

K-ONE 기술 문서 #24

K-ONE 공용개발환경을 위한 Tricircle을 이용한 Multi-Site 클라우드 구축

Document No. K-ONE #24

Version 0.1

Date 2018-02-16

Author(s) 강문중, 신준식

■ 문서의 연혁

버전	날짜	작성자	내용
초안 - 0.1	2018. 02. 23	강문중, 신준식	초안 작성
0.2	2018. 02. 27	강문중, 신준식	오탈자, 목차 수정

본 문서는 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신
기술진흥센터의 지원을 받아 수행된 연구임 (No. 2015-0-00575, 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(MSIT) (No. 2015-0-00575, Global SDN/NFV
OpenSource Software Core Module/Function Development)

기술문서 요약

본 기술문서는 K-ONE 컨소시엄 참여기관인 광주과학기술원, 고려대학교, 숭실대학교, 포항공과대학교, 한국과학기술원, 총 5개 기관에서 각자 목표로 상정한 다양한 오픈소스 소프트웨어의 개발 및 검증환경을 공통된 하드웨어 묶음 형태로 제공하기 위하여 Edge-Cloud 패러다임에 대응하는 클러스터형 테스트베드인 K-Cluster 다수가 여러 사이트에 걸쳐 분산 배포된 상황에서 이 K-Cluster들을 네트워크로 연동한 멀티 사이트 K-ONE 공용개발환경(또는 공용개발환경)에 대해, 멀티 사이트간 네트워킹을 OpenStack Tricircle 프로젝트를 이용해 자동화시키기 위한 방안에 대해서 서술한다.

Contents

K-ONE #24. K-ONE 공용개발환경을 위한 Tricircle을 이용한 Multi-Site 클라우드 구축

1. 서론	5
1.1. 목적	5
1.2. K-Cluster	6
1.3. K-ONE 공용개발환경	7
2. 관련 기술 배경	9
2.1. VLAN	9
2.2. VXLAN	10
2.3. VPN	13
3. OpenStack Tricircle	16
3.1. OpenStack Tricircle의 필요성	16
3.2. OpenStack Tricircle의 구조	18
4. Tricircle을 활용한 멀티사이트 K-ONE 공용개발환경 구축	20
4.1. Tricircle을 이용한 K-ONE 공용개발환경 설계	20
4.2. Tricircle을 이용한 K-ONE 공용개발환경의 부분적인 검증	23

그림 목차

그림 1 K-Cluster 구성요소	6
그림 2 K-Cluster Hardware 구성	7
그림 3 소프트웨어 정의 인프라 패러다임의 MultiX Challenges	8
그림 4 K-ONE 공용개발환경의 구축 현황도	8
그림 5 이더넷 프레임 내에 삽입되는 802.1Q 태그	9
그림 6 VLAN Trunk를 통한 멀티사이트 K-ONE 공용개발환경 구축 시 문제	10
그림 7 VXLAN 패킷의 구조	11
그림 8 VXLAN를 통한 멀티사이트 K-ONE 공용개발환경 구축	12
그림 9 실제 VXLAN 트래픽에서 평문으로 노출되는 패킷의 내용물	13
그림 10 OS 내부에서의 OpenVPN을 통한 패킷의 흐름	14
그림 11 L2TP/IPSec에서의 트래픽 캡슐화	15
그림 12 현재 K-ONE 공용개발환경의 사이트간 네트워크 연결	17
그림 13 Control Plane, Data Plane 관점에서 보는 OpenStack Tricircle	18
그림 14 Tricircle의 내부 구성도	19
그림 15 Tricircle에서의 사이트간 네트워킹 자동화	21
그림 16 Tricircle을 이용한 K-ONE 공용개발환경 상의 사이트간 터널링	22
그림 17 Tricircle 테스트 DevStack 설정 파일 중 수정 필요부분	24

K-ONE #1. K-ONE 공용개발환경을 위한 Tricircle을 이용한 Multi-Site 클라우드 구축

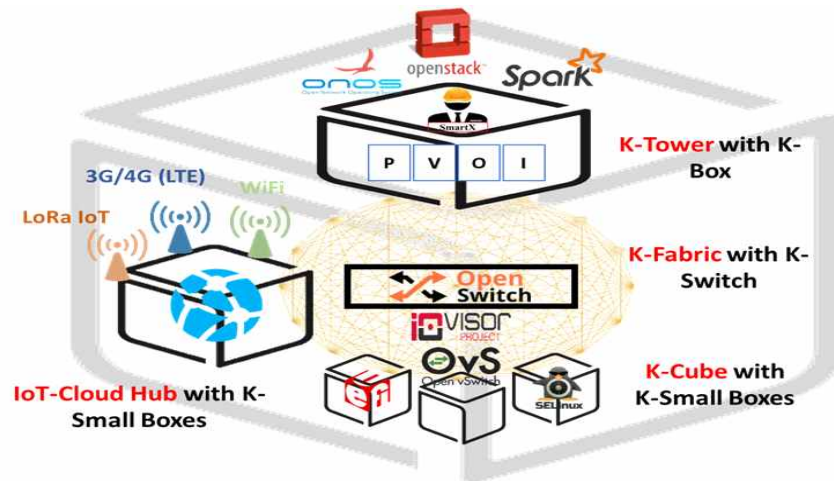
1. 서론

1.1. 목적

- o 본 기술문서는 K-ONE 컨소시엄 참여기관인 광주과학기술원, 고려대학교, 숭실대학교, 포항공과대학교, 한국과학기술원, 총 5개 기관에서 각자 목표로 상정한 다양한 오픈소스 소프트웨어의 개발 및 검증환경을 공통된 하드웨어 묶음 형태로 제공하기 위하여 Edge-Cloud 패러다임에 대응하는 클러스터형 테스트베드인 K-Cluster 다수가 여러 사이트에 걸쳐 분산 배포된 상황에서 이 K-Cluster들을 네트워크로 연동한 멀티 사이트 K-ONE 공용개발환경(또는 공용개발환경)에 대해, 멀티 사이트간 네트워킹을 OpenStack Tricircle 프로젝트를 이용해 자동화시키기 위한 방안에 대해서 서술한다.
- o OpenStack Neutron은 소프트웨어-정의 네트워킹을 활용해 OpenStack 클러스터 내의 네트워킹을 자동화하는 프로젝트로, 클러스터의 사용자가 원하는 형태의 네트워크 구조를 소프트웨어적으로 구축할 수 있다. 그러나 OpenStack Neutron은 어디까지나 OpenStack 클러스터가 위치한 하나의 지역(예를 들어 데이터센터 하나)에 대해서만 고려해 설계되었다는 단점이 있다. 여러 지역에 걸친 OpenStack 클러스터를 구축하는 것이 불가능한 것은 아니나, 이렇게 구축된 OpenStack 클러스터는 내부적으로 멀티 사이트에 대한 구분이 불가능해 자원의 실제 인접성을 고려하지 않는 비효율적인 네트워킹이 발생할 수 있다.
- o OpenStack Tricircle은 OpenStack의 하위 프로젝트 중 하나로 앞서 언급한 멀티 사이트간 네트워킹 자동화의 한계점을 넘는 것을 목적으로 한다[1]. Tricircle은 OpenStack 외의 별도 설정 없이 Neutron의 API를 확장시켜 OpenStack의 다른 프로젝트들에서도 매끄럽게 멀티 사이트 네트워킹을 활용할 수 있고, 클러스터간 지역 구분을 통해 자원의 실제 인접성이 잘 고려된 효율적인 네트워킹을 구축할 수 있게 한다.
- o K-ONE 공용개발환경은 단일 K-Cluster로는 실증이 어려운 멀티사이트/도메인 관련 실증을 통합적으로 제공하기 위하여 K-Cluster 시제품들을 K-ONE 컨소시엄의 참여 사이트에 각각 배포하고 이를 KREONET 연구망으로 연결함으로써 구축한 5개의 멀티사이트에 걸친 실증 테스트베드이다. 본 기술문서에서는 K-ONE 공용개발환경의 OpenStack Tricircle을 활용한 네트워킹 자동화 설계 및 구축 과정에 대해 상세히 기술한다.

1.2. K-Cluster

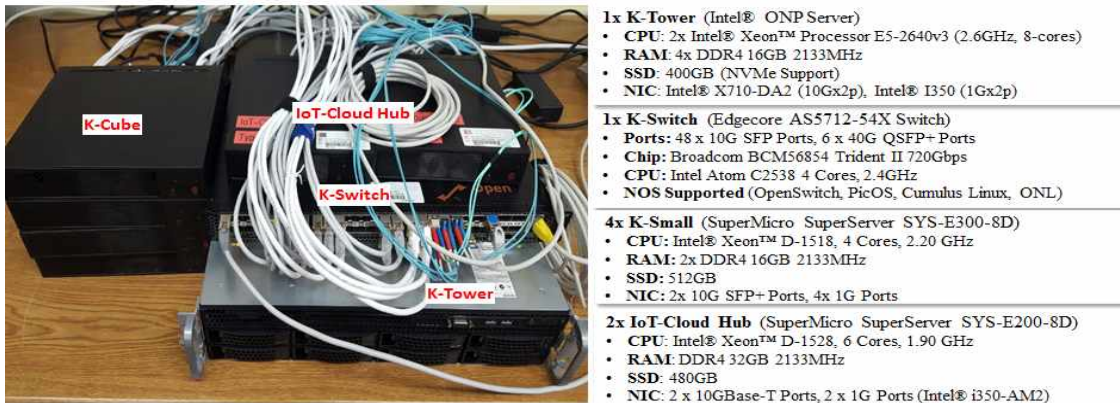
- o K-Cluster 및 K-ONE 공용개발환경의 자세한 설명은 'K-ONE 기술문서 16#_멀티사이트 클라우드 실증환경에 대응하는 K-Cluster 중심의 K-ONE 공용개발환경 설계 및 활용 방안'을 참조하면 되고, 본 기술문서에는 간략하게 기술한다[2].



<그림 1: K-Cluster 구성요소>

- o <그림 1>과 같이 K-Cluster는 소프트웨어 정의 인프라 패러다임에 대응하는 다양한 SDN/NFV/Cloud 실증을 제공하기 위한, Edge-Cloud 모델에 대응하는 경제적인 소규모 클러스터 형태의 테스트베드로 정의한다. K-Cluster의 예상 활용처는 대규모의 ICT 서비스를 제공하는데 필요한 안정적이고 고성능의 인프라를 구성하는데 쓰이는 것이 아니라, Edge-Cloud와 연관된 다양한 SDI 실증을 및 연구개발을 위한 적정 규모의 테스트베드를 확보하기 어려운 연구자, 개발자를 위한 소규모 테스트베드 환경을 기본 활용처로 상정한다. 물론 K-Cluster도 안정적이고 고성능의 자원을 제공하는 것이 주요 요구사항 중 하나로 실제 Production 환경에 적용하는데 부족함이 없으나, 요구사항의 우선순위 측면에서 다양한 Edge-Cloud 기술을 하나의 테스트베드에 수용하기 위한 자원 측면의 유연성과 구축 경제성을 설계 시 가장 높은 우선순위로 한다.
- o 먼저 K-Tower는 K-Cluster 전체를 DevOps(개발운영병행체제) 관점에서 관리/운영의 자동화를 위한 단일 또는 복수의 K-Box들로 구성된 관제(monitor & control)을 전담한다. K-Tower는 내부적으로 오픈소스 DevOps 도구를 활용하여 자동화를 지원하는 Provisioning 센터, Visibility 센터, Orchestration센터, 그리고 Intelligence 센터를 위한 지원역할을 담당하는 소프트웨어 적인 기능들을

포함한다. K-Cluster의 하단부에 위치한 K-Cube는 다수의 동종 K-Small 박스 자원들로 구성된 K-Cluster 내부의 클러스터로써 계산/저장/네트워킹을 담당하는 공유된 자원집합을 지칭한다. K-Fabric은 K-Cluster의 중앙에 위치하여 물리/가상 네트워킹 기능들을 활용하여 패브릭 형태의 밀결합된 네트워킹을 구성한다. 이를 통해 K-Cluster 구성 요소들인 K-Tower, K-Cube, IoT-Data Hub 간의 빠르고(fast) 유연하며(flexible) 안정된(reliable) 연결성을 제공한다.

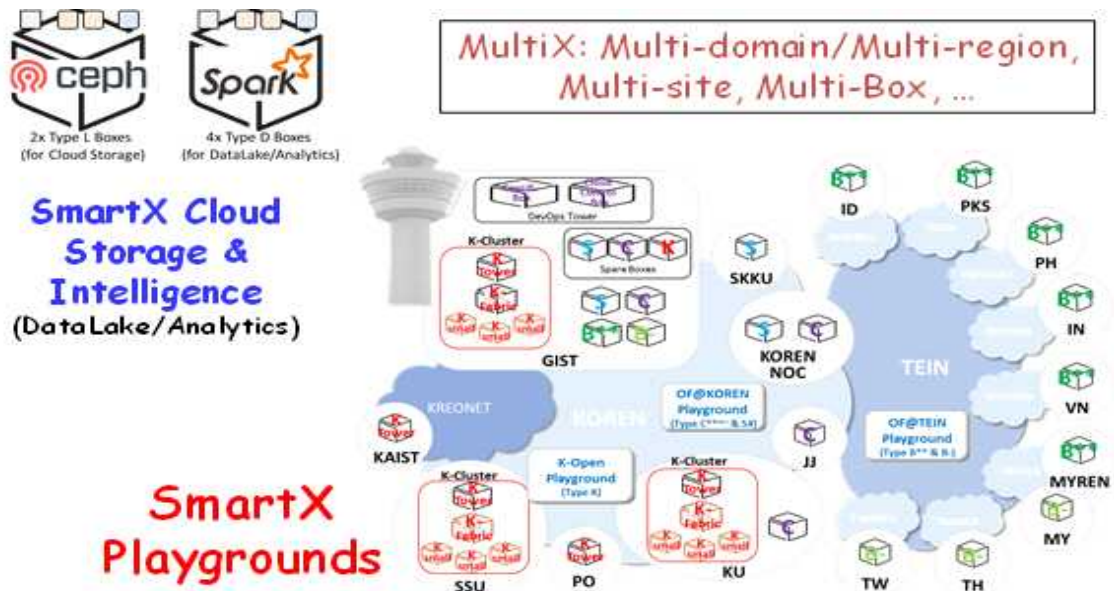


<그림 2: K-Cluster Hardware 구성>

- o K-Tower, K-Fabric, K-Cube, IoT-Data Hub등으로 구성된 K-Cluster 모델을 활용하면 에지 클라우드에서 요구하는 SDN/NFV/Cloud 통합과 FastData/BigData/ HPC 기반의 다양한 서비스를 저렴한 상용 화이트박스들을 활용하여 유연하게 대응할 수 있다.

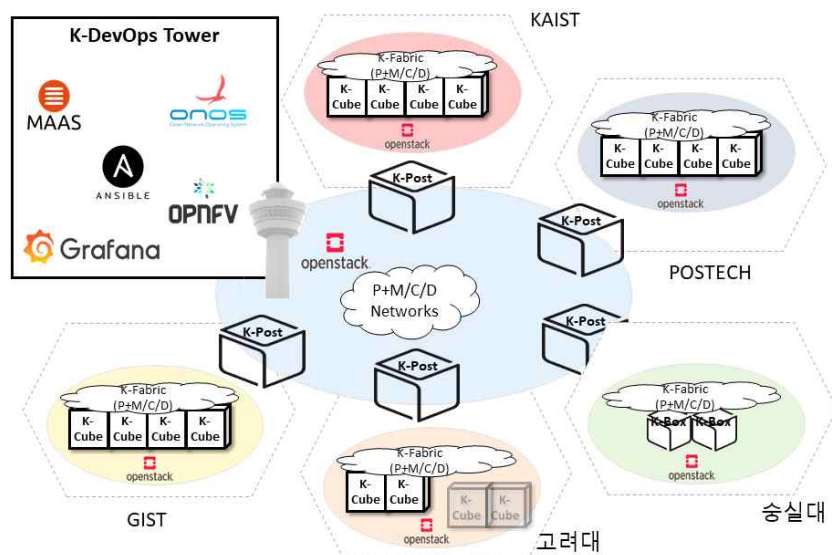
1.3. K-ONE 공용개발환경

- o 고립된 하나의 박스, 하나의 클러스터, 하나의 사이트를 고도화 하는 전략은 기술적으로 이미 많이 성숙되어 자원의 용량 및 성능 측면에서 큰 향상을 기대하는 것은 어려우며, 근본적으로 자원 활용의 유연성/확장성 측면의 커다란 단점을 내포하고 있다. 게다가 다수의 멀티 자원요소들 간의 연결성을 제공하는 클러스터링, 네트워킹 기술이 고도화 됨에 따라 다수의 요소를 연결함으로써 발생하는 병목현상, 성능저하 보다 자원의 클러스터링을 통한 유연하고 확장성 있는 활용이 큰 가치를 갖게 되었다. 따라서 최근에는 <그림 3>의 우측 상단과 같이 멀티박스를 넘어서 멀티리전, 멀티사이트, 심지어 멀티도메인과 같이 다양한 멀티 이슈에 대응하는 MultiX가 ICT 인프라의 연구 키워드의 하나로 부상하고 있다.



<그림 3: 소프트웨어 정의 인프라 패러다임의 MultiX Challenges>

- o K-ONE 컨소시엄 기관을 대상으로 멀티사이트 SDN/NFV/Cloud 통합 실증환경인 K-ONE 공용개발환경을 <그림 4>과 같이 구성하였으며, 이를 위해 다음과 같은 노력을 수행하였다.
- o K-Cluster 및 K-ONE 공용개발환경의 자세한 설명은 'K-ONE 기술문서 16#_멀티사이트 클라우드 실증환경에 대응하는 K-Cluster 중심의 K-ONE 공용개발환경 설계 및 활용 방안'을 참조하면 되고, 본 기술문서에는 간략하게 기술한다.



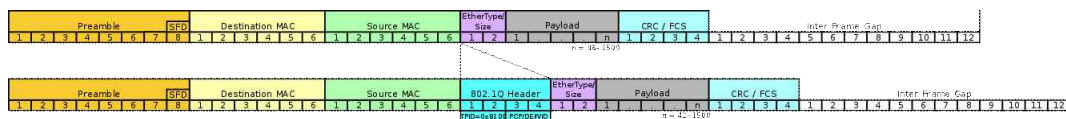
<그림 4: K-ONE 공용개발환경의 구축 현황도>

2. 관련 기술 배경

- o 2절은 멀티사이트 K-ONE 공용개발환경을 구축하기 위해 필요한 기술들과 관련 오픈소스 프로젝트들에 대해 간략하게 기술한다.

2.1. VLAN

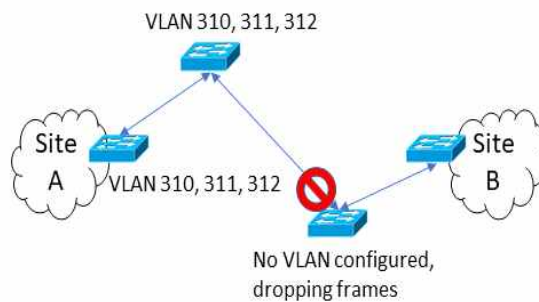
- o VLAN은 OSI 2계층의 기술로, 브로드캐스트 도메인을 분할하는 기술이다[3]. VLAN은 네트워크 패킷에 태그를 추가하고 이 태그들을 네트워킹 장비들에서 다루는 방식으로 동작한다. 이를 통해 물리적으로는 단일 네트워크에 연결되어 있지만, 마치 서로 다른 형태의 네트워크로 갈라져 있는 것처럼 동작할 수 있으며, 여러 케이블이나 네트워킹 장치들을 사용할 필요성을 줄인다.
- o IEEE 802.3 이더넷 네트워크 상의 VLAN을 위한 표준으로 IEEE 802.1Q가 있으며, 이 표준에서는 이더넷 프레임을 위한 VLAN 태깅과 부착된 태그들을 브릿지와 스위치에서 처리하는 절차를 정의한다. 802.1Q에서는 <그림 5>와 같이 소스 MAC 주소와 EtherType 필드 사이에 32비트 크기의 필드값을 정의하고, 12비트 크기의 VLAN ID값을 가지며 따라서 최대 4048(2^{12})개의 VLAN을 동시에 정의할 수 있다. 일반적으로 네트워크 구조의 말단(Tree 구조의 Leaf)에 해당하는 서버 등의 장비에서는 VLAN 태그를 다루지 않고 미부착 상태(Untagged)로 프레임을 전송하며, 그 장비가 연결되는 스위치에서는 그 장비가 연결된 포트에 대해 들어오는 모든 Untagged 프레임을 특정 VLAN에 소속된 프레임으로 다루고, 반대로 그 포트로 나가는 모든 Tagged 프레임에서 VLAN 태그를 제거해서 Untagged 프레임 상태로 내보내는 Access 모드를 주로 활용한다.



<그림 5: 이더넷 프레임 내에 삽입되는 802.1Q 태그>

- o 스위치 간 VLAN 확장은 가장 손쉽게는 한 스위치에서 하나의 VLAN이 전용 할당된 Access 모드 포트를 다른 스위치에 꽂는 식으로 가능하지만, 확장하는 VLAN이 많아질수록 두 스위치 간 케이블링도 비례해서 복잡해진다. 따라서 일반적으로는 두 스위치 간 하나의 링크만을 유지하고, 여러 개의 VLAN을 전파할 수 있도록 하는 Trunk 모드로 설정한 포트를 통해 VLAN을 전파한다. Trunk 모드에서는 Untagged 프레임이 전송되는 대신, Tagged 프레임을 그대로 내보내므로 한 포트에 여러 VLAN의 프레임이 섞여 들어오더라도 구분이 가능하다.

- o VLAN Trunk를 활용한다면 단일 네트워크를 멀티 사이트에 걸쳐 확장시키는 것이 가능하므로, 멀티 사이트를 단일 네트워크로 통일시켜 멀티사이트 K-ONE 공용개발환경을 구축하는 것도 가능하다. 그러나 VLAN을 이용해 멀티사이트 K-ONE 공용개발환경을 구축하는 것은 큰 단점이 있는데, 두 사이트 간 VLAN 네트워크가 거치는 모든 홉의 스위치들에 대해 VLAN Trunk 설정이 업데이트되어야 한다는 점이다. VLAN Trunk 설정이 되지 않은 스위치에서는 Untagged 프레임을 제외한 Tagged 프레임들이 전송될 경우 <그림 6>과 같이 해당 프레임들을 모두 폐기해버린다. 따라서 지나가는 구간의 모든 스위치에 대해 OpenStack 클러스터를 구축하고자 하는 주체가 접근권한을 가지고 있는 전용망으로 원거리 구간이 연결된 네트워크 환경에서만 사용가능한 기술이며, 지나가는 경로의 모든 스위치들에 대한 VLAN Trunk 설정 업데이트가 필요하므로 설정이 까다롭고 실수로 인한 오류가 발생하기 매우 쉬우며, 네트워킹 자동화가 매우 어렵다.



<그림 6: VLAN Trunk를 통한 멀티사이트
K-ONE 공용개발환경 구축 시 문제>

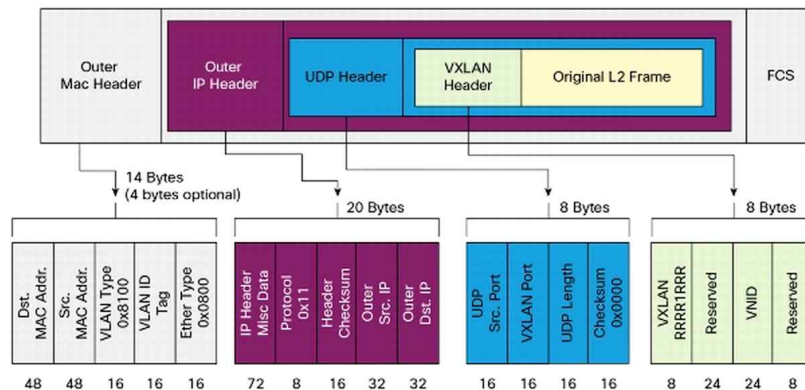
- o 또한 VLAN은 동시에 지정할 수 있는 VLAN의 수가 적다는 단점이 있다. 클라우드 컴퓨팅 환경에서 모든 테넌트들은 각자 독점하는 논리적인 네트워크가 필요하며, 따라서 각 모든 논리적인 네트워크들은 서로를 식별할 수 있는 수단(=네트워크 ID)이 필요하다. 본래 VLAN을 사용해 클라우드 환경에서의 어플리케이션과 테넌트들이 격리되었지만, VLAN 규격에 따르면 고작 4,096개의 네트워크 ID만이 지원되므로 대규모 클라우드 컴퓨팅 환경을 지탱하기에는 한계점이 있다.

2.2. VXLAN

- o VXLAN(Virtual Extensible LAN)은 기존의 OSI 3계층 인프라에서 오버레이 네트워크를 운영하기 위한 네트워크 가상화 기술이다[4]. 오버레이 네트워크란 기존의 OSI 2계층, 3계층 기반 네트워크 위에 겹친 상태로 존재하는 가상 네트워크로, 유

연한 네트워크 구조를 제공하기 위한 기술이다. VXLAN은 앞 장에서 언급한 VLAN의 한계점을 넘는 클라우드 컴퓨팅 환경을 위한 확장성을 위해 고안되었다.

- o VXLAN(Virtual Extensible LAN)의 핵심은 VLAN ID 필드 크기를 24비트로 확장시킨 VXLAN Segment ID로, 이를 통해 지원가능한 ID의 수가 약 1,600만개로 증가했다[5]. 그러면서도 VLAN과 같이 ID를 통해서 모든 네트워크가 구별 가능하므로 기존의 OSI 3계층 인프라 내에서 수많은 격리된 OSI 2계층 VXLAN 네트워크가 공존할 수 있다. VLAN과 같이 오로지 같은 네트워크에 소속된 장비만이 서로 통신할 수 있다.



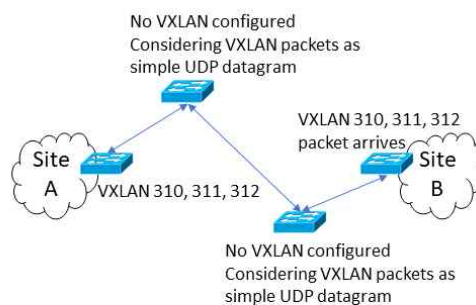
<그림 7: VXLAN 패킷의 구조>

- o <그림 7>과 같이 VXLAN은 OSI 2계층의 이더넷 프레임을 캡슐화(encapsulation) 하기 위해 OSI 4계층의 UDP 데이터그램을 사용하여 VLAN과 유사하게 ID를 부여해 캡슐화하는 방식을 사용하며 IANA에서 할당받은 4789 포트를 사용한다[6]. 기존의 VLAN Trunk 방식이 링크가 거치는 경로 중간의 모든 스위치들에 대한 VLAN Trunk 설정이 업데이트되어야 했다면, VXLAN은 정상적인 UDP 데이터그램 형태로 캡슐화가 되었기 때문에 아무 VXLAN 관련 설정이 되지 않은 스위치에서도 정상적인 전송이 가능하다. VXLAN 터널의 양 끝단인 VTEP(VXLAN Tunnel Endpoint)들은 가상 또는 물리적인 스위치 포트일 수 있으며, 여기서 캡슐화와 캡슐화의 해제가 이루어진다.

- o VXLAN은 최초 표준 당시 전파 방식으로 멀티캐스트(Multicast)를 채택했으며 이후 유니캐스트(Unicast) 방식이 추가되었다. 먼저 멀티캐스트는 멀티캐스트 전용으로 할당된 특정 IP 주소 대역을 통한 방식으로, 224.0.0.0에서 239.255.255.255까지의 Class D 주소 대역을 사용한다. 이 대역 내의 각 주소들은 하나의 멀티캐스트 그룹을 담당하며, IGMP(Internet Group Management

Protocol)을 통해 멀티캐스트 그룹에 가입할 수 있다. 이를 통해 특정 멀티캐스트 그룹에 전송되는 패킷은 가입된 모든 서버에게 전송된다. 이러한 특징을 가지는 멀티캐스트 방식을 사용한 VXLAN은 하나의 터널에 여러 개의 VTEP이 존재할 수 있으므로 OSI 2계층 네트워크 구조를 분할시키면서도 브로드캐스팅 등의 특징을 고스란히 살릴 수 있어 기존의 네트워크 분할이라는 취지에 있어 VXLAN의 전파방식에 적격이었으나, 멀티캐스트는 같은 OSI 2계층 네트워크 내에서만 전파되고, 다른 네트워크로 전파되기 위해서는 별도의 설정이 필요하며 멀티캐스트가 차단된 네트워크도 많았기 때문에 물리 네트워크에 따른 제약이 심해 여전히 전용망을 벗어나서 운용하기 어렵다는 단점이 있다[7].

- o 이에 대응해 추가된 유니캐스트 방식은 전파 방식의 특성상 브로드캐스팅이 불가능하고, 점대점(Point-to-Point) 연결만이 가능한 방식으로 양끝단에 1개씩 있는 2개의 VTEP만이 하나의 터널에 존재할 수 있다. 이 방식에서는 OSI 2계층 네트워크를 구성하기 위해 VTEP들을 가상 또는 물리 스위치에 붙여 터널들을 이음으로서 하나의 OSI 2계층 네트워크를 구성한다. 이 방식은 해당 오버레이 네트워크에 소속된 장비에서의 설정만이 필요하고 <그림 8>과 같이 중간 경유 장비에 대해서는 완전히 생략해도 되므로 설정의 난이도는 훨씬 낮으며, 이에 따라 스위치나 네트워크에 대한 제어권이 전혀 없는 공용망을 경유하는 오버레이 네트워킹도 가능하다는 장점이 있다. 그러나 점대점 연결만이 가능하다는 점으로 인해 하나의 오버레이 OSI 2계층 네트워크를 구성하는 터널들을 이어주기 위해서 가상화된 스위치를 많이 사용하게 되고, 또 VTEP의 숫자가 늘어남으로서 캡슐화로 인한 오버헤드 및 가상화된 스위치로 인한 오버헤드가 늘어나는 단점이 있다.

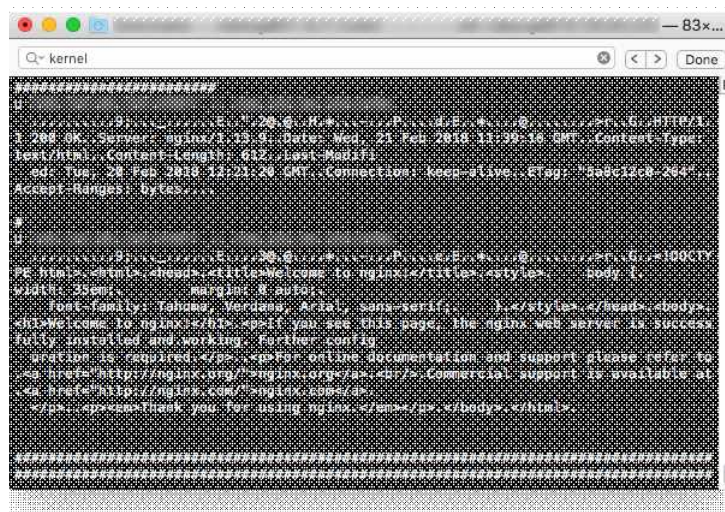


<그림 8: VXLAN를 통한 멀티사이트
K-ONE 공용개발환경 구축>

- o 멀티사이트 K-ONE 공용개발환경에서는 제어권이 없는 공용망을 통한 사이트간 연결이 필연적이다. 또한, 멀티사이트 네트워킹에 있어서도 중간 경유 스위치들의 설정까지 다루게 되는 경우 자동화가 매우 어려운 실정이다. 따라서 멀티사이트 K-ONE 공용개발환경 구성을 위해서는 유니캐스트 방식의 VXLAN이 적합하다.

2.3. VPN

- o VPN은 가상 터널링 기술, 트래픽 암호화를 통해 가상의 점대점 연결을 만들어낸다[8]. 일반적인 VPN은 철저한 점대점 연결로, 브로드캐스트 도메인을 지원하지 않거나 연결하지 않으며, 따라서 Microsoft Windows NETBIOS와 같은 서비스들이 일반적인 LAN에서와 같이 정상 지원되지 않을 수 있다. 이를 극복하기 위해 VPLS(Virtual Private LAN Service), L2TP(Layer 2 Tunneling Protocol) 등이 도입되었다.
- o VPN(Virtual Private Network)의 기본 목적은 공용망을 통해 사설 네트워크를 확장하는 것으로, 사용자들이 공유된 망 또는 공용망을 통해 사설 네트워크에 직접 연결되어 있는 컴퓨팅 장치들과 안전하게 통신하는 것이다. VPN을 통해 동작하는 어플리케이션들은 사설 네트워크의 관리, 보안, 기능성 등으로부터 이점을 얻을 수 있다.

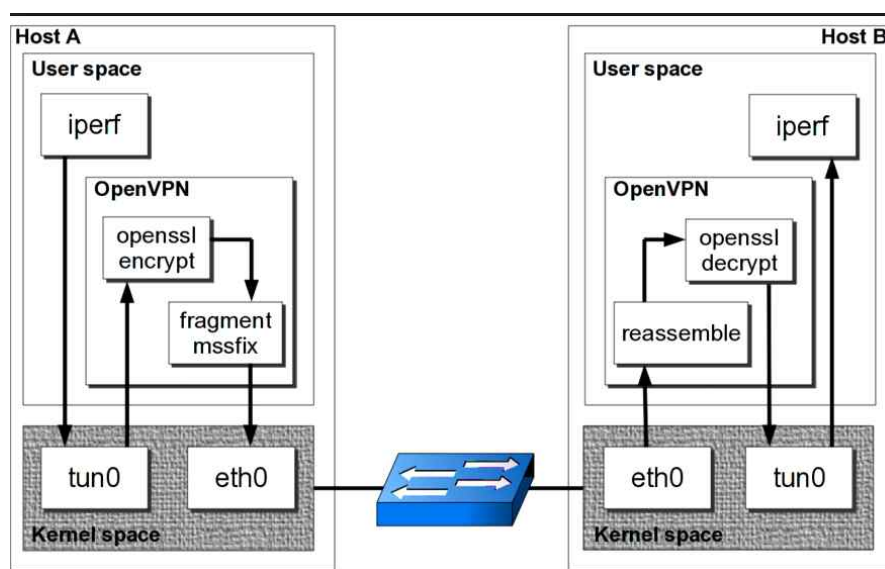


<그림 9: 실제 VXLAN 트래픽에서 평문으로
노출되는 패킷의 내용물>

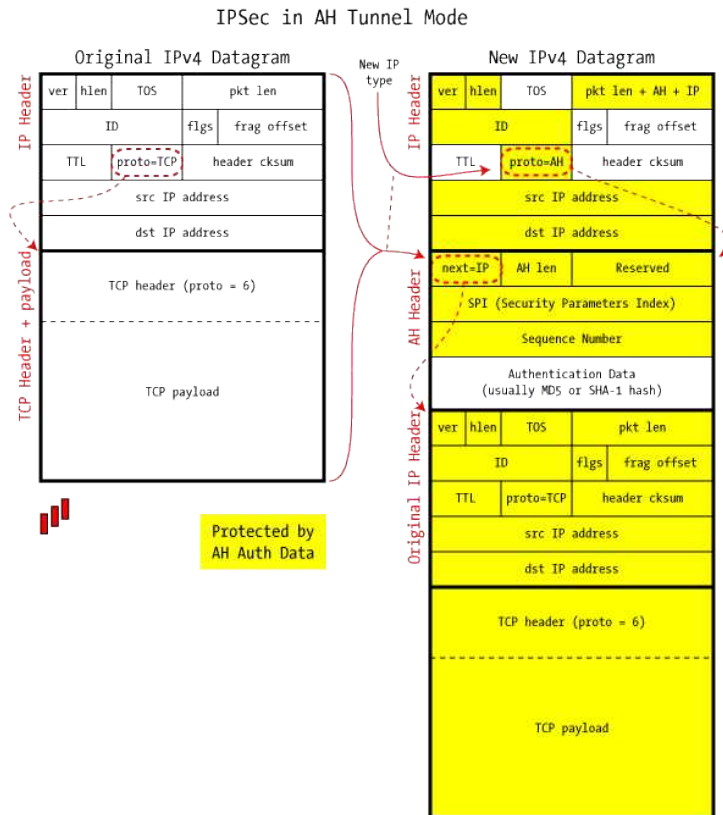
- o 이러한 목적을 충족시키기 위해서는 사설 네트워크를 공용망으로 확장하기 위해 네트워크 가상화가 필요하고, 또한 네트워크 가상화를 통해 형성된 가상 네트워크를 안전하게 만드는 수단이 필요하다. 네트워크 가상화를 위해 VPN은 VXLAN과 유사하게 기존의 OSI 3계층 인프라에서의 오버레이 네트워크를 운영하기 위한 가상화 기술을 포함한다. 그러나 VXLAN의 경우 단순히 패킷에 대한 캡슐화/캡슐화 해제만을 다루는 기술로, <그림 9>와 같이 VXLAN 트래픽은 캡슐화된 패킷의 내용이 고스란히 평문으로 노출되는 단점이 있다. 데이터센터와 같이 철저하

게 관리되는 전용망 내에서 오버레이 네트워킹을 통해 네트워크를 나누는 데에는 적합하지만 공용망을 경유하는 오버레이 네트워킹에는 다소 부적합한 면이 있다.

- o VPN은 공용망을 거치는 것을 염두에 둔 기술로, 캡슐화 과정에서 암호화 기법을 활용한다. VPN 점대점 터널의 양 끝단은 미리 약속된 방식에 따라 비밀키를 공유하고, 이 비밀키로 내부에 탑재되는 패킷을 암호화한다. 이를 통해 VPN 패킷이 경유해 지나가는 공용망에서 패킷을 추출해 읽더라도 모든 데이터는 암호화되어 있으므로 읽어낼 수 없고, 오로지 반대편에서만 해독키를 가지고 있으므로 데이터를 읽어낼 수 있다.
- o <그림 10>과 같이 VPN은 이러한 터널링 및 암호화를 위해 커널의 가상 인터페이스(tun/tap)를 통해 동작한다. 가상 인터페이스는 일반적인 물리 인터페이스와 달리 인터페이스에 대응되는 물리 어댑터가 존재하지 않는다. 가상 인터페이스는 유저영역 또는 커널영역에 있는 VPN 어플리케이션에 대응되어 있다. VPN 터널을 통해 데이터를 전송하고자 하는 어플리케이션은 해당 데이터가 담긴 패킷을 VPN의 가상 인터페이스로 보낸다. 가상 인터페이스에 들어온 패킷은 제일 먼저 VPN 어플리케이션에 의해 터널링 캡슐화 및 암호화 처리를 거친다. 이렇게 캡슐화된 패킷은 다시 커널의 IP 스택을 통해 물리 인터페이스에 들어와 물리 어댑터를 통해 노드 밖의 네트워크로 보내진다. 그리고 반대편에서 물리 인터페이스로 들어온 패킷은 해당 VPN 터널을 위해 Listen 상태로 대기 중인 VPN 어플리케이션에게 수신되고, VPN 어플리케이션에 의해 캡슐화 해제 및 복호화가 진행된 후 원래 데이터가 담긴 패킷이 역으로 가상 인터페이스를 통해 빠져나와 원래 목표한 어플리케이션에게 전달된다.



<그림 10: OS 내부에서의 OpenVPN을 통한 패킷의 흐름>



<그림 11: L2TP/IPSec에서의 트래픽 캡슐화>

- o VPN 기술은 여러 종류가 있으나, 기술의 발전으로 충분한 암호화 수준이 보장되지 않는 것들을 제외한다면 L2TP/IPSec, OpenVPN 등의 기술이 있다. L2TP/IPSec은 정확하게는 VPN 터널 양 끝단에 가상화된 OSI 2계층 장치를 만들고, 이를 가상화된 브릿지 등을 통해 물리 네트워크와 연결하는 터널링 기술인 L2TP(Layer 2 Tunneling Protocol)와 네트워크로 보내지는 패킷에 대한 인증과 암호화를 수행하는 프로토콜 모음인 IPSec(Internet Protocol Security) 암호화 기술의 조합으로, IPSec은 세션의 시작 시 터널의 양끝단간 상호 인증과 암호키 교환을 수행한다[9]. OpenStack Neutron의 VPNaaS 플러그인의 경우 L2TP/IPSec에 키교환 방식으로는 NAT 환경에서도 동작 가능하고, 연결이 끊어졌을 때 재빠른 재구성이 가능한 IKEv2를 사용한다.
- o OpenVPN은 VPN을 구현하는 오픈 소스 프로젝트 중 하나로, 안전한 점대점 연결 또는 사이트간 연결을 OSI 2계층 또는 3계층 수준에서 지원한다[10]. OpenVPN은 SSL/TLS 키 교환 방식을 활용하는 자체적인 보안 프로토콜을 사용한다. OpenVPN은 NAT 환경과 대부분의 방화벽을 통해서도 동작할 수 있으며, GPL 라이선스로 공개되었다. OpenVPN은 노드간 미리 공유된 비밀키(Pre-shared secret key), TLS 인증서, ID/PW, 그리고 서드파티 플러그인을 통한

LDAP(Lightweight Directory Access Protocol) 또는 SQLite/MySQL 등의 DB를 쓰는 방식으로 서로를 인증할 수 있다. 일대다 환경에서는 서버가 자체 CA(Certification Authority)를 갖추고 그 CA에 기반한 인증서를 모든 클라이언트마다 발급하는 식으로 인증이 가능하다. 이 프로토콜은 OpenSSL 라이브러리를 집중적으로 사용해 OpenSSL이 지원하는 TLS 프로토콜과 거의 모든 256비트 암호화까지를 지원한다.

- o 멀티사이트 K-ONE 공용개발환경에서는 확장성을 확보하기 위해서는 전용망 뿐만 아니라 공용망을 거치는 사이트간 네트워킹이 거의 필수적이다. 따라서 장기적인 관점에서 멀티사이트 K-ONE 공용개발환경의 확장에 있어 보안성이 보장되지 않는 VXLAN이 아닌 VPN을 통해 사이트간 네트워킹을 구성할 수 있는 방향으로 나아가야 한다.

3. OpenStack Tricircle

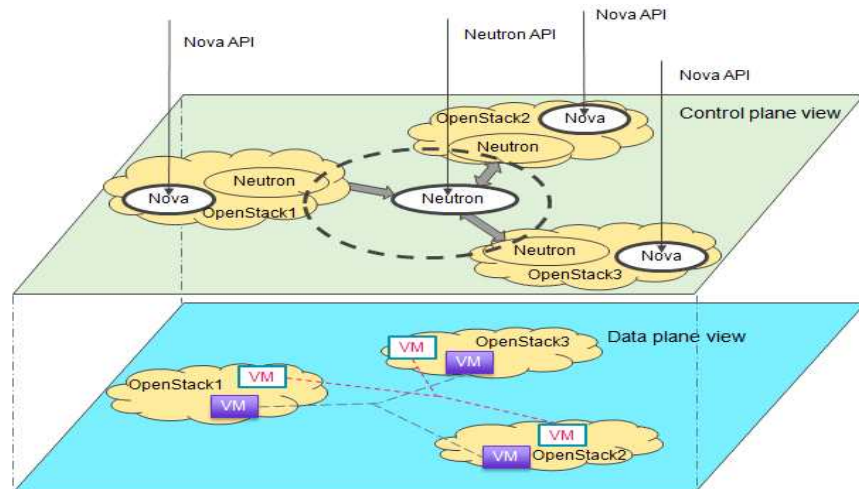
- o 3절은 OpenStack Tricircle를 이용한 멀티사이트 K-ONE 공용개발환경의 구축 방안을 논하기에 앞서 OpenStack Tricircle 프로젝트에 대해 간략하게 기술한다.

3.1. OpenStack Tricircle의 필요성

- o 소프트웨어 정의 인프라 패러다임의 핵심 기술인 SDN, NFV, Cloud의 경우에도 그동안의 연구개발을 통해 단일 클러스터, 단일 사이트를 대상으로 기술이 상당 수준 성숙되었다. 그리고 이를 시장의 대규모 인프라 상에 도입하기 시작하면서 상기 기술들이 다수의 사이트, 다수의 소유자들을 연결하는 멀티리전, 멀티사이트, 멀티도메인 기술로 근본적인 이슈 도메인을 변화해 나가고 있다. 따라서 오늘날의 SDN/NFV/Cloud 기술은 곧 MultiX 이슈와 연계한 연구 개발이 요구되며, 이를 위한 테스트베드가 필요하다.
- o 단일 K-Cluster를 활용하면 단일 클러스터/사이트를 범위로 한 SDN/NFV/Cloud의 통합적인 실증이 가능하나, 앞서 언급한 MultiX에 대응하는 실증을 지원하는 것이 어렵다. 따라서 K-Cluster들을 지리적, 네트워크 적으로 구분된 사이트들에 분산 배포하고 이를 유연하고 신뢰성 있는 네트워크로 연계함으로써 하나의 MultiX 테스트베드로 구성하는 것이 필요하다. 그리고 소프트웨어 도구들을 활용해 인프라의 구성을 자동화함으로써 다양한 SDN/NFV/Cloud 실증 환경을 쉽고, 빠르고, 유연하게 제공하는 테스트베드가 필요하다.
- o 따라서 상기 요구사항에 대응하여 각 K-Cluster를 다수의 사이트에 분산 배포하고 이를 연구망으로 엮어 구성한 멀티사이트 테스트베드인 K-ONE 공용개발환경

3.2. OpenStack Tricircle의 구조

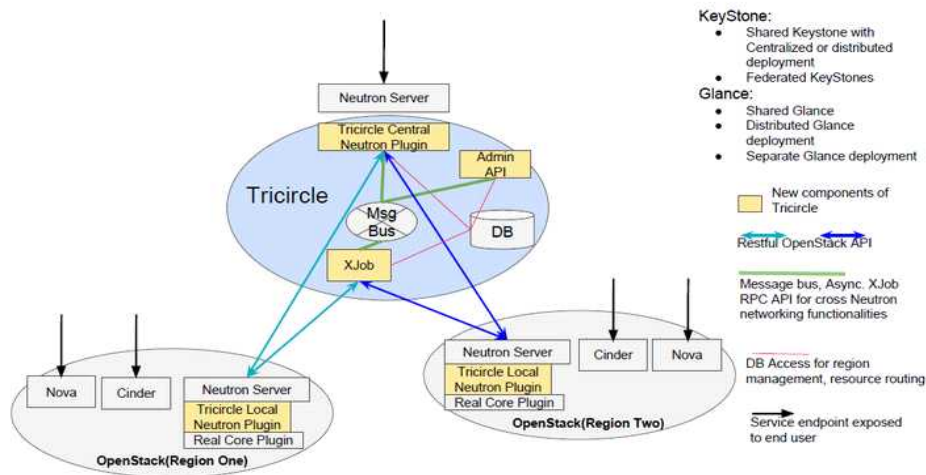
- o OpenStack Tricircle은 멀티사이트 OpenStack 구성에서의 각 사이트간 Neutron 사이의 자동화된 네트워킹을 목적으로 하는 프로젝트이다[1]. 클라우드를 관리하는 Control Plane 관점에서 보면 OpenStack Tricircle은 각 사이트에 각자 독립된 Neutron이 동작하는 OpenStack 클라우드들을 하나의 클러스터로 동작하도록 하며, 여러 사이트에 걸친 전역 네트워크와 라우터 외 기타 여러 추상화된 네트워크 자원들을 생성할 수 있도록 해준다. 최종 사용자 관점인 Data Plane 관점에서는, 모든 VM 또는 베어메탈 서버 또는 컨테이너들이 서로 다른 클라우드에 할당되지만 모든 사이트를 아우르는 추상화된 전역 네트워킹 자원을 통해 상호연결되며 동시에 테넌트 수준의 격리까지 보장해준다.



<그림 13: Control Plane, Data Plane 관점에서 보는 OpenStack Tricircle>

- o 중앙집중형 또는 분산형의 공유된 Keystone 인스턴스 또는 Federated Keystone은 OpenStack Tricircle과 여러 OpenStack 인스턴스들을 위한 Identity 관리를 위해 사용될 수 있다.
- o OpenStack Tricircle Local Neutron 플러그인은 OpenStack Neutron 서버 아래에서 OVN/Dragonflow Neutron 플러그인처럼 같은 프로세스로 동작한다. OpenStack Tricircle Local Neutron 플러그인은 사이트간 OpenStack Neutron 네트워킹을 위한 트리거 역할을 하며, 실제 코어 플러그인과 Neutron API 서버간의 아주 얇은 레이어 계층 역할을 한다. 이 OpenStack Tricircle Local 플러그인이 설치된 OpenStack Neutron 서버는 “Local Neutron”라고 불리며, 이는 중앙의

“Central Neutron” 과 대응되는 개념이다. 이 Local Neutron과 함께 동작하는 Nova/Cinder 또한 “Local Nova/Cinder”로 불린다.



<그림 14: OpenStack Tricircle의 내부 구성도>

- o <그림 14>와 같이 OpenStack Tricircle Central Neutron 플러그인은 OpenStack Tricircle Local Neutron 플러그인과 똑같이 OpenStack Neutron 서버 아래에서 OVN/Dragonflow Neutron 플러그인처럼 같은 프로세스로 동작한다. 이 플러그인은 사이트간 OpenStack 인스턴스들의 테넌트 수준의 OSI 2계층/3계층 네트워킹 자동화를 수행하는 역할을 한다. OpenStack Tricircle Central Neutron 플러그인이 설치된 Neutron 서버는 “Central Neutron”라고 불린다.
- o Admin API는 OpenStack 인스턴스와 Availability Zone의 매핑을 관리하며, 객체 UUID 라우팅 정보 수집 및 유지보수를 위한 API를 제공한다.
- o Xjob은 Admin API 및 OpenStack Tricircle Central Neutron 플러그인으로부터 사이트간 OpenStack의 기능과 다른 비동기 작업들에 대한 데이터를 받고 작업을 처리하는 역할을 담당한다. 예를 들어 어떤 Project를 위해 어떤 인스턴스를 처음으로 부팅하게 될 때, 아직까지 그 인스턴스를 위해 필요한 라우터, Security Group 규칙, FIP 외 다른 자원들이 아직까지 생성되지 않았을 수 있다. 이 자원들은 최대한 인스턴스의 최초 부팅 요청에 대한 반응 속도를 빠르게 하기 위해 비동기적으로 생성되기 때문이다. 네트워크, 서버넷, Security Group 자원과는 달리 앞서 언급한 자원들은 반드시 해당 자원들을 사용하는 인스턴스가 부팅되기 전에 만들어져야 한다. Admin API, OpenStack Tricircle Central Neutron 플러그인은 이러한 비동기 작업을 Xjob에 의해 제공되는 RPC API 지원 메시지 버스로 Xjob으로 보내진다.

- o 데이터베이스는 OpenStack Tricircle이 자체 OpenStack Tricircle Central Neutron 플러그인 및 Admin API, Xjob를 위해 Pods, 작업, 자원 라우팅 테이블을 보관하기 위해 필요하다. OpenStack Tricircle Central Neutron 플러그인은 또한 Central Neutron 서버의 데이터베이스를 재활용해 테넌트 내 IP/MAC 주소, 네트워크, 라우터 등의 자원들에 대한 전역 관리를 수행한다. OpenStack Tricircle Local Neutron 플러그인은 코어 플러그인과 Neutron API 서버와의 얇은 레이어로서만 동작하기 때문에, Local Neutron의 데이터베이스는 Neutron API 서버와 코어 플러그인을 위해서만 사용된다.
- o Glance 설치에 여러 가지 옵션이 있다. 첫째가 공유형 Glance로, 모든 OpenStack 인스턴스가 넓은 대역폭, 짧은 지연시간이 보장된 단일 사이트에 몰린 경우 추천된다. 두 번째는 분산형 백엔드를 가진 공유형 Glance로, OpenStack 인스턴스가 여러 사이트에 걸쳐 있을 때 사용하기 좋다. 세 번째는 분산형 Glance로, Glance 서비스와 Glance의 백엔드가 여러 사이트에 분산 배치되는 경우에 좋다. 마지막으로 서로 분리된 Glance 설치 방식이 있는데, 각 사이트는 서로 분리된 Glance와 백엔드가 구성되는 것으로 이 경우 사이트간 이미지 공유가 불필요한 경우 쓰는 방식이다.

4. Tricircle을 활용한 멀티사이트 K-ONE 공용개발환경 구축

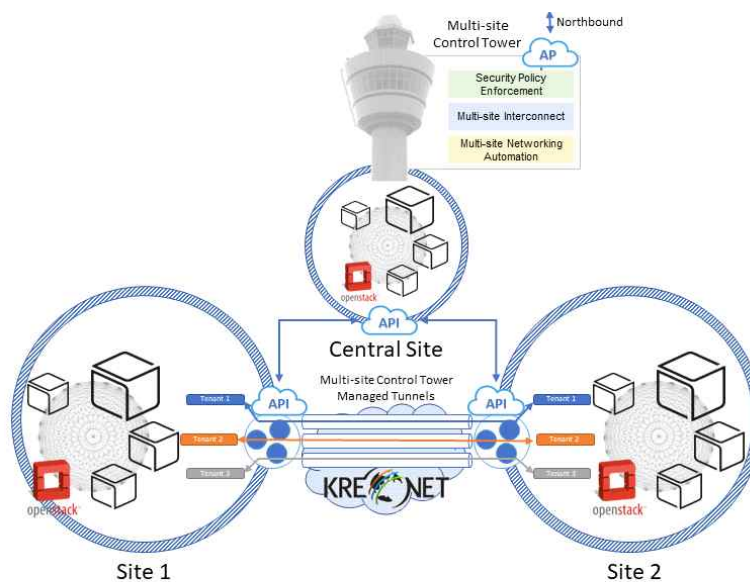
- o 4절에서는 본격적으로 K-ONE 공용개발환경에 OpenStack Tricircle을 적용해 멀티사이트 클라우드 환경을 구축하는 방법에 대해서 논한다.

4.1. Tricircle을 이용한 K-ONE 공용개발환경 설계

- o 다양한 멀티사이트 SDN/NFV/Cloud 들은 기본적으로 지리적, 네트워크 적으로 분할된 다수의 사이트들에서 발생하는 다양한 문제를 다루는 것을 목표로 한다. 따라서 이러한 멀티사이트 실증을 지원하기 위해 K-ONE 공용개발환경을 하나의 L2 네트워크로 묶는 것 보다는, 각 사이트 별로 구분된 L3 네트워크 서브넷에 연결된 구성이어야 한다. 독립적인 네트워크 환경을 기반으로 K-ONE 공용개발환경의 각 사이트들은 독립적인 SDN/NFV/Cloud 인프라를 구성하며, 이러한 각 사이트를 하나의 인프라로 연결하는 실제 환경에서 기인하는 멀티사이트 이슈를 K-ONE 공용개발환경 상에서 개발/실증할 수 있도록 지원해야 한다.
- o 비록 각 사이트는 독립적인 인프라 환경으로 구성되어야 하지만, 멀티사이트 환경인 K-ONE 공용개발환경을 효율적으로 운영하기 위해서는 중앙에서 전체 사이트들을 다양한 오픈소스 DevOps 도구들을 활용하여 자동화된 관제를 담당하는 Tower가 필요하다. K-ONE 공용개발환경에서는 특별히 확보된 박스를 활용하는 것이 아니라 각 K-Cluster의 K-Box (K-Tower)를 묶어 하나의 논리적인 Tower로

구축하는 것을 기본 설계로 하며, 이 논리적인 Tower를 K-DevOps Tower로 정의한다. 이 때 각 사이트에 분산된 K-Box들을 하나의 논리적인 Tower로 클러스터링하기 위해 OpenStack 클라우드를 활용해 K-Box들을 하나의 자원 풀로 클러스터링 한다. K-DevOps Tower의 각 K-Box는 자신이 속한 K-Cluster만 전담으로 관리하게 된다. 즉, Site A의 K-Box 상에 생성된 관계 VM은 Site B의 K-Cluster의 관계를 수행할 수 없게 구성되어야 한다. 이 때 K-DevOps Tower 또한 상기 제시한 P+M/C/D 네트워크 설계에 따라 구성되어야 한다.

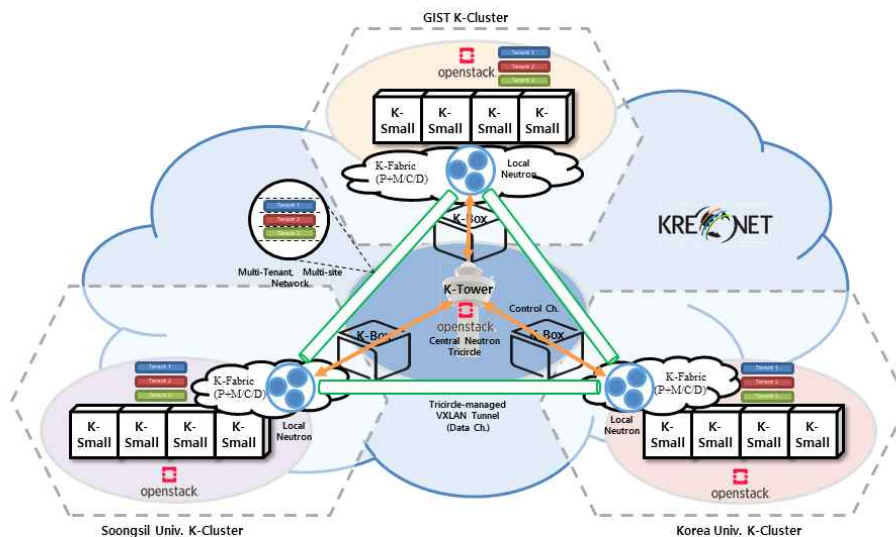
- o 이를 위해 현재 K-ONE 공용개발환경은 다수의 사이트에 배포된 K-Box들을 하나의 OpenStack 클라우드로 구성하기 위해 각 K-Box들을 VLAN을 통해 L2 네트워크로 직접 묶어놓은 상태이다[2]. 이 L2 네트워크는 상시 연결된 상태로 이 L2 네트워크 상에서 K-Box간 VXLAN 터널이 테넌트 별로 구성된다. 현재의 K-ONE 공용개발환경은 L2 네트워크 없이는 OpenStack Neutron에 의한 VXLAN 기반 네트워킹 자동화가 되지 않으므로 L2 네트워크가 필수였으나, OpenStack Tricircle을 도입할 경우 L2 네트워크 없이도 OpenStack Neutron에 의한 VXLAN 기반 네트워킹 자동화가 가능하다.



<그림 15: OpenStack Tricircle에서의 사이트간
네트워킹 자동화>

- o 상기 내용을 바탕으로 OpenStack Tricircle을 활용한 멀티사이트 K-ONE 공용개발환경의 설계를 <그림 15>와 같이 정리할 수 있다. OpenStack Tricircle을 이용할 경우 각 사이트는 Keystone과 같은 몇몇 요소 서비스를 제외하면 K-ONE 공용개발환경의 요구조건에 맞춰 독립적인 클러스터로 구성할 수 있다. 그러나 각

사이트가 독립적인 클러스터로 구성된다면 해당 사이트들은 사이트간의 네트워킹이 필요할 때 상대방에 대한 연결정보가 없으므로, 상호연결이 힘들다. 따라서 이러한 각 사이트의 독립적인 클러스터들을 중앙 제어하는 주체가 필요하다. OpenStack Tricircle에서는 이 중앙의 주체를 Central Region이라고 부르며, Central Region은 OpenStack Tricircle Central Neutron 플러그인을 통해 각 사이트에 설치된 OpenStack Tricircle Local Neutron 플러그인과 API 기반의 통신을 주고받으며 각 사이트의 네트워킹을 관제한다. OpenStack Tricircle에서는 이 API 기반의 통신을 주고받는 네트워크에 대해서는 언급하지 않는데, 기존의 이미 구성된 전용망 또는 별도로 구성된 터널링을 이용하여 구성하는 것이 일반적이다. 그러나 이 API 기반의 통신을 주고받는 사이트간 연결은 어디까지나 사이트간 네트워킹 자동화를 위한 제어 데이터만을 주고받는 용도로 큰 대역폭은 필요하지 않으며, OpenStack Tricircle은 테넌트 내의 사이트간 네트워킹에 한정된 자동화를 제공하며, 이를 위해 생성되는 VXLAN 터널들은 앞서 언급한 제어용 사이트간 연결과는 별도로 생성/삭제된다.



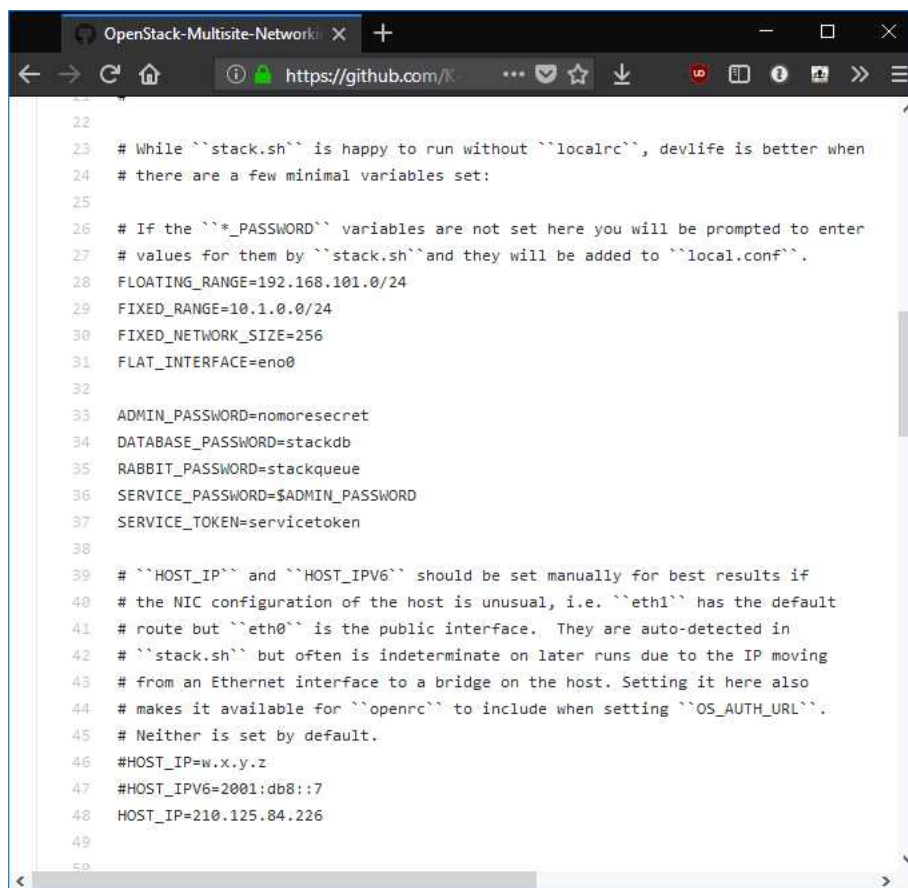
<그림 16: OpenStack Tricircle을 이용한 K-ONE 공용개발환경 상의 사이트간 터널링>

- o 따라서 <그림 16>과 같이 K-ONE 공용개발환경에 OpenStack Tricircle을 적용함에 있어, 먼저 사이트간 네트워킹 자동화를 중계하는 Central Region 역할은 이미 비슷한 역할을 맡고 있는 K-ONE 공용개발환경에서 중앙에서 전체 사이트들을 다양한 오픈소스 DevOps 도구들을 활용하여 자동화된 관제를 담당하는 Tower에 구성한다. 또한 OpenStack Tricircle을 이용하여 얻을 수 있는 가장 큰 이점은 중간경유 네트워크에 대한 설정 단순화이므로, 먼저 별도 연결이 필요한 해당

4.2. Tricircle을 이용한 K-ONE 공용개발환경의 부분적인 검증

-
- 23

대역을 할당해주면 된다. 'FIXED_NETWORK_SIZE'는 'FIXED_RANGE'에서 할당된 주소 대역의 크기로, /24 CIDR 대역일 경우 크기는 256이다. 'FLAT_INTERFACE'는 'FLOATING_RANGE'의 주소 대역이 할당되어 있는 OS 상의 실제 인터페이스 이름을 넣어야 한다. 'ADMIN_PASSWORD'는 OpenStack 클라우드의 최고관리자 admin 계정의 비밀번호이고, 'DATABASE_PASSWORD'는 OpenStack 클라우드에서 사용할 데이터베이스의 비밀번호이다. 'RABBIT_PASSWORD'는 요소 서비스간 API 콜을 주고받는데 사용할 메시지 큐의 비밀번호이다. 'HOST_IP' 옵션은 'FLAT_INTERFACE'에 실제 할당되어 있는 IP 주소를 입력하면 된다.



```
22
23 # While ``stack.sh`` is happy to run without ``localrc``, devlife is better when
24 # there are a few minimal variables set:
25
26 # If the ``*_PASSWORD`` variables are not set here you will be prompted to enter
27 # values for them by ``stack.sh`` and they will be added to ``local.conf``.
28 FLOATING_RANGE=192.168.101.0/24
29 FIXED_RANGE=10.1.0.0/24
30 FIXED_NETWORK_SIZE=256
31 FLAT_INTERFACE=eno0
32
33 ADMIN_PASSWORD=nomoresecret
34 DATABASE_PASSWORD=stackdb
35 RABBIT_PASSWORD=stackqueue
36 SERVICE_PASSWORD=$ADMIN_PASSWORD
37 SERVICE_TOKEN=servicetoken
38
39 # ``HOST_IP`` and ``HOST_IPV6`` should be set manually for best results if
40 # the NIC configuration of the host is unusual, i.e. ``eth1`` has the default
41 # route but ``eth0`` is the public interface. They are auto-detected in
42 # ``stack.sh`` but often is indeterminate on later runs due to the IP moving
43 # from an Ethernet interface to a bridge on the host. Setting it here also
44 # makes it available for ``openrc`` to include when setting ``OS_AUTH_URL``.
45 # Neither is set by default.
46 #HOST_IP=w.x.y.z
47 #HOST_IPV6=2001:db8::7
48 HOST_IP=210.125.84.226
49
50
```

<그림 17: OpenStack Tricircle 테스트를 위한 DevStack 설정
파일 중 테스트를 위해 수정이 필요한 부분>

- o 정상적으로 DevStack 설치를 진행하기 위해서는 OpenStack Pike 버전을 위한 DevStack 설치 매뉴얼(<https://docs.openstack.org/devstack/pike/>)에서 제시된 DevStack 다운로드까지의 일련의 과정을 수행할 것을 권장한다. 먼저 “sudo useradd -s /bin/bash -d /opt/stack -m stack” 명령을 통해 stack 사용자를 추가한다. 그 다음 “echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee

/etc/sudoers.d/stack" 명령을 통해 stack 사용자를 비밀번호 없이 sudo 명령이 사용하도록 등록한 뒤 "sudo su - stack" 명령어로 stack 사용자로 전환한다. 마지막으로 "git clone <https://git.openstack.org/openstack-dev/devstack>" 명령을 통해 DevStack을 /opt/stack 디렉토리에 다운로드받는다. 그 후 /opt/stack/devstack 디렉토리에 앞서 작성한 설정 파일을 local.conf로 이름을 바꿔 넣고 "./stack.sh" 명령을 통해 OpenStack 클러스터를 설치한다. 이 설치과정은 꽤 시간이 걸리며, OpenStack 클러스터를 멈출 때에는 "./unstack.sh", 완전히 삭제할 때는 "./clean.sh" 명령을 사용하면 된다. OpenStack Tricircle은 여러 OpenStack 클러스터간 네트워킹 자동화를 다루므로, 다른 노드에 'FLOATING_RANGE', 'FIXED_RANGE'를 다르게 설정하여 OpenStack 클러스터를 하나 더 구성한다.

- o 두 OpenStack 클러스터가 모두 정상적으로 동작한다면, 이제 OpenStack Tricircle을 실제로 검증하기 위해 OpenStack 명령어만을 활용해 두 클러스터간 네트워킹을 구성하고 실제로 두 클러스터마다 VM을 만들어 두 VM간 연결이 제대로 됐는지 테스트할 수 있다. 이를 위해 미리 작성한 'automate.sh' 스크립트 파일을 활용할 수 있다. 'automate_scrub.sh' 스크립트 파일은 'automate.sh' 스크립트의 동작을 역순으로 초기화하는 동작을 수행한다. 'automate.sh' 파일의 동작은 K-ONE 구글 드라이브에 업로드된 동영상을 통해 확인할 수 있다.
- o 'automate.sh' 명령어는 다음과 같은 행동을 한다. 먼저 OpenStack Neutron에 두 클러스터 및 중앙제어에 대응하는 RegionOne, RegionTwo, CentralRegion Region 객체를 등록한다. 그리고 RegionOne을 Availability Zone az1, RegionTwo를 Availability Zone az2에 등록한다. 그리고 RegionOne에 테넌트 네트워크 net1 객체를 만들고, RegionTwo에는 net2 객체를 만들고, 각각의 테넌트 네트워크에 대응되는 'FIXED_RANGE' 옵션에 주어졌던 IP 주소 대역을 가지는 서브넷 객체 subnet1, subnet2 객체를 만든다. 마지막으로 az1, az2에 모두 걸치는 사이트간 네트워크인 net3와 거기에 대응하는 subnet3를 VXLAN 타입으로 생성한다. 그 다음에는 두 클러스터의 Floating IP 주소 대역을 쓰는 외부 네트워크를 각 클러스터에 대응하는 ext-net1, ext-net2를 만들고, ext-net1과 subnet1, ext-net2와 subnet2를 연결하는 라우터 R1, R2가 각자 대응되는 클러스터에 생성된다.
- o 이제 두 클러스터의 내부 테넌트 네트워크 및 두 테넌트를 연결하는 사이트간 네트워크가 생성되었으니, 두 클러스터에 cirros VM을 생성한다. 이 때 두 VM은 각자 클러스터에 따라 net1, net2 네트워크, 그리고 공통으로 net3 네트워크를 할당되며 각 네트워크에 대응되는 인터페이스가 추가되어 각각 2개씩의 인터페이

스를 가진다. 마지막으로 스크립트에서 가장 중요한 부분은 Security Group 규칙을 업데이트하는 부분으로, TCP/UDP/ICMP에 대한 모든 룰 허용은 평범하지만 net3에 대응되는 Router 객체의 포트에 대해 "--allowed-address-pair ip_address=0.0.0.0/0" 옵션을 주어 업데이트한다. 이 설정 업데이트가 되지 않으면 net3 포트는 아무리 VM 내에서 라우팅 룰을 제대로 설정하여도 상대 VM을 net1 또는 net2 주소로 호출할 경우 net3 네트워크 상에 net3에 할당되지 않은 IP 주소를 가지는 패킷이 있음을 감지하고 패킷을 폐기한다. 따라서 정상적으로 호출이 되지 않게 된다. 위의 옵션은 해당 포트에 대해 모든 주소대역에 대한 패킷을 허용하도록 하여 이런 상황이 발생하지 않도록 한다.

- o 이후 스크립트에서는 두 cirros VM에 대해 인터페이스에 IP 주소 할당을 거치고 상대방의 net1, net2 주소를 통해 핑 테스트를 실시한다. 이 때 두 VM은 라우팅 테이블이 업데이트된 상태로 상대방의 net1, net2 주소에 대한 게이트웨이는 상대방의 net3 주소로 정해져 있다. 따라서 해당 핑 패킷은 net3 인터페이스를 통해 상대방으로 전달되고, 상대 VM의 핑 응답이 정상적으로 돌아오게 된다. 이를 통해 OpenStack Tricircle에 의해 net3 네트워크, subnet3 서브넷이 정상적으로 구축되었고 정상적인 통신이 가능함을 확인할 수 있다.

5. 결론

- o 본 기술문서를 통해 소프트웨어 정의 인프라 패러다임에 준비하기 위한 Edge-Cloud에 대응하는 테스트베드 모델인 K-Cluster을 설계에 대해 상세히 설명하였다. 그리고 설계에 따라 구축한 K-Cluster 하드웨어 시제품의 구성 및 상세 스펙에 대해 기술하였다.
- o 본 기술문서를 통해 K-ONE 컨소시엄이 목표로 하는 멀티사이트 SDN/NFV/Cloud 실증을 유연하고 효율적으로 지원하기 위한 K-ONE 공용개발 환경의 설계를 멀티사이트 관점에서 OpenStack Tricircle을 이용해 구축하는 방안을 제시하였다.
- o 제시한 멀티사이트 K-ONE 공용개발환경을 OpenStack Tricircle을 이용해 구축하는 방안을 DevStack을 활용해 부분적으로 검증하였다. 따라서 본 기술문서에서 제안한 K-Cluster/K-ONE 공용개발환경 모델에 따라 인프라 환경을 구축하면 현재 SDI 패러다임에 대응하는 다양한 기술을 적용할 수 있음을 확인할 수 있다.

References

- [1] OpenStack Tricircle, <https://wiki.openstack.org/wiki/Tricircle>
- [2] K-ONE 기술문서 #16 '멀티사이트 K-ONE Playground' 활용을 위한 K-DevOps Tower의 구축 및 활용 방법
- [3] IEEE 802.1Q standard 2005 version
<http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf>
- [4] Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks
<https://tools.ietf.org/html/rfc7348>
- [5] VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks <http://dx.doi.org/10.17487%2FRFC7348>
- [6] Virtual Extensible LAN (VXLAN) Overview - Arista
https://www.arista.com/assets/data/pdf/Whitepapers/Arista_Networks_VXLAN_White_Paper.pdf
- [7] Enhanced VXLAN: Who needs Multicast?
<https://adamraffe.com/2013/06/24/enhanced-vxlan-who-needs-multicast/>
- [8] Virtual Private Network: An Overview
<https://technet.microsoft.com/en-us/library/bb742566.aspx>
- [9] IP Encapsulating Security Payload (ESP) <https://tools.ietf.org/html/rfc2406>
- [10] OpenVPN Open-source Project
<https://openvpn.net/index.php/open-source.html>

K-ONE 기술 문서

- K-ONE 컨소시엄의 확인과 허가 없이 이 문서를 무단 수정하여 배포하는 것을 금지합니다.
- 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: 2018/02