K-ONE 기술 문서 #1

Multi-Region 오픈스택의 터널링 기반 오버레이 가상 네트워킹 구성 자동화를 수행하는 OvN-Manager 설계 및 개발

Document No. K-ONE #1
Version 1.0
Date 2016. 05. 08.
Author(s) 신준식

K-ONE

ONK
OpenNetworkingKorea

■ 문서의 연혁

버전	날짜	작성 내용	비고
-	2016. 03. 27	문서 틀 작성	
0.1	2016. 04. 28	1 절 작성	
0.2	2016. 05. 05	2 절 작성	
1.0	2016. 05. 08	1.0 버전 완성	

본 문서는 2015년도 정부(미래창조과학부)의 재원으로 정보통신 기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-15-2012, 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. B0190-15-2012, Global SDN/NFV OpenSource Software Core Module/Function Development)

기술문서 요약

다수의 해외 사이트로 구성된 OF@TEIN Playground는 각 사이트별로 독립적인 L2 네트워크를 갖는 환경으로써 각 사이트 별로 독립적인 클라우드 Region을 구성하는 Multi-Region 오픈스택 구성으로 구축하였다. 따라서 OF@TEIN Playground의 한 SmartX Box 내에는 오픈스택 Neutron 네트워크와 박스 간 연결을 위해 SDN 제어기로 제어하는 SDN 네트워크가 구성된다. 이 때 SDN 네트워크의 설정을 수동으로 하는 경우 그 복잡성으로 인해 박스의 수 및 SDN 네트워크의 구성요소의 수에 따라 시간적인 비효율성 및 에러 발생률이 급증하게 된다. 따라서 이러한 문제를 해소하기 위하여 SDN 네트워크 의 오버레이 가상 네트워킹 프로비저닝을 자동화하기 위한 OvN-Manager (Overlay vNetworking Manager)를 제안한다.

본 기술문서에서는 OvN-Manager는 Software-Defined Infrastructure의 자원 관리 자동화 구도인 Abstraction, Accessible, Automation의 3A 구도와, 네트워킹 템플릿 기반의 프로비저닝 자동화를 핵심으로 하는 OvN-Manager의 설계 및 구현 과정에 대해 상세히 설명한다. 그리고 구현된 OvN-Manager를 실제 OF@TEIN Playground를 대상으로 적용함으로써 템플릿 기반 자동 프로비저닝 기능의 검증 결과를 제시하는 것을 목적으로 한다.

Contents

K-ONE #1. Multi-Region 오픈스택의 터널링 기반 오버레이 가상 네트워킹 구성 자동화를 수행하는 OvN-Manager 설계 및 개발

그림 목차

K-ONE #1. Multi-Region 오픈스택의 터널링 기반 오버레이 가상 네트워킹 구성 자동화를 수행하는 OvN-Manager 설계 및 개발

1. Multi-Region 오픈스택의 터널링 기반 Inter-connection

1.1. 목적 및 개요

o 본 기술문서에서는 Multi-Region 기반 오픈스택 클라우드 Playground 운영을 위해 발생하는 VxLAN 기반 Overlay Virtual Networking 설정/관리 복잡성을 해소하기 위한 OvN-Mananger (Overlay vNetworking Manager)의 설계/구현/검증 과정을 상세히 기술한다.

o 국제 연구자들을 대상으로 SDN/Cloud 기술이 통합된 실증 환경을 제공하기 위해 운영 중인 OF@TEIN SmartX Playground는 환경적인 특성 상 각 사이트별로 오픈스택 Region을 구성하는 Multi-Region 기반의 오픈스택 클라우드[1]로 구성되어 있다. 하지만 이 경우 각 Region은 각각 독립적인 오픈스택 환경으로써 기본 오픈스택 네트워킹 설정으로는 다른 Region 간 VM 연결성을 제공하지 않으므로, 이를 해결하기 위한 방안으로 대표적인 터널 기반의 오버레이 네트워킹 기술인 VXLAN[2]을 활용하여 각 사이트의 모든 박스들을 메쉬 토폴로지로 상호 연결하였다. 이와 같은 메쉬 토폴로지의 터널 연결로 인해 브리지, 터널, (오픈플로우[3]) 플로우 설정이 복잡해짐에 따라 Playground 구축을 위한 네트워킹 설정, 관리의 비효율성을 갖게 되었다. 따라서 메쉬 토폴로지 형태의 VXLAN 구성을 소프트웨어로 자동화하는 DevOps[4] 자동화 도구의 개발이 요구되었으며, 이와 같은 요구사항을 만족하기 위하여 OvN-Manger를 설계/개발하였다.

o OvN-Manager는 궁극적으로 L2, L3, SDN을 모두 망라하는 네트워킹 구성요소들의 설정, 보안, 에러 감지/복구, 기본 가시성 제공을 DevOps 관점에서 소프트웨어 기반으로 자동화함으로써 소프트웨어 정의 인프라 (Software-Defined Infrastructure) 구도의 네트워킹 측면을 구체화하는 것을 목표로 한다. 본 문서에서는 OvN-Manager의 첫 번째 구현 버전을 기술하며, 이는 OVS (Open vSwitch)[5]를 중심으로 VXLAN을 활용하여 박스 간을 연동하는 터널링 기반 오버레이 가상 네트워킹의 설정, 제한된 문제 탐지/복구 기능을 템플릿 기반으로 자동화하는 것을 구체적인 구현 목표로 한다.

1.2. OF@TEIN SmartX Playground

o 본 문서에서 사용하는 Playground란 놀이(Play)로 대응되는 사용자들이 원하는 다양한 실증실험들을 자유롭게 수행할 수 있는 공간을 상징하며, 이를 위한 구성, 보안, 인증, 자원 관리와 같은 인프라 요소 기능들을 제공하는 테스트베드를 의미한다.

o OF@TEIN SmartX Playground란 연구자들을 대상으로 미래 인터넷 실증 테스트

베드를 제공하려는 전 세계적인 추세에 발맞추기 위하여 2012년 광주과학기술원 NetCS 연구실을 중심으로 국내외 사이트를 대상으로 구축을 시작하여 2016년 4월 기준 총 10개 사이트를 하나로 묶어 SDN/Cloud 실증 환경을 대외적으로 제공하고 있는 Playground를 의미한다[6]. 이를 위하여 각 사이트에 초 융합형 자원 박스인 SmartX Box Type B들을 배포하고 이들을 TEIN 망으로 상호 연동하여 하나의 자원 풀을 만들었으며 현재 오픈스택 클라우드를 구성하여 실증 환경을 제공하고 있다.

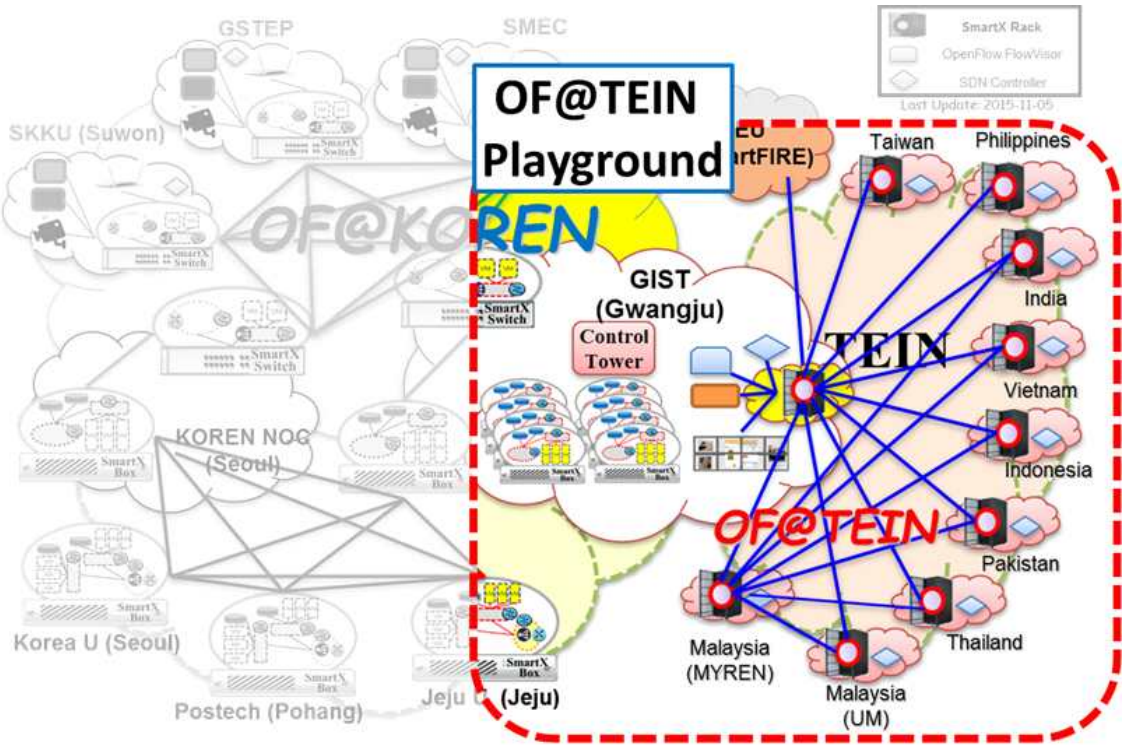


그림 1 OF@TEIN Playground 구성도

o 이 때 SmartX Box는 클라우드 컴퓨팅의 3대 자원 요소인 컴퓨팅/네트워킹/스토리지 지원을 하나의 박스 단위로 확보하고, 이 위에 SDN/NFV/클라우드 기술을 얹음으로써, 기존의 특수 목적용 하드웨어(서버, 네트워킹 장비 등)와 달리 다양한 서비스에 적합한 자원을 유연하고 신속하게 제공할 수 있는 한국형 초 융합 자원 박스로 정의한다. 따라서 SmartX Box를 활용하면 하나의 공통된 박스 모델을 기반으로 SDN/NFV/Cloud를 포함하는 다양한 실증이 가능하다.

o OF@TEIN을 구성하는 각 사이트들이 세계적으로 분포되어 있음에 따라 각각의 SmartX Box들은 공통된 서브넷을 갖는 L2 네트워크로 구성되지 않고 각기 상이한 연구망에 연결되어 있다. 예를 들어 광주과학기술원의 SmartX Box는 한국의 KOREN 연구망에 연결되어 있으며, 말레이시아 사이트에 위치한 SmartX Box는 MYREN 연구망에, 파키스탄 사이트에 위치한 SmartX Box는 PERN 연구망에 직

접적으로 연결되어 있는 상황이다. 이러한 환경적인 제약으로 인하여 각 Box 는 직접적으로 연결되어 있는 네트워크의 IP 주소 대역을 갖는 형태로 OF@TEIN Playground의 인프라가 구성되어 있다.

1.3. Multi-Region 오픈스택 상의 오버레이 가상 네트워킹

o 독립적인 L2 네트워크를 갖는 다수의 사이트들을 대상으로 하나의 공통된 관리 인터페이스를 통해 오픈스택 클라우드를 운영하기 위해서는 하나의 Region으로 모든 자원을 묶어 관리하는 기본적인 오픈스택의 설치/운영 법으로는 한계점이 있다. 기본적인 오픈스택의 구성은 자원 풀을 구성하는 다수의 박스들이 한 곳에 집중되어 위치하며 공통된 L2 네트워크에 연동되어 있는 환경을 가정한다. 따라서 L3 환경 상에 기본적인 형태의 오픈스택을 구성하게 되면 Floating IP 할당 문제 등의 이슈가 발생하게 된다.

o 오픈스택에서는 Zoning을 위해 크게 Region, Availability Zone, Host Aggregate 라는 세 가지 구분을 정의하였다. 우선 Region은 API endpoints, Networks/ Compute 자원 등을 모두 갖고 있는 독립적인 오픈스택 클라우드가 구성되어 있는 구역을 의미하는 단위이다. 이 때 하나의 인터페이스를 통해 전체 관리를 가능케 하기 위하여 다른 Region 간에는 인증, 대시보드 관련 프로젝트인 OpenStack Keystone, OpenStack Horizon 프로젝트를 공유한다. Availability Zone는 하나의 Region 안에 존재하는 Compute 노드들을 물리적인 위치에 따라 논리적으로 그룹핑한 것을 의미하며, Host Aggregate는 AZ 보다 좀 더 유연하게 Compute 노드들을 기능/특성에 따라 Logical하게 Grouping 하는 단위를 의미하게 된다. 아래 그림은 오픈스택의 Zoning 개념을 보다 시각화한 그림이다[7].

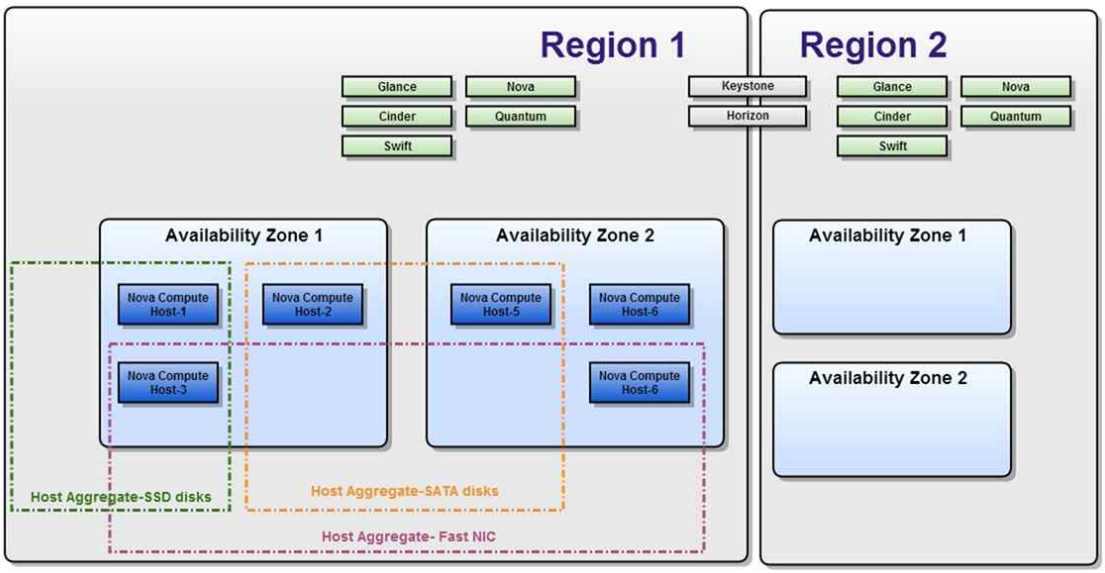


그림 2 오픈스택 Zone 구분 예시

o 상기한 오픈스택 Zone의 특성을 고려해 본 결과 OF@TEIN Playground Type B* 인프라를 대상으로 오픈스택을 운영하기 위한 방안으로써 각 사이트별로 독립적인 오픈스택 클라우드를 구성하되, 하나의 인터페이스를 통해 전 사이트의 활용을 가능케 하는 Multi-Region 방식의 오픈스택을 구축하기로 결정하였다. 하지만 Multi-Region 기반의 오픈스택 클라우드로 구성할 경우 각 Region은 독립적인 클라우드 이므로 서로 다른 Region에 위치한 VM 간에는 통신이 불가능하다는 제약점을 갖고 있다.

o 상기한 Multi-Region 오픈스택의 한계점을 해결하기 위하여 Region 내 Box들은 오픈스택 Neutron의 VLAN 기본 설정에 따라 구성하고, 서로 다른 Region의 Box 간에는 VXLAN으로 상호 연결한다. 이를 위하여 오픈스택 Neutron이 관리하는 OVS 브릿지(br-int, br-vlan, br-ex) 외에 네트워킹 제어를 위한 OVS 브릿지를 추가로 설정하고 이 브릿지 간을 VXLAN 터널로 연결한다. 그리고 추가한 브릿지는 OpenDaylight[8]와 같은 SDN 컨트롤러로 제어하도록 설정함으로써 각 박스별로 오픈스택 Neutron에 의해 제어되는 오픈스택 네트워크 부분과 SDN 컨트롤러로 제어되는 SDN 네트워크가 설정된다. 아래 그림은 설명한 네트워크 구성을 그림으로 도식화 한 것이다. OF@TEIN Playground Type B* 를 구성하는 박스들은 아래 그림과 같이 네트워크가 구성되어 있다.

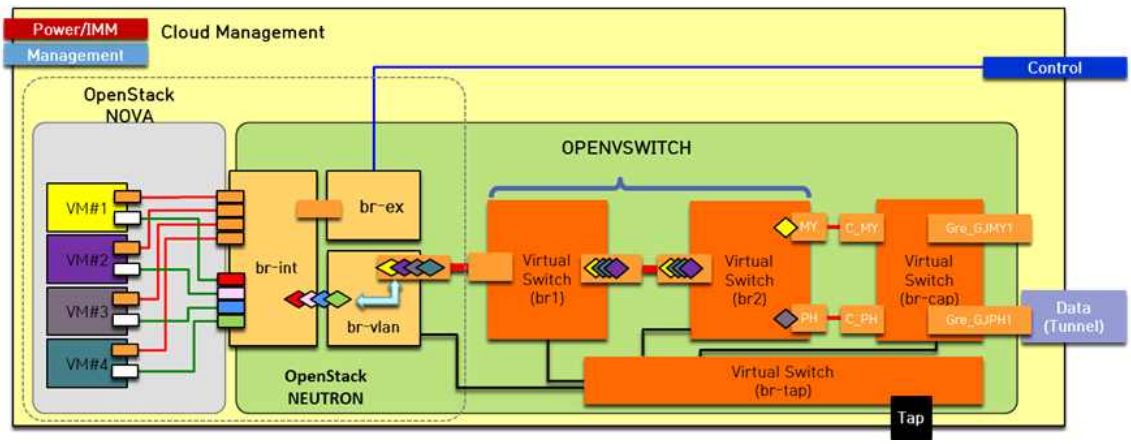


그림 3 OF@TEIN Playground Type B* 박스 내부 네트워크 구성도

o 그림 3과 같이 네트워크를 구성하고, 오픈스택에서 연결하고자 하는 VM들이 연결된 VLAN ID, IP Subnet을 일치시키게 되면 다른 Region에 있더라도 상호 통신이 가능하다.

1.4. 자동화 도구의 필요성

o OF@TEIN Playground 상에서 오픈스택 기반의 SDN/클라우드 환경에서는 오픈스택 Neutron이 관리하는 기본 브릿지 외에 추가 네트워킹 요소들이 존재하므로 정상적인 Inter-connection을 위해서는 OVS 브릿지, VXLAN 터널, OpenFlow 플로우를 일일이 설정해 주어야 한다. 각각의 박스를 대상으로는 이러한 설정이 큰 문제가 되지 않지만, Playground를 구성하는 박스가 다수가 되었을 경우 복잡성이 크게 증가한다. 특히 VXLAN 터널의 경우 박스의 Pair 마다 터널이 생성되는 메쉬 토폴로지 형태로 구성되므로 박스 수의 증가에 따라 터너르이 수가 급격히 증가한다($O(n^2)$)

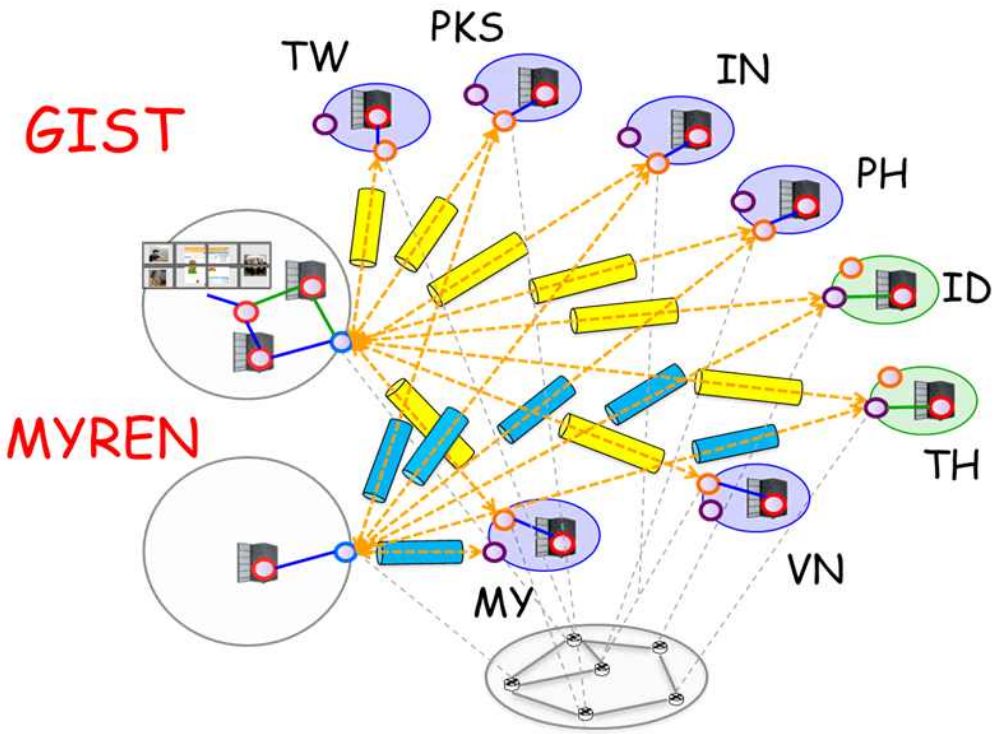


그림 4 OF@TEIN Playground 네트워킹 구성

o 각 박스마다 공통적인 기본 구성은 동일하나 세부 파라메터가 상이하기 때문에 이와 같은 복잡한 설정을 수동으로 하는 것은 운영 측면의 시간적인 비효율성이 상당하며, 사람이 설정하는 것이므로 휴먼 에러의 가능성을 내포하고 있다. 따라서 브릿지, 터널, 플로우의 설정을 자동화하는 오버레이 가상 네트워킹 프로비저닝 도구가 필수적이다.

2. OvN-Manager 설계 및 구현

2.1. 기존 개발된 자동 Inter-connection 도구

o OF@TEIN Playground의 오버레이 가상 네트워킹 프로비저닝의 자동화를 위한 노력으로 2014년도 OVSDB 프로토콜을 활용하여 브릿지, 터널, 플로우의 설정을 자동화하는 스크립트 기반 Inter-connect 프로비저닝 도구를 개발하였다.[9] 하지만 기존 Inter-connect 프로비저닝 도구의 경우 OF@TEIN Playground의 네트워킹 요소들의 설정 자동화에 초점이 맞추어져 있어 고정된 네트워킹 구성으로만 프로비저닝이 가능하였다. 또한 특정 박스만을 대상으로 하거나 특정 네트워킹 요소만을 프로비저닝 하는 기능이 없었기에 보다 세부적이고 유연한 형태의 Inter-connection 프로비저닝이 불가능하다는 한계점을 갖고 있었다. 프로비저닝 방식 및 제한적인 자동화로 인하여 본 도구를 활용하기 위해서는 OF@TEIN Playground에 대한 이해는 물론이고, OVSDB, 그리고 프로비저닝 도구에 대한 이해가 반드시 선행되어야 했다.

2.2. 소프트웨어 설계

o OF@TEIN Playground의 효율적인 운영을 위해서 상기한 기존 Inter-connect 프로비저닝 도구의 Inflexibility, Coarse-grained, Inusability 적인 한계를 해소하는 OvN-Manager (Overlay vNetworking Manager)의 개발이 요구되었다. 기존 한계를 개선하기 위해 OvN-Manger는 아래와 같은 요구사항을 충족하도록 설계했다.

- 오버레이 가상 네트워킹의 모든 프로비저닝 과정은 전 자동화 되어야 함
- 누구나 쉽게 오버레이 가상 네트워킹의 구성을 디자인하고 이를 적용할 수 있어야 함
- 브릿지, 포트, 터널, 플로우와 같은 네트워킹 세부 요소 각각을 설정 가능해야 함. (Fine-grained Configuration)
- 프로비저닝 이후에도 모든 Inter-connect 상태를 주기적으로 모니터링 하여 문제점을 자동으로 판별할 수 있어야 함. 자동으로 발견한 문제점을 운영자에게 알리고 이를 제한적인 범위 내에서 복구해야 함.

o 상기한 요구사항을 반영하는 OvN-Manger의 전체적인 컨셉 및 역할은 아래 그림과 같다. 운영자가 정의한 네트워킹 템플릿의 내용에 따라, OVS, OpenDaylight SDN 컨트롤러 등의 오픈소스 소프트웨어들을 활용하여, 다른 사이트 내 박스들을 대상으로 VXLAN 터널링 중심의 오버레이 가상 네트워킹을 자동화 하는 것이 OvN-Manager의 주요 역할이다. 따라서 본 역할을 이행하는 OvN-Manager를 구현할 수 있도록 아래와 같이 설계를 진행하였다.

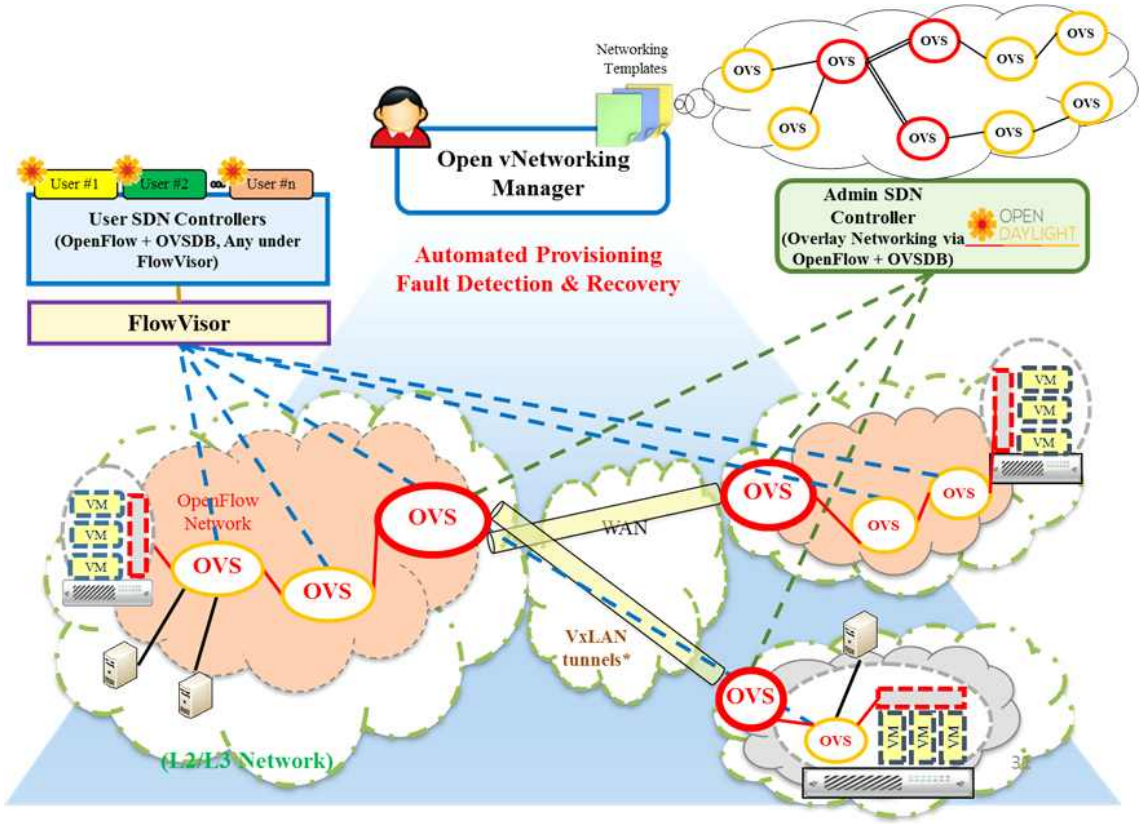


그림 5 OvN-Manager 컨셉 및 역할

o OvN-Manager의 설계에 있어 SDI (Software-Defined Infrastructure)의 인프라 관리 자동화 개념을 적용하고자 하였다. SDI에서 인프라 관리 자동화를 위해 서버, 스위치 등과 같은 하드웨어를 포함하는 모든 자원 요소들은 API를 통해 DevOps 자동화 소프트웨어를 활용하여 구축/운영을 전 자동화 할 수 있어야 한다. 즉, 모든 하드웨어/소프트웨어 자원을 논리적으로 추상화해야 하고(Abstraction), 이러한 자원들을 API를 통해 접근 가능해야 하며(Accessible), 소프트웨어로 자원들의 관리를 자동화해야 한다(Automation).

OvN-Manager 또한 상기한 Abstraction, Accessible, Automation의 3A 구도에 따라 필요한 요소들을 정리하고, 이를 기반으로 소프트웨어를 단순·명료하게 설계하고자 하였다. 우선 오버레이 가상 네트워킹을 구성하는 자원으로는 박스, VXLAN 터널, SDN 제어기, 플로우, OVS 브릿지가 있으며, 이는 논리적으로 추상화 되어야 한다. 그리고 추상화된 각 자원들을 관리하기 위한 인터페이스로 박스 접속을 위한 SSH, OVS 브릿지 설정을 위한 OVSDB, SDN 제어기에서 플로우 제어를 위한 OpenFlow, 그리고 SDN 제어기 관리를 위한 REST API가 필요하다. 마지막으로 오버레이 가상 네트워킹의 프로비저닝 단계를 'Bridge - Tunnel - Flow'의 세 Phase로 정의하고 각 Phase마다 '정보 수집 - 템플릿과 비교 - 설정' 의 과정을

반복함으로서 전체 프로비저닝을 자동화한다. 이와 같이 3A 구도에 따라 OvN-Manager의 설정 대상, 방법 및 프로비저닝 절차를 상위관점에서 설계하였다. (그림 5)

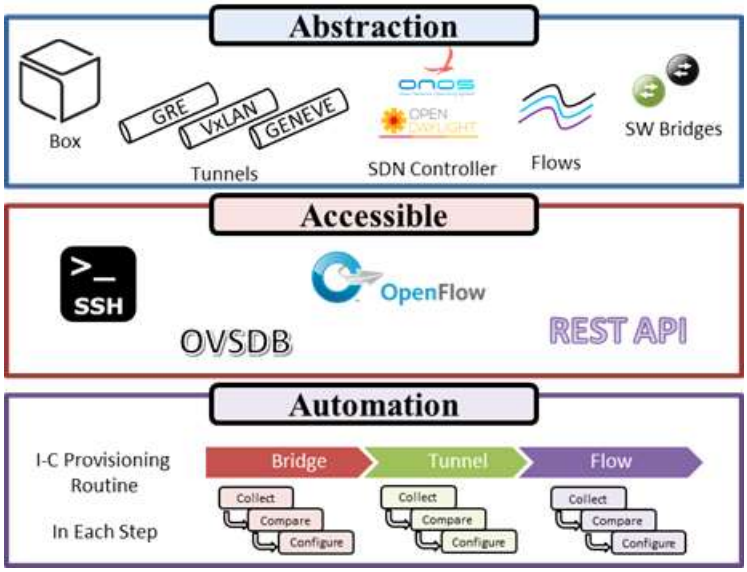


그림 6 OvN-Manager의 3A 기반 설계

o OvN-Manager의 주요 개발 원리 중 하나는 DevOps 개발 방법론이다. 개발된 소프트웨어를 큰 주기로 구현하고 마는 것이 아니라 개발하는 과정에서 실제 Playground 환경에 상시 적용하고, 이를 운영하는 과정에서 발생하는 크고 작은 이슈들을 끊임없이 개선함으로써 도구를 보다 실용적이고 빠르게 개선할 수 있는 DevOps 방법론을 기본으로 삼고 있다.

o OvN-Manger의 가장 중요한 설계 요소는 네트워킹 템플릿 기반의 자동화이다. 앞서 기재한 세부 요소 설정, 유연한 프로비저닝, 쉽게 사용가능해야 한다는 Inter-connection 프로비저닝 요구사항을 충족하기 위해서는 인프라 운영자가 프로비저닝 하고자 하는 네트워킹 구성을 세부적으로 정의할 수 있는 수단과 OvN-Manager가 프로비저닝 하는 과정에서 참고할 대상이 필요하다. 이를 위하여 OvN-Manger는 네트워킹 템플릿이라는 개념을 도입하였다.

- 인프라 운영자는 각 박스 별로 네트워킹을 구성하는 브릿지, 터널의 원하는 형태를 정해진 포맷에 따라 네트워킹 템플릿에 정의함으로써 자원을 논리적으로 추상화한다. 정의된 네트워킹 템플릿들을 OvN-Manager의 입력 파일로 전달하게 되면 특정 박스의 네트워킹을 설정할 때에 네트워킹 템플릿에 정의된 네트워킹 구성을 해석하고 앞서 설명한 Accessible한 인터페이스로 접근하여 “브릿

지 - 터널 - 플로우" 순서에 따라 하나씩 설정을 자동으로 수행함으로써 결과적으로 프로비저닝 전 과정을 자동화 한다.

- 에러 검출 및 복구 기능을 위하여 30초 마다 OvN-Manager가 자동으로 프로비저닝 과정을 반복하며, 박스로부터 수집된 현재 구성 정보를 템플릿 정보와 비교하여 템플릿 정보와 상이한 설정을 에러로 판별하게 된다. 그리고 에러가 발견되었을 경우 일차적으로 에러를 수정하는 메커니즘이 진행되며, 에러의 내역은 메일을 통해 운영자에게 자동으로 발송한다.

- 네트워킹 템플릿 개념의 도입으로 인해 인프라 운영자는 각 요소의 세부 설정을 직접 하거나 알 필요가 없으며, 네트워킹 템플릿 작성법에 따라 작성만 하면 쉽게 오버레이 가상 네트워킹 프로비저닝이 가능하며, 네트워킹 템플릿을 통해 개별 브릿지, 포트, 터널, 플로우와 같은 세부 수준의 설정이 가능하다. 게다가 템플릿을 기반으로 에러를 검출함으로써 제한적이나마 자동으로 네트워킹의 문제를 검사/복구하는 기능을 제공할 수 있다.

2.3. 소프트웨어 구현

o 본 기술문서에서 기술하는 OvN-Manager 도구는 Ubuntu 12.04 LTS 버전의 리눅스 운영체제 상에서 BASH 쉘 스크립트 언어로 작성되었다. 각 박스의 브릿지에 OpenFlow 플로우를 내려주기 위해 OvN-Manager는 오픈소스 SDN 컨트롤러인 OpenDaylight를 사용한다. 그 중 OpenDaylight Hydrogen 버전을 기준으로 구현을 진행하였다. 따라서 OvN-Manager의 Flow 설정 기능을 사용하기 위해서는 OpenDaylight 컨트롤러를 설치해야 한다.

o OvN-Manager는 기본적으로 브릿지, 터널, 플로우의 세 Phase를 순차적으로 수행하여 프로비저닝 한다. 그리고 각 Phase 내에서는 박스 별로 "Collect - Compare - Configure"의 동일한 세 단계를 반복한다.

o 브릿지 Phase의 초입에서는 특정 박스를 대상으로 OVS 브릿지 상태 정보를 수집하기 위하여 OVSDB(Open vSwitch DataBase) 프로토콜을 활용하여 "ovs-vsctl show" 커맨드 실행시 출력되는 브릿지 정보를 가져온다(Collect). 가져온 정보에서 기재된 브릿지 순서대로 설정 Loop를 반복하는데 설정 Loop 내에서는 해당 브릿지의 OVS 정보(브릿지, 포트, SDN 컨트롤러, DPID 등)를 파싱하여 변수로 저장한다. 그리고 박스의 템플릿 파일로부터 해당 브릿지 정보를 추출한 뒤 정보를 파싱하여 변수에 저장한다. 그리고 수집한 정보와 현 OVS 상태 정보를 서로 비교하여 템플릿의 설정 값과 상이한 부분이 발견되었을 경우 OVSDB 프로토콜을 이용하여 곧바로 재설정 후 리포트 파일에 해당 내역을 남겨 놓는다

(Configure). 본 Loop는 모든 브릿지의 작업이 끝날 때까지 반복되기 때문에 브릿지 Phase를 완료하면 모든 브릿지에 대한 프로비저닝이 완료된다.

o 터널은 브릿지와 플로우 설정과는 달리 하나의 박스를 대상으로 에러를 판단할 수 있는 요소가 아니라 터널의 양 종단에 위치한 박스를 동시에 확인해야 한다. 따라서 터널 Phase의 가장 초입에서는 모든 박스의 템플릿들을 모두 조회한 후 이를 이용하여 터널 리스트를 생성한다. 그리고 나서 각 박스별이 아닌 터널 별로 Loop를 반복한다. 우선 터널의 양 종단에 위치한 박스로부터 OVSDB 프로토콜을 통해 OVS 정보를 수집하고 터널 관련 정보만을 파싱하여 변수로 저장한다(Collect). 그리고 템플릿으로부터 생성한 터널 리스트와 수집한 터널 정보를 상호 비교하여 템플릿 대비 잘못된 설정을 확인한다(Compare). 확인된 에러는 OVSDB를 이용하여 복구한 후 리포트에 내역을 남긴다 (Configure). 본 Loop는 모든 터널에 대하여 순차적으로 반복되므로 터널 Phase가 완료되면 모든 터널에 대한 설정이 완료된다.

o 플로우 Phase에서는 각 박스별로 Loop 문을 실행함으로써 전 박스를 순차적으로 설정한다. 이 때 각 박스별로 User 및 Operator를 위한 OpenDaylight SDN 컨트롤러가 각각(또는 하나로) 존재하므로 템플릿 파일을 통해 해당 사이트의 컨트롤러 정보를 추출한다. 그리고 브릿지별 플로우 설정을 네트워킹 템플릿에 기재된 브릿지 순으로 수행한다. 브릿지 Loop에서는 브릿지의 플로우 정보를 OpenDaylight REST API로 수집한 후 이를 파싱한다(Collect). 그리고 네트워킹 템플릿의 브릿지 정보로부터 플로우 정보를 자동으로 생성하여 OpenDaylight로부터 수집한 플로우 정보와 비교한다(Compare) 그리고 상이한 플로우는 OpenDaylight의 REST API를 이용하여 재설정한다(Configure).
플로우 Phase에서는 브릿지를 위한 플로우 설정 후 터널을 위한 플로우 설정 과정이 별도로 존재한다. 브릿지 설정이 완료된 후 이전 터널 Phase에서 정리한 터널 리스트에 따라 터널 별로 Loop문을 반복하게 된다. 각 터널의 양 종단을 위한 플로우 정보를 OpenDaylight REST API로 수집하고(Collect), 이를 브릿지 정보로부터 자동 생성한 플로우 정보와 비교한 후(Compare), 상이한 부분은 재설정하고 에러 리포트에 기재한다(Configure).
플로우 Phase의 경우 다른 Phase와 가장 다른 점은 템플릿 자체에 설정하려는 플로우를 일일이 정의하지 않는다는 점이다. 대신 플로우가 적용되는 브릿지, 포트 정보에 따라 자동으로 플로우 정보(플로우 명, 플로우 ID 등)가 결정된다.
결과적으로 플로우 Phase를 통해 브릿지간 네트워킹을 위한 플로우, VXLAN 터널을 위한 플로우를 OpenDaylight를 이용해 자동으로 설정할 수 있다.

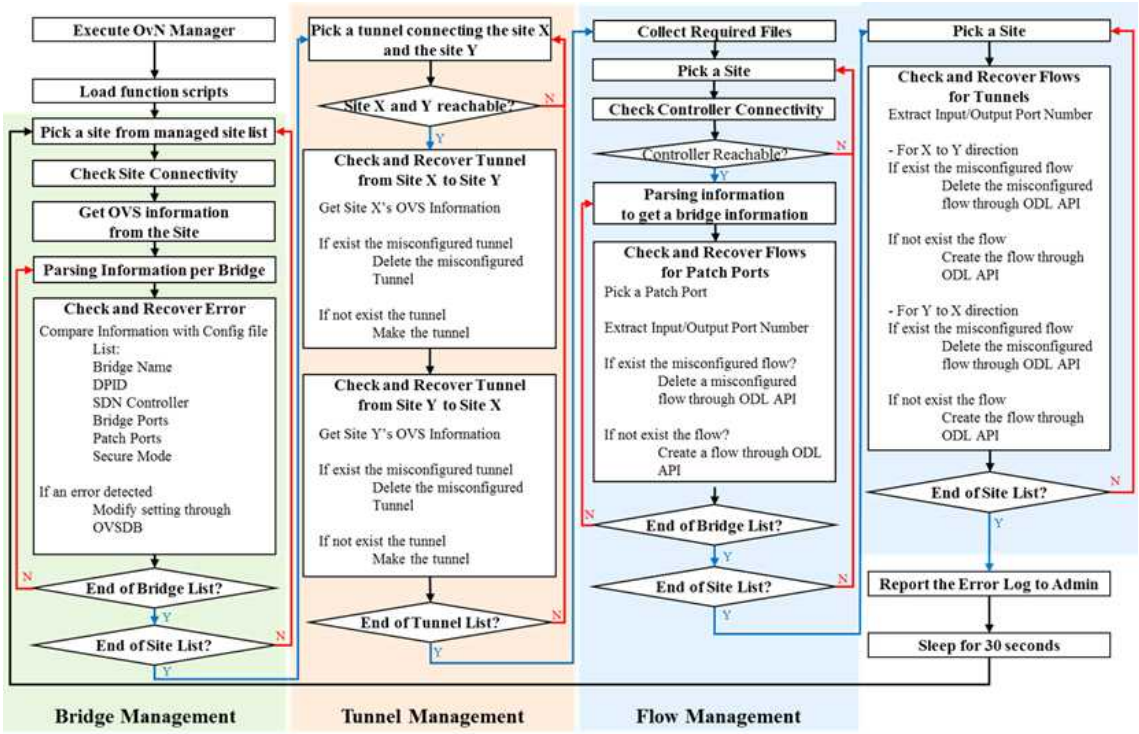
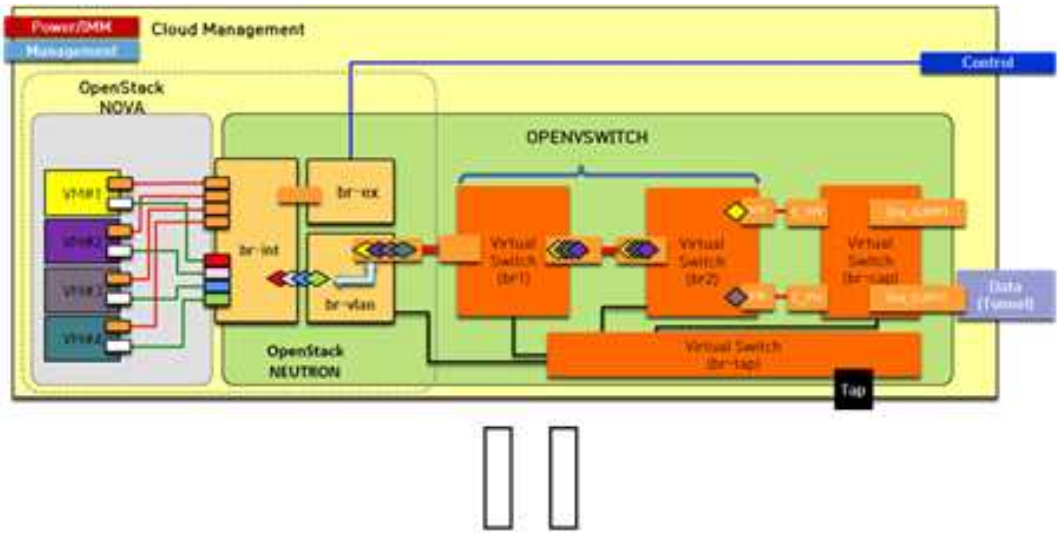


그림 7 OvN-Manager 자동 프로비저닝 절차

o 세 Phase가 완료된 이후 기재된 리포트를 OF@TEIN 운영자에게 메일로 자동으로 전달하게 된다. 그리고 나서 30초 마다 해당 Phase를 다시 반복함으로써 운영자가 일일이 프로그램을 실행하지 않아도 자동으로 에러 복구/검출, 리포트, 프로비저닝을 정기적으로 수행한다. 그림 7은 앞서 설명한 OvN-Manager의 전체 실행 과정을 플로우 차트 형태로 정리한 것이다.

o OvN-Manager에서 입력파일로 전달되는 네트워킹 템플릿은 자체적으로 정의한 포맷에 따라 작성해야 한다. 포맷 자체는 OVS의 "ovs-vsctl show" 커맨드 입력 시 출력되는 정보 포맷을 많은 부분 차용하였다. 대신 설정 정보 구역을 구분하기 위하여 <이름> </이름> 으로 구역을 기본적으로 구분하도록 구현하였다.

o 그림 8은 OF@TEIN의 기본 네트워킹 구성과 이를 템플릿으로 정의하였을 때의 예시 템플릿을 보여주는 그림이다. 그림 8을 분석해 보면 네트워킹 구성도의 각 자원 요소들 (브릿지, 포트, 터널, SDN 제어기 등)을 정해진 포맷에 따라 그대로 템플릿에 묘사해 놓았음을 볼 수 있다. 이를 통해 네트워킹 템플릿이 실제 자원들을 추상화하여 원하는 오버레이 가상 네트워킹 구성 결과물을 묘사한 파일임을 쉽게 확인할 수 있다.




```
<BRIDGE>
        <NAME>xenbr0</NAME>
        <PORT>eth5</PORT>
</BRIDGE>

<BRIDGE>
        <NAME>xenbr1</NAME>
        <PATCH_PORT>
                <FROM>xbr1_br1</FROM>
                <TO>br1_xbr1</TO>
        </PATCH_PORT>
</BRIDGE>

<BRIDGE>
        <NAME>br1</NAME>
        <DPID>00:00:11:11:11:11:11:07</DPID>
        <PORT>eth0</PORT>
        <PORT>eth1</PORT>
        <PORT>eth2</PORT>
        <PORT>eth3</PORT>
        <PATCH_PORT>
                <FROM>br1_xbr1</FROM>
                <TO>xbr1_br1</TO>
        </PATCH_PORT>
        <PATCH_PORT>
                <FROM>br1_br2</FROM>
                <TO>br2_br1</TO>
        </PATCH_PORT>
</BRIDGE>
```

```
<BRIDGE>
        <NAME>br2</NAME>
        <DPID>00:00:11:11:11:11:11:08</DPID>
        <PATCH_PORT>
                <FROM>br2_br1</FROM>
                <TO>br1_br2</TO>
        </PATCH_PORT>
        <PATCH_PORT>
                <FROM>GJ</FROM>
                <TO>C_GJ</TO>
        </PATCH_PORT>
        <PATCH_PORT>
                <FROM>MYREN</FROM>
                <TO>C_MYREN</TO>
        </PATCH_PORT>
</BRIDGE>

<BRIDGE>
        <NAME>brcap</NAME>
        <DPID>00:11:11:11:11:11:11:06</DPID>
        <PATCH_PORT>
                <FROM>C_GJ</FROM>
                <TO>GJ</TO>
        </PATCH_PORT>
        <PATCH_PORT>
                <FROM>C_MYREN</FROM>
                <TO>MYREN</TO>
        </PATCH_PORT>
</BRIDGE>
```

그림 8 OF@TEIN SmartX Box Type B* 기본 네트워킹 템플릿 예제

o OvN-Manager는 네트워킹 템플릿에 기재된 네트워킹 구성과 동일하도록 항상 프로비저닝을 수행하므로, 상기 템플릿을 OvN-Manager에 전달하게 되면 항상 동일한 오버레이 가상 네트워킹이 프로비저닝 되어 있음을 보장할 수 있다.

3. OvN-Manager 기능 검증

o OvN-Manager의 프로비저닝, 에러 검출/복구 기능을 검증하기 위하여 아래 시나리오에 따라 간단한 기능 검증을 수행하였다.

- 박스 내의 브릿지, 터널, SDN 제어기 연결과 같은 세부 설정에 대한 자동 프로비저닝 및 에러 복구 기능
- 박스 간을 연결하는 VXLAN 터널의 에러 검증 및 복구 기능

o OvN-Manager의 실증을 위해 OF@TEIN Playground의 SmartX Box들을 활용한다. OvN-Manager는 GIST에 위치한 박스에서 실행되어 파키스탄과 MYREN NOC 사이트에 위치한 두 박스를 대상으로 자동 프로비저닝 기능을 검증한다. 이 때 두 박스는 그림 8에서 제시한 OF@TEIN Playground Type B* 박스의 기본 네트워킹 설정 및 이에 따른 네트워킹 템플릿을 기본으로 한다.

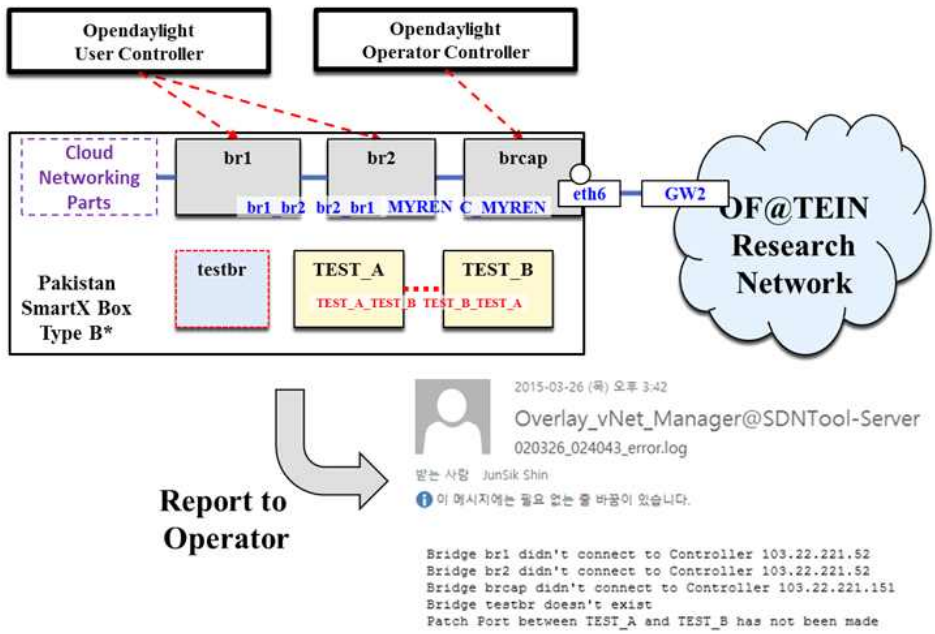


그림 9 OvN-Manager 기능 검증: 브릿지, 포트, 컨트롤러 세부설정

o 첫 번째 검증 시나리오는 파키스탄 사이트 SmartX Box에 브릿지, 포트, SDN 컨트롤러 연결이 설정되어 있지 않을 때 OvN-Manager를 이용해 자동으로 에러를 찾고, 이를 자동으로 프로비저닝 하는 것이다. (그림 9) 이를 위하여 기본 네트워킹 템플릿에 브릿지로 testbr, TEST_A, TEST_B를 생성하고 TEST_A와 TEST_B를 연결하는 패치 포트를 추가적으로 정의한다. 정의한 후 OvN-Manager를 실행시

키면 자동으로 해당 브릿지와 패치 포트가 설정되는데, 브릿지, 포트 설정상의 에러 탐지 및 복구 기능을 검증하기 위해 testbr 브릿지와 TEST_A TEST_B 브릿지의 패치 포트, 그리고 br1, br2, brcap의 SDN 제어기 연결을 해제한다. OvN-Manager는 30초 마다 반복 실행되므로 자동으로 전체 프로시저가 다시 실행되는데 이 때 제거한 설정 값을 템플릿 내용과 비교하는 과정에서 자동으로 발견하고 이를 운영자에게 이메일로 리포트 하는 것을 볼 수 있다. 또한 상기 화면에서는 표기가 되어 있지 않으나, 리포트를 하기 전 해당 에러를 자동으로 복구한다.

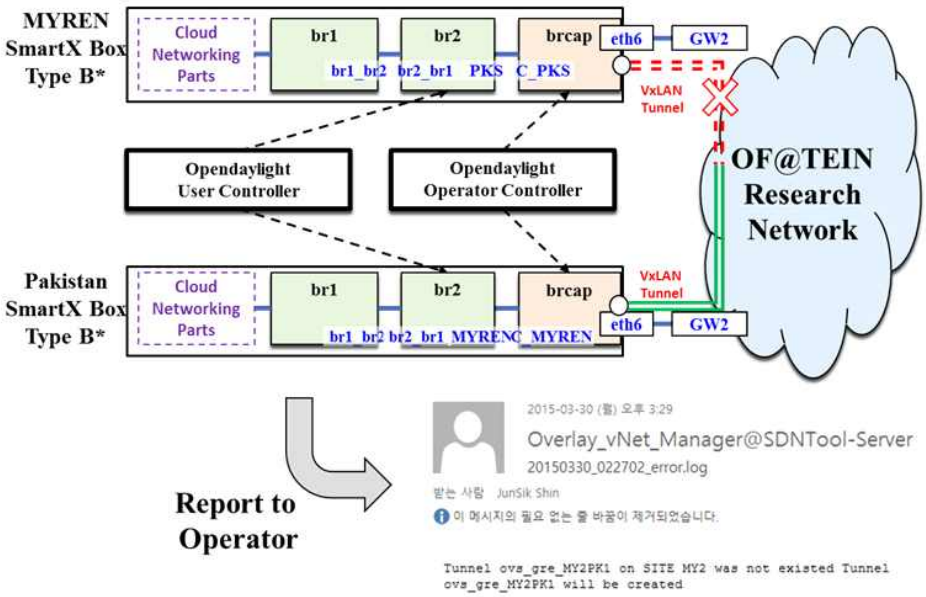


그림 10 OvN-Manager 기능 검증: VXLAN 터널 에러 자동 검증 및 복구

o 첫 번째 검증 시나리오는 하나의 박스를 대상으로 세부적인 설정(Fine-grained Configuration)이 가능한지를 검증하는 것이었다면, 두 번째 검증 시나리오는 메쉬 형태로 구성된 복잡한 VXLAN 터널을 자동으로 관리할 수 있는 지를 보이는 것이다. (그림 10) 본 검증을 위하여 OF@TEIN Playground의 MYREN, 파키스탄 사이트 SmartX Box를 대상으로 기본 네트워킹 설정을 한 후, MYREN-파키스탄 SmartX Box를 연결하는 VXLAN 터널의 MYREN 쪽 endpoint를 강제로 제거하였다. 이 때도 OvN-Manager는 VXLAN의 연결 상태를 ping으로 체크하고, 현재 SmartX Box의 네트워킹 설정 상태와 템플릿을 비교하여 VXLAN 터널의 이상 유무를 체크한다. 위의 시나리오에서는 터널을 통해 ping이 전달되지 않고, MYREN 상에 터널링을 위한 포트가 제거되었으므로 OvN-Manager는 이를 통해 MYREN의 설정에 문제가 있음을 파악할 수 있다. 따라서 터널에 에러가 있음을 메일로 리포트하고 이를 자동으로 복구하게 된다.

o 두 시나리오를 통해 OvN-Manager가 네트워킹 템플릿을 중심으로 오버레이 가상 네트워킹을 구성하는 브릿지, 터널, 플로우를 대상으로 매우 세밀한 수준까지의 프로비저닝 자동화가 가능함을 확인할 수 있으며, 추가적으로 템플릿과의 비교를 통해 에러를 자동으로 감지하고 복구 가능함을 확인할 수 있다. 따라서 OF@TEIN Playground를 대상으로 OvN-Manager를 적용하면 오버레이 가상 네트워킹 프로비저닝 측면의 운영 효율성을 증대시킬 수 있음을 알 수 있다.

4. Conclusion

SDN - Cloud 실증을 통합적으로 지원하는 OF@TEIN Playground가 독립적인 복수의 L2 네트워크가 상호 연결된 환경임에 따라 Multi-Region 기반의 오픈스택 클라우드를 구축해야 한다. 이 때 다른 Region에 위치한 VM간 통신을 위해서 오픈스택 Neutron이 관리하는 네트워크 외에 박스 간을 VXLAN 터널링으로 상호 연결하는 오버레이 가상 네트워킹을 추가적으로 설정해야 한다. 이 때 박스가 다수임에 따라 발생하는 프로비저닝 복잡성을 해소하고자 개발한 OvN-Manager (Overlay vNetworking Manager)의 설계 및 구현 과정을 본 기술문서를 통해 세밀히 설명하였다. 그리고 개발한 OvN-Manager의 네트워킹 템플릿 기반 자동 프로비저닝 기능을 실제 OF@TEIN Playground를 대상으로 검증해 보았다.

현재 기술한 OvN-Manager은 BASH 스트립트 기반의 도구로써 소프트웨어 측면의 확장성에 한계를 갖고 있으며, 재사용성이 떨어진다는 한계점을 갖고 있다. 또한 오버레이 가상 네트워킹의 프로비저닝 자동화를 목표로 하고 있으나, 현재까지는 OVS를 활용하는 브릿지, 터널 자원과 OpenDaylight를 활용하는 플로우 자원의 프로비저닝 만을 다루고 있다. 또한 실제 Field에서 운영 도구가 가져야 할 필수 기능인 보안, 고가용성 등의 기능은 전혀 고려하지 않았다. 따라서 앞으로 이러한 한계점들을 점차적으로 해소하고 기능을 추가해 나가면서 오버레이 가상 네트워킹의 많은 영역을 안정적으로 자동 프로비저닝할 수 있는 도구로 개선해 나갈 예정이다.

References

[1] OpenStack, http://openstack.org/.

[2] Mahalingam, Mallik, et al. "Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks." Internet Req. Comments (2014).

[3] McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.

[4] M. Httermann, DevOps for developers. Apress, 2012.

[5] Open vSwitch, http://openswitch.org/.

[6] J. Kim et. al., "OF@TEIN: An OpenFlow-enabled SDN testbed over international SmartX Rack sites," in *Proc. APAN –Networking Research Workshop*, Aug, 2013.

[7] K. Zhang, "OpenStack Zoning - Region/Availability Zone/Host Aggregate", https://kimizhang.wordpress.com/2013/08/26/openstack-zoning-regionavailability-zonehost-aggregate/.

[8] Medved, Jan, et al. "Opendaylight: Towards a model-driven sdn controller architecture." 2014 IEEE 15th International Symposium on. IEEE, 2014.

[9] Na, Taeheum, and JongWon Kim. "Inter-connection automation for OF@ TEIN multi-point international OpenFlow islands." Proceedings of The Ninth International Conference on Future Internet Technologies. ACM, 2014.

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: 2016/05