비의 낭비를 가져오며, 앞으로는 각 영역에 적합한 동적이며 유연하고 특화된 네트워크 구조와 제품이 요구된다. 이와 같이 네트워크 영역에서 요구되는 변화에 효율적이고 유연하게 대응할 수 있는 새로운 네트워킹 방식으로 네트워크 기능 가상화(NFV: Network Functions Virtualization)[2] 분야가 연구되고 개발되기 시작했다.

NFV는 주로 미들박스와 같은 네트워크 서비스 장비 내의 네트워크 기능들을 하드웨어 전용 장비로부터 고성능 범용 서버 등으로 분리시켜 소프트웨어로 유연하게 제어와 관리가 가능하도록 가상화시키는 기술[3,4]을 말한다. NFV의 가장 중요한 요소는 네트워크 서비스 기능을 표준화된 형태로 어떻게 가상화를 지원하느냐에 달려있다. NFV는 사실상 새로운 기술이라고는 볼 수 없으며, 종래의 학계나 산업체에서 계속적으로 연구돼 오던 네트워크 가상화의 현실화된 통신 사업자 버전이라고 할 수 있다. 일반적인 NFV의 구현과 동작 방법은 기존 하드웨어 내에 구현된 네트워크 기능을 컴퓨팅/네트워크 자원의 클라우드 형태로부터 소프트웨어 형태로 가상화해 이를 VNF (Virtualized Network Function)로 만들고, 이를 포워딩 그래프 혹은 서비스 기능 체이닝 형태로 연결하는

표 1. NFV 세계 시장 현황과 전망
Table 1. NFV Golobal Market Status and Outlook

분류	제품 종류		2013	2014	2015	2016	2017	2018	2019
하드웨어	NFVI 서버, 스토리지		81.0	152.8	363.7	650.9	1,030.2	1,449.6	1,806.0
소프트웨어	NFV MANO		2.9	13.2	79.4	154.8	294.1	529.4	767.6
	VNF	vRouter	0.3	2.5	20.9	67.9	128.9	193.4	270.8
		모바일 코어	32.4	74.4	217.8	492.1	941.7	1,584.6	2,088.2
		IMS	57.1	89.3	395.9	795.9	1,142.2	1,488.7	1,719.5
		PCRF & DPI	305.7	553.0	1,029.7	1,567.0	2,213.3	2,693.1	3,153.1
		Security	6.3	37.0	78.6	139.4	223.9	321.5	420.6
		비디오 CDN	-	-	9.9	33.0	107.5	228.9	301.2
		소계	401.9	757.7	1,767.5	3,181.1	5,020.2	6,985.1	8,641.7
	기타		0.1	1.5	14.7	87.9	263.7	474.7	688.4
	소계		404.8	770.9	1,846.8	3,337.9	5,314.3	7,514.5	9,409.3
서비스	NFV 아웃소스 서비스		-	26.8	53.4	109.9	159.2	256.3	386.8
NFV 합계			485.79	950.51	2,264.01	4,098.65	6,503.72	9,220.35	11,602.07

형태다. 실제로 NFV를 구현하는 방법은 다양하며, 이를 표준화하기 위해 유럽 표준화 기구인 ETSI에서는 통신사업자 중심으로 2012년 말, NFV ISG(Industry Specification Group)를 구성해 활발한 표준 개발 작업을 진행하고 있다.

ETSI에서는 2014년까지 NFV 개념과 표준 모델, 그리고 PoC를 통한 가능성 확인을 주로 추진해 왔으며 통신 서비스 사업자들이 NFV 시범 서비스를 시작하고 있고, 네트워크 벤더, IT 벤더, 서비스 사업자들이 협력해 오픈데이라이트나 OPNFV와 같은 오픈소스 기반의 플랫폼 개발에 주력하고 있다. 표 1은 NFV 세계 시장 현황과 전망에 대해서 그래프 형식으로 나타내고 있다. NFV는 사업자들이 부분적으로 기술 적용이 용이하다는 장점이 있고, 급속도로 성장하고 있는 클라우드 시장의 영향으로 빠른 도입 가능성을 보이고 있다. 2017년까지 급성장한 후 안정적인 성장세가 돼, NFV 관련 하드웨어, 소프트웨어, 서비스를 모두 합하여 2019년에는 116억불 규모의 시장이 될 것으로 전망[15]된다.

NFV에서 기본 개념 중 하나인 네트워크 서비스(Network Service, NS)는 하나 또는 그 이상의 NFV로 구성돼 특정 네트워크를 지원할 수 있다. 네트워크 서비스는 일반 정보를 비롯한 모든 정보들이 NSD(Network Service Descriptor)에 기술된다. 네트워크 서비스가 정의하는 VNF 포워딩 그래프의 정보도 NSD에 포함해서 VNF 사이의 관계성을 기술하게 된다.

본 논문에서는 먼저, 클라우드 서비스와 어플리케이션을 위한 정보 모델링 언어인 TOSCA를 사용하여 NFV에 대한 정보 모델링을 서술하고 있는 OASIS 표준 문서를 분석하고, 이를 기반으로 NSD에 필요한 정보들을 모델링한다. 그리고 정의된 정보 모델링을 이용하여 오픈 소스 기반의 클라우드 오케스트레이션 프레임워크인 Cloudify를 이용하여 NSD를 설계 및 구현한다. 마지막으로, 구현된 NSD를 오픈스택 위에 배치하고 정상적으로 실행되는지 검증한다.

본 논문의 구성은 다음과 같다. 2장에서는 네트워크 기능 가상화 기술인 NFV에 대해 서술하고, NFV를 정보 모델링하기 위해 최근 표준화 단체에서 많이 언급되고 있는 정보 모델링 언어들을 살펴본다. 3장에서는 NFV의 기본 개념인 NSD에 대해서 살펴보고, 표준 단체인 OASIS에서 작성된 TOSCA를 이용한 NFV에 대한 정보 모델링 문서를 분석한다. 또한 분석한 문서를 기반으로 NSD에 대한 정보 모델링을 실시한다. 4장에서는 클라우드 오케스트레이션 프레임워크인 Cloudify에서 제공하는 Composer를 이용하여 NSD를 설계 및 구현한다. 구현된 NSD를 Cloudify Manager를 통하여 오픈스택위에 배치하고 정상적으로 실행되는지 검증한다. 마지막으로 5장에서는 결론과 함께 향후 연구를 제시한다.

Ⅱ. 관련연구

본 장에서는 최근 각광받고 있는 기술인 NFV에 대하여 자세히 살펴보고 NFV를 모델링하기 위하여 표준화 단체들이 많은 관심을 갖고 있는 정보 모델링 언어에 대해 살펴본다. 2.1절에서는 네트워크 기능 가상화 기술인 NFV에 대하여 서술한다. 2.2절에서는 정보 모델링 언어들을 Yang, JSON, TOSCA 순서로 서술한다.

2.1 NFV(Network Function Virtualization)

본 절에서는 네트워크 기능 가상화 기술인 NFV에 대하여 서술하고, 기본 개념 구조에 대해서 서술한다. 또한 NFV의 정의와 NFV 기본 개념인 네트워크 서비스와 VNF에 대해 다룬다.

2.1.1 NFV 정의

NFV는 네트워킹에 필요한 모든 유형의 자원들을 추상화하여, 소프트웨어로 자동 관리 및 제어가 가능케하는 기술을 의미한다. 통신 사업자들은 이러한 NFV 기술을 도입하면서 점점 복잡해지는 네트워크 관리 용이성, 관리 비용 절감, 네트워크의 민첩성 등의 장점과 효율성을 얻고자 한다. NFV 환경에서는 여러 장비 개발업체들이 개발한 매우 다양한 네트워킹 기능들이 완벽하게 상호호환이 가능해야 하며, 기존의 비가상 네트워킹 기능들과의 연동도 매우 중요한 고려사항이다. 멀티벤더들이 개발한 다양한 NFV 네트워크 서비스/기능간 상호호환성을 보장하기 위해 ETSI NFV 그룹에서는 네트워크 가상화가 지원되는 프레임워크를 정의하였는데, 가상화를 수행하는 기능 블록들을 정의하고, 이들 간의 인터페이스와 오퍼레이션을 정의하였다.

NFV 기술이 본격적으로 적용되면서 많은 네트워크 장비 또는 소프트웨어 개발업체들은 NFV 표준에 맞추어 개발하거나 기존의 장비를 재개발함으로 새로운 시장 진출을 꾀하고 있다. 예를 들면, OSS/BSS 제공자, NFV 오케스트레이터 제공자(NFVO), NFV 플랫폼 개발자, VNF 개발자 등 여러 영역으로 나뉘어 NFV 시장을 공략하고 있다. 특히 서버, 컴퓨팅, 네트

워크 가상화 분야 대형 업체들간 인수합병이나 협력을 통해 통합 솔루션을 제공하고자 시도하고 있는 현황이다. 국내 이동통신사업자들도 NFV 도입을 위한 기술 개발 및 에코 서비스 발굴 작업을 한창 진행 중에 있다. 또한, ETSI ISG NFV 그룹은 멀티 벤더 환경에서 네트워크 기능 가상화 기술 분야 산업규격을 마련하기 위해 Phase-2 표준화를 매우 활발히 진행시키고 있으며, 실제 NFV 기술 개발 활성화를 위해 오픈 소스 커뮤니티와의 밀접한 협력을 추진하고 있다. 그러나, 통신사업자들이 NFV를 도입하여 상용망에 적용하기까지는 성능 저하 방지, 멀티 벤더 환경 지원, 완벽한 상호호환성 보장, 기존의 가상 및 비가상 자원 간 연동 등 해결해야 할 기술 이슈를 남겨놓고 있는 실정이다.

NFV 기본 개념 구조는 크게 네트워크 서비스 계층, 가상 네트워크 기능 계층 그리고 NFV 인프라 구조 계층으로 구성될 수 있다. 그림 1은 NFV의 개념 구조를 도식화하여 나타내었다.

- 네트워크 서비스 계층 : 제공할 서비스 목적에 따라 사용하는 가상 네트워크 기능(VNF: Virtual Network Functions)들을 이용하여 네트워크 서비스를 추상화하는 계층
- 가상화 네트워크 기능(VNF) 계층 : 벤더들이 개발한 VNF들이 등록되고, 네트워크 서비스를 위해 사용될 수 있는 인스턴스가 존재하는 계층
- NFV 인프라 구조 계층 : 가상 자원, 가상화 지원에 필요한 소프트웨어, 물리적 자원이 존재하는 계층

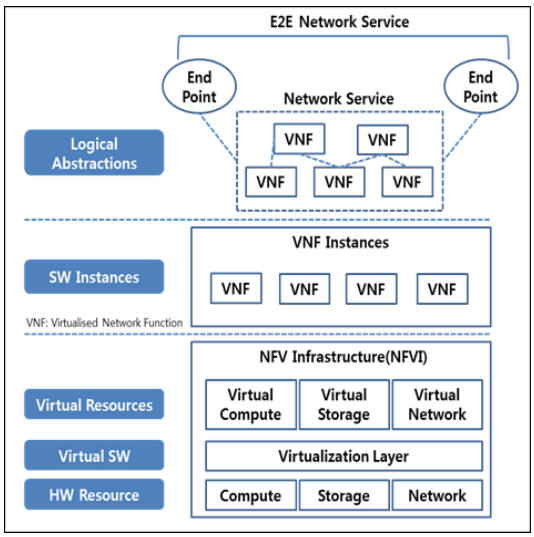

그림 1. NFV 개념 구조
Fig. 1. NFV Concept Structure

2.1.2 네트워크 서비스와 VNF 개념

NFV에서 가장 기본이 되는 개념에는 네트워크 서비스와 VNF(Virtual Network Functions)가 해당된다. 하나의 VNF는 NFV 인프라 구조내에서 배치되어 동작할 수 있는 특정의 네트워크 서비스/기능을 구현한 소프트웨어로 정의할 수 있다. 예를 들면, 가상 스위치, 방화벽, DPI, 트랜스코드, 게이트웨이 등이 VNF에 해당한다. 이러한 VNF가 동작하기 위해서는 NFV 프레임워크 즉, 오케스트레이터(NFVO)와 가상 네트워크 기능 관리자(VNFM)가 제공하는 라이프 사이클 관리 기능, 자원 관리 등을 포함하는 오케스트레이션 기능을 제공 받아야 한다. 하나의 VNF는 하나의 컴포넌트 또는 여러 개의 컴포넌트들로 구성될 수 있다. 하나의 VNF 안에 존재하는 VNFC(Virtual Network Functions Chaining)들은 내부적으로 상호 관계성을 나타내는 그래프 형태로 연결된 구조를 갖고 있다.

그림 2는 네트워크 서비스의 개념 구조를 나타낸다. 네트워크 서비스는 하나 또는 그 이상의 VNF들로 구성되어 특정의 네트워크 서비스(예: 가상 EPC)를 지원할 수 있다. 네트워크 서비스는 일반 정보(예: 개발업체, 버전 등)를 비롯하여 해당되는 VNF 리스트, 해당 PNF(Physical Network Function) 정보, 가상 링크, 성능(예: SLA) 요구사항, 모니터링 파라미터, 자동 스케일링 조건과 액션 등 모든 정보를 NSD(Network Service Description)에 기술한다. 네트워크 서비스가 정의하는 VNFFG(VNF Forwarding Graph) 정보 또한 NSD에 포함되어 VNF들 간의 관계성을 기술하게 된다.

네트워크 서비스에 포함되어 있는 VNF들은 그래프 형태로 상호 연관성에 맞추어 구성될 수 있으며, 이를 VNFFG(VNF Forwarding Graph)라 정의한다. VNFFG는 패킷이 처리되어야 하는 VNF들의 순서를 표기하는 것으로, 반드시 네트워크 서비스에 포함되어

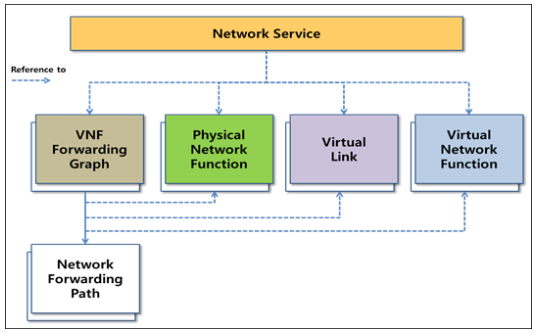

그림 2. 네트워크 서비스 개념 구조
Fig. 2. Conceptual Structure of Network Service

구성되어야 하며, VNFFG 생성, 업데이트, 삭제 및 정보 검색 기능들이 가능하다.

2.2 TOSCA

본 절에서는 최근 많은 표준화 단체들이 정보 모델링에 활용하고 있는 정보 모델링 언어에 대하여 서술한다. 네트워크 환경설정 프로토콜을 위한 데이터 모델링 언어인 Yang과 속성/값 쌍으로 이루어진 개방형 데이터 표준 포맷인 JSON을 간단히 소개하고, 클라우드의 토폴로지에 대한 설명을 명세하기 위한 TOSCA를 서술한다.

Yang은 Netconf 네트워크 환경설정 프로토콜을 위한 데이터 모델링 언어이다. Yang 데이터 모델링 언어는 IETF 소속의 NETMOD(Netconf Data Modeling Language) WG가 개발했으며, 2010년 10월에 RFC 6020[6]으로 정의되었다. Yang은 환경 설정 데이터뿐만 아니라 네트워크 상태 모니터링 데이터의 모델링에도 이용될 수 있다. JSON은 속성/값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다. 비동기 브라우저/서버 통신을 위해 사용되며 XML을 대체하는 주요 데이터 포맷이다. JSON은 프로그래밍 언어와 플랫폼에 독립적이므로, 서로 다른 시스템 간에 객체를 교환하기에 용이하다.[7-9]

TOSCA(Topology and Orchestration Specification for Cloud Applications)[10]는 클라우드의 토폴로지에 대한 설명을 명세하기 위하여 표준화기구인 OASIS(Organization for the Advancement of Structured Information Standards)에서 제정한 표준 언어이다. 클라우드 컴퓨팅은 어플리케이션 서비스들에 대하여 자동적으로 생성, 관리, 상호 운용성에 대한 관계를 포함한 모든 라이프 사이클에 대한 정의가 가능하다면 더욱 활용적일 수 있다. TOSCA는 서비스 토폴로지를 사용하여 서비스 컴포넌트들에 대한 정의와 각 컴포넌트들에 대한 관계 정의를 제공하며, 서비스 생성 및 수정에 대한 관리의 처리 과정을 통합적인 프로세스를 이용하여 나타낼 수 있다.

또한, TOSCA는 IT 서비스를 정의하기 위한 메타 모델을 정의[10]한다. 메타 모델은 서비스들을 관리하는 방법뿐만 아니라 서비스의 구조에 관한 것들도 정의할 수 있다. 그림 3은 TOSCA 핵심 개념[11]인 TOSCA Service Template의 컴포넌트를 나타낸 것이다[12]. TOSCA Service Template 구조에서 Topology Template는 서비스의 구조를 정의하며, Plane은 전체 생명주기 동안에 서비스의 관리와 생성, 삭제 등에 사

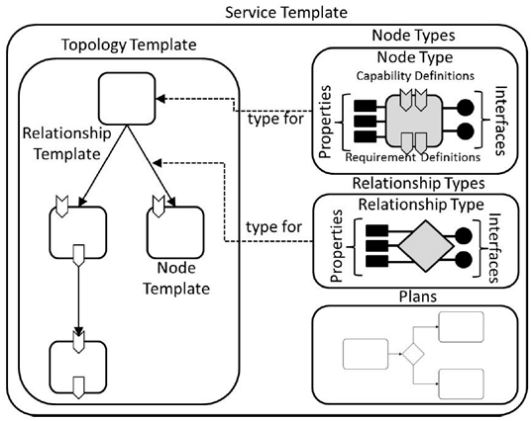

그림 3. TOSCA 핵심 개념
Fig. 3. TOSCA Core Concept

용되는 프로세스 모델을 정의한다. Topology Template는 Node Template와 Relationship Template로 구성되어 있으며 각각은 방향 그래프로서 서비스의 토폴로지 모델을 정의한다. Node Template는 그래프의 노드를 표현하며 각 서비스 컴포넌트의 Node Types에 대하여 명세하고 있다. Node Type은 컴포넌트들의 성질을 정의하며, 컴포넌트를 조작하기 위하여 필요한 오퍼레이션들을 정의하는데 사용된다. Relationship Template는 노드들의 관계를 명세하는데 사용되며 각각의 Relationship Template는 관계의 성질이나 의미를 정의하는데 사용되는 Relationship Type을 나타낸다.

Ⅲ. NSD 정보 모델링

본 장에서는 NFV에서 네트워크 서비스에 대한 모든 정보를 기술하고 있는 NSD에 대한 정보 모델링을 서술한다. 3.1절에서는 표준화 기구인 OASIS에서 정의한 TOSCA를 이용한 NFV 정보 모델링 문서(TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0)[14,15]를 분석한다. 3.2절에서는 앞서 분석한 문서를 기반으로 NSD를 정의한다.

3.1 OASIS TOSCA NFV 표준 문서 분석

NFV에서는 배포 템플릿을 통하여 네트워크 서비스를 실현하는데 필요한 속성과 요구사항을 완벽하게 구현할 수 있다. NFV에서 네트워크 서비스를 위한 배포 템플릿(deployment template)은 NSD로 정의되며, 이는 VNF와 해당 PNF 사이의 관계와 VNF 연결에

필요한 링크를 설명하고 있다. 최상위 네트워크 서비스는 VNF, PNF, VL(Virtual Link), VNFFG와 같은 네 가지 정보 요소들로 이루어져 있다. VNF Descriptor는 배포 및 작동 동작 요구 사항의 측면에서 VNF를 설명하는 배포 템플릿이다. VNFFG Descriptor는 VNF와 PNF 및 이들을 연결하는 VL를 참조하여 네트워크 서비스의 부분 또는 전체 토폴로지에 대한 설명을 하는 배포 템플릿이다. VL Descriptor는 VNF와 PNF, 네트워크 서비스의 종단 간 연결에 필요한 요구사항을 설명하는 배포 템플릿으로, NFVI(NFV Infrastructure)에서 사용할 수 있는 다양한 연결 옵션들에 의해 충족될 수 있다. PNF Descriptor는 물리적인 네트워크 기능에 적용된 가상화 링크의 요구사항에 대한 연결성과 인터페이스, KPI(Key Performance Indicator)에 대한 서술을 하고 있다.

TOSCA 정보 모델에서 가장 상위 레벨은 Service template이다. Service template은 다른 타입을 갖는 Node template을 포함하고 있다. NFV에서 NSD는 가장 상위 레벨이며, NSD는 VNFD, VNFFGD, VLD, PNFD를 포함하고 있다. 그림 4는 TOSCA와 NFV 사이의 정보모델링 측면에서의 매핑 관계를 보여주고 있다. NSD는 Service template을 이용하여 표현하고 있으며, VNFD, VNFFGD, VLD, PNFD는 적절한 Node type과 함께 Node template으로 구성되어 있다. VNFD는 재사용이 가능한 Node type과 함께 또 다른 Service template을 사용함으로써 더 자세히 표현될 수 있다.

네트워크 서비스에 대한 TOSCA 정보 모델에서 NFV에서 표현한 것과 같이 NSD는 네트워크 서비스를 구현하는데 필요한 속성과 요구사항을 설명한다. 그림 5는 NFV MANO 규격 [ETSI GS NFV-MAN

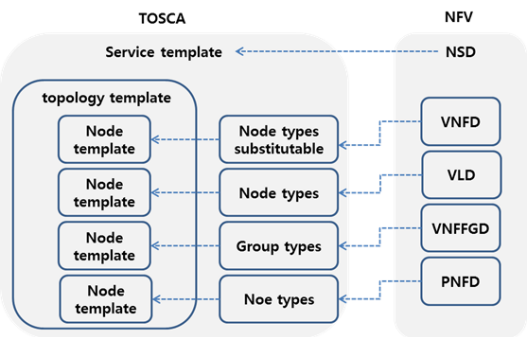

그림 4. TOSCA와 NFV 간의 매핑 단계
Fig. 4. Mapping step between TOSCA and NFV

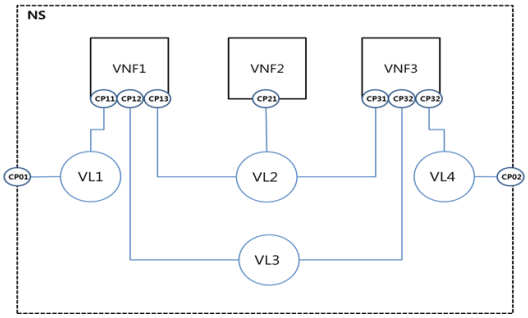

그림 5. NFV에서 네트워크 서비스의 예
Fig. 5. Network service example on NFV

001 v1.1.1]에서 정의한 네트워크 서비스에 대한 예제이다. 이 예제에서 네트워크 서비스는 세 개의 VNF를 포함하고 있다. 각 VNF는 연결점, 즉 CP(Connection Points)에 의해 연결되어 있으며, VNF의 가상적인 또는 물리적인 인터페이스에 대한 표현을 나타내고 있다. 가상 링크, 즉 VL(Virtual Link)은 가상화 링크에 연결된 하나 또는 그 이상의 VNF와 다른 필수적인 파라미터들 간의 연결성에 대한 기본적인 토폴로지를 설명하고 있다.

네트워크 서비스에 포함돼 있는 VNF들은 그래프 형태로 상호 연관성에 맞춰 구성될 수 있으며, 이를 VNF 포워딩 그래프(VNFFG)라고 한다. 그림 6은 그림 5에서 설명한 네트워크 연결 토폴로지 위에 정의된 두 개의 VNF 전달 그래프의 예를 보여주고 있다. VNFFG1에는 두 개의 네트워크 포워딩 경로(VNFFG1:NFP1, VNFFG1:NFP2)가 있지만 VNFFG2에는 단일 네트워크 포워딩 경로(VNFFG2:NFP1)만 있다.

[ETSI GS NFV-MAN 001 v1.1.1] 문서에 따르면 VNFFG는 네트워크 서비스의 일부 또는 전체의 토폴로지를 설명하는 배포 템플릿이다. TOSCA 메타 모델을 사용하는 경우, TOSCA에서 정의된 그룹 개념을 사용하여 VNFFGD(VNF Forwarding Graph

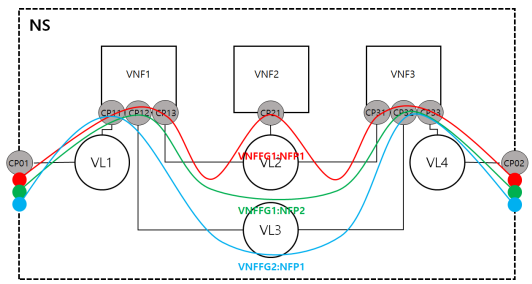

그림 6. VNF 포워딩 그래프의 예
Fig. 6. VNFFG example

Descriptor)를 모델링 할 수 있다. 참조된 VNF, PNF, VL 그리고 CP는 VNFFG 그룹에서 속성(properties) 값으로 정의될 수 있으며, 네트워크 포워딩 경로 요소는 VNFFG 그룹의 목표로 정의되어야 한다.

그림 7은 VNF의 내부 구조에 대한 예를 보여준다. VNF2는 3개의 VDU(Virtual Deployment Units) 이루어져 있으며 각각 내부 가상 링크로 연결되어 있다. VNF의 서브셋인 VDU는 하나의 싱글 가상 머신에 매핑될 수 있다. CP의 일부는 오직 내부 가상화 연결에 사용되는 반면, 다른 CP는 외부 가상화 연결을 위해 사용될 수 있다. 또한 CP는 VDU와 결합되어야 한다. 내부 가상 링크 VL의 주요 기능은 네트워크 서비스 레벨에서 정의된 가상화 링크와 같은 역할을 하지만 VNF내의 VDU 간의 연결을 제공하는 역할에만 사용된다. 첫 번째 VDU는 두 개의 연결점(CP21, CP22)을 갖는다. 첫 번째 연결점 CP21은 외부 가상 링크 연결에 사용되며, 다른 연결점 CP22는 내부 가상 링크 연결에 사용되고 있다. VDU는 연결점과의 관계를 위해 Capability Bindable을 제공한다. 연결점은 두 개의 요구사항 bindable과 virtualLinkable을 갖는다. 외부 가상 링크에 대한 요구사항을 갖는 연결점은 VNF의 virtualLinkable 요구사항을 외부와 연결한다. 외부 연결점은 또한 Forwarder capability를 갖는데 이는 네트워크 포워딩 경로를 형성하는데 사용된다. 그림 7에서 연결점 CP21은 VNF2의 외부 연결점이다.

VNF에 대한 TOSCA 기반의 예제에서는 ID, vendor 및 version을 VNFD 특정 용도에 대해 service_properties로 정의한다. topology_template은 VNF2의 내부 구조를 정의한다. subsititution_mappings에서 Node Type은 Service template에 정의된 재사용이 가능한 타입인 tosca.nodes.nfv.vnf2로 정의되어 있다. virtualLinkable requirement는 CP21을

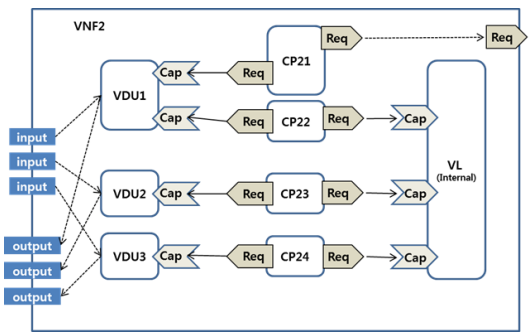

그림 7. 서비스 템플릿에서 VNF에 대한 매핑 예
Fig. 7. VNF mapping example on Service template

사용하여 외부와 연결이 가능하다. VNF의 컴퓨팅 구성요소인 VDU는 컴퓨팅 및 메모리에 대한 속성값을 갖으며 artifact로 서술되고 있는 가상 머신 이미지 파일도 포함하고 있다. VNF2의 종점인 CP21은 VDU1의 requirement에 대한 바인딩을 갖으며 외부 가상 링크를 위한 virtualLinkable requirement를 포함하고 있다. CP22, CP23 및 CP24는 모두 VNF 내부 가상 링크에 사용되는 연결점이다.

VNFD(VNF Descriptor)에서 표현할 수 있는 Node Types에 대하여 서술한다. 첫째 'tosca.nodes.nfv.VNF'는 VNF에 대한 Node Types을 표현하고 있다. VNF Node Types는 [ETSI GS NFV-MAN 001 v1.1.1] 문서에서 정의된 가상 네트워크 기능을 나타낸다. 이는 다른 모든 VNF Node Types에서 파생되는 기본 타입으로 모든 VNF 노드가 모델링 및 관리를 위한 일관성 있는 기능을 가질 수 있게 한다. "tosca.nodes.nfv.VDU" Node Type은 [ETSI GS NFV-MAN 001 v1.1.1]에서 정의된 논리적인 VDU에 대한 전체적인 서술을 나타낸다. VDU는 배치될 인프라의 HW 파라미터(cpu, memory, PCIe, network)와 SW 파라미터(hypervisor, virtual switch)를 가지고 있다. TOSCA 기반으로 VDU에 대한 NFV와 가상머신의 매핑관계를 표현하여 VDU 노드가 생성될 때 가상머신에 대한 이미지 정보, 필요한 자원 정보 등 가상머신에 배치될 때 필요한 정보들을 나타내고 있다. "tosca.nodes.nfv.CP" Node Type은 [ETSI GS NFV-MAN 001 v1.1.1]에서 정의된 논리적인 연결점에 대한 엔티티를 나타내고 있다. 예를 들어 CP 타입에서는 virtual port, virtual NIC address, physical port, physical NIC address 등 네트워크 연결에 필요한 종점을 나타내고 있다. 각각의 연결점은 CP 타입의 서브 타입으로 모델링된다.

3.2 TOSCA 기반의 NSD 정보 모델링

앞서 분석한 문서를 기반으로 NSD에 정보 모델링에 필요한 요소인 가상화된 네트워크 기능인 VNF, VNF가 실제 가상 머신에 배치될 때 필요한 정보들을 담은 VDU, 마지막으로 각 노드들을 연결하는데 필요한 연결점인 CP를 재사용이 가능한 Node Type으로 정의한다. 그림 8은 VNF Node Type에 대한 정보 모델링이다. VNF의 "id", VNF를 제공하는 "vendor", VNF의 "version"을 표현하고 있다.

그림 9는 VDU Node Type에 대한 정보 모델링이다. 가상 머신에 배치 시 필요한 속성들을 정의하였다. image는 가상 머신의 이미지 id, flavor는 가상 머신의 flavor id, Cloudify_agent는 가상 머신의 agent 이름

```
# Virtual Network Function Component
tosca.nodes.etri.nfv.vnf:
    derived_from: tosca.nodes.Root
    properties:
        id:
            type: string
            required: false
        vendor:
            type: string
            required: false
        version:
            type: string
            required: false
```

그림 8. VNF 정보 모델링
Fig. 8. VNF information modeling

```
# Virtualisation Deployment Unit
tosca.nodes.etri.nfv.vdu:
    derived_from: cloudify.nodes.Root
    properties:
        image:
            default: ' '
        flavor:
            default: ' '
        cloudify_agent:
            default: { }
```

그림 9. VDU 정보 모델링
Fig. 9. VDU information modeling

이다. 이는 input값을 입력으로 받아 실제 환경에 배치 시 속성값이 부여된다.

그림 10은 CP Node Type에 대한 정보 모델링이다. CP는 논리적인 연결점에 대한 엔티티를 표현하는 Connection Point의 약어이다. 속성값으로는 연결점에 대한 id, 연결점에 대한 type 두 가지로 정의하였다. type으로 올 수 있는 값은 virtual port, virtual NIC address, physical port, physical NIC address 등이며 필수적으로 명시되어야 한다.

```
# Connection Point
tosca.nodes.etri.nfv.cp:
    derived_from: cloudify.nodes.Root
    properties:
        id:
            type: string
            required: false
        type:
            type: string
            required: false
```

그림 10. CP 정보 모델링
Fig. 10. CP information modeling

```
topology_template:
    inputs:
        flavor ID:
        image ID:
        agent_user:
    vADC:
        type: tosca.nodes.nfv.vnf
        properties:
            id:
            vendor:
            version:
        relationship:
            forwarder1 # the subsititution
            mappings in VNF1 has
            forwarder1: [CP11, forwarder]
            forwarder1 # the subsititution
            mappings in VNF1 has
            forwarder1: [CP12, forwarder]
    vADC_VDU:
        type: tosca.nodes.etri.nfv.vdu
        properties:
            Cloudify_agent:
            flavor:
            image:
        relationship:
            type: tosca.relationships.contained_in
            target: vADC
    vIPS:
        type: tosca.nodes.nfv.vnf
        properties:
            id:
            vendor:
            version:
        relationship:
            forwarder1 # the subsititution
            mappings in VNF1 has
            forwarder1: [CP21, forwarder]
            forwarder1 # the subsititution
            mappings in VNF1 has
            forwarder1: [CP22, forwarder]
    vADC_IPS:
        type: tosca.nodes.etri.nfv.vdu
        properties:
            Cloudify_agent:
            flavor:
            image:
        relationship:
            type: tosca.relationships.contained_in
            target: vIPS
    CP01 #endpoints of NS
        type: tosca.nodes.nfv.CP
        properties:
            id:
            type:
    CP02 #endpoints of NS
        type: tosca.nodes.nfv.CP
        properties:
            id:
            type:
```

그림 11. NSD 정보 모델링
Fig. 11. NSD information modeling

그림 11은 네트워크 서비스에 대한 정보 모델링을 Service template로 정의한 것이다. 네트워크 서비스는 input 값으로 가상 머신의 이미지 ID, 가상 머신의 flavor ID, 가상 머신의 agent 이름을 갖으며, 이는 실제 가상 머신 배치시 적용되는 값이다. topology template을 보면 VNF는 가상 ADC, 가상 IPS 두 개로 구성되고 각 VNF 내에는 실제 가상 머신 배치 시 적용 VDU를 한 개씩으로 정의하였다. 마지막으로 네트워크 종단점 역할을 하는 CP01, CP02를 정의하였다.

Ⅳ. Cloudify를 이용한 정보 모델링 구현 및 검증

본 장에서는 Cloudify를 이용한 정보 모델링 구현 및 검증에 대하여 서술한다. 4.1절에서는 오픈 소스 기반의 클라우드 오케스트레이션 프레임워크인 Cloudify에 대하여 서술하고, Cloudify에서 제공하는 Composer와 Manager에 대하여 서술한다. 4.2절에서는 앞서 정의한 NSD 정보 모델링을 Cloudify의 Composer를 이용하여 구현하고, 구현된 모델을 Manager를 이용해 오픈스택 위에 배치하여 검증한다.

4.1 Cloudify

Cloudify는 오픈소스로 진행되고 있는 클라우드 오케스트레이션 프레임워크[16][17]이다. Cloudify는 OASIS TOSCA 정보 모델링 언어를 사용하며 Python 프로그래밍 언어를 기반으로 하고 있다. 라이센스는 Apache License 버전 2.0이며 Github를 통하여 오픈소스 파일들을 찾아볼 수 있다. Blueprints라는 YAML DSL(Domain Specific Language) 구성 파일로 구축되어 어플리케이션의 구성관리, 서비스 및 서비스의 관계, 라이프 사이클 등을 정의할 수 있다. 이러한 Cloudify를 사용하면 클라우드 컴퓨팅 및 가상화 인프라 구조의 배포 단계를 자동화할 수 있다. Blueprints는 구성관리를 실행하는 API를 통하여 데이터 센터에서 어플리케이션의 상호 작용 방법에 대하여 서술하고 있다[18].

Cloudify는 클라우드 또는 데이터센터 환경에서 배포, 배포된 어플리케이션의 모든 측면의 모니터링, 문제 및 실패 감지, 수동 또는 자동 교정 및 유지관리 작업처리 등 어플리케이션 및 서비스를 모델링하고 전체 라이프 사이클에 관하여 자동화할 수 있다. 어플리케이션 전체(인프라스트럭처, 미들웨어, 어플리케이션 코드, 스크립트, 도구 구성관리, 통계와 로그)는 Blueprint에 서술된다. 사람이 쉽게 읽을 수 있도록 작성되는 YAML 형식으로 작성되는 Blueprint를 사용하면 어플리케이션 구성관리를 세분화하여 쉽고 편리하게 구성이 가능하며, Blueprint에서 어플리케이션의 각 부분의 라이프사이클을 정의할 수 있다.

Cloudify는 컴퓨팅 인스턴스를 실행하고 네트워크, 저장소 및 보안을 구성하여 어플리케이션에 필요한 인프라 구조의 자원을 제공한다. SSH를 통해 원격 또는 로컬 머신에서 스크립트를 실행하거나 구성관리 도구를 호출하여 서버를 구성하고 미들웨어 및 코드를 배포할 수 있다. 어플리케이션 관리 측면에서 Cloudify의 사용자 워크플로우를 사용하면 어플리케이션의 구조를 간단하게 변경 가능하다. 또한 스트림 데이터를 처리하기 위하여 통계 및 로그 수집 기능을 제공하며 데이터 집계 및 시각화 기능을 통해 서로 다른 워크플로우를 실행할 수 있으므로 비즈니스 및 어플리케이션 KPI를 기반으로 스마트한 처리가 가능하다. 이러한 측면을 이용하여 수동적 또는 자동적으로 워크플로우를 트리거할 수 있으며 어플리케이션 분석에도 도움이 될 수 있다. Cloudify는 플러그인 기능도 제공한다. Cloudify 플러그인은 스크립트, CM 툴, 통계 및 로그표현, 그 밖의 다른 툴을 실행할 수 있다. 플러그인은 툴이 설치되고 실행되는 추상화도구이며, Python 기반으로 작성되어 있다. 또한 Cloudify의 사용자들이 직접 플러그인을 작성할 수 있고, 배포하여 Blueprint에 적용할 수 있다.

Blueprint 파일은 설치, 시작, 종료, 오케스트레이션 및 모니터링 등 어플리케이션의 라이프사이클에 대한 실행 계획을 서술하고 있다. Cloudify는 어플리케이션의 배포 계획을 서술하거나 클라우드 환경에서 실행하는데 적합한 입력으로서 Blueprint를 사용한다. 또한 Blueprint는 클라우드 드라이버 구성관리 파일을 배포할 수 있으며, 선택된 클라우드에 대한 가상 머신이나 필요한 이미지 파일을 서술할 수 있다. 소스코드를 이용하여 인프라 구조에 대한 관리를 가능하게 한다. 각 구성요소마다 바이너리의 위치, 설치 및 모니터링 구성 관리에 대하여 서술할 수 있다. 기본적인 인프라 구조에서 코드를 분리하는 추상화 계층을 생성함으로써 Cloudify는 모든 클라우드를 지원할 수 있게 된다. Cloudify는 어플리케이션 서비스를 구성하거나 배치할 수 있는 함수로서 Chef, Puppet, Ansible과 같은 구성관리에 대한 툴을 지원한다.

Cloudify는 Blueprint Composer를 제공하고 있다. Composer는 드래그 앤 드롭 인터페이스를 사용하여 Blueprint YAML 파일을 동적으로 생성할 수 있는 그래픽 편집기이다. Composer는 복잡한 어플리케이션을 위한 토폴로지 모델링이 가능하게 하고 외부 플러

그인 및 스크립트를 통해 관련된 라이프사이클 운영에 대한 구현을 추가할 수 있는 방법을 제공한다. 드래그가 가능한 구성요소에는 컴퓨트(compute), 노드, 데이터베이스, 웹 서버 등 플랫폼 및 네트워크 항목들이 있으며, 사용자가 정의한 노드 타입, 플러그인, 인터페이스도 추가할 수 있다. 마지막으로 Composer를 이용하여 생성한 결과물은 tgz 파일로 다운이 가능하다.

Cloudify Manager는 Docker, Script, Chef, Puppet Plugins 등 다양한 플러그인을 활용하여 어플리케이션 호스트를 관리할 수 있다. 생성된 Blueprint를 다양한 환경에 배포하고 설치할 수 있는 기능을 제공한다. 설치된 어플리케이션에서 Healing, Scaling 및 사용자가 정의한 워크플로우를 실행할 수 있다. 웹 UI를 사용하여 통계 및 로그를 쉽게 파악할 수 있으며 어플리케이션 토폴로지를 이용하여 다양한 작업을 수행할 수 있다. 또한 플러그인 인증 및 인가 메커니즘을 통해 어플리케이션을 관리하기 위한 보안적인 환경을 제공하며, 통계 및 이벤트에 대한 기록도 나타내고 있다. 어플리케이션의 호스트 시스템에서 실행중인 에이전트를 관리할 수 있다. Cloudify는 CLI로 직접 연결하여 사용할 수 있고, Cloudify Manager를 이용하여 더욱 쉽게 연결하여 관리할 수도 있다.

Cloudify Manager는 Nginx, Gunicorn, Flask, Elasticsearch, Logstash, RabbitMQ, Riemann, Celery, InfluxDB, Grafana와 같은 오픈소스를 이용하여 구성되어 있다. 그림 12는 Cloudify Manager에 사용되는 오픈소스 구조도이다. 먼저 Nginx는 고성능의 웹서버이다. Cloudify Manager에서는 REST 서비스 및 웹 UI를 위한 프록시 역할을 하며 Cloudify의 특정 리소스, 에이전트 패키지 및 Blueprint 리소스를 호스팅할 파일 서버로서의 역할을 한다. Gunicorn은 WSGI HTTP 서버이며 Flask는 웹 프레임워크이다. Gunicorn과 Flask는 Cloudify의 REST 서비스를 제공한다. Gunicorn이 서버 역할을 하는 동안 REST 서비스 자체는 Flask를 사용하여 작성된다. Elasticsearch는 JSON 기반의 문서 저장소이다. Elasticsearch는 어플리케이션의 모델을 보유하고 있는 기본 데이터베이스 역할이며 로그 및 이벤트를 저장 및 인덱싱하는 역할을 한다. Logstash는 데이터 처리기이다. 여러 입력을 사용하여 메시지를 처리하며 필터를 적용하거나 다른 출력의 출력값으로 활용할 수 있다. Logstash는 Cloudify가 RabbitMQ에서 로그 및 이벤트 메시지를 가져와 Elasticsearch에서 인덱스를 생성하는데 사용된다. RabbitMQ는 메시지 플랫폼 기반의 큐이다. RabbitMQ는 배치 작업, 로그 및 이벤트, 통계 등의 큐잉 역할을 하는데 사용된다. Riemann은 주로 모니터링을 위해 사용되는 이벤트 스트림 프로세서이며 Cloudify에서 정책 기반의 결정자로 사용된다. Celery는 분산 작업 큐이다. Cloudify의 관리 작업자, 배포 특정 에이전트 및 호스트 에이전트를 Celery기반으로 한다. InfluxDB는 시계열 데이터베이스이며 Grafana는 InfluxDB의 그래픽 대쉬보드이다. RabbitMQ에서 통계를 가져와 InfluxDB에 제출하는데 사용되며, InfluxDB는 Cloudify가 어플리케이션 호스트의 통계

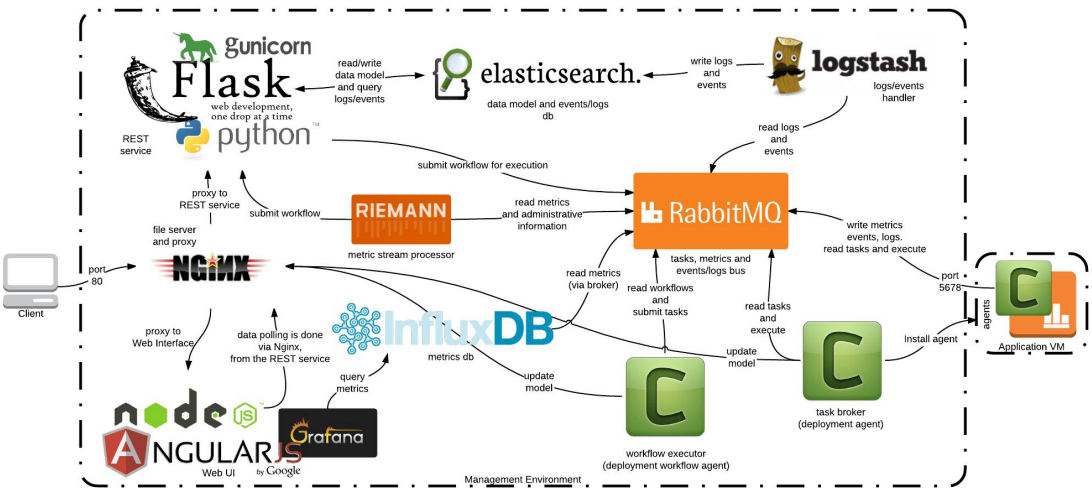

그림 12. Cloudify Manager에 사용되는 오픈소스 구조도
Fig. 12. Open Source structure in Cloudify Manager

에 대한 정보를 저장하는데 사용된다. Grafana는 InfluxDB에 저장된 메트릭을 그래프로 나타내기 위해 Cloudify의 웹 UI에 내장되어 있다.

4.2 구현 및 검증

3.2절에서 정의한 NSD를 표현하기 위하여 Cloudify Composer를 이용하여 NSD를 구현하였다. 앞서 정의한 Node Type VNF, VDU, CP를 import하여 각각의 노드를 생성한다. 생성된 Node Type을 기반으로 하여 정의한 NSD Service template를 드래그 앤 드롭 방식으로 정의한다. 정의된 템플릿을 검증하기 위하여 오른쪽 상단의 Validation 버튼을 적용 후 검증을 한다. 그림 13은 Composer를 이용하여 NSD를 생성하는 Composer 웹 UI이다.

그림 13과 같이 composer를 사용하여 드래그 앤 드랩 방식으로 모델링을 수행하면 NSD 소스코드가 자동으로 생성되는 것을 확인할 수 있다. 생성된 NSD에 대한 소스코드를 3.2절에서 정의한 NSD Service template 소스코드와 비교하여 확인한다. input 값으로 vADC, vIPS 각각의 가상머신 배치정보를 입력받으면 입력 받은 정보들은 각 VNF의 VDU의 속성값 image, flavor, Cloudify_agent에 적용되어 가상머신 배치시 사용된다. 그림 14는 Composer를 이용한 NSD 생성 코드이다. 실제로 3.2절에서 정의한 NSD 모델과 거의 유사하며, Cloudify 내의 TOSCA 기반 정보 모델링의 차이점으로 인한 몇 가지 필드만 다르고, 직접 정의한 TOSCA 기반의 NSD 정보 모델과 서로 교환하여 사용하여도 되는 수준으로 생성이 되었다.

비교 및 검증 후 정의된 NSD 정보 모델링을 오픈스택의 가상머신에 적용하기 위하여 Cloudify Manager를 이용한다. Cloudify Manager를 오픈스택을 이용하여 부트스트랩 후 할당된 유동 IP주소를 이용하여 Manager 웹 UI에 접속한다. 그리고 Blueprint 탭에서 NSD 정보 모델을 업로드한다. Upload된 정보 모델을 실제 Manager에 배치한다. Create Deployment 기능을 클릭하게 되면 input 값을 입력해야 한다. input 값으로는 각 VNF에 필요한 가상머신 이미지 id, 가상머신 flavor id, 가상머신 agent 이름을 입력해야 한다. 그림 15는 Manager에 배치된 NSD 정보 모델에 대한 Topology 뷰를 보여준다. 오른쪽 상단의 Execute Workflow 버튼을 클릭하여 install을 선택하면 Manager를 통해 오픈스택의 가상머신에 배치한다[19].

Openstack[20] 위에 입력한 정보를 바탕으로 가상머신 이미지 생성을 확인한다. 실제 VNF 서비스에 필요한 이미지가 없기 때문에 본 논문에서는 Openstack에 설치되어 있는 Ubuntu 14.04 LTS 이미지 id를 이용하여 설치하였다. flavor 크기는 m1.large(8GB RAM, 80GB DISK, vCPU 4개)로 지정하여 설치하였다. 오

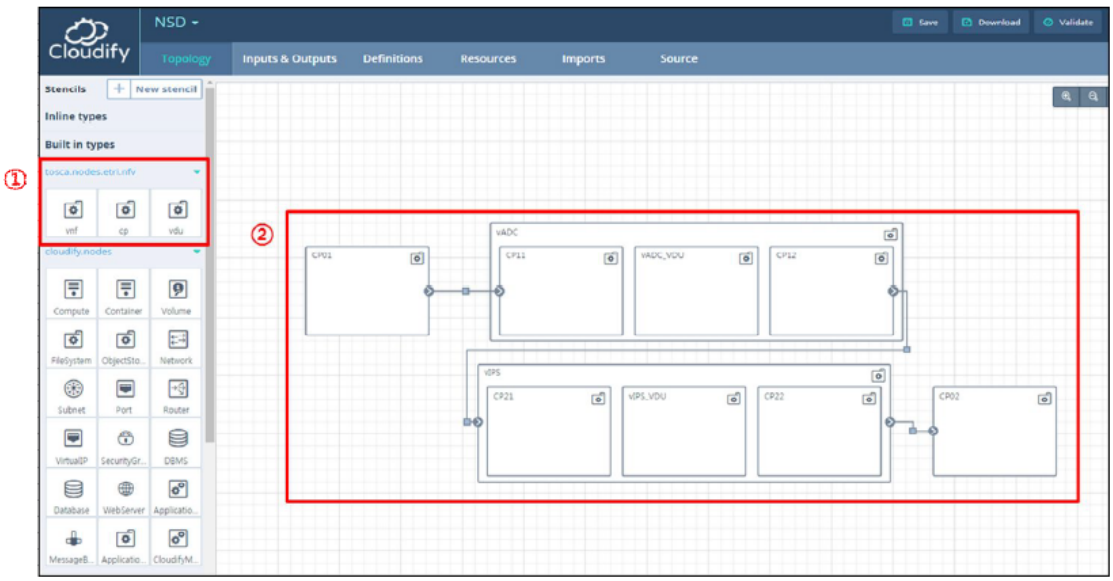

그림 13. Composer를 이용한 NSD 생성
Fig. 13. NSD generation using Composer

```
tosca_definitions_version: cloudify_
imports:
  - 'http://getcloudify.org.s3.amazo
  - >-
    https://raw.githubusercontent.co
  - types/nsd.yaml
inputs:
  vADC_image:
    description: Image to be used wh
  vADC_flavor:
    description: Flavor of the agent
  vADC_agent_user:
    description: User for connecting
  vIPS_image:
    description: Image to be used wh
  vIPS_flavor:
    description: Flavor of the agent
  vIPS_agent_user:
    description: User for connecting
node_templates:
  CP01:
    type: tosca.nodes.etri.nfv.cp
    relationships:
      - type: cloudify.relationships.
        target: CP11
    properties:
      id: CP01
      type: external_cp
  CP02:
    type: tosca.nodes.etri.nfv.cp
    properties:
      type: external_cp
      id: CP02
```

```
vADC:
  type: tosca.nodes.etri.nfv.vnf
  properties:
    id: vADC1
    vendor: NMLAB
    version: 1
CP11:
  type: tosca.nodes.etri.nfv.cp
  relationships:
    - type: cloudify.relationships.contained
      target: vADC
  properties:
    id: CP11
    type: external_cp
vADC_VDU:
  type: tosca.nodes.etri.nfv.vdu
  relationships:
    - type: cloudify.relationships.contained
      target: vADC
  properties:
    cloudify_agent: { get_input: vADC_agent_
    flavor: { get_input: vADC_flavor }
    image: { get_input: vADC_image }
CP12:
  type: tosca.nodes.etri.nfv.cp
  relationships:
    - type: cloudify.relationships.contained
      target: vADC
    - type: cloudify.relationships.connected
      target: vIPS
  properties:
    id: CP12
    type: external_cp
```

```
vIPS:
  type: tosca.nodes.etri.nfv.vnf
  properties:
    id: vIPS
    vendor: NMLAB
    version: 1
  relationships:
    - type: cloudify.relationships.connected_to
      target: CP02
CP21:
  type: tosca.nodes.etri.nfv.cp
  relationships:
    - type: cloudify.relationships.contained_in
      target: vIPS
  properties:
    type: external_cp
    id: CP21
vIPS_VDU:
  type: tosca.nodes.etri.nfv.vdu
  relationships:
    - type: cloudify.relationships.contained_in
      target: vIPS
  properties:
    cloudify_agent: { get_input: vIPS_agent_user }
    flavor: { get_input: vIPS_flavor }
    image: { get_input: vIPS_image }
CP22:
  type: tosca.nodes.etri.nfv.cp
  relationships:
    - type: cloudify.relationships.contained_in
      target: vIPS
  properties:
    type: external_cp
    id: CP22
```

그림 14. Composer를 이용한 NSD 생성 코드 확인
Fig. 14. Generated NSD code using Composer

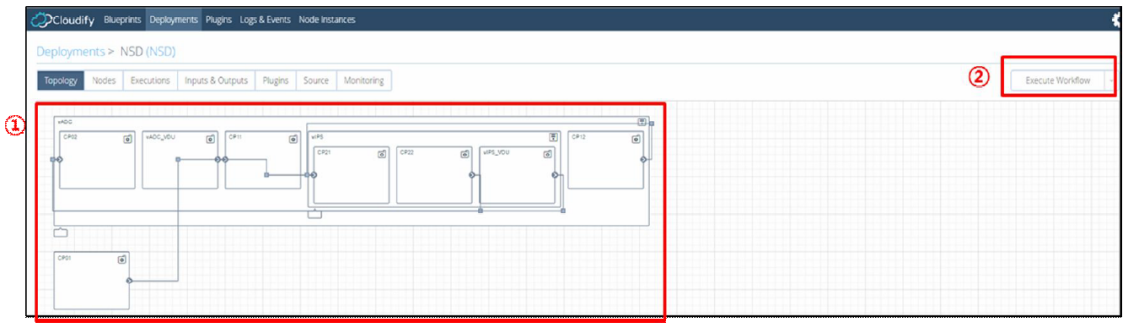

그림 15. Manager에 배치된 NSD 정보 모델
Fig. 15. NSD information model in manager

그림 16. CPU 및 네트워크 성능에 대한 실시간 모니터링
Fig. 16. Real-time monitoring of CPU and network performance

픈스택 웹 기반 대쉬보드 Horizon에서 VM들의 전원 상태가 정상적으로 실행(running)된 것을 확인할 수 있다[21]. 그림 16은 VM 등이 정상적으로 실행되면서 자원 사용량에 대한 모니터링 결과를 보여준다. 그림 16은 하나의 VM에 대한 CPU 사용량을 시스템과 사용자 관점에서 보여주고, 물리 메모리와 디스크 IO(Input/Output)양을 보여준다. 맨 아래 그래프는 네트워크 IO에 대해서 RX(Receive)와 TX(Transfer) 양을 보여주는 것이다.

V. 결 론

본 연구에서는 네트워크 시장의 화두가 되고 있는 기술로서 소프트웨어 기반의 기능을 별도로 분리시켜 운영하여 통신사업들이 기존보다 효율적으로 인프라

를 관리할 수 있는 기술인 NFV에 대하여 살펴보았다. NFV는 기존 네트워크 장비들의 하드웨어와 소프트웨어가 한 곳에 위치하고 있는 의존성 있는 구조를 해결해 줄 수 있다. NFV 기본 개념 중 하나인 네트워크 서비스는 일반 정보를 비롯하여 어플리케이션 서비스, 라이프 사이클 등 모든 정보를 NSD(Network Service Descriptor)에 기술한다. 본 논문에서는 NSD를 클라우드의 토폴로지에 대한 설명을 명세하기 위하여 표준화 기구인 OASIS에서 정의한 TOSCA 표준 언어를 이용하여 정보 모델링을 정의하고, 오픈소스 기반의 클라우드 오케스트레이션 프레임워크인 Cloudify를 이용하여 구현 및 검증하였다.

OASIS에서 정의한 TOSCA를 이용한 NFV 정보 모델링 문서 [TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0]를 분석하여 NFV와 TOSCA 언어 사이의 매핑 관계를 정의한다. NSD는 가상화된 네트워크 기능 VNF(Virtual Network Function), 물리적인 네트워크 기능 PNF(Physical Network Function), 가상 연결에 필요한 링크 VL(Virtual Link), VNF의 논리적인 실행 순서를 서술하는 VNFFG(VNF Forwarding Graph)로 구성된다. 분석한 표준 문서를 기반으로 네트워크 서비스에 필요한 정보 모델링을 정의하였다. NSD 정보 모델링에 필요한 VNF, VDU, CP를 Node Type으로 정의하였으며, 정의된 Node Type을 이용하여 가상 IPS와 가상 ADC를 VNF로 정의하고 VNF가 실제 가상머신에 배치 시, 필요한 정보를 VDU 내에 명시하였다. 또한 VNF 간의 연결을 위한 연결점인 CP를 각각 정의하여 네트워크 서비스의 엔드 포인트를 CP로 표현하였다.

다음으로 정의된 NSD 정보 모델링을 구현 및 검증하기 위하여 오픈소스 프로젝트인 Cloudify를 이용하였다. Cloudify Composer 웹 UI에 Node Type을 추가하고, 추가된 Node Type을 기반으로 앞서 정의한 NSD를 드래그 앤 드롭 형식으로 구현하였다. Composer를 이용하여 정의된 NSD 파일을 Manager를 이용하여 오픈스택 위에 배치하고, 각 VNF에 대한 인스턴스가 자동적으로 생성되어 실행되는 것을 통하여 정의된 NSD 모델이 수행까지 이루어지는 것을 검증하였다.

향후 연구로는 NSD 정보 모델링에 대한 표준화를 통하여 다양한 인프라에서 정보 모델링의 적용 가능성을 검증할 것이다. 실제 가상화된 네트워크 서비스를 구현하여 앞서 정의된 NSD에 적용 및 서비스에 대한 검증을 할 계획이며, NSD의 표현 요소 중 하나인 VNFFG 또한 정보 모델링을 통해 구현 및 검증할 계획이다.

References

[1] M. D. Hong and A. J. Young, “Market trends of SDN/NFV supply and demand,” *Electronics and Telecommun. Trends*, vol. 31, no. 2, pp. 28-40, Apr. 2016.

[2] ETSI ISG NFV, *Network Functions Virtualization - Introductory White Paper*(2012), Retrieved Jul., 18, 2018, from http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[3] ETSI ISG, *ETSI GS NFV 001 v1.1.1 Network Functions Virtualization (NFV) User Cases*(2013), Retrieved Jul., 18, 2018, from http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[4] ETSI ISG, *ETSI GS NFV-MAN 001 v1.1.1 Network Function Virtualization (NFV) Management and Orchestration*(2014), Retrieved Jul., 18, 2018, from http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[5] SDN/NFV Forum, *SDN and NFV based OpenSource, OpenStandard*, acorn, 2016

[6] M. Bjorklund, *YANG - A Data Modeling Langulage for the Network Configuration Protocol (NETCONF)*(2010), Retrieved July., 18, 2018, from https://tools.ietf.org/html/rfc6020

[7] M. Bjorklund, *The YANG 1.1 Data Modeling Language*(2016), Retrieved Jul., 18, 2018 from https://tools.ietf.org/html/rfc7950

[8] *NETCONF Central*, Retrieved Jul., 18, 2018, from http://www.netconfcentral.org/

[9] *YANG Central*, Retrieved Jul., 18, 2018, from http://www.yang-central.org/

[10] OASIS, *Topology and Orchestration Specification for Cloud Applications Version 1.0*(2013), Retrieved Jul., 18, 2018, from http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.pdf

[11] Matt Rutkowski, *TOSCA - An Open Standard for Cloud Application Portability*(2014), Retrieved Jul., 18, 2018, from http://www.cloud-coun

cil.org/CSCC-Webinar-OASIS-TOSCA-Enabling-Application-Portability-in-the-Cloud-10-22-14.pdf

[12] M. J. Choi, "NETCONF/YANG technology for building and managing SDN/NFV," *OSIA Standards & Technol.* vol. 28, no. 2, pp. 62-74, Jun. 2015.

[13] R. Penno, P. Quinn, D. Zhou, and J. Li, *Yang Data Model for Service Function Chainning draft-penno-sfc-yang-13*(2015), Retrieved Jul., 18, 2018, from https://tools.ietf.org/html/draft-penno-sfc-yang-13

[14] OASIS, *TOSCA Simple Profile for Network Functions Virtualization (NFV)* Version 1.0(2017), Retrieved Jul., 18, 2018, from http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html

[15] OASIS, *TOSCA Simple Profile in YAML* Version 1.1(2017), Retrieved Jul., 18, 2018, from http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.1/TOSCA-Simple-Profile-YAML-v1.1.pdf

[16] M. Cinque, D. Cotroneo, F. Frattini, and S. Russo, "To cloudify or not to cloudify: The question for a scientific data center," *IEEE Trans. Cloud Computing*, vol. 4, no. 1, pp. 90-103, Jan. 2016.

[17] Cloudify, *What Is Cloudify?*(2016), Retrieved Jul., 18, 2018, from http://docs.cloudify.co/4.3.0/

[18] L. Y. Al-Harthy and A. H. Al-Badi, "To Cloudify or Not to Cloudify," *World Academy Sci., Eng. and Technol. Int. J. Humanities and Soc. Sci.*, vol. 8, no. 8, 2014.

[19] S. T. Graham and X. Liu, "Critical evaluation on jclouds and cloudify abstract APIs against EC2, azure and HP-cloud," *2014 IEEE 38th COMPSACW*, pp. 510-515, Vasteras, Sweden, 2014.

[20] Openstack, *What is Openstack?*(2017), Retrieved Jul., 18, 2018, from https://docs.openstack.org/

[21] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: Toward an open-source solution for cloud computing," *Int. J. Comput. Appl.*, vol. 55, no. 3, Oct. 2012.

최 수 길 (Soo-Gil Choi)

2015년 2월 : 강원대학교 컴퓨터과학과 졸업
2017년 2월 : 강원대학교 컴퓨터과학과 이학석사
<관심분야> 네트워크 관리, TOSCA/YANG 기반 정보 모델링, NFV 관리

최 성 윤 (Seong-Yun Choi)

2016년 2월 : 강원대학교 컴퓨터과학과 졸업
2018년 2월 : 강원대학교 컴퓨터과학과 이학석사
2018년~현재 : 오로스테크놀로지 연구원
<관심분야> 네트워크 관리, IoT, NFV, 클라우드

최 미 정 (Mi-Jung Choi)

1998년 2월 : 이화여자대학교 공학사
2000년 2월 : 포항공과대학교 공학석사
2004년 2월 : 포항공과대학교 공학박사
2004년~2005년 : 프랑스 INRIA 연구소 박사후 연구원
2005년~2006년 : 캐나다 워터루대학 박사후 연구원
2006년~2008년 : 포항공대 컴퓨터공학과 연구 조교수
2008년~현재 : 강원대학교 컴퓨터학부 컴퓨터과학전공 부교수
<관심분야> 네트워크 관리, 정보보안, SDN/NFV 관리 및 통합