KOREN SmartX Platform 환경에서 개발자 맞춤형 인프라를 제공하기 위한 인프라 슬라이싱 설계 및 검증

Document No. 1
Version 0.1
Date 2017-11-29
Author(s) GIST Team

■ 문서의 연혁

버전	날짜	작성자	비고
초안 - 0.1	2017. 09. 10	한정수, 김종원, 박선	
1.0	2017.11.29.	한정수, 김종원, 박선	

본 연구는 한국정보화진흥원(NIA)의 미래네트워크연구시험망(KOREN) 사업 지원과제의 연구결과로 수행되었음 (17-951-00-001)

This research was one of KOREN projects supported by National Information Society Agency (17-951-00-001)

Contents

#5. KOREN SmartX Platform 환경에서 개발자 맞춤형 인프라를 제공하기 위한 인프라 슬라이싱 설계 및 검증

그림 목차

#5. KOREN SmartX Platform 환경에서 개발자 맞춤형 인프라를 제공하기 위한 인프라 슬라이싱 설계 및 구현

1. KOREN SmartX Platform

1.1. 목적 및 개요

o 본 문서에서는 여러 사이트에 분산되어 있는 박스들을 활용하여 구성하고 있는 KOREN SmartX Platform에서 여러 개발자들이 공유되는 환경 속에서 개발자 맞춤형 인프라를 제공하기 위해서 인프라 (클라우드 및 네트워크)를 논리적으로 할당 하고 고립시켜 제공해주는 슬라이싱에 대해 설계 및 구현하고 검증하는 과정을 다루고 있다. 이러한 슬라이싱은 상부 플랫폼의 관할 센터 중 하나인 Orchestration 센터에서 API 형태로 개발자에게 제공해 손 쉽게 자신만의 인프라를 KOREN SmartX Platform 환경 상에서 구성할 수 있도록 지원해준다. 본 절에서는 KOREN SmartX Platform에 대해 먼저 설명을 하고, 다음절부터는 인프라 슬라이싱 개념 및 요구 사항, 그리고 디자인 설계에서부터 검증까지 내용을 다루고자 한다.

1.2. KOREN SmartX Playground

o Playground란 사용자들이 하고 싶은 실험 및 환경을 자유롭게 할 수 있게 제공되는 공간을 상징하고 사용자들이 Playground 위에서 하는 다양한 실증 실험들은 Play로 표현할 수 있다.

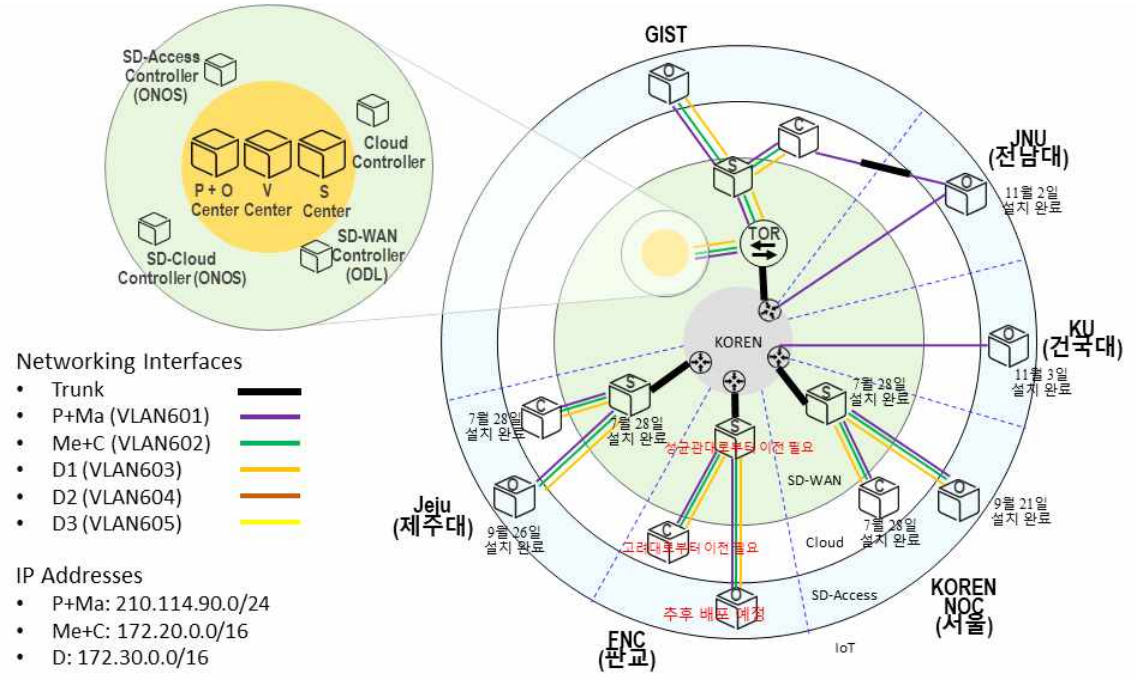

그림 1 KOREN SmartX Playground 및 관제타워

o KOREN SmartX Playground는 사용자들이 원하는 실증 실험을 할 수 있도록 다양한 역할을 지원하는 박스들을 유연하게 엮어 컴퓨팅, 네트워킹, 스토리지로 대표되는 IT 자원들을 알맞게 제공할 수 있는 물리적인 인프라를 의미한다. 현재 KOREN SmartX Playground를 구성하는 인프라는 크게 그림 ?과 같이 4가지 사이트로 구성되어 있다. GIST, KOREN NOC, 판교 FNC, Jeju 사이트로 구성된 인프라들을 구성하고 있는 Type O, Type C, Type S 박스들과 이를 관제하고 있는 센터 및 컨트롤러로 구성된다. 그리고 이러한 사이트들 사이의 내부형태를 보면, 각 벨트 별로 영역이 존재하며 (SD-WAN, Distributed Cloud, SD-Access) 자세한 설명은 아래와 같다.

o SD-WAN (Software-defined Wide Area Network: 소프트웨어-정의 광역 통신망)은 단일 사용자 집단에 소속된 분산된 사이트들을 연결하는 WAN을 소프트웨어-정의 기반으로 관제하는 방식이며, 고가의 단일 MPLS (Multi Protocol Label Switching) WAN에 의존하지 않고 저렴한 다수 WAN들을 통합하여 분산된 사이트들을 연결하는 사설 오버레이 네트워킹을 지원하는 영역을 말한다.

o Distributed Cloud (분산 클라우드) 는 단일 사용자 집단에 소속된 분산된 사이트들을 연결하는 통합적으로 제공하는 클라우드 영역을 말한다.

o SD-Access (Software-defined Access: 소프트웨어-정의 액세스) 는 분산된 IoT 디바이스들을 연결하면서, 다양한 통신 인터페이스를 사용하는 유/무선 IoT 네트워크들을 소프트웨어-정의 기반으로 관제하는 방식을 말한다. IoT 디바이스들과 근접한 위치에서 IoT-Cloud 형태로 분산 클라우드와의 안정된 연결을 지원하면서, SD-Access 대응 서버-스위치들의 계산/저장 자원들을 활용하여 IoT 데이터에 대한 전처리를 담당하는 영역을 말한다.

o 이러한 영역들은 각자의 역할을 수행하기 위해 필요한 기능들을 제공하고 있는 박스 (서버나 스위치가 어떠한 기능을 담고 있는 하드웨어적인 표현)가 필요하게 된다. 예를 들면, SD-WAN 영역에서는 분산된 사이트들을 연결하는 오버레이 네트워킹 및 스위칭 기능을 제공하는 박스가 필요하며, 분산 클라우드 영역에서는 클라우드 기능을 제공할 수 있는 박스가 필요하며, SD-Access 영역에서는 IoT 디바이스들이 연결될 수 있는 다양한 통신 인터페이스를 담고 있는 박스가 필요하다. 이렇게 각 영역에서 필요한 기능들을 담아서 제공하는 박스를 Type-S (SD-WAN 영역을 커버), Type-C (Cloud 영역을 커버), Type-O (SD-Access 영역

을 커버)라고 명칭하며 이러한 영역에 대응하는 박스들을 구성하고 있다.

o 이렇게 주어진 박스들을 한 단계 위에 레벨에서 관장할 수 있는 요소가 필요하다. 이러한 역할을 하기 위해 P+O (Provisioning & Orchestration) 센터와 Visibility 센터 그리고 Cloud 및 SDN 제어기들이 이러한 요소를 담당하고 있다. P+O Center에서는 분산된 사이트들에 박스들을 원격으로 세팅 및 준비하며 개발자들이 사용할 수 있는 여러 기능들을 제공하는 역할을 하며, Visibility 센터는 이러한 인프라들의 상태를 가시적으로 파악할 수 있도록 가시화 정보를 수집하고 이러한 데이터들을 가시화할 수 있는 데이터로 변환해 운영자에게 제공해주는 역할을 한다. 그리고 Cloud 및 SDN 제어기들은 실제적으로 이러한 센터들이 시키는 주요한 일들을 직접 처리하는 역할을 하며, Cloud 및 SDN에 관련된 전반적인 일을 센터들의 지시를 받아 수행하는 역할을 한다.

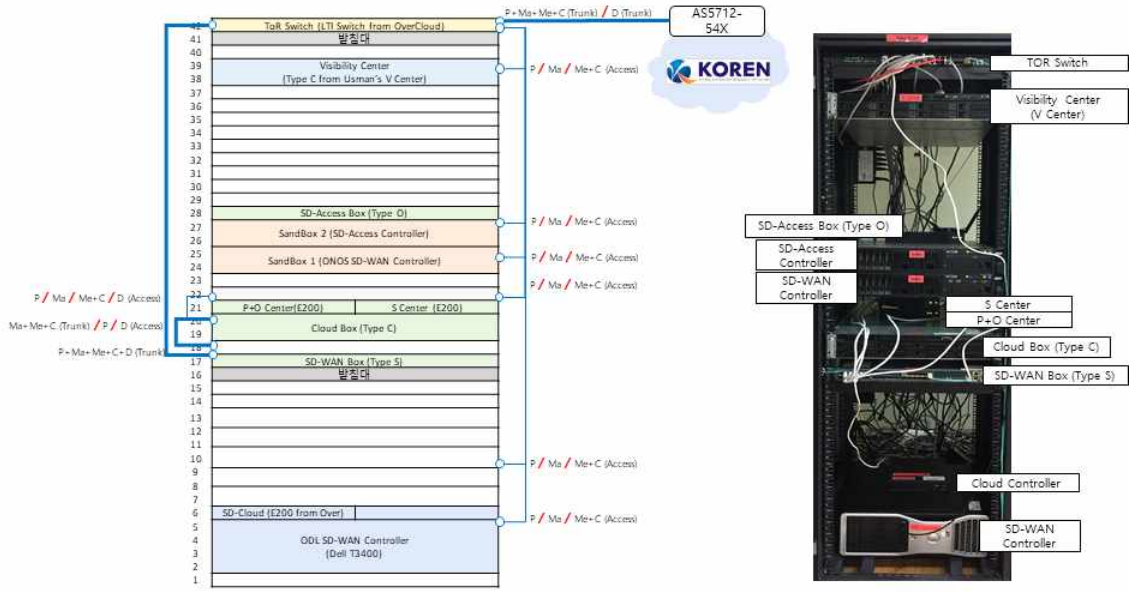

그림 2 GIST KOREN Rack 및 구조

o GIST KOREN Rack에서는 GIST 내부의 관제타워 및 컨트롤 노드를 포함한 인프라들로 구성되어 있다. 위에서 제시한 여러 유형의 박스들을 잘 제어하기 위해서 Cloud 제어 노드, SD-WAN 네트워크 제어 노드, SD-Access 네트워크 제어기노드 등이 포함되어 있다. 또한 이런 제어기들을 관장하면서 직접 일을 시키는 센터들도 Rack에서 포함이 되어있다. Type S, C, O는 한 세트로 관리가 되면서 Type S가 스위치 역할을 할 수 있기 때문에, ToR 스위치 역할을 하면서 Type C와 Type O를 지원한다.

o Rack에 구성된 박스들은 기본적으로 여러 개의 네트워크 인터페이스를 활용해

유기적으로 연결하고 제어할 수 있는 기반을 마련하고 있다. 원격지에서 박스들을 대상으로 접속하고 제어하기 위한 P (Power) 네트워크, 박스들을 관리하거나 패키지들을 받는 용도의 M (Management) 네트워크, 박스 간의 제어 메시지를 위한 C (Control) 네트워크, 가상머신이나 서비스 간의 데이터 이동 트래픽을 위한 D (Data) 네트워크로 구성되어 있다. 다양한 용도에 따른 복수 개의 네트워크 인터페이스를 활용함으로써 보다 트래픽들을 분리시키고, 문제 발생 시 빠르고 즉각적인 대응이 가능할 수 있다.

1.3. KOREN SmartX platform으로의 확장 및 개선

o KOREN SmartX Playground에서 다양한 개발자들이 자신의 실험들을 실증하기 위한 인프라는 구축이 됐으나, 개발자들이 손쉽게 KOREN SmartX Playground를 사용할 수 있도록 Platform 차원에서 인터페이싱 역할이 부족하다.

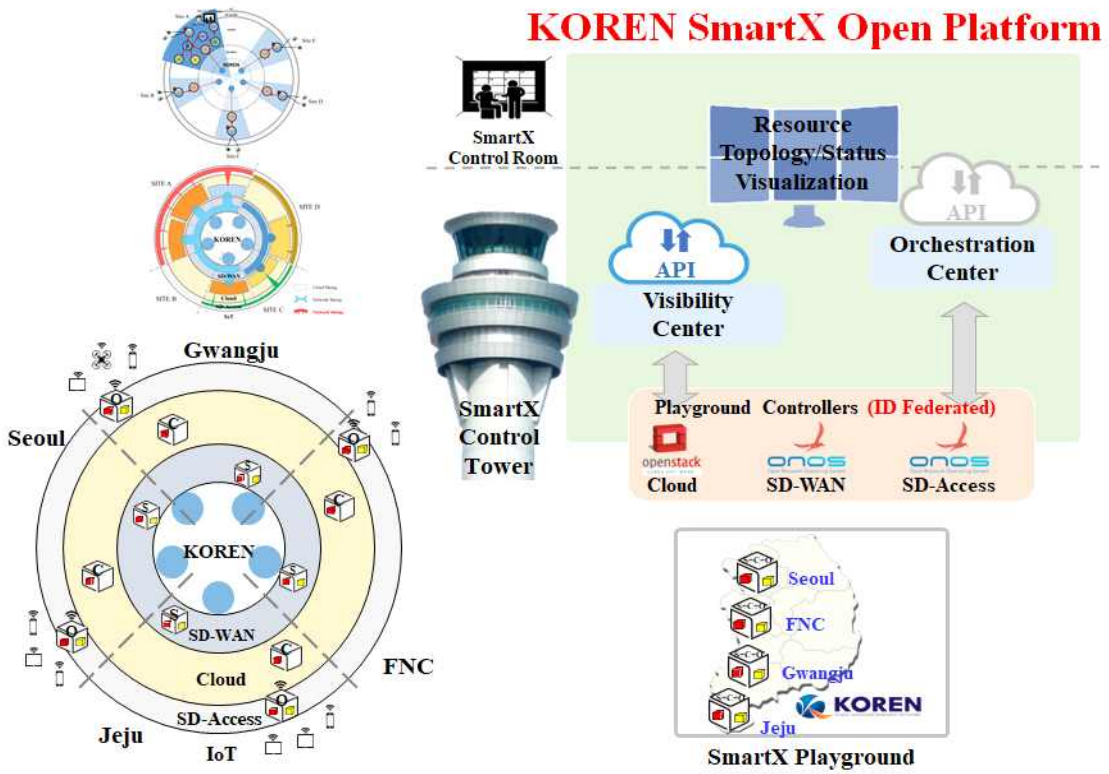

그림 3 KOREN SmartX Open Platform 개념도

o 이를 위해 개발자가 API 형태로 사용가능 할 수 있도록 지원하는 센터들을 구축하고, 초기 단계에서는 개발자에게 Orchestration 기능과 Visibility 기능을 제공하는 센터들을 시작으로 향후에 점차 확장되는 형태로 개선하도록 한다. Visibility Center에서 제공하는 API는 현재 개발자가 가지고 있는 인프라에 대한 전체적인

상황을 파악할 수 있는 데이터들을 가져오고 시각화해주는 기능을 제공해주고, P+O Center에서는 개발자들이 자신만의 독립적인 인프라를 가지기 위해서 클라우드 및 네트워크를 할당해주는 기능을 제공하도록 한다. 이러한 기능들을 통해 개발자들은 좀 더 손쉽게 KOREN SmartX Playground를 활용할 수 있도록 인터페이싱 역할이 가능하게끔 개선 및 확장하는 형태로 KOREN SmartX Platform을 구축한다.

2. 개발자 맞춤형 인프라 슬라이싱

2.1. 인프라 슬라이싱 개요

o 국내 여러 사이트의 분산형 Box를 활용하여 구성된 KOREN SmartX Platform 환경에서 구성된 인프라를 모든 개발자들이 활용하는 공유 형태이지만, 사용자 별로 마치 자신만의 독립적인 인프라를 사용하는 것처럼 느낄 수 있도록 인프라를 나누고, 고립시키고, 할당하고 관리하는 기술이 필요하다. 이러한 인프라 즉, 클라우드 및 네트워크를 논리적으로 나누고, 할당하고 관리하는 기술 형태를 인프라 슬라이싱이라고 지칭한다.

2.2. 인프라 슬라이싱 요구사항

o 여러 사이트에 분산되어 있는 KOREN SmartX Platform 환경에서 인프라 슬라이싱은 개발자들이 Cloud 뿐만 아니라 IoT 기기들까지도 연결되는 서비스들을 고려한 IoT-Cloud 서비스에 대응하기 위해서는 다음과 같은 요구사항을 포함한다.

o 첫 번째로 인프라 슬라이싱의 범위는 클라우드에서 제공하는 컴퓨팅 스토리지 뿐만 아니라 IoT 기기 까지 확장된 네트워킹을 전제로 한다. 클라우드 같은 경우 컴퓨팅 및 스토리지 그리고 부분적인 네트워킹을 사용자 테넌트 별로 할당하고 관리할 수 있는 기술을 제공하고 있으나, 이는 클라우드로 묶인 인프라에 한정해서 다루고 있으며, IoT-Cloud 까지 확장하는 서비스에 대응할 경우에는 인프라를 제공하는 부분에 있어서 제한적일 수밖에 없다. 즉 Cloud 범위 관할을 벗어나는 IoT 기기들의 경우 네트워킹 인프라에 관해서 개발자에게 슬라이싱 형태로 제공해주는 것이 제한적이다. 따라서 IoT-Cloud 까지 확장된 인프라를 테넌트 별로 할당하고 관리할 수 있는 기술이 필요하다.

o 두 번 째로, KOREN SmartX Platform을 사용하는 개발자들이 손쉽게 사용하면서 자신의 서비스들을 실증할 수 있도록 API 형태로 제공해야 한다. API 형태로 제공되면, 개발자들은 내부적인 기술적인 수준을 몰라도 손쉽게 원하는 클라우드 인프라의 형태와 연결할 IoT 디바이스들을 파라미터로 입력해 클라우드 리소스를

제공받고 IoT 기기들과의 네트워킹 연결을 할 수 있게 된다.

o 마지막으로 각 사용자들이 할당한 슬라이싱은 독립적이며, 다른 영역을 침범해선 안 되며, 고유의 할당 영역을 보장받아야 한다. 즉 사용자 별로 할당된 인프라 슬라이싱은 서로의 존재를 모르게끔 고립된 (Isolated) 형태로 존재해야 하며, 이들이 다른 영역을 침범해서는 안 되도록 보장해야 한다. 이를 위해서는 테넌트별로 고유한 아이디와 이와 연결되는 클라우드 리소스 및 네트워크 리소스를 관리해야하며, 이러한 테넌트를 지원할 수 있는 기술적인 요소들을 활용해야 한다.

3. 개발자 맞춤형 인프라 슬라이싱 설계

3.1. 개발자 맞춤형 인프라 슬라이싱 설계

o KOREN SmartX Platform 환경에서 개발자 맞춤형 인프라 슬라이싱을 위해 다음 그림과 같은 구도로 인프라 슬라이싱을 디자인 하였다.

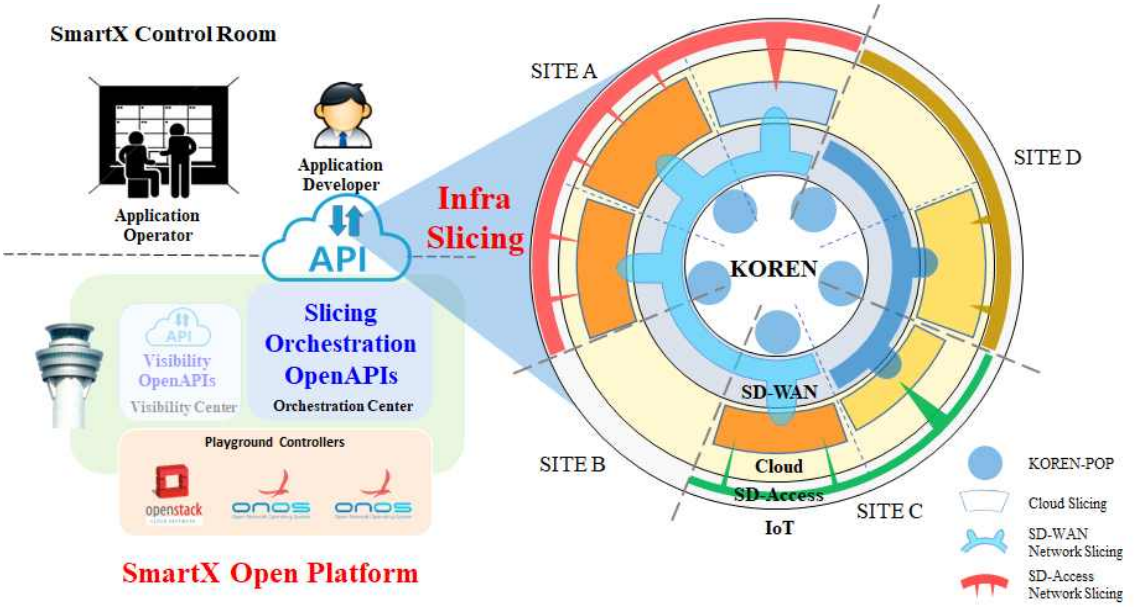

그림 4 개발자 맞춤형 인프라 슬라이싱의 설계

o KOREN SmartX Platform의 Orchestration Center에서는 개발자들이 사용할 수 있는 슬라이싱 API가 지원된다. 인프라를 구성하고 있는 영역 벨트 별로 SD-WAN, Cloud, SD-Access와 같이 세분화된 영역에 대한 슬라이싱을 단계적으로 완성하고, 이들을 엮어서 개발자가 사용할 수 있는 최종적인 형태의 API를 제공해주는 것을 목표로 하고 있다. 즉 SD-WAN 슬라이싱은 사이트와 사이트를 넘어가는 네트워킹을 중심으로 슬라이싱을 하는 것을 말하며, 클라우드 슬라이싱은 주로 클라우드 자원을 중심으로 슬라이싱 하는 기술을 말한다. 마지막으로 SD-Access 슬라이싱은 IoT 기기들과 클라우드 자원과의 네트워크 연결 중심의 슬라이싱을 말하며, 각자의 영역에서 필요한 클라우드 SDN 제어기들을 활용해 슬라이싱을 기술적으로 진행하는 역할을 한다.

o 다음 그림은 2절에서 위와 같은 슬라이싱 구현을 위해 실제적인 인프라의 네트워크 구성을 어떻게 하였는지를 나타내는 그림이다.

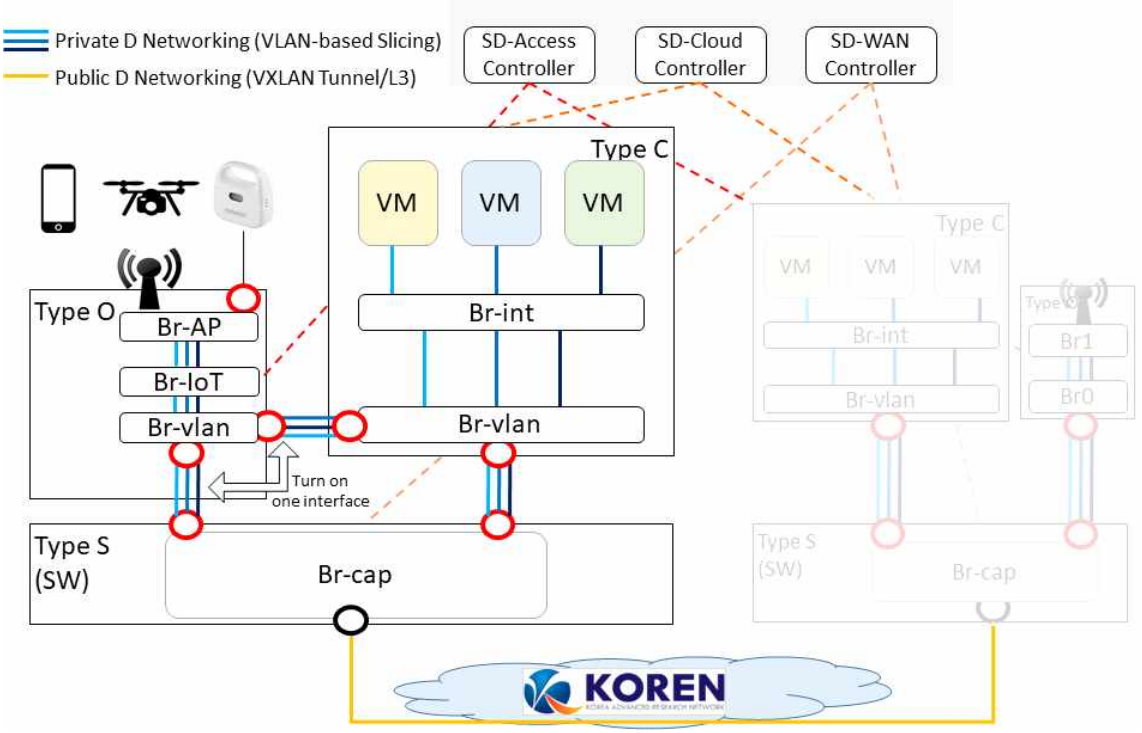

그림 5 인프라 슬라이싱을 위한 네트워킹 설계도

o 각 사이트 별로 Type S, Type C, Type O에 대해서 내부적으로 네트워크를 어떻게 구성하는지에 대해서 설계하였다. Type C 박스 같은 경우 Cloud OS인 오픈스택을 활용하여 컴퓨팅 리소스와 부분적인 네트워킹에 대해서 지원을 받는 형태로 OVS를 구성하였다. 즉 여기서 만들어진 Br-int와 Br-vlan OVS 브릿지는 오픈스택의 Neutron이 관할하게 되며, Neutron에서 테넌트를 위한 네트워킹 형태는 VLAN 네트워크로 구성하였다. 즉 테넌트 네트워킹 별로 VLAN ID가 태깅이 되면서 Type S의 박스로 나가는 형태로 구성하고 있다.

o Type O 박스의 경우, 무선과 유선을 관리할 수 있는 가상 스위치가 필요하며, 이 가상 스위치는 VLAN ID를 제어가 가능해야 한다. 즉, 테넌트 간의 슬라이싱을 지원하기 위해서는 가상 스위치가 IoT 디바이스의 패킷들을 받아서, 적절하게 VLAN 태깅을 거친 후, 연결된 Type S에게 전달되도록 구성해야 한다. 따라서 무선을 받는 가상스위치 Br-AP와 무선 및 유선을 통합해서 받는 가상 스위치 Br-IoT를 만들고, 그리고 이러한 통합 Br-IoT 가상 스위치에서 나가는 모든 패킷들을 확인하고 VLAN 태깅을 해주는 Br-vlan 가상 스위치를 추가적으로 구성을 하였다.

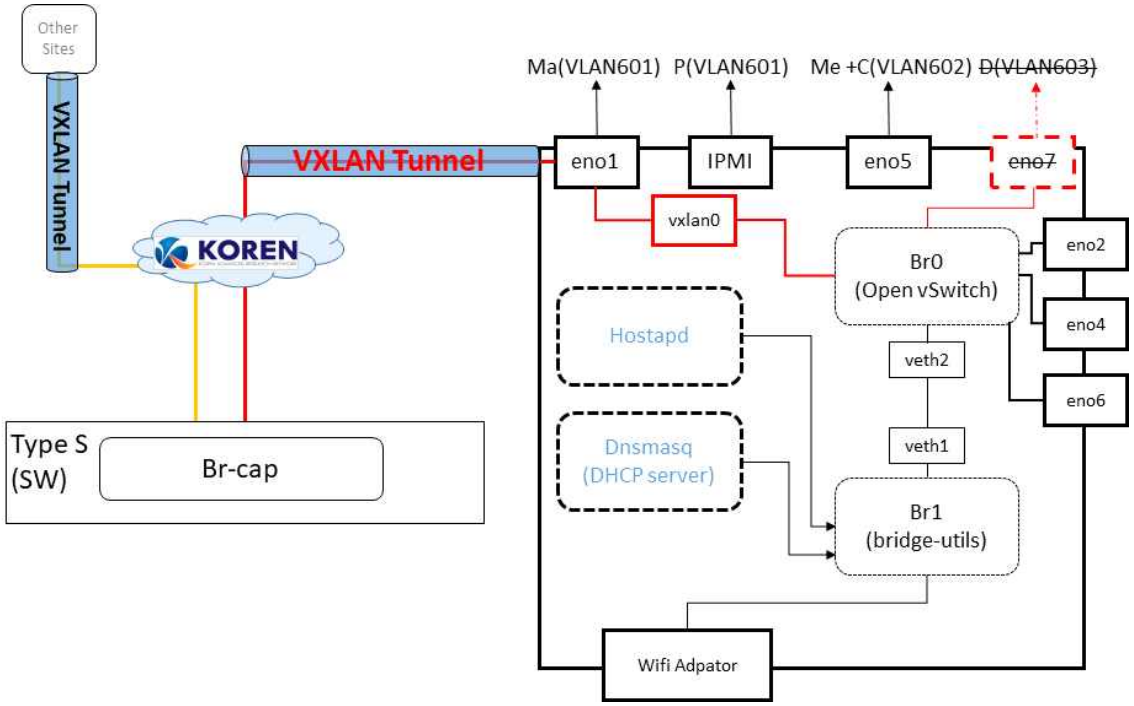

그림 6 Type O'의 구조 및 Type S와의 네트워킹 연결 구성도

o 마지막으로 Type S 박스는 Type C와 Type O에서 태깅 되어서 들어오는 포트를 OVS로 묶어서 서로를 연결해주는 역할을 하며 또한 다른 사이트로 패킷을 전달하기 위해서 터널링 기법인 VXLAN 기술을 활용해서 다른 사이트로 패킷을 전달하는 역할을 하게 된다.

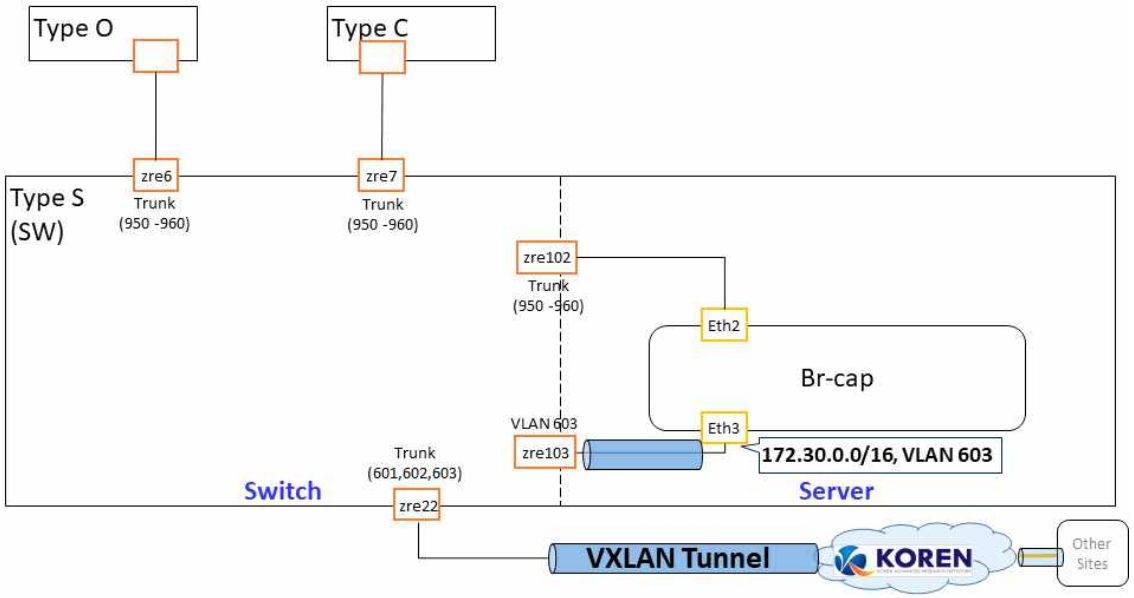

그림 7 Type S 스위치-박스의 내부적인 네트워크 구성도

o 각 사이트에 위치한 Type S 스위치-박스는 터널링 형태로 서로 연결이 가능하도록 구성하였으며, Type S 스위치-박스의 컴퓨팅 파트에 가상 스위치 Br-cap를 구성하고, 이를 통해 다른 사이트에 위치한 Type S 스위치와 터널링으로 연결이 된다. Type S 스위치-박스의 스위치 파트에서는 Type C, Type O가 직접적으로 연결이 돼 있으며, Type S가 없는 분산된 사이트에 위치한 Type O'는 GIST에 위치한 Type S에 직접적인 터널링 연결을 통해 분산된 사이트 간 연동을 완료하였다.

o 다음 그림 7은 슬라이싱 ID 및 다른 엔티티와의 관계도를 정리한 그림이다. 2절에 슬라이싱 요구사항에서도 언급되었듯이, 각 테넌트별로 슬라이싱을 지원하기 위한 ID들이 정리가 되어야 한다. 즉 슬라이싱 ID 중심으로 다른 요소들이 적절하게 연결이 된 관계를 관리 및 유지해야 한다. 슬라이싱 ID에 연결되는 다른 요소들은 그림과 같이 테넌트 (유저), SDN 제어기, VLAN ID, IoT 기기, VM 인스턴스 들이 있다. 즉 어떤 한 슬라이싱 ID에 연결된 테넌트는 누구이며, 어떤 VM 인스턴스와 어떤 IoT 기기가 연결이 되어 있으며, 이 슬라이싱 ID는 SDN 제어기의 권한은 어떻게 구성되어 있으며, VLAN ID는 어떤 것을 사용하는지를 관리하고 있어야 한다. 여기서 Tenant ID를 위해 오픈스택의 Keystone 프로젝트에서 제공하는 Tenant ID를 그대로 사용하면서, 각 슬라이싱과 관련된 엔티티와의 관계 정보를 관계형 DB로 관리하는 형태로 설계하였다. 즉, 슬라이싱과 관련된 여섯 개의 엔티티와의 관계 DB 테이블을 생성하였다.

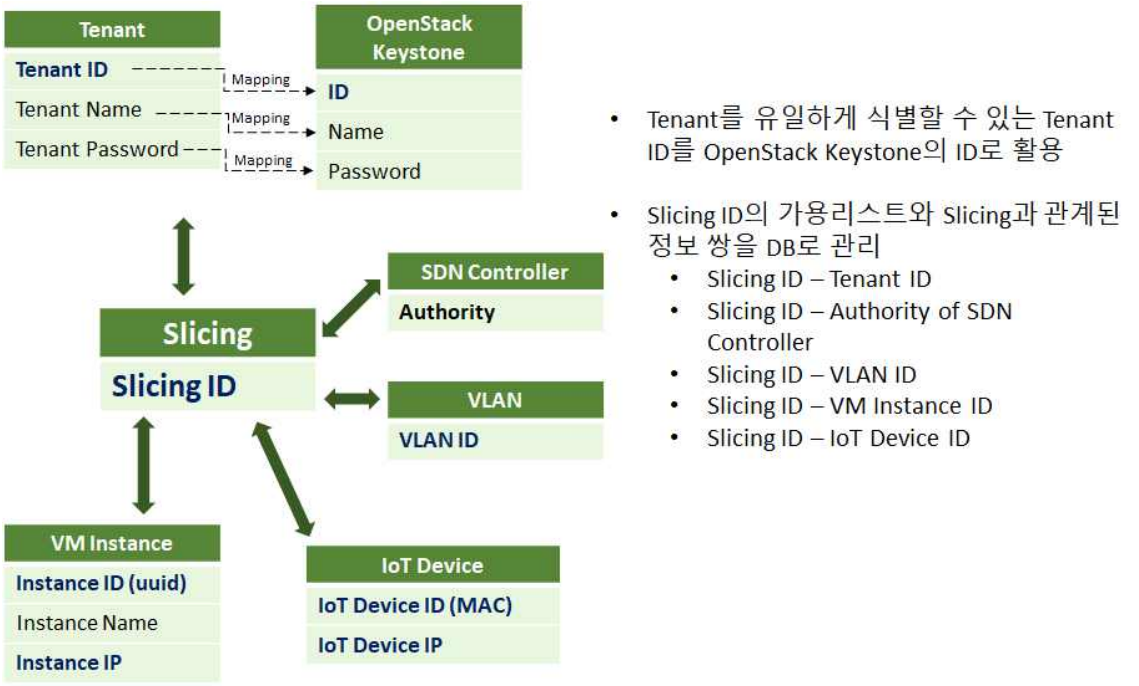

그림 8 슬라이싱 ID 및 다른 엔티티와의 관계도

```
CREATE TABLE Slicing (
Slicing_ID VARCHAR(100) NOT NULL,
Tenant_ID VARCHAR(100) Not NULL,
Authority VARCHAR(100) Not NULL,
VLAN_ID VARCHAR (100) Not NULL,
PRIMARY KEY(Slicing_ID)
);

CREATE TABLE Instance (
Instance_ID VARCHAR (100) NOT NULL,
Instance_Name VARCHAR (100) NOT NULL,
IP VARCHAR (100) NOT NULL,
PRIMARY KEY(Instance_ID)
);

CREATE TABLE IoT (
MAC VARCHAR (100) NOT NULL,
IP VARCHAR (100) NOT NULL,
PRIMARY KEY(MAC)
);

CREATE TABLE Slicing_Instance (
Slicing_ID VARCHAR (100) NOT NULL,
Instance_ID VARCHAR (100) NOT NULL,
PRIMARY KEY (Slicing_ID, Instance_ID),
FOREIGN KEY (Slicing_ID) references Slicing(Slicing_ID),
FOREIGN KEY (Instance_ID) references Instance(Instance_ID)
);

CREATE TABLE Slicing_IoT (
Slicing_ID VARCHAR (100) NOT NULL,
MAC VARCHAR (100) NOT NULL,
PRIMARY KEY (Slicing_ID, MAC),
FOREIGN KEY (Slicing_ID) references Slicing(Slicing_ID),
FOREIGN KEY (MAC) references IoT(MAC)
);
```

그림 9 슬라이싱 관리 DB 설계

o 슬라이싱이 VLAN 기술을 기반으로 만들기 때문에 Slicing ID 별 가용 네트워크를 미리 사용자들과 약속하는 것이 필요하다. 따라서 다음과 같이 VLAN (951~960)에 대한 가용 네트워크 리스트를 미리 지정하였다.

- 951 : 10.10.10.0/24
- 952 : 10.10.20.0/24
- 953 : 10.10.30.0/24
- 954 : 10.10.40.0/24
- 955 : 10.10.50.0/24
- 956 : 10.10.60.0/24
- 957 : 10.10.70.0/24
- 958 : 10.10.80.0/24
- 959 : 10.10.90.0/24
- 960 : 10.10.100.0/24

그림 10 VLAN 별 가용
네트워크 리스트

o 위에서 설계한 슬라이싱 ID와 주변 관계 요소들을 관리하기 위한 부분과 각각의 영역 별 슬라이싱 (SD-WAN/Cloud/SD-Access)를 연결해주는 오케스트레이션 부분이 필요하다. 따라서 SD-WAN 슬라이싱, Cloud 슬라이싱, SD-Access 슬라이싱

과 이들을 엮고 슬라이싱 ID와 주변 요소들의 관계를 정리하는 슬라이싱 통합 이렇게 네 가지로 세분화해서 인프라 슬라이싱을 정리하는 형태로 소프트웨어를 디자인하였으며, 전체적인 인프라 슬라이싱의 워크플로우 순서도는 다음 그림과 같다.

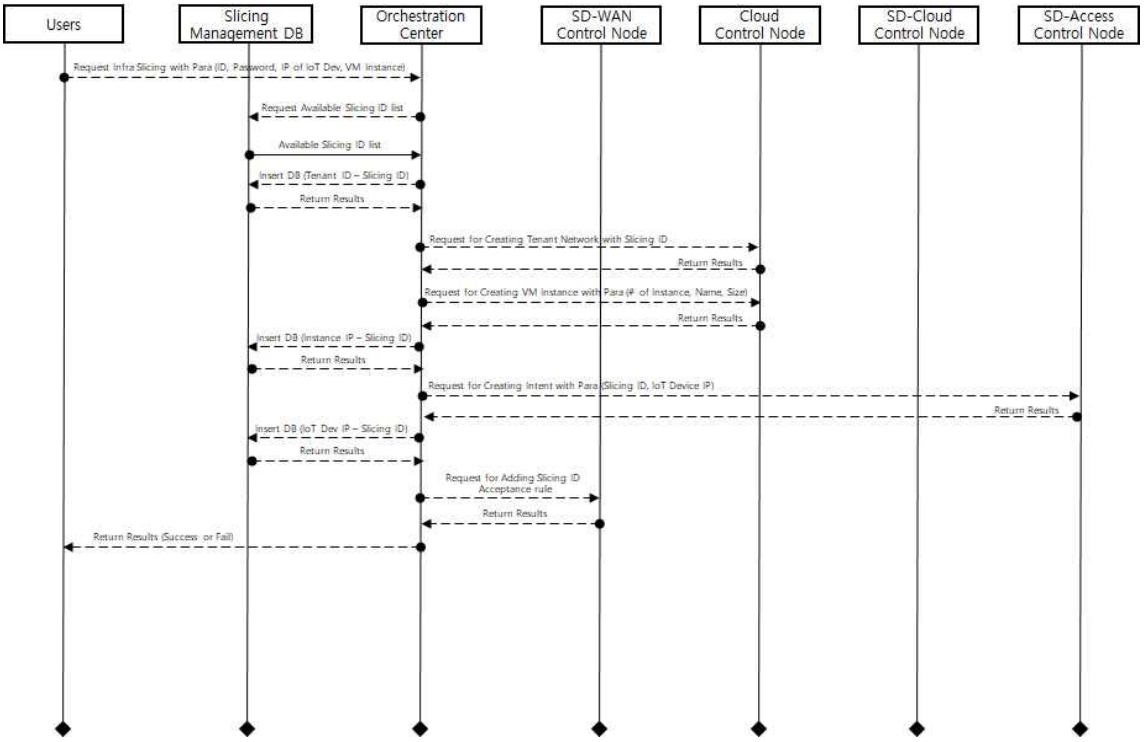

그림 11 인프라 슬라이싱에 대한 전체적인 워크플로우 순서도

o 초기 설계 시에는 통합적인 인프라 슬라이싱을 제공하는 하나의 API를 사용자가 콜하는 형태로 구축을 진행하였으나, 이렇게 된다면, 사용자별로 슬라이싱을 유연하게 연결하고 관리해주는 부분이 결여되기 때문에, 아래와 같이 API들을 부분적으로 나누어서 정리하였다.

Slicing_Create (Tenant ID, Password)
Cloud_Slicing (Tenant ID, Password, Slicing ID, # of VM, Flavor, Image)
SD-Access_Slicing (Tenant ID, Password, Slicing ID, IoT MAC)
Slicing_Delete (Tenant ID, Password, Slicing ID)
Slicing_List (Tenant ID, Password)
Slicing_Show (Tenant ID, Password, Slicing ID)

3.2. KOREN SmartX Platform 상의 인프라 슬라이싱 검증 및 활용

o 본 절에서는 3장에서 설계한 인프라 슬라이싱을 구현하고, 이를 실제 KOREN SmartX Playground 환경에서 검증하는 것을 보인다. 최종 사용자들이 자신들의 IoT-Cloud 서비스를 수행하기 위해서는 IoT 기기와 Cloud 가상 머신간의 네트워크 적으로 연결성을 보장받아야 한다. 이를 위해서 구현한 슬라이싱을 최종사용자가 아래 그림과 같은 순서로 API를 활용해야 한다.

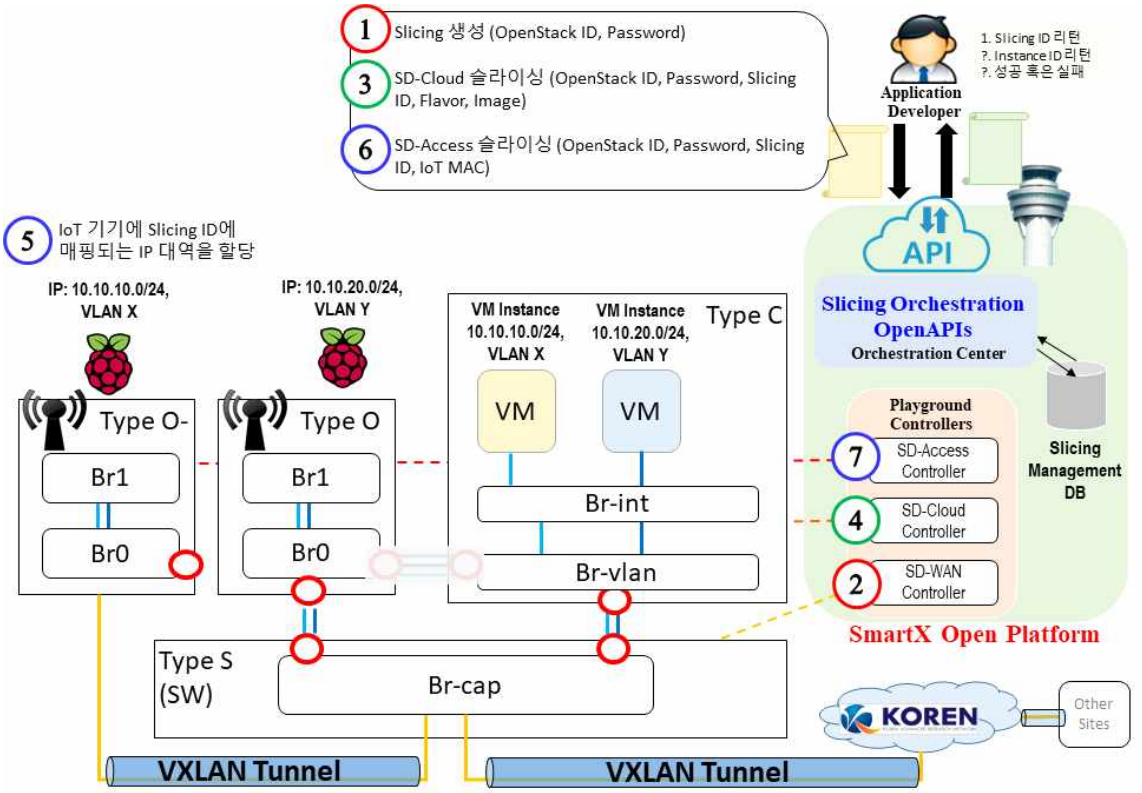

그림 12 인프라 슬라이싱 수행을 위한 시나리오 구성

o 첫 번째로는 사용자가 자신의 Slicing ID를 할당받기 위해서 Slicing 생성 API를 호출해야 한다. Slicing_ID를 호출하면 자신의 ID와 Password를 입력해야하고, 인증이 정상적으로 완료가 된다면, 다음 그림과 같이 자신의 슬라이싱 ID가 생성이 된다.

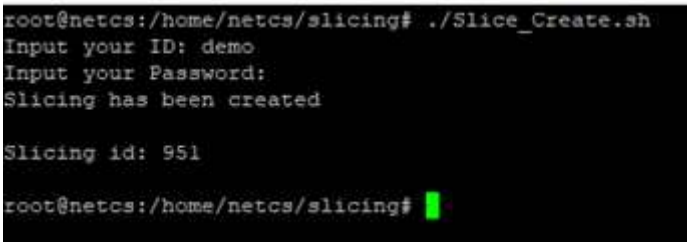

그림 13 Slicing_Create 호출 화면

o 위 그림에서 demo라는 사용자가 Slicing ID 951을 할당 받는 장면이다. demo라는 사용자는 오픈스택의 Keystone의 등록된 사용자이며, Password를 입력받으면 사용자의 인증과정이 오픈스택의 Keystone을 통해 이루어지는 것을 볼 수 있다.

o 그리고 1번 과정이 진행이 되면, 각 사이트를 연결해주는 SD-WAN 영역을 위한 길을 열어주게 된다. 즉, 각 분산된 사이트의 Type S 서버-스위치 박스에서 해당 VLAN ID에 대해서 터널간의 트래픽 허용을 위한 플로우 룰이 내려지게 된다.

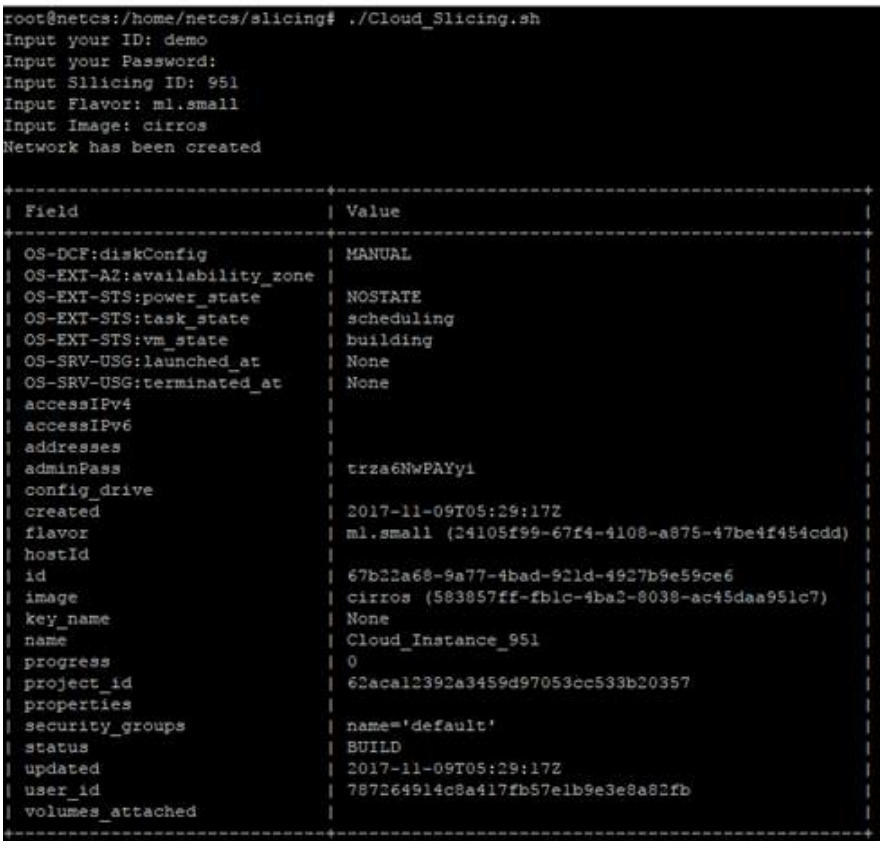

그림 14 Cloud_Slicing 호출 화면

o 두 번 째 과정으로 SD-Cloud 슬라이싱을 진행한다. 이 과정에서는 Slicing ID를

갖는 가상 머신을 생성하는데, 가상 머신에 대한 크기 및 이미지를 설정하고, 자신의 ID 및 Slicing ID를 입력하게 되면 가상머신 생성이 진행이 된다. 이 때, Slicing ID와 같은 VLAN ID를 갖는 테넌트 네트워크가 오픈스택 Neutron 서비스를 통해 생성이 되며, 가상머신에 이 네트워크를 연결하는 과정을 거치게 된다.

o 그림 14를 보면, 사용자 ID, Password를 입력받고, 인증이 완료되면 Slicing ID를 입력받는다. 해당 Slicing ID가 사용자 ID가 갖고있는 것이 정상적으로 확인이 되면, 가상 인스턴스를 만들기 위해서 필요한 정보들을 (Flavor, Image) 입력한다. 입력이 완료가 되면, 그림 14 하단에 결과 창을 볼 수 있으며, Cloud_Instance_951이라는 이름으로 가상머신이 생성이 되었고, 오픈스택의 대쉬보드에서 해당 사용자 ID로 로그인을 하면 생성된 가상머신을 확인이 가능하다.

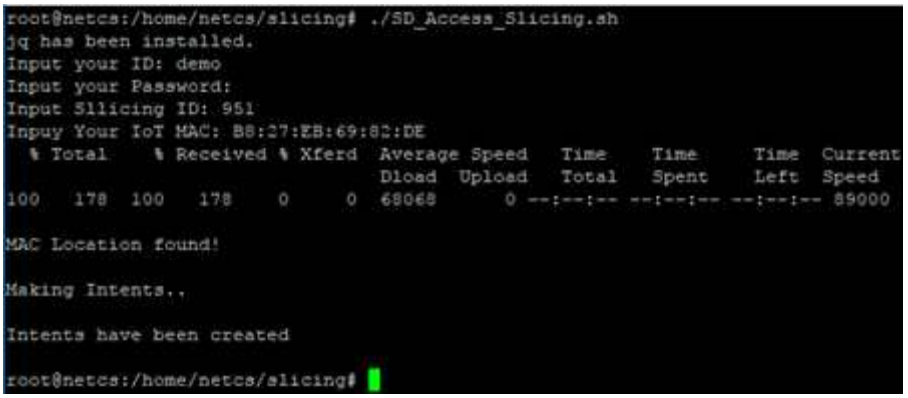

그림 15 SD_Access_Slicing 호출 화면

o 마지막으로 사용자는 자신의 IoT 기기들을 Slicing 으로 연결하기 위한 작업을 진행한다. 이 때, 사용자는 미리 자신이 할당받은 Slicing ID에 맞는 네트워크 대역대로 기기의 IP를 설정하여야 한다. 그 이후 SD_Access_Slicing을 호출하면, 인증작업이 끝나고, 등록하려는 MAC 주소에 대한 슬라이싱을 진행한다.

o 그림 15는 사용자 Demo가 자신이 할당받은 Slicing ID 951번에 IoT 기기 MAC 주소 B8:27:EB:69:82:DE를 등록하는 과정을 나타내고 있다. 입력받은 MAC 주소가 인프라를 관장하고 있는 범위 내에 있는 지 체크 하는 작업을 거친 후, MAC의 위치를 알아내고, 이를 기반으로 ONOS의 Intent가 설치가 진행이 된다. Intent는 한 기기 당 양방향의 쌍으로 진행이 된다. IoT 기기에서 나가는 패킷의 경우, Slicing ID에 맞는 VLAN을 태깅하는 플로우 룰을 내려주는 Intent를 설치하며, IoT 기기로 들어가는 패킷에 대해서는 태깅된 VLAN ID를 벗겨내는 플로우 룰을 내려주는 Intent를 설치하게 된다. 이러한 작업이 끝나면, 사설 IP를 가진 IoT 기기들과 가상머신들이 서로간의 통신이 가능하며, 사용자의 IoT 기기가 어

떤 위치에 있든 간에, Type O Access 박스를 통해 연결되면, 서로간의 통신이 가능하다.

o 이러한 세 단계의 과정이 모두 끝나면 사용자가 자신의 서비스를 구동하기 위한 인프라 및 네트워크가 준비가 된 것이다. 이후에는 적절한 서비스를 선택해 서비스 합성을 수행시키면 된다. 슬라이싱의 검증을 위해서 Smart Air IoT-Cloud 서비스를 KOREN SmartX Platform 상에 검증해보았다. 이 서비스는 지리적으로 분산되어 있는 몇 개의 사이트에 위치한 IoT 디바이스에 센서들을 붙여서 먼지 값들을 측정하고, 드론이나 핸드폰을 통한 비디오 스트리밍으로 실시간으로 상황을 파악 가능하게 하는 서비스이다. 이렇게 생성된 데이터들은 구축된 SmartX Platform 상의 클라우드를 통해 만들어진 가상머신으로 들어가게 되고 최종적으로 통합 대쉬보드를 통해 그림 16과 같이 전체적인 상황을 파악할 수 있다.

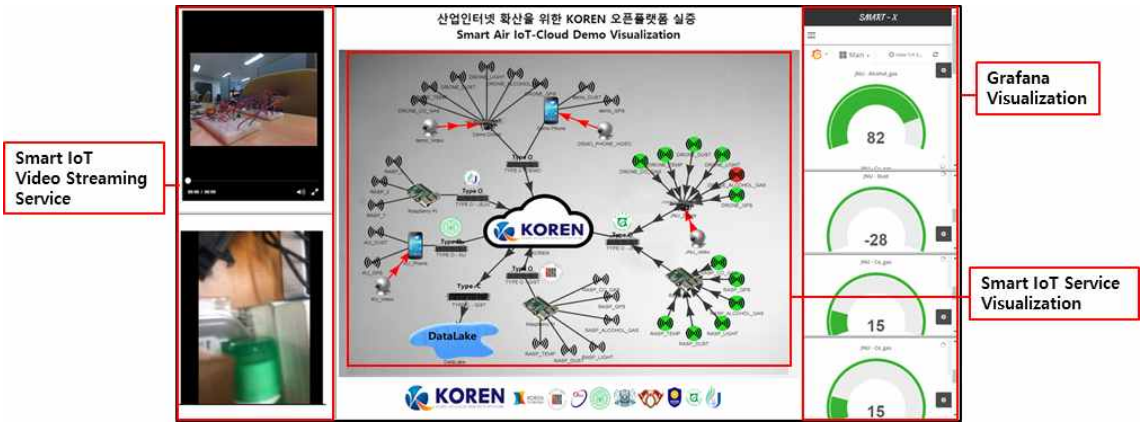

그림 16 Smart Air IoT-Cloud 서비스 통합 시연 데모

o 본 시연을 위해 하나의 슬라이싱 ID를 만들고, 하나의 가상머신 그리고 여러 IoT 디바이스들에 대해 슬라이싱을 진행하였다. 인프라 슬라이싱을 통해, 각 IoT 기기들은 Private IP 범위를 가지고 있지만 지리적으로 떨어져 있어서 정상적으로는 접근이 안되는 GIST에 위치한 가상머신 인스턴스에게 접근하는게 가능한 것을 볼 수 있었고, 이를 통해 위의 서비스가 정상적으로 동작하는 것을 확인하였다.

References

SmartX 기술 문서

- 본 문서와 관련된 대한 문의 사항은 아래의 정보를 참조하시길 바랍니다. (Homepage: https://nm.gist.ac.kr, E-mail: ops@smartx.kr)

작성기관: 광주과학기술원
작성년월: 2017/11