https://doi.org/10.22670/knom.2016.19.1.1

K-Slice : SDN/SFA 기반 테스트베드 네트워크 연동 시스템

김 성 민◆, 심 규 석*, Baraka D. Sija**, 김 명 섭°

K-Slice : SDN/SFA-based Testbeds' Network Federation System

Sung-Min Kim◆, Kyu-Seok Sim, Baraka D. Sija, Myung-Sup Kim°

요 약

전 세계적으로 네트워크를 활용한 서비스가 다양해지고 그 중요도가 커짐에 따라 안정적이며, 고성능의 컴퓨팅 네트워크 환경이 요구되고 있다. 하지만 그러한 고성능의 장비는 대부분이 고가이기 때문에 개인 또는 소규모 연구진이 구축하기에는 여러가지 측면에서 제약사항이 발생하게 된다. 이러한 제약사항을 해결하기 위하여 세계 각국의 연구 기관에서는 다양한 플랫폼의 테스트베드를 구축하였으며, 대규모 실험 환경을 구축을 위하여 각 테스트베드를 연동하는 연구가 활발히 진행이 되었다. 하지만 현재까지의 테스트베드 연동은 사용자의 요구에 맞춰 네트워크 환경을 동적으로 설정하지 못하는 한계점이 여전히 존재한다. 따라서 본 논문에서는 다수의 테스트베드와 SDN기술을 연동하여 연구자에게 동적인 네트워크 자원을 제공할 수 있는 시스템을 구축하였으며 그 구조를 제안한다.

Key Words : Future Internet, Testbeds Federation, SFA, MySlice, SDN

ABSTRACT

As the services utilizing the network are becoming more diverse and more important in the world, a stable and high performance computing network environment is required. However, most of these high performance equipment are expensive, so there are limitations on how to build an personal or small scale research team. In order to solve these limitations, research institutes around the world have constructed test beds of various platforms, and studies for linking each test bed have been actively conducted in order to construct a large scale experimental environment. However, until now, there is a limitation that the testbed Federating can not dynamically set the network environment according to the user's demand. In this paper, we propose a system that can provide dynamic network resources to researchers by linking multiple testbeds with SDN.

※ 이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구(No.2015R1D1A3A01018057) 및 2016년 한국과학기술정보연구원 연구과제(첨단연구망기반 협업플랫폼 서비스 및 글로벌 연동, K-16-L01-C02-S03) 지원으로 수행됨.
◆ First Author : Dept. of Computer and Information Science, Korea University. gogumiking@korea.ac.kr
° Corresponding Author : Dept. of Computer and Information Science, Korea University. tmskim@korea.ac.kr
* Dept. of Computer and Information Science, Korea University. kusuk007@korea.ac.kr
** Dept. of Computer and Information Science, Korea University. sijabarakajia25@korea.ac.kr
논문번호 : KNOM2016-01-002, Received April 20, 2016; Revised June 2, 2016; Accepted July 3, 2016

Ⅰ. 서 론

현재 네트워크를 활용한 서비스가 다양해지고 그 규모가 커짐에 따라 미래인터넷 연구를 위한 대규모 네트워크 구축은 불가피한 상황이다. 하지만 미래인터넷 연구를 위한 대규모 테스트베드 구축을 위한 PC와 스위치장비 등은 대부분 고가이기 때문에 원하는 환경을 구성하고 실험을 수행하기에 현실적이지 못하는 문제가 발생하며, 실제 실험을 위한 테스트베드는 원하는 네트워크 구성을 최소한으로 표현하여 중•소규모의 테스트베드를 구축하는 것이 일반적이다. 따라서 클라우드 컴퓨팅과 같은 미래 인터넷 인프라를 활용한 새로운 서비스가 요구되고 있다. 이러한 기술적 요구가 반영된 서비스를 제공하기 위해서는 수많은 실측실험과 이를 통해 얻어진 측정 데이터가 필요하며, 이를 위해 세계 곳곳에 다수의 미래인터넷 실험 인프라가 구축되어 있다.

미래인터넷 분야에서 가장 중요한 요구사항 중 하나로 사용자는 클라우드 컴퓨팅을 통해 전 세계 어디서든 인터넷만 연결되어 있다면 클라우드에서 사용 가능한 컴퓨팅 자원을 할당 받아 사용할 수 있어야 한다. 현재 대부분의 연구자원 플랫폼은 중소규모의 Stand-alone형태로 제공되고 있으며 대규모의 실험환경 구축을 위해서는 이들 연구자원 플랫폼의 효과적인 연동방안이 필수적이며, 대규모의 클라우드 컴퓨팅이 가능하기 위해서는 서로 다른 영역의 클라우드 망과 연동할 수 있는 기술이 필요하다.

이와 같은 문제를 해결하기 위해 전 세계적으로 다양한 구조와 기능을 가진 다수의 연구자원 플랫폼의 통합 연동을 위해 관련 연구, 교육 및 공동 프로젝트 수행 등 다양한 연구자원 플랫폼 간의 연동 연구가 활발히 진행 중에 있으며, 미국의 GENI[1, 2]와 Emulab[3], 유럽영역의 FIRE[4, 5]와 Planetlab[6, 7] 등이 그 대표적인 예시이다. 국내에서는 KISTI의 KreonetEmulab[8]이 있다. 이렇듯 다양한 기관에서 테스트베드를 운영하여 연구자들에게 서비스를 제공하고 있지만 여전히 대규모의 실험환경을 구축하여 실험을 진행하지 못하는 한계가 있어 각 테스트베드들의 연동 필요성이 증대되고 있다.

이에 따른 미래인터넷 테스트베드들의 연동을 위한 연구도 활발히 진행이 되고 있는데 서로 다른 플랫폼으로 구성된 테스트베드 간 커뮤니케이션의 괴리를 극복하기 위해 표준연동규약 SFA(Slice-based Federation Architecture)[9]를 정의하여 서로 다른 테스트베드들의 컴퓨팅 자원을 Slice단위로 사용할 수 있다. SFA를 활용하여 테스트베드들을 연동하는 대표적인 시스템으로 프랑스 UPMC의 Onelab에서 개발하여 서비스 중인 MySlice[10]가 있다. 하지만 현재의 테스트베드 연동은 각 기관에서 운영중인 다수의 컴퓨팅 자원을 빌려 제공하는 것은 가능하지만, 네트워크를 사용자의 요구에 맞게 설정하지 못하여 동적인 실험 환경을 구축하지 못하는 한계가 있다.

따라서 본 논문에서는 서로 다른 플랫폼으로 구축된 테스트베드 연동시스템의 컴퓨팅 자원을 이용하여 사용자에 요구에 맞춰 동적인 네트워크 환경을 구축은 물론 안정적인 네트워크 환경을 제공할 수 있는 방법과 그 구조를 제안한다. 미래 인터넷 테스트베드의 동적인 네트워크 자원 연동을 위해서는 SDN(Software Defined Networking)기술과의 연동이 필수적이다.

본 논문의 2장에서는 테스트베드와 표준연동규약 SFA 및 테스트베드 연동 시스템인 MySlice, SDN 기술에 대하여 기술하며, 3장에서는 현재까지의 테스트베드 연동 분야의 한계점을 정의한다. 또한 4장에서는 본 논문에서 제안하는 테스트베드 연동 시스템의 구조에 대하여 기술하며, 마지막 5장에서는 결론과 향후 연구 계획에 대하여 기술한다.

Ⅱ. 관련 연구

본 장에서는 테스트베드의 네트워크 자원을 연동하기 이전에 활발히 진행되어온 연구 및 실제 연구자들에게 서비스를 제공하고 있는 시스템에 대하여 기술하며, 또한 본 논문에서 제안하는 구조에 추가된 SDN기술에 대하여 기술한다. 연구자들에게 안정적인 실험 환경을 제공하기 위해서 세계의 다양한 연구기관에서는 원격 접속이 가능한 테스트베드를 구축하여 연구자들에게 제공하였으며, 대규모 실험 환경의 수요에 의해 각 테스트베드들 간의 연동을 위해 표준연동규약 SFA를 정의하여 테스트베드들의 연동을 이루었다. 이에 대한 내용은 이어서 기술한다.

1. Stand-alone 형식의 테스트베드

소규모 연구진에서 대규모의 컴퓨팅 실험환경을

구축하기에는 비용적, 공간적인 측면에서 많은 제약사항이 발생한다. 때문에 세계의 다양한 연구기관에서는 별도의 테스트베드를 구축하여 연구자들에게 연구자원 이용 권한을 주어 컴퓨팅 자원을 제공하고 있다.

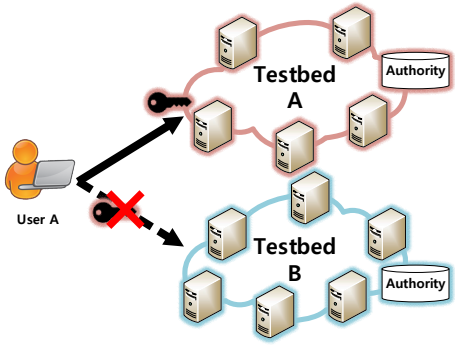


그림 1. Stand-alone 테스트베드
Fig. 1. Stand-alone Testbed

그림 1은 현재 각 연구기관에 구축되어 있는 테스트베드의 가장 기본적인 구조로 각 테스트베드는 독립적으로 운영이 되며, 사용자는 이용하고자 하는 테스트베드의 기관에 계정을 등록하여 권한을 부여받은 후에 연구자원을 이용할 수 있다. 만약 계정을 등록하지 않은 테스트베드의 컴퓨팅 자원을 이용하기 위해서는 새로운 계정을 생성해야 하며, 당연하게도 서로 다른 플랫폼의 테스트베드는 사용자의 계정을 독립적인 Authority에서 관리하기 때문에 계정의 연동은 이루어질 수 없다. 또한 각 테스트베드의 컴퓨팅 자원을 이용하기 위해서는 별도의 플랫폼을 이용해야 하므로 접근이 용이하지 않다.

미래 인터넷 테스트베드에 관한 연구는 미국과 유럽이 주축이 되어 연구를 진행하고 있는데, 미국 영역의 테스트베드로 Emulab과 GENI(Global Environment for Network Innovations)등이 있으며, 유럽 영역의 테스트베드로는 FIRE(Future Internet Research and Experimentation)와 PlanetLab등이 있다. Emulab은 미국 Utah대학에서 개발된 연구자원 프레임워크로 연구자에게 시스템을 개발, 디버그와 평가를 할 수 있는 광범위한 환경을 사용자가 요구하는 네트워크 구조로 구성하여 제공하는 연구자원 플랫폼이다. Emulab은 PlanetLab과 비교되는 대표적인 미래인터넷 연구자원 플랫폼으로 현재까지도 Emulab을 이용한 연구가 활발히 진행중에 있다. 그리고 GENI는 대표적인 미래인터넷 연구자원 플랫폼 프로젝트로 Emulab을 확장하여 망 가상화를 통해 응용 계층 및 물리/제어계층 실험이 가능하다. 또한 FIRE는 유럽의 미래 인터넷 관련 연구를 지원하는 프로젝트인 FP7(Future Internet in Framework Programme 7) 내에서 미래인터넷을 위한 기술개발 및 다양한 실험 인프라를 구축하는 프로젝트로 GENI와 유사하게 실험적이고 혁신적인 아이디어를 구상하고 실험적으로 증명하기 위한 연구자원 플랫폼을 구축하는 것을 목표로 한다. FIRE 계열의 프로젝트는 유럽의 다양한 국가에서 운영하고 있는 연구용 컴퓨팅 자원을 통합, 연동하는 연구를 진행 중에 있다.

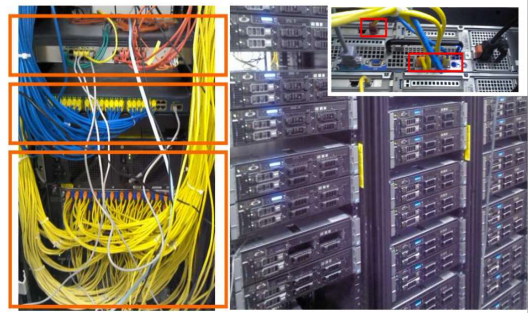


그림 2. 한국과학기술정보연구원의 Kreonet-Emulab
Fig. 2. Kreonet-Emulab of KISTI

국내에서는 한국과학기술정보연구원(KISTI)에서 운영중인 Kreonet-Emulab(그림 2)이 대표적인 테스트베드로, 100여개의 노드를 운용중에 있다. Kreonet-Emulab은 현재 미국영역의 ProtoGENI와 연동되어 서비스 중에 있으며, 본 논문에서는 MySlice와 Kreonet-Emulab 즉, ProtoGENI와 연동을 하여 그 가능성을 검증하였다.

미래인터넷 연구를 위하여 새롭고 창의적인 네트워킹 개념들의 실험적 실증 수요가 증가함에 따라 새로운 서비스를 지향하는 다양한 실험자들의 요구사항을 수용해야 하며, 다수의 실험자들이 여러 실험을 동시에 수행할 수 있어야 한다. 또한 안정적인 실험자원을 공급해야하며, 컴퓨팅 중심적이고 네트워킹 중심적인 자원을 제공해야 한다. 이러한 연구자원 플랫폼을 제공하기 위해서 다양한 연구자원 플랫폼 간의 연동 필요성이 증가되고 있다. 미국영역의 GENI계열과 유럽영역의 PlanetLab, FIRE, Onelab등의 연구자원 플랫폼 연동 시스템이 그 대표적인 예시이다. 하지만 대부분의 연구자원 플랫폼은 Stand-alone의 형태로 서비스를 제공하기 때문에 대규모 실험환경을 구축하기 위해서는 이들 연구자원 플랫폼의 효과적인 연동 방안이 필수적이다. 이

에 미국영역의 GENI와 유럽영역의 FIRE가 공동 재정한 SFA표준연동 프로토콜을 통한 연구자원 플랫폼 연동이 해답이 될 수 있다.

2. SFA(Slice-based Federation Architecture)

SFA는 Princeton 대학의 Larry Peterson을 주축으로 미국영역의 GENI와 유럽영역의 FIRE의 연구원들이 함께 정의하고 개발한 연구자원 플랫폼 연동 규약으로 2009년 7월 Draft Version 1.04를 시작으로 현재 3.1-18까지 개발이 진행 중에 있다. SFA는 GENI, FIRE등 동일 계열의 연구자원 플랫폼의 연동은 물론 이기종 간의 연동에도 사용된다. 테스트베드를 연동을 하는데 있어서 더 쉽고 효과적인 방법을 통하여 연동하는 것은 중요하다. 이에 따라 다양한 요구사항이 존재 하는데 기본적으로 사용자는 본인이 가입한 기관의 연구자원 플랫폼에 자유롭게 접근하여 그 자원을 사용 할 수 있어야 함은 물론이고 다른 기관에서 운영하지만 같은 종류의 연구자원 플랫폼의 자원 역시 접근하여 서비스를 제공받을 수 있어야 한다. 또한 더욱 큰 규모의 네트워크 실험을 제공 받기 위해서는 다른 종류의 연구자원 플랫폼의 자원을 이용 할 수 있어야 한다. 그리고 이렇게 다양한 연구자원 플랫폼으로부터 서비스를 제공받기 위해서 모든 기관에 가입할 필요 없이 한 곳에만 계정을 등록해도 제공받을 수 있어야 하며, 사용자 입장에서 사용성이 좋고 통합된 도구를 통하여 자원이용이 가능해야 한다. 마지막으로 다양한 기관에 소속되어 있는 연구자들이 대규모의 프로젝트를 협업할 수 있어야 한다. 자원 플랫폼 연동의 요구사항을 종합적으로 정리하면 그림 3과 같이 다양한 기관에 소속되어 있는 사용자들은 하나의 기관에만 가입하면 통합된 도구를 통하여 다양한 종류의 연구자원 플랫폼의 자원을 이용할 수 있어야 하고, 대규모 프로젝트를 협업할 수 있어야 한다.

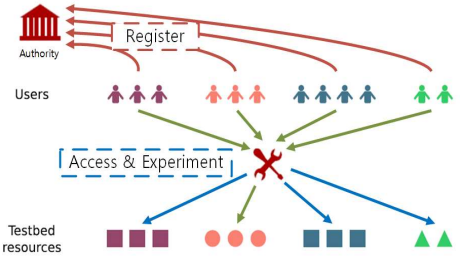


그림 3. SFA 기반 연동 요구사항
Fig. 3. Requirement for Federation based on SFA

SFA에는 각각의 영역에서 역할을 하는 개념적 주체를 크게 4가지로 나눌 수 있는데, 표1은 SFA의 개념적 주체들과 각 주체들의 역할을 정리한 것이다. SFA의 개념적 주체로는 자원을 실제로 소유하고 그 정책을 수립하는 Owner와 자원들을 원활하게 제공하고, 물리적 자원을 관리하는 실질적 관리자인 Operator, 실질적으로 자원을 이용하여 다양한 실험을 진행하는 Researcher, 그리고 공동의 연구를 함께 진행하는 Researcher들을 대표하여 해당 연구의 Researcher에게 권한을 승인하는 Identity Anchor가 있다.

표 1. SFA의 주체
Table 1. The Principal of SFA

주체	역할
Owner	자원의 실제 소유 주체, 해당 자원들의 이용 정책을 수립 수립된 정책을 적용하기 위한 메커니즘 제공
Operator	자원들의 원활한 제공을 위해 수립된 정책에 따른 운영 주체 제공되는 자원의 오남용 및 악용의 소지가 있는 경우 사용제한 가능 새로운 물리적 자원을 설치하고 오래 되었거나 결함이 있는 장비 교체
Researcher	제공되는 자원의 이용 주체 슬라이스를 만들고 자원을 할당 받아 일련의 실험 환경 구성 목적달성을 위한 실험 진행
Identity Anchor	연구별 Researcher들의 대표자, Principal Investigator(PI)라고도 불림 Researcher들의 특성 또는 역할에 따른 자원 접근 및 사용 승인

Owner는 자원을 실제로 소유하고 있는 주체로 해당 자원들에 대한 이용 정책을 수립할 수 있으며, 이때 SFA는 Owner가 수립한 정책을 적용하기 위한 메커니즘을 제공한다. Operator는 Owner가 소유하고 있는 자원들의 원활한 제공을 위해 수립된 정책에 따른 운영 주체로 제공되는 자원의 오•남용 및 악용의 소지가 있는 경우 사용에 제한을 줄 수 있다. 또한 Researcher는 제공되는 자원들을 실제로 이용하는 주체로 슬라이스를 생성하고 자원을 할당 받아 일련의 실험 환경을 구성하여 실험을 진행한다. Researcher는 SFA를 통하여 자원을 제공받는

데에 있어서 그 동작 과정을 모두 알고 있을 필요는 없다. 마지막으로 Identity Anchors는 연구 별 Researcher들의 대표자로 PI(Principal Investigator)라고도 하며, Researcher들의 특성 또는 역할에 따른 자원 접근 및 사용 권한을 승인하는 역할을 한다.

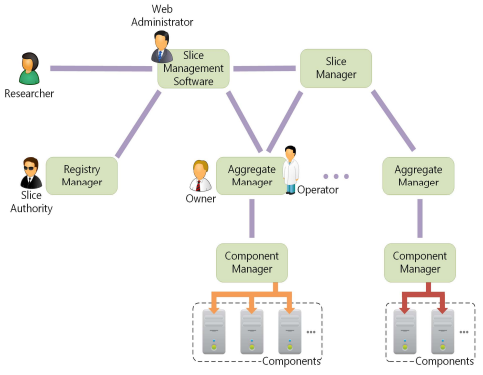


그림 4. SFA 기반 연동의 개념
Fig. 4. Conception of Federation based on SFA

다양한 종류의 연구자원 플랫폼을 연동하는데 단지 SFA 표준연동규약을 사용하는 것만으로 연동이 되는 것은 아니다. SFA 표준연동규약에 정의되어 있는 물리적, 소프트웨어적 요소들이 필요한데 그림 4는 SFA 표준연동규약으로 연동한 Federation-System의 구성요소들의 이상적인 연결 구조이다. CM(Components Manager)은 Owner가 소유하여 운영하고 있는 다수의 자원(Component)을 수집 또는 분리하여 슬리버 기반으로 관리하고 제어하는 역할을 수행한다. 여기서 슬리버란 컴퓨팅 자원을 추상화 하는 최소 단위를 의미한다. 또한 자원에 대한 정보를 AM(Aggregate Manager)에 전달한다. 또한 AM은 CM으로부터 전달받은 자원을 사용자 조건별로 분리를 하고, 슬라이스를 생성하는 역할을 수행한다. AM에서 수행하는 Management Interface는 사용자 조건에 맞춘 다수의 CM을 관리하며, AM API는 각 CM에서 슬리버로 정의된 인프라 자원을 수집한다. 또한 AM은 사용자 조건에 맞춰 묶은 슬라이스를 SM(Slice Manager)에 전달한다. SM은 AM에서 생성된 슬라이스를 관리하고 웹 인터페이스를 통해 사용자에게 친숙한 환경으로 슬라이스 모니터링 기능을 제공한다. 또한 슬라이스는 물론 CM에서 관리하는 자원들의 현황 모니터링 기능 역시 제공한다. 마지막으로 Registry는 사용자 계정 정보와 로그 정보 및 슬라이스 정보 등을DB 또는 파일형식으로 유지하고 해당 정보를 SM으로 전달하여 슬라이스 제어가 가능토록 한다. SM은 다수의 AM을 관리하고, AM은 다수의 CM을 관리한다. 또한 CM은 다수의 Component를 관리하며, 사용자는 SM에 직접적으로 접근하는 것이 아니라 별도의 소프트웨어를 통하여 접근한다.

2.3. SFA기반 테스트베드 연동 시스템(MySlice)

SFA기반의 테스트베드 연동 시스템은 MySlice, SFI, flack, OMNI 등 다양하게 있다. 여기서 Mysilce와 SFI는 유럽영역의 테스트베드에서 자주 활용되는 시스템이고, flack과 OMNI는 미국영역의 테스트베드에서 자주 활용되는 시스템이다. 본 논문에서는 유럽영역의 MySlice를 이용하여 실험환경을 구성하였으며, MySlice를 기반으로 서술하였다.

MySlice는 파리의 UPMC대학의 연구진들이 Onelab2 프로젝트를 통해 개발한 SFA 기반의 테스트베드 연동 플랫폼이다. MySlice는 SFA 표준연동규약 프로토콜을 이용하여 타 도메인 간의 사용자 인증 및 권한부여, 사용 가능한 자원 검색 및 할당, 반환 등의 서비스를 제공한다. 이러한 서비스는 MySlice의 실제 사용자인 연구원이 사용자 관점의 맞춤형 실험 환경 구축을 가능하게 하며 Web UI 형태로 제공된다.

MySlice의 핵심 기능인 인증 및 권한 관리 기능을 통해 MySlice는 사용자의 인증서를 안전하게 관리해주며 사용자는 인증서를 통해 MySlice와 연동된 테스트베드의 접근 권한을 보다 쉽고 간편하게 취득할 수 있다. 이렇게 취득한 권한을 이용하여 사용자들은 다양한 테스트베드의 자원에 접근하여 슬라이스 생성이 가능하며 사용자들은 자신에게 필요한 컴퓨팅 자원의 용도와 지역성을 고려하여 직접 특정 자원을 선택하여 슬라이스를 생성할 수도 있다.

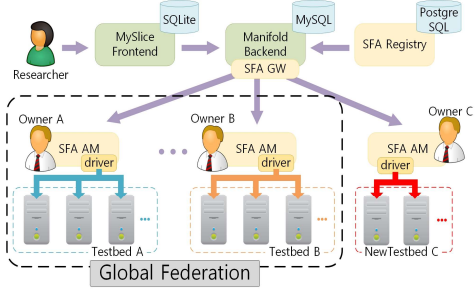


그림 5. MySlice 기반 테스트베드 연동 구조
Fig. 5. The Structure of Testbeds Federation based on MySlice

그림 5는 연구자와 Testbed 소유자 관점에서의 MySlice와 Testbed의 연동을 나타낸 그림이다. 신규 테스트베드의 소유자는 MySlice를 통해 기 연동된 국제적 테스트베드들(Global Federation)과의 연동에 대한 부담을 줄일 수 있어서 연동에 대한 진입장벽을 낮출 수 있다. 국제적 테스트베드와의 연동은 자신이 소유하고 있는 자원들과 연동된 테스트베드의 자원들을 MySlice를 사용하는 연구자들에게 다양하고 많은 자원들과의 협업 서비스 제공을 가능하게 한다.

MySlice는 사용자에게 기 연동된 테스트베드의 자원에 대한 검색, 예약, 할당, 반환 등 일련의 서비스를 Web 인터페이스를 통해 제공하는 SFA 기반의 테스트베드 연동 플랫폼 (표준연동 제어 오픈 소프트웨어)이다. MySlice연동 플랫폼은 사용자의 인터페이스를 담당하는 MySlice 와 여러 테스트베드와의 연결을 담당하는 Manifold로 구성되어 있다. MySlice 연동 플랫폼은 MySlice Frontend와 Manifold Backend로 나눠진다. MySlice 와 Manifold 사이의 통신은 https 프로토콜을 사용하며 XMLRPC를 이용해 데이터를 주고받는다.

MySlice는 Django 웹 프레임워크를 기반으로 개발되었고 웹으로 접속하는 사용자와 django 프로젝트 관련 데이터를 저장하고 관리하기 위해 SQLite를 기본 데이터베이스로 사용한다. Manifold 는 Python 기반으로 개발되었고 관리자, 플랫폼 및 테스트베드들의 데이터를 저장하고 관리하기 위해 MySQL을 기본 데이터베이스로 사용한다. Manifold는 SFA Registry와 여러 테스트베드들을 하나로 연동해주는 허브 역할을 한다.

2.4. REST(Representational State Transfer) API

REST API[11]란 네트워크 기반 자원을 정의하고 자원에 대한 주소를 지정하는 방법을 정의한 아키텍처로 HTTP의 주요 저자 중 한사람인 Roy Fielding에 의해 정의되었다. REST API는 서비스 지향적 구조인 SOAP(Simple Object Access Protocol)와 달리 자원 지향적 구조로 web 사이트의 컨텐츠, DB의 내용 등을 하나의 자원으로 인식하여 각 자원들의 고유한 URI(Uniform Resource Identifier)를 부여하고, 각 자원에 대한 생성, 일기, 수정, 삭제 작업을 HTTP의 기본 명령어인 POST, GET, PUT, DELETE를 이용하여 처리할 수 있다.

REST API의 특성은 Uniform Interface, Stateless, Cacheable, Self-descriptiveness, Client-Server Structure로 크게 다섯 가지로 생각할 수 있다. 먼저 Uniform Interface는 HTTP 표준에만 따르면 어떠한 플랫폼에서 사용이 가능한 인터페이스 스타일이라는 의미다. C나 Java등 특정 언어나 기술에 종속받지 않고 HTTP와 JSON을 사용할 수 있는 플랫폼에서 사용이 가능하다. 다음으로 Stateless는 사용자나 클라이언트의 Context를 서버 쪽에 유지 하지 않는다는 의미로 상태 정보를 저장하지 않으면 각 API 서버는 들어오는 요청만을 들어오는 메시지로만 처리하면 되며, 세션과 같은 Context 정보를 신경 쓸 필요가 없기 때문에 구현이 단순해질 수 있다. 다음으로 Cacheable은 HTTP 기반의 로드 밸런서나 SSL은 물론이고, HTTP가 가진 특징 중 하나인 Caching 기능을 적용할 수 있기 때문에 용량이나 성능 면에서 많은 장점이 있다. 또한 Self-descriptiveness는 API 메시지 자체만 보고도 API를 이해할 수 있는 구조를 갖는 다는 것이다. 자원과 Method를 이용해서 어떤 행위를 하는지를 알 수 있으며, 메시지 포맷 역시 JSON을 이용하여 직관적으로 이해가 가능한 구조이다. 마지막으로 Client-Server Structure는 서버에서 API를 제공하고, 제공된 API를 이용하여 행위 및 저장을 책임지며, 클라이언트의 경우 사용자 인증이나 Context 등을 직접 관리하고 책임지는 구조로 역할이 나뉜다.

표 2. REST API 구조
Table 2. The Structure of REST API

Element	Description
Resource	자원에 대한 정의 URI를 통해서 개별 자원은 독립적인 URI를 가짐
Action	자원에 대한 처리 방법 정의 HTTP Method(POST, GET, PUT, DELETE) 사용
Message	자원에 대한 처리 방법에 대하여 정의 HTTP Message Payload를 통하여 표현 다양한 Content Type을 선언하여 사용 가능

표 2는 REST API의 구성에 대하여 정리한 것으로 REST API는 크게 Resource, Action, Message 세 단계로 구성이 되어 있다. Resource는 자원에 대하여 정의를 하는 것으로 각 자원들은 독립적인 URI를 갖기 때문에 다른 자원들과 구별될 수 있다. Action은 자원에 대한 처리 방법을 정의하는 것으로 다른 명령체계 없이 HTTP에서 기본적으로 사용

하는 명령어인 POST, GET, PUT, DELETE를 사용한다. 마지막으로 Message는 자원에 대한 처리 방법에 대하여 정의하는 것으로 HTTP와 동일한 방식으로 표현이 되며 다양한 Content Type으로 선언이 가능하다. 주로 JSON형태를 사용하게 된다.

표 3. REST API의 기능
Table 3. The Function of REST API

Method	CRUD	SQL	Description
POST	Create	INSERT	자원 생성
GET	Read	SELECT	자원 및 정보 조회
PUT	Update	UPDATE	자원 수정
DELETE	Delete	DELETE	자원 삭제

표 3은 REST API에서 사용 가능한 Method를 정리한 것으로 HTTP와 동일하게 POST, GET, PUT, DELETE 명령어를 사용할 수 있다. POST를 통해 해당 URI에 대하여 요청을 하게 되면 리소스를 생성하게 된다. 다음으로 GET은 해당 리소스에 대한 정보를 조회 할 수 있으며, PUT을 통해 해당 리소스의 내용을 수정할 수 있다. 마지막으로 DELETE를 통해 해당 리소스를 삭제할 수 있다.

이러한 REST API는 OPEN API를 제공하고 구조가 비교적 단순하기 때문에 누구나 사용할 수 있으며, 플랫폼에 종속적이지 않고 HTTP 표준에만 따르면 MultiPlatform을 지원하고 연동에 용이하다. 또한 데이터 형식이 독립적이며, 원하는 타입으로 데이터를 전달 가능하며, 기존에 사용하던 HTTP의 포맷을 그대로 사용할 수 있다는 장점이 있다.

Ⅲ. 기존 테스트베드 연동의 한계

테스트베드를 연동함에 있어서 가장 중요한 요구사항은 지리적으로 분산되어 구축된 테스트베드의 자원을 자신의 자원처럼 사용할 수 있어야 하는 점이다. 이에 대해서 사용자의 위치에 따른 제약이 없어야 하며, 사용자가 요구하는 실험환경 규모에 제약이 없어야 한다. 또한 사용자가 요구하는 실험환경의 구조적 제약이 없어야 한다. 이와 같은 요구사항을 충족시키기 위하여 원격지에서 접속하여 자신의 컴퓨터처럼 사용 가능한 다양한 테스트베드를 통합하여 대규모 실험을 진행 할 수 있도록 하는 테스트베드 연동 시스템 등 연구자들에게 양질의 실험 환경을 제공하기 위한 다양한 연구가 진행되어 왔지만, 여전히 모든 연구자들의 요구하는 실험환경을 제공하지 못하는 한계점이 몇 가지 존재 한다.

첫째, 현재의 테스트베드 연동 시스템을 통해서는 통합된 연구환경을 생성할 수 없다. 다수의 컴퓨팅 자원을 슬라이스로 생성할 수는 있지만 그들은 네트워크 상에서 독립적인 자원으로 인식이 되기 때문에 다수의 컴퓨팅 자원들을 이용한 네트워크 실험을 진행하지 못한다. 사용자가 다수의 컴퓨팅 자원을 동시에 요구하는 것은 그 만한 컴퓨팅 파워가 필요하다는 것이며, 독립적인 자원 하나가 그에 합당한 작업을 수행 하지 못한다면 테스트베드를 연동하는 의미가 없다.

둘째, 사용자의 요구에 맞는 네트워크 환경을 구성하지 못한다. 사용자는 네트워크 실험을 진행할 때 필요한 토폴로지를 구성하여 실제로 구현하기 힘들기 때문에 가상으로 구현해야만 한다. 이를 위해서는 각 연구자원 플랫폼의 스위치의 플로우 테이블 설정을 변경하여 네트워크 트래픽의 경로를 설정할 수 있어야 한다.

셋째, 실험을 진행함에 있어서 높은 QoS를 보장하지 못한다. 현재의 서로 다른 플랫폼의 테스트베드는 직접적인 네트워크 선로가 연결이 되어 있지 않기 때문에 안정적인 QoS를 보장하지 못함은 물론 정확한 실험 결과를 도출하기 어렵다. 이러한 문제를 해결 하기 위해서는 각 연구자원 플랫폼의 스위치 장비를 통합하여 제어할 수 있는 기술이 필요하다.

마지막으로 네트워크에 문제 발생 시 발 빠른 대응이 어렵다. 원거리로 떨어져 있는 연구자원의 상태를 모니터링하는 것은 결코 쉬운 일이 아니며, 문제 발생 시 이를 원격으로 복구하는 것 또한 불가능에 가깝다.

따라서 미래인터넷 테스트베드의 네트워크 자원의 연동을 위해서는 실제 구성되어 있는 네트워크 구조와 관계 없이 사용자의 요구에 맞춰 동적으로 트래픽의 흐름을 변경하여 가상의 네트워크 구조를 생성해야 한다. 이를 위해서는 라우터, 스위치 등 네트워크 장비의 역할을 가상화 하여 소프트웨어로 정의할 수 있는 SDN기술과의 연동이 필수적이다.

Ⅳ. 테스트베드 네트워크자원 연동 방법

본 장에서는 본 논문에서 제안하는 향상된 기능의 테스트베드 연동 시스템과 그 구조[12]에 대하여 서술한다. 테스트베드의 네트워크 자원을 연동하기

에 앞서 ONOS API Controller라는 ONOS SDN Controller에 사용자의 요구를 전달하는 모듈을 제작하였으며, 최종적으로 해당 모듈을 MySlice 에 삽입을 하여 네트워크 자원의 연동을 이루었다.

1. ONOS API Controller

다양한 테스트베드의 네트워크 자원을 제어하기 위해서는 네트워크를 소프트웨어로 정의하고 제어할 수 있는 SDN기술과의 연동이 필수적이다. SDN Controller에는 ODL(OpenDaylight), FloodLight 등 다양한 종류가 있지만 본 논문에서는 ONOS SDN Controller[12]를 기반으로 테스트베드 연동을 진행하였다. 다만 ONOS SDN Controller를 직접적으로 제어하는 것이 아닌 외부에서 메시지를 전달하여 ONOS SDN Controller를 제어하는 방식으로 시스템을 개발하였다. 그 이유는 차후 MySlice와 연동 작업 진행 시 웹페이지에서 간단한 조작으로 네트워크 제어가 가능해야 하기 때문이다.

본 논문에서는 외부에서 ONOS SDN Controller를 제어하는 방법으로 REST API를 채택하였다. ONOS는 Intent라는 기능을 통하여 쉽게 네트워크를 제어할 수 있으며, REST API를 지원하여 ONOS SDN Controller를 제어하기에 적합하다.

그림 6은 ONOS SDN Controller와 ONOS API Controller를 연동한 구조로 ONOS API Controller는 ONOS SDN 컨트롤러에 사용자의 요구를 REST API의 형식에 맞게 전달하게 되면 ONOS SDN 컨트롤러는 이를 수용하여 네트워크의 정보를 출력하거나 제어하게 된다. ONOS API Controller가 ONOS SDN 컨트롤러의 제어권을 얻기 위해서는 ONOS SDN 컨트롤러의 사용자 계정과 비밀번호를 통해 인증을 받아야 한다.

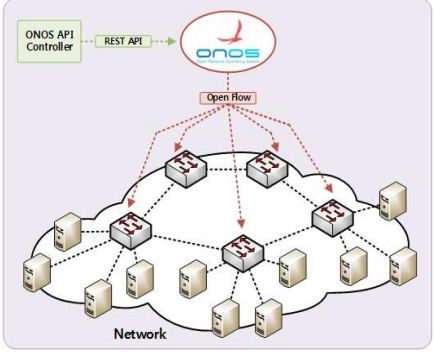


그림 6. ONOS API Controller의 구조
Fig. 6. The Structure of ONOS API Controller

ONOS API Controller는 기본적으로 정보확인, Intent 추가, Intent 삭제로 크게 세 가지 기능을 사용할 수 있다. 그림 7은 ONOS API Controller의 전체 기능을 정리한 것으로 확인 가능한 정보로는 현재 네트워크의 토폴로지 정보, 스위치 정보, 노드 정보, 설정되어 있는 Intent정보 등이 있으며, 추가 가능한 Intent는 Host-to-Host Intent와 Point-to-Point Intent가 있으며 이렇게 생성된 Intent를 삭제할 수 있다.

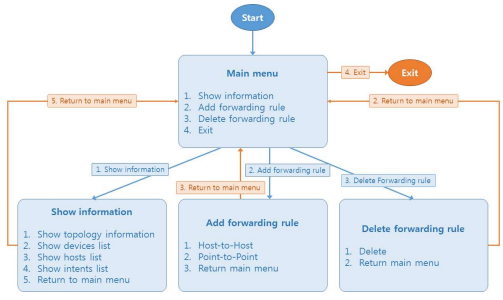


그림 7. ONOS API Controller의 기능
Fig. 7. The Fuction of ONOS API Controller

2. ONOS API Controller on MySlice

앞서 설명한 ONOS API Controller는 Python언어로 제작한 간단한 프로그램으로 MySlice와의 연동을 위해서는 PHP 스크립트로 웹페이지 상에서 ONOS API Controller의 각 기능을 실행시킬 수 있어야 한다. 또한 사용자의 입장을 고려하여 직관적이고 간단한 조작만으로 네트워크를 제어할 수 있어야 한다.

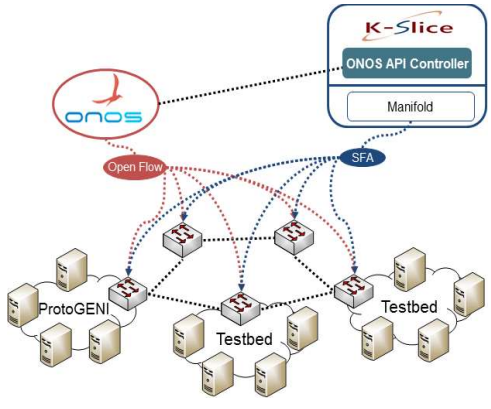


그림 8. ONOS API Controller가 추가된 K-Slice의 구조
Fig. 8. The Structure of K-Slice with ONOS API Controller

그림 8은 본 논문에서 실제 구축한 시스템의 전체 구조로 ProtoGENI와 ONOS SDN Controller,

K-Slice를 연동하는 구조를 나타낸 것으로 여기서 K-Slice는 MySlice를 본 연구 환경에 맞게 재설정한 시스템이다.

기본적인 구조는 기존 컴퓨팅 자원의 연동이 완료된 K-Slice와 ONOS SDN 컨트롤러, 각 스위치들이 연결된 형태이다. 이때, ONOS SDN 컨트롤러는 연동이 약속된 모든 테스트베드의 스위치들과 연결이 되어 있어야 한다. 연동 방법은 K-Slice와 마찬가지로 표준연동규약 SFA를 통하여 다양한 테스트베드의 컴퓨팅자원을 슬라이스로써 생성한다. 이어서 사용자의 요구에 맞는 네트워크 토폴로지를 입력 받아 ONOS SDN에 경로에 맞는 Intent를 추가하여 슬라이스에 포함되어 있는 컴퓨팅 자원들 간의 트래픽 흐름을 제어하여 가상 토폴로지를 생성함으로 네트워크 자원의 연동을 이룬다. 이때에는 컴퓨팅 자원을 연동하는 것과 같이 네트워크 자원을 연동함에 있어서 SFA를 통해 사용자의 요구를 전달하는 것이 아닌 REST API로 ONOS SDN 컨트롤러에 전달하여 네트워크 자원을 연동한다.

그림 6과 같은 구조로 테스트베드 연동이 이루어지면, 각 테스트베드는 상용 인터넷 망과는 별개로 독립적인 물리 네트워크로 직접 연결이 되기 때문에 다른 트래픽이 흐름의 영향을 받지 않아 안정적인 QoS를 보장받을 수 있다. 또한 각 테스트베드의 스위치 장비는 ONOS SDN 컨트롤러에 의해 대역폭, 플로우 테이블 등이 제어되기 때문에 사용자가 요구하는 토폴로지를 ONOS SDN 컨트롤러에서 입력 받아 슬라이스의 네트워크환경을 구성할 수 있다. 또한 네트워크 장비에 문제가 발생하면 ONOS SDN 컨트롤러에서 동적으로 새로운 네트워크 경로를 설정할 수 있기 때문에 네트워크 문제에 따른 실험 진행이 불가능한 상황에 대하여 발 빠른 대응이 가능하다.

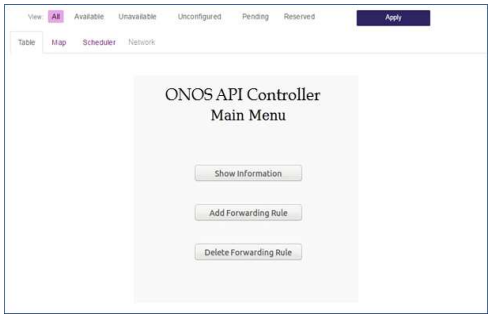


그림 9. K-Slice의 네트워크 자원 설정 탭
Fig. 9. The Network Resource Configuration Tab of K-Slice

그림 9는 K-Slice에 삽입한 ONOS API Controller의 메인 화면으로 사용자는 간단한 버튼 클릭으로 네트워크 정보를 GUI로 확인할 수 있으며, 진행하고자 하는 실험의 성격에 맞게 Intent를 추가 또는 제거하여 네트워크를 설정할 수 있다.

V. 결론 및 향후 연구

본 논문에서는 테스트베드 연동 시스템 K-Slice를 이용하여 생성한 슬라이스의 네트워크 자원을 연동하는 구조와 그 방법으로 SDN Controller와의 연동을 통한 네트워크 제어방법에 대하여 제안하였으며, 이를 실제 구현하였다. 이를 통하여 각 컴퓨팅 자원들의 네트워크 경로를 재설정하여 안정적인 QoS를 보장받을 수 있었으며, 소프트웨어적으로 네트워크 대역을 제어하면 더욱 안정적인 QoS를 지원할 수 있는 가능성은 여전히 열려있다.

Bandwidth를 제어하여 더욱 향상된 네트워크를 보장할 수 있으며 이미 소스코드를 삽입을 해 놓은 상태이다. 하지만 현재 ONOS 개발진은 “bug report ONOS-5002”를 통해 네트워크 대역을 설정함에 있어서 JSON 포맷으로 요청을 하는 작업에 치명적인 버그가 발생함을 알리고 이를 보수하는 작업을 수행 중에 있다. 향후 ONOS SDN 컨트롤러 버그의 보수가 완료된 후, ONOS API Controller에 네트워크 대역 설정 기능을 추가하여 실험자들에게 향상된 QoS의 네트워크 실험 환경을 제공할 수 있을 것이다.

또한 현재까지의 테스트베드 연동은 컴퓨팅 자원에 대해서 슬라이스를 생성하고 자원을 할당함에 있어서 실제 구축되어 있는 자원의 개수까지 밖에 구성을 하지 못하는 한계점이 존재하였다. 하지만 미래의 인터넷은 실제 자원들을 이용하여 구축된 테스트베드를 넘어서 컴퓨팅 자원 가상화 기술을 이용하는 방향으로 발전하고 있다. 이러한 기능을 지원해주는 기술로 “Open Stack” 기술이 대표적이다. Open Stack이란 퍼블릭 및 프라이빗 클라우드 컴퓨팅 플랫폼을 구축하는데 필요한 소프트웨어 블록을 개발하는 오픈 소프트웨어 프로젝트로, 서버, 스토리지, 네트워크와 같은 물리자원을 가상화시켜 대쉬보드 및 REST API로 제공하고 이를 관리할 수 있는 클라우드 플랫폼 구축용 오픈소스 프로젝트 이다. Open Stack은 사용자에게 컴퓨팅 자원을 실제가 아닌 이미지 형태의 가상머신으로 제공하기 때문에 사용자가 구성할 수 있는 컴퓨팅 자원 규모

에 제한이 없다. 또한 컴퓨팅 환경에 장애가 발생 시 빠른 복구가 가능하기 때문에 기존 구축되어 있는 물리적 테스트베드에 비해 장점이 많다. 따라서 본 연구진은 본 연구과제의 향후 연구개발 계획으로 K-Slice와 Open Stack의 연동 연구를 진행할 계획이다.

References

[1] Elliott, Chip. "GENI-global environment for network innovations." LCN. 2008.

[2] Berman, Mark, et al. "GENI: A federated testbed for innovative network experiments." Computer Networks 61 (2014): 5-23.

[3] Hibler, Mike, et al. "Large-scale Virtualization in the Emulab Network Testbed."USENIX Annual Technical Conference. 2008

[4] Gavras, Anastasius, et al. "Future internet research and experimentation: the FIRE initiative." ACM SIGCOMM Computer Communication Review 37.3 (2007): 89-92.

[5] Schwerdel, Dennis, et al. "Future Internet research and experimentation: The G-Lab approach." Computer Networks 61 (2014): 102-117.

[6] Chun, Brent, et al. "Planetlab: an overlay testbed for broad-coverage services."ACM SIGCOMM Computer Communication Review 33.3 (2003): 3-12.

[7] Lo, Shihmin, Ellen Zegura, and Marwan Fayed. "Virtual network migration on real infrastructure: A planetlab case study." Networking Conference, 2014 IFIP. IEEE, 2014.

[8] Lee, Minsun, Woojin Seok, and Kwan-Jong Yoo. "Challenges and Opportunities in the KREONET-Emulab Network Testbed." 한국콘텐츠학회 ICCC 논문집(2014): 213-214.

[9] Fdida, Serge, Timur Friedman, and Thierry Parmentelat. "OneLab: An open federated facility for experimentally driven future internet research." New Network Architectures. Springer Berlin Heidelberg, 2010. 141-152.

[10] Baron, Loic, et al. "OneLab: Major computer networking testbeds open to the IEEE INFOCOM community." Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on. IEEE, 2015.

[11] Berde, Pankaj, et al. "ONOS: towards an open, distributed SDN OS." Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014.

[12] Berde, Pankaj, et al. "ONOS: towards an open, distributed SDN OS."Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014.

[13] 김성민, 이수강, 석우진, 김명섭, "SDN/SFA 기반 미래 인터넷 연구자원 플랫폼 연동 구조", 2016년 통신망운용관리 학술대회 (KNOM 2016), 강원대학교, 춘천, May. 12-13, 2016, pp.80-84.

김 성 민 (Sung-Min Kim)

2014년 : 고려대학교 컴퓨터정보학과 학사

2014년~현재 : 고려대학교 컴퓨터정보학과 석사과정

<관심분야> 네트워크 관리 및 보안, 데이터 암호화, 클라우드 컴퓨팅

심 규 석 (Kyu-Seok Shim)

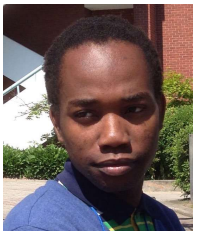
2014년 : 고려대학교 컴퓨터 정보학과 졸업

2016년 : 고려대학교 컴퓨터정보학과 석사과정 졸업

2016년~현재 : 고려대학교 컴퓨터정보학과 박사과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

Baraka D. Sija



2016년~현재 : 고려대학교 컴퓨터 정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 트래픽 분류

김 명 섭 (Myung-Sup Kim)

1998년 : 포항공과대학교 전자계산학과 학사
2000년 : 포항공과대학교 컴퓨터공학과 석사
2004년 : 포항공과대학교 컴퓨터공학과 박사
2006년 : Dept. of ECS, Univ. of Toronto, Canada
2006년~현재 : 고려대학교 컴퓨터정보학과 교수
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크