논문 20-45-09-19 The Journal of Korean Institute of Communications and Information Sciences '20-09 Vol.45 No.09
https://doi.org/10.7840/kics.2020.45.9.1632

Openflow기반 오케스트레이션 프로토콜 설계 및 구현

이 길 호•, 권 원 식*, 천 세 준*, 최 진 식°

Design and Implementation of Openflow-Based Orchestration Protocol

Gil-ho Lee•, Won-sic Kwon*, Se-joon Chun*, Jin-seek Choi°

요 약

Software-defined networking (SDN) 은 논리적 중앙 집중형 구조이기 때문에 트래픽이 컨트롤러에 과도하게 집중된다. 이러한 문제를 해결하기 위해 여러 개의 SDN 컨트롤러가 사용된 다중 도메인을 관리할 수 있는 오케스트레이션 프로토콜이 요구된다. 오케스트레이션 프로토콜은 다중 도메인 SDN망의 토폴로지 탐색, 모니터링뿐만 아니라 종단간 흐름을 프로비져닝하기 위한 핵심 기술이다. 본 논문에서는 오픈플로(Openflow) 프로토콜 기반의 오케스트레이션 프로토콜을 구현하고 토폴로지 탐색 및 토폴로지 업데이트 실험 결과를 기반으로 오픈플로 오케스트레이션 프로토콜의 성능을 분석한다. 실험 결과 토폴로지 탐색 성능이 REST(Representational State Transfer) 방식과 비교하여 최대 5.5배 트래픽이 추가적으로 발생했지만 다이나믹 매핑(Dynamic Mapping)을 이용한 추상화 토폴로지(Abstraction Topology) 구현으로 트래픽 발생량 차이를 5.5배에서 3.2배로 줄여 문제점을 완화했다. 반면 토폴로지 이벤트 모니터링 성능에서 이벤트 처리 시간은 REST 방식과 비교하여 최소 51배 빠르게 처리했다. 또한 네트워크 크기나 토폴로지 탐색의 주기 조건 변화에서도 평균 27ms 속도로 이벤트 성능이 일정하였다. 뿐만 아니라 기존 SDN 컨트롤러의 재사용과 신뢰성 높은 컨트롤러 기반의 토폴로지 관리가 가능함을 입증하였다.

Key Words : SDN, Openflow, REST API, Orchestration, Protocol, Monitoring

ABSTRACT

In software-defined networking (SDN), the traffic is excessively concentrated on the controller due to the logically centralized architecture. To solve this problem, an orchestration protocol is required to manage multiple domains with multiple SDN controllers. The orchestration protocol is a core technology for topology discovery and fault monitoring as well as end-to-end flow provisioning of multi-domain SDN networks. This paper implements an OpenFlow-based orchestration protocol and evaluates its effectiveness as an orchestration protocol through the experimental results of topology discovery and update. The results show that the traffic generation increases by up to 5.5 times compared to the REST (Representative State Transfer) approach, but reduces it by at least 3.2 times with the implementation of abstraction topology using Dynamic Port Mapping. On the other hand, event processing time can be improved at least 51 times faster than the REST approach. The proposed protocol also has fast monitoring performance at an average rate of 27 ms in various experimental conditions while changing network sizes and topology monitoring periods. In addition, the proposed protocol enables to reuse the open-source controllers and reliable controller-driven topology management.

※ 이 연구는 2020년도 산업통상자원부 및 산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임('20010825').
• First Author : Hanyang University Department of Computer Science, leeeeeegilho@gmail.com, 학생(석사), 학생회원
° Corresponding Author : Hanyang University Department of Computer Science, jinseek@hanyang.ac.kr, 정교수, 종신회원
* Hanyang University Department of Computer Science, wolfwathc@naver.com, 학생(박사), 정회원; sniperjoon@hanyang.ac.kr, 학생(박사), 학생회원
논문번호 : 2020007-176-D-RE, Received July 30, 2020; Revised August 25, 2020; Accepted August 25, 2020

Ⅰ. 서 론

SDN(Software Defined Networking)은 빠르고 다양하게 변화하는 사용자들의 요구를 유동적으로 충족시킬 수 있는 논리적 네트워크 제어 기술이다. SDN은 데이터 플레인(Data Plane)과 컨트롤 플레인(Control Plane)을 분리하고 중앙에 위치한 컨트롤러를 사용하여 토폴로지 관리, 정책 및 네트워크 기능 가상화 서비스를 제공한다. 그러나 네트워크 장비 수가 증가함에 따라 하나의 컨트롤러가 담당하는 장비 수가 증가하게 되어 컨트롤러에 부하와 트래픽이 집중되는 현상이 발생한다.[1] 특히 클라우드 컴퓨팅 기술이 보편화되면서 가상화 및 클라우드 서비스에 대한 기대와 의존도가 높아지고 망의 확장성에 대한 필요성이 늘어나고 있기 때문에 복수의 SDN 컨트롤러로 구성된 분산형 다중 도메인 SDN망 기술이 요구된다.[1]

분산형 다중 도메인 SDN망을 구축하고 관리하기 위해서는 반드시 오케스트레이션(Orchestration) 프로토콜이 필요하다. 오케스트레이션 프로토콜(Orchestration Protocol)은 다중 도메인에 대한 토폴로지 탐색(Topology Discovery), 토폴로지 이벤트 모니터링(Topology Event Monitoring), 프로비져닝(Provisioning) 기능을 제공하는 망 대 망 프로토콜로 분산 다중 도메인 SDN망을 하나의 망처럼 관리하기 위한 통합제어 기능을 제공한다.

분산 다중 도메인 SDN망을 위한 오케스트레이션 프로토콜에 대한 연구는 대표적으로 다음과 같이 진행되어 왔다. CTTC(Centre Tecnològic de Telecomunications de Catalunya)에서는 웹소켓 기반의 YANG/RESTCONF를 이용한 COP(Control Orchestration Protocol)을 제안하였다.[2] 웹소켓 방식은 양방향 End-to-End 전송 서비스에 대한 품질(Quality of Service : QoS)과 성능을 보장하지만 망 대 망 소켓 프로토콜의 표준화 과정 및 오케스트레이터 구현을 위한 추가적인 개발 절차가 필요하다.[2]

GTOP(Generalized Topology discovery, operational monitoring, and provisioning) 은 표준화된 PCEP(path computation element communication protocol)을 확장하여 인터도메인 토폴로지를 기반으로 경로 계산 기능을 제공하는 오케스트레이션 프로토콜로 제안되었다.[3] 그러나 COP와 마찬가지로 PCEP 확장에 대한 표준화가 이루어지고 있지 않아 이기종 네트워크 간에 오케스트레이션 프로토콜 사용에 있어 호환성 문제가 발생한다.

REST(Representational State Transfer)[4]의 경우 HTTP(HyperText Transfer Protocol)기반 서비스로 범용성이 보장되며 uniform resource indicator(URI) 규칙만 정의하면 쉽게 오케스트레이션이 가능하기 때문에 많이 연구되고 왔다. 하지만 데이터 모델에 대한 표준화 부재로 REST 적용이 필요한 모든 구성요소에 공통된 YANG 기반의 데이터 모델링 작업 및 URI 정의가 필요하다. 또한 HTTP 기반으로 실시간 응답 서비스가 어려워 프로토콜에 대한 추가적인 개발이 필요하다. 또한 정보기반(Information-driven) 프로토콜로써 정보 제공자의 장애나 정보의 부정확성으로 토폴로지 관리에 문제가 발생 할 수 있다.

본 논문에서는 SDN 표준인 오픈플로 프로토콜을 오케스트레이션 프로토콜(Openflow-based Orchestration Protocol)로 제안하여 표준화 및 데이터 모델링 문제를 해결하고자 한다. 오픈플로 프로토콜은 Controller-driven[3] 프로토콜로 컨트롤러가 폴링 방식으로 토폴로지를 직접 검색하기 때문에 REST와 같은 정보기반 프로토콜과 비교해 신뢰성 높은 도메인 관리가 가능하다. 하지만 주기성을 가진 폴링 방식 사용으로 트래픽이 많아지고 최대 주기만큼 토폴로지 갱신에 대한 지연시간이 발생한다. 본 논문에서는 실시간으로 전달 받는 이벤트 정보에 대한 즉각적인 처리로 토폴로지 갱신에 대한 지연시간 발생 문제를 해결하고 다이나믹 매핑(Dynamic Mapping) 기술을 이용한 추상화 토폴로지 구현으로 트래픽 발생량 문제를 해결하기 위한 오픈플로 기반 오케스트레이션 프로토콜을 제안한다. 오픈플로 프로토콜을 사용함으로써 기존 오픈소스 컨트롤러의 재사용이 가능해지고 신뢰성 높은 Controller-driven 방식의 도메인 관리가 가능한 컨트롤러 간의 접속프로토콜로 사용이 가능하다. 또한 이를 통해 오케스트레이션 환경 구축 시 컨트롤러 간에 상호 호환성 및 네트워크 확장성 문제를 해결한다.

실험검증을 위해 오케스트레이션 프로토콜을 사용할 수 있는 에이전트를 오픈소스 기반의 SDN 컨트롤러에 구현한다. 에이전트가 구현된 SDN 컨트롤러들을 이용해 분산 다중 도메인 오케스트레이션 실험 환경을 구축하고 오픈플로 기반 오케스트레이션 프로토콜의 성능을 분석한다. 주요 성능 메트릭(Metric)은 토폴로지 탐색 트래픽과 토폴로지 이벤트 모니터링에서의 이벤트 처리 시간을 측정한다. 또한, 다이나믹 매핑(Dynamic Mapping) 기술을 이용하여 토폴로지 정보를 추상화 한다. 추상화 정도에 따라 가상화된 Implicit와 Explicit 토폴로지[5] 형태로 데이터 구성과 관리가 가능하도록 확장하여 트래픽 발생량 감소와 호환성을 가지는 오케스트레이션 구조의 생성이 가능

함을 입증한다.

본 논문은 다음과 같이 구성이 되어 있다. 2장에서는 오픈플로 기반 오케스트레이션 프로토콜 구조와 오픈플로 에이전트 구현 그리고 다이나믹 매핑 기술에 대해 설명한다. 3장에서는 토폴로지 탐색(Topology Discovery) 및 토폴로지 이벤트 모니터링(Topology Event Monitoring) 실험을 통해 제안된 오케스트레이션 프로토콜의 성능을 분석한다. 마지막으로 4장에서는 결론 및 향후 연구 방향으로 끝을 맺고자 한다.

Ⅱ. 본 론

2.1 분산형 다중 도메인 SDN 오케스트레이션 구조

그림 1은 본 논문에서 사용하는 분산형 다중 도메인을 위한 오케스트레이션 계층구조이다. 오케스트레이션 계층 구조는 상위 계층의 오케스트레이터(Orchestrator)와 하위 계층의 SDN 컨트롤러로 다계층 구조를 가진다.[6] 하위 도메인 A, B, C들은 세 개의 SDN 컨트롤러가 따로 관리하는 분산형 다중 도메인을 형성한다.

오케스트레이터 역할을 하는 SDN 컨트롤러는 하위 계층의 SDN 컨트롤러에게 전달 받은 정보를 바탕으로 전체 도메인 토폴로지를 관리한다. 뿐만 아니라 외부에서 전달 받은 프로비저닝(Provisioning) 요청에 대해 전체 도메인 토폴로지를 고려하여 종단간 경로를 하위 계층 SDN 컨트롤러에게 전달한다. 하위 계층의 SDN 컨트롤러는 설치된 에이전트를 이용하여 도메인내의 정보를 수집하고 오케스트레이터에게 전달한다. 그리고 도메인내의 토폴로지 변화를 실시간으로 감지하여 오케스트레이터에게 전달한다. 뿐만 아니라 오케스트레이터에게 전달 받은 경로를 관리하는 도메인내의 스위치에게 전달하여 해당 스위치의 플로우 테이블을 수정하게 한다. 본 논문에서는 SDN 컨트롤러와 오케스트레이터 간의 통신을 표준화된 오픈플로 프로토콜을 사용하여 이기종 컨트롤러간의 상호 운용성을 보장한다. 또한 기존 SDN 컨트롤러의 오픈플로 서버를 이용하여 SDN 컨트롤러를 추가 구현 없이 오케스트레이터로 재사용이 가능하다.

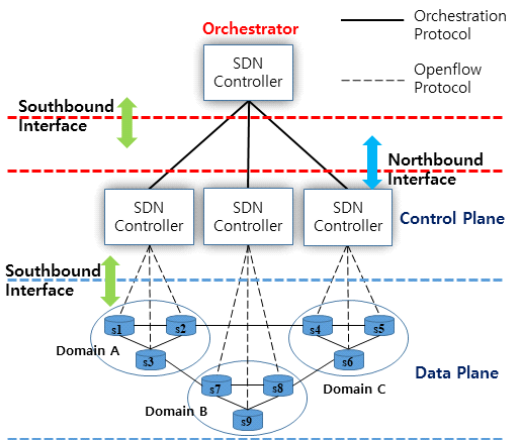

그림 1. 오픈플로 기반 오케스트레이션 프로토콜을 이용한 분산형 다중 도메인 SDN 오케스트레이션 계층 구조
Fig. 1. Distributed Domain SDN Orchestration hierarchy architecture using Openflow-based Orchestration Protocol

2.1.1 Openflow Agent

그림 2에서 보는 바와 같이 하위 계층의 SDN 컨트롤러와 오케스트레이터 간의 오픈플로 기반 오케스트레이션 프로토콜통신을 위해 SDN 컨트롤러에 Northbound 프로토콜로 사용이 가능한 오픈플로 에이전트(Openflow Agent)를 구현한다. 이로 인해 SDN 컨트롤러들로 이루어진 다중 도메인 오케스트레이션 환경에서 컨트롤러와 오케스트레이터간의 호환성이 보장된다.

SDN 컨트롤러는 하위 도메인 정보를 수집하고 인텐트 (작업 수행을 위한 정보)로 생성하여 에이전트에게 전달한다. 수신한 인텐트 정보를 바탕으로 에이전트는 가상 스위치를 생성한다. 가상 스위치(Virtual Switch)는 오케스트레이션 프로토콜을 이용하여 분산 다중 도메인 내 스위치 정보를 오케스트레이터에게 전달한다. 그리고 오케스트레이터는 전달된 정보에 따른 제어 인텐트를 생성하여 SDN 컨트롤러에게 제공한다. 이처럼 에이전트는 도메인 내의 가상 스위치 정보를 관리하고 가상 스위치를 통해 제안된 프로토콜을 이용하여 오케스트레이터와 통신함으로써 SDN 컨트롤러로 구성된 오케스트레이션 환경을 구축한다.

오픈플로 에이전트는 도메인 내 스위치 정보를 기반으로 Port, Flow 정보를 List 형태로 관리한다. Port 정보를 통해 오케스트레이터는 도메인 내외부의 연결 정보를 관리 하면서 토폴로지 정보를 유지한다. Flow 정보는 프로비저닝 기능 제어의 기반이 된다. 그리고 DPID(OpenFlow Datapath ID)는 도메인 내의 스위치 이름을 관리하는 기능이다. 가상 스위치를 통해 DPID가 오케스트레이터에 등록되면 오케스트레이터는 DPID를 중심으로 토폴로지 형성을 위한 과정을 진행한다. 오픈플로 기반 통신 인터페이스 구축을 위해 Openflow J 라이브러리[7]를 사용해 오픈플로 프로토콜 메시지 처리 기능을 구현했다. 클러스터링 구성 및

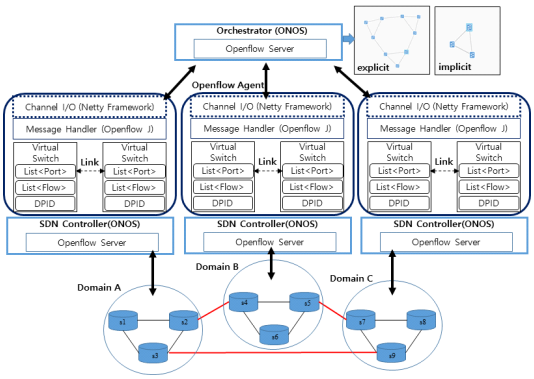

그림 2. 오픈플로 에이전트 구조
Fig. 2. Openflow Agent Architecture

고성능 입출력 기능사용을 위해 Netty 프레임워크[8]를 사용하여 TCP 소켓 처리에서 쓰레드 풀 기반의 Nonblocking Input and output (NIO) 기능을 구현했다.

2.1.2 Data Structure

기존 오픈플로의 데이터 구조에 도메인 내외부의 연결정보만 추가 구현함으로써 에이전트와 오케스트레이터간의 통신으로 쉽게 다중 도메인을 관리한다. 데이터 구조는 Host, Port, Flow등으로 구성되며 스위치의 연결정보를 관리하는 Link를 추가한다. 에이전트는 SDN 컨트롤러의 API(application program interface)를 이용하여 수집된 정보를 Host, Port, Flow, Link 형태의 객체로 생성한다. 생성된 객체는 Agent 오브젝트 타입의 객체 형태로 도메인 전체정보를 관리하고 있는 HashMap 타입의 agentsMap 객체에 추가한다. AppComponent는 Appconfig의 topologyType 값을 이용해 추상화 토폴로지 구축의 유무를 결정한다. 추상화 토폴로지 구축을 위한 다이나믹 매핑 기능으로 새로운 가상 스위치인 imAgent를 만들고 에이전트가 관리하고 있는 도메인 내의 스위치들의 DPID 및 Port 정보를 매핑하여 imAgent에 저장한다. 추상화 토폴로지 유무가 결정되고 나서 Agent는 통신 인터페이스 생성 부분으로 전달되고 Openflow J, Netty를 사용하여 추상화된 토폴로지 내 가상 스위치들을 생성한다.

2.2 Openflow-based Orchestration Protocol

본 논문은 분산형 다중 도메인 SDN 오케스트레이션 계층구조를 구성하기 위해 SDN 컨트롤러와 오케스트레이터 간 인터페이스인 오케스트레이션 프로토콜의 구현에 있다. 오픈플로 기반 오케스트레이션 프로토콜(Openflow-based Orchestration Protocol)을 SDN 컨트롤러의 Northbound 인터페이스와 상위 오케스트레이터(SDN 컨트롤러)의 Southbound 인터페이스 프로토콜로 사용한다. 오케스트레이션 프로토콜은 메시지 절차를 통해 SDN 컨트롤러 간에 종단간 플로우(Flow) 프로비저닝, 토폴로지 탐색 및 토폴로지 이벤트 모니터링 기능을 제공한다.

2.2.1 Protocol Procedure

오케스트레이터와 SDN 컨트롤러는 제안하는 프로토콜의 3단계의 절차를 통해 계층 구조 형성을 위한 주기적 정보교환을 수행하며, 분산형 다중 도메인을 관리한다. 그림 3은 오픈플로 기반 오케스트레이션 프로토콜 절차[9]로 Initialization, Periodic 그리고 Event Stage들로 구성된다.

① Initialization Stage 과정은 가상 스위치를 오케스트레이터에 등록하고, 도메인의 정보를 파악하는 과정이다. 예를 들어 OFPT_FEATURE_REPLY 메시지는 스위치의 기능 정보를 가상 스위치가 오케스트레이터에게 전송 하는 메시지이다. 가상 스위치가 연결을 시도하면 오케스트레이터는 OFPT_FEATURE_REQUEST 메시지를 가상 스위치에게 전송한다. 메시지를 수신한 가상 스위치는 DPID를 OFPT_FEATURE_REPLY 메시지에 실어서 보낸다. 오케스트레이터는 전달 받은 메시지로 가상 스위치의 DPID를 파악하고 이후 절차에서 발생하는 정보를 해당 가상 스위

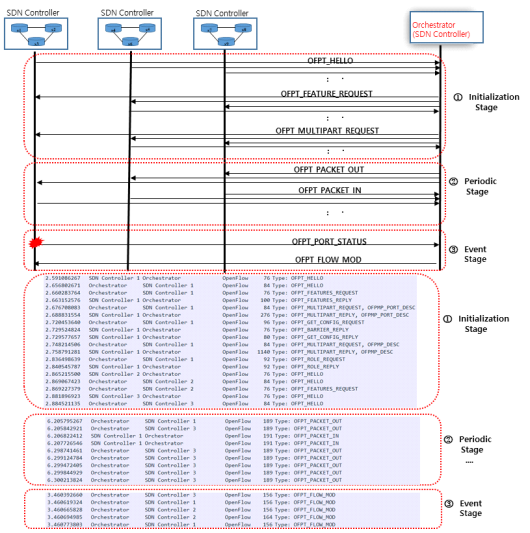

그림 3. 오픈플로 기반 오케스트레이션 프로토콜 메시지 절차 & 오케스트레이션 통신 과정 와이어샤크 캡처
Fig. 3. Openflow-based Orchestration Protocol Message procedure & Orchestration process wireshark capture

치에 저장한다.

② Periodic Stage에서 오케스트레이터는 Controller-driven 커반 방식으로 토폴로지 정보 검증을 위한 probe packet(OFPT_PACKET_OUT)을 주기적으로 사용하여 분산 다중 도메인의 상태를 파악한다. 그리고 Link Layer Discovery Protocol(LLDP) 프로토콜에서 사용하는 데이터를 probe packet에 실어 연결된 모든 가상 스위치에게 보낸다. 가상 스위치는 수신 받은 probe Packet 내의 LLDP 프로토콜 데이터를 이용하여 도메인 내 스위치의 포트 정보(링크정보)가 반영된 LLDP 프로토콜 데이터를 생성하고 'OFPT_PACKET_IN' 메시지에 실어 오케스트레이터에게 전송한다. 오케스트레이터는 연결된 모든 가상 스위치로부터 'OFPT_PACKET_IN' 메시지를 수신 받게 되면 메시지의 LLDP 프로토콜 정보에서 얻게 된 링크 정보와 등록된 가상 스위치 정보를 활용하여 토폴로지 전체 정보를 파악하게 되고 이후 일정 주기로 도메인 정보를 수집한다.

③ Event Stage는 특정 상황에서만 발생한다. 발생 상황은 다음과 같다. 첫 번째는 토폴로지의 정보 변화가 발생하면 가상 스위치로부터 오케스트레이터가 포트 정보 변경 메시지인 OFPT_PORT_STATUS를 수신 받는다. 오케스트레이터는 수신 받은 메시지의 정보를 통해 스위치의 포트 상태를 감지하고 토폴로지를 갱신한다. 두 번째는 가상 스위치가 오케스트레이터로부터 새로운 OFPT_FLOW_MOD 메시지를 수신 받는다. 오케스트레이터는 OFPT_FLOW_MOD 메시지를 기반으로 새로운 플로우를 생성하여 가상 스위치의 플로우(Flow) 테이블을 수정한다.

2.2.2 Topology Discovery & Topology Event Monitoring

제안하는 프로토콜의 토폴로지 관리 특징은 안정 상태(Steady State)에서 폴링 방식을 사용하는 Controller-driven 커반의 토폴로지 탐색 기능과 정보기반(Information-driven) 프로토콜의 이벤트 기반(Event-based) 토폴로지 갱신 기능을 함께 사용 하는 것이다. 네트워크는 그림 3의 절차 ①과 같은 초기 상태(Initialization Stage)가 끝나게 되면 주기적인 반복과 이벤트를 처리하기 위한 안정 상태를 유지한다. 안정 상태는 절차 ②와 같이 probe Packet을 이용하여 컨트롤러가 주기적 정보 교환 규칙성을 가지며 지속적으로 네트워크를 관리하는 평형 상태를 의미한다. 이 주기적인 정보교환은 네트워크 트래픽에서 많은 부분을 차지하고 있다. 따라서 안정 상태에서의 Controller-driven 운영 방식은 네트워크에서 트래픽 발생의 주요 요인이 된다. 제안하는 프로토콜은 안정 상태에서 Controller-driven 토폴로지 탐색 방법으로 도메인을 관리한다. 폴링 방식으로 Controller-driven 토폴로지 탐색 기능을 사용하여 각 도메인이 동일하게 관리됨으로써 부정확한 정보에 따른 비신뢰성과 서로 다른 정보에 따른 네트워크 간의 충돌을 방지하여 신뢰성이 높은 도메인 관리를 보장한다.

다음은 이벤트 기반(Event-based) 토폴로지 갱신 기능으로 안정 상태에서 그림 3의 절차 ③과 같이 임의의 시간에 이벤트(장애)를 감지하고 신속하게 처리한다. 네트워크 관리에서 이벤트를 처리하고 다시 안정 상태로 전환하는 것은 관리 측면에서 매우 중요한 성능이다. 그리고 전환 시 발생하는 트래픽 량 보다 신속하게 안정 상태로 전환하는지가 중요 성능으로 제시되어야 한다. 이벤트 기반 토폴로지 갱신 기능은 오케스트레이터가 다중 도메인 SDN망 장애를 실시간으로 감지하고 즉각적으로 처리하여 빠르게 안정 상태로 전환한다.

오케스트레이션 프로토콜로써 오픈플로 프로토콜의 Controller-driven 토폴로지 탐색 기능과 이벤트 기반 토폴로지 갱신 기능을 이용하기 때문에 신뢰성 높은 도메인 관리가 가능하다. 또한 토폴로지 변화 정보를 실시간으로 인지하고 처리함으로써 신속한 모니터링 성능을 가진다. 결국 오케스트레이션 프로토콜로써 최소 성능을 확보하고 표준 프로토콜 사용으로 이기종 SDN 컨트롤러로 구성된 네트워크에서 호환성도 함께 제공한다.

2.2.3 Dynamic Mapping Mechanism

오픈플로 기반 오케스트레이션 프로토콜에서 신뢰성 있는 도메인 관리를 위해 스위치들 사이의 연결 링크 정보를 탐색하는 Controller-driven 토폴로지 탐색 방식을 사용했다. 하지만 컨트롤러가 주기적으로 모든 링크에 대하여 probe packet을 보내기 때문에 많은 트래픽이 발생하는 단점이 있다. 본 논문에서는 다이나믹 매핑(Dynamic Mapping) 기술이 적용된 추상화 토폴로지를 구축하고 스위치와 링크의 개수를 줄여 발생하는 트래픽 문제의 해결책을 제시한다.

다이나믹 매핑 추상화 토폴로지는 Explicit 또는 Implicit 형태가 있다. Explicit 형태는 도메인 내 스위치와 링크들의 정보를 그대로 표현한 방법이며 Implicit 형태는 도메인을 하나의 가상 스위치에 매핑하여 도메인 내의 연결 정보가 축약된 추상화된 토폴로지 표현 방법이다. 그림 4는 제안된 프로토콜을 이용하여 추상

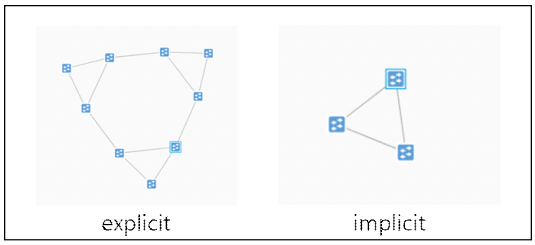

그림 4. Abstraction Topology의 결과
Fig. 4. Result of abstraction topology

화 토폴로지를 구현한 예를 보여준 것이다. 그림처럼 Implicit Topology는 Explicit Topology 구성과 비교하여 9개의 스위치 대신 도메인별로 하나씩 3개의 가상 스위치만을 사용하여 적은 자원으로 토폴로지가 관리 된다. 따라서 추상화 토폴로지 구성은 오픈플로 프로토콜의 probe packet 사용으로 발생하는 트래픽 문제의 대안이 된다.

다이나믹 매핑을 구현하기 위해 'OFPT_FEATURE_REPLY ', 'OFPT_MULTIPART_REPLY', 'OFPT_PACKET_IN' 메시지에 다이나믹 매핑을 적용한다. 매핑 과정을 통해서 생성된 가상 스위치의 ID(DPID) 값을 OFPT_FEATURE_REPLY 메시지 구조의 'datapath-id' 부분에 대입한다. 그림 5는 다이나믹 매핑이 적용된 정보를 이용하여 OFPT_FEATURE_REPLY 메시지를 만들어 오케스트레이터에게 전송하는 과정이다. Explicit의 경우 모든 스위치의 DPID가 오케스트레이터에게 전달되며 Implicit의 경우 도메인마다 도메인의 정보들이 매핑 되어있는 하나의 가상 스위치의 DPID 정보가 전송된다.

다이나믹 매핑이 적용된 가상 스위치의 포트 정보를 스위치의 상태 정보를 가지고 있는 OFPT_MULTIPART_REPLY 메시지의 'body' 부분에 대입하여 보내준다. 그림 6은 가상 스위치의 포트 정보를 이용하

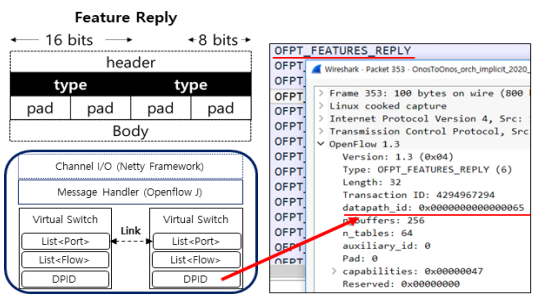

그림 5. 다이나믹 매핑이 적용된 OFPT_FEATURE_REPLY 메시지
Fig. 5. OFPT_FEATURE_REPLY message with dynamic mapping applied

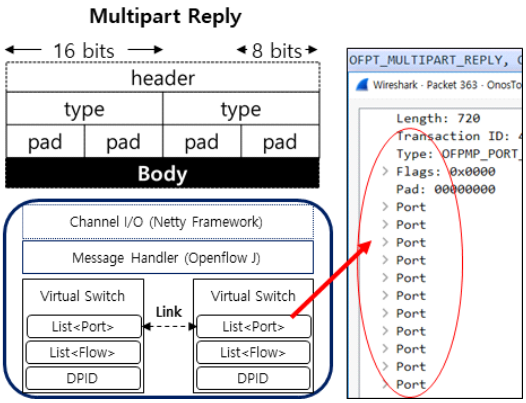

그림 6. 다이나믹 매핑이 적용된 OFPT_MULTIPART_REPLY 메시지
Fig. 6. OFPT_MULTIPART_REPLY message with dynamic mapping applied

여 OFPT_MULTIPART_REPLY 메시지를 만들어 오케스트레이터에게 전송하는 과정이다. 메시지의 'body'에 있는 데이터를 이용하여 오케스트레이터는 데이터베이스에 저장되어 있는 가상 스위치의 포트 정보를 설정한다.

가상 스위치는 OFPT_PACKET_OUT(probe packet) 메시지에 있는 LLDP 프로토콜 정보를 다이나믹 매핑이 적용된 LLDP 프로토콜 형태의 정보로 OFPT_PACKET_IN 메시지에 담아 오케스트레이터에게 전송한다. 오케스트레이터는 메시지의 LLDP 프로토콜 데이터를 활용하여 도메인 내부의 연결 정보 및 도메인 간의 외부 연결 정보를 이용해 Explicit 또는 Implicit 형태로 토폴로지를 형성한다.

Ⅲ. 실험 및 결과

이 장에서는 제안된 오픈플로 기반 오케스트레이션 프로토콜의 성능을 측정하기 위해 테스트베드를 구현하고 토폴로지 탐색(Topology Discovery)과 토폴로지 이벤트 모니터링(Topology Event Monitoring) 실험을 진행했다. 측정결과를 이용하여 REST API의 정보기반(Information-Driven) 폴링 방식을 이용한 실험 결과와 성능 비교를 수행하고 제안된 프로토콜의 성능의 장단점을 밝힌다. 그리고 추상화 토폴로지(Abstraction Topology)를 구현하여 트래픽 성능을 측정하고 오픈플로 프로토콜의 트래픽 문제 해결방안을 검증한다. 실험을 통해 제시했던 Controller-driven[3] 폴링 방식 토폴로지 탐색 방식과 이벤트 기반 토폴로지 갱신 기능 처리가 가능한 오픈플로 기반 오케스트레이션

프로토콜이 오케스트레이션 프로토콜로 사용 가능함을 입증한다.

3.1 실험 구조

그림 7 왼쪽은 본 논문에서 진행하는 실험 구조 및 결과를 캡쳐한 화면으로 Explicit 방식과 Implicit 방식을 보여준다. 오른쪽은 제안된 오케스트레이션 프로토콜 기반으로 계층화된 3개의 다중 도메인 SDN 망 환경 구성이다. 표 1과 같이 한 개의 오케스트레이터(상위 계층 SDN 컨트롤러), 3개의 SDN 컨트롤러(하위 계층 SDN 컨트롤러), 스위치 망을 에뮬레이션하는 Mininet[10]으로 구성되어 있다. 여기서는 소단원에 관한 내용을 간단히 살펴본다. 여기서는 소단원에 관한 내용을 간단히 살펴본다.

표 1은 실험환경에 대한 정보이다. 가상 머신은 5개를 사용했으며 CPU는 Intel Core i7-7700 HQ 2.80 GHz를 공유 하고 최대 사용 Core의 수는 4개를 할당받아 사용했다. OS는 Ubuntu 16.04를 사용하고 RAM은 6GB 그리고 디스크의 용량은 20GB을 할당받아 구성되었다. SDN 컨트롤러는 ONOS 1.7[11] 버전을 사용했다. 실험은 Wireshark 도구를 이용하여 프로토콜의 성능을 측정하였다.

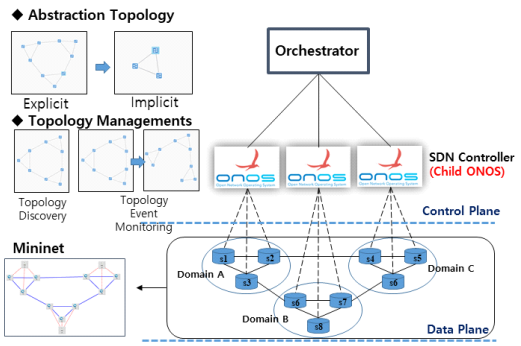

그림 7. 실험 구조 및 결과 화면
Fig. 7. Experiment Architecture and screenshots

표 1. 실험환경 사양
Table 1. Experiment environment specification

Protocol	CPU	RAM	Hard Disk	OS
Orchestrator	Intel Core i7-7700HQ 2.80GHz	6GB	20GB	Ubuntu 16.04
SDN Controller (Child ONOS)	Intel Core i7-7700HQ 2.80GHz	6GB	20GB	Ubuntu 16.04
Mininet	Intel Core i7-7700HQ 2.80GHz	6GB	20GB	Ubuntu 16.04

3.2 성능 Metric

본 실험 검증에서는 안정 상태에서의 토폴로지 탐색과 토폴로지 이벤트 모니터링 성능을 측정하기 위해 토폴로지 탐색과 Port down 이벤트에 대한 성능을 이용하여 제안된 프로토콜이 최소 성능이 보장 가능함을 입증한다. 오케스트레이션 프로토콜의 정확하고 유의미한 성능 측정을 위해 IETF에서 제시한 SDN 컨트롤러 성능 측정 표준 메트릭[12]을 기준으로 성능을 측정한다.

먼저 '토폴로지 탐색(Topology Discovery)' 메트릭은 RFC8456 문서[12]의 Network Topology Discovery Time Metric을 따르며 그림 8과 같다. 그림8은 절차 ① Initialization stage 이후 토폴로지 링크의 정보를 수집하기 위해 오케스트레이터에서 보내는 첫 번째 메시지(OFPT_PACKET_OUT)부터 하위 SDN 컨트롤러가 보낸 링크 정보가 담긴 메시지(OFPT_PACKET_IN)의 마지막 메시지 도착까지의 메트릭 절차를 의미한다. 실험을 통해 해당 메트릭 절차 동안 소모된 트래픽량을 측정하고 probe Packet(OFPT_PACKET_OUT)을 사용하여 신뢰성 있는 도메인 관리가 가능함을 보인다.

다음으로 토폴로지 이벤트 모니터링 메트릭은 Network Topology Change Detection Time Metric[12]을 따르며 그 절차는 그림9와 같다. 그림 9는 포트가 단절되는 이벤트(Port Down) 발생 시 포트 변경 알림 메시지(OFPT_PORT_STATUS)를 실시간으로 오픈플로 서버가 수신하게 된다. 기존에는 토폴로지 갱신을 Controller-driven 방식의 주기성을 가진 폴링 방식을 사용해 지연시간이 발생했다. 하지만 본 논문에서는 오픈플로 서버가 이벤트 기반 토폴로지 갱신 기능을 사용하여 실시간으로 토폴로지 변화를 인지하게

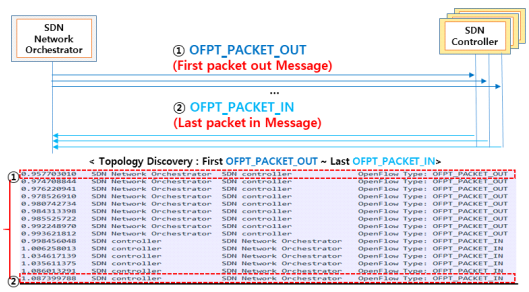

그림 8. Openflow based Orchestration Protocol의 토폴로지 탐색
Fig. 8. Topology Discovery of Openflow based Orchestration Protocol

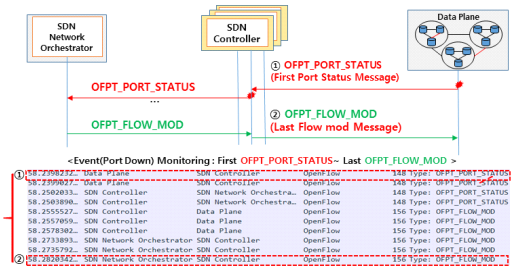

그림 9. Openflow based Orchestration Protocol의 이벤트 (Port Down) 모니터링
Fig. 9. Event(Port Down) Monitoring of Openflow based Orchestration Protocol

되고 즉각적으로 토폴로지를 갱신한다. 또한 지연시간이 발생되지 않고 스위치의 플로우 테이블 변경 메시지(OFPT_FLOW_MOD)를 스위치에게 전송하고 안정 상태로 전환한다. 따라서 해당 이벤트의 성능 측정 구간은 인용한 메트릭의 시작인 이벤트 감지부터 오케스트레이터가 보낸 플로우 테이블 변경 메시지가 데이터 플레인의 스위치까지의 도착한 시간을 측정하여 신속한 모니터링 기능이 가능함을 측정한다. 토폴로지 탐색의 경우 도메인 개수를 늘려 각각 5번 실험을 진행한다. 토폴로지 이벤트 모니터링의 경우에는 폴링 주기의 변화로 각 주기마다 15초 간격으로 15번 실험을 하고 95% 신뢰구간으로 표시되어 진행한다. 실험에 대한 결과는 데이터의 평균을 기반으로 제시한다.

3.3 실험 결과

그림 10은 토폴로지 탐색 트래픽 성능 비교 그래프다. 도메인 개수를 증가시켜 개수에 따라 토폴로지 탐색을 수행할 때 발생하는 트래픽량 변화를 측정했다. 도메인 구성은 하나의 컨트롤러와 컨트롤러가 관리하는 3개의 스위치로 구성되었다. REST와 비교하여 최소 5.1배, 최대 5.5배 트래픽을 더 소모했다. 하지만 다이나믹 매핑을 이용해 추상화 토폴로지를 구현함으로써 단점인 트래픽량을 최소 5.1배 차이에서 최소 3.3배, 최대 3.9배 감소시켰다. 제안하는 프로토콜은 토폴로지 탐색에 있어 probe packet(OFPT_PACKET_OUT(LLDP))을 사용한 폴링 방식 사용으로 REST의 폴링 방식과 비교하여 트래픽과 시간을 더 소모한다. 하지만 다이나믹 매핑 방식을 이용한 추상화 토폴로지 구현으로 REST와의 성능 차이를 줄였다.

그림 11은 토폴로지 이벤트 모니터링 성능 비교 그래프다. 이벤트(Port Down) 발생 상황에서 주기가 증가함에 따라 REST에 비해 51배, 155배 169배, 296배

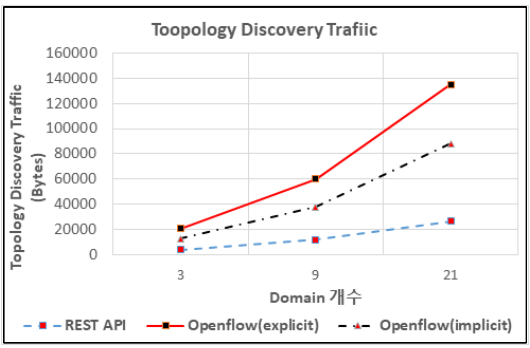

그림 10. REST API와 오프플로 기반 오케스트레이션 프로토콜을 이용한 토폴로지 탐색 트래픽 실험 결과
Fig. 10. Topology Discovery Traffic experiment result using REST API and Openflow-based Orchestration Protocol

빠르게 이벤트 검출 후 안정 상태로 전환했다. 뿐만 아니라 이벤트 기반의 토폴로지 갱신 기능을 통해 Explicit, Implicit 토폴로지 구성과 상관없이 실험 전체에서 평균 27 ms (95% 신뢰구간 23 ms-30 ms)에서 100 ms 이내 이벤트를 처리했다. 실험을 통해 REST에 비해 주기와 상관없이 실시간으로 토폴로지 이벤트 모니터링 기능 제공이 가능함을 확인했다.

오픈플로 기반 오케스트레이션 프로토콜은 실험 결과를 통해 주기변화와 상관없이 실시간으로 정보기반 토폴로지 이벤트 처리 기능을 제공했다. 또한 오픈플로의 Controller-driven 토폴로지 탐색 기능 사용으로 REST의 폴링방식 비해 토폴로지 관리에 신뢰성이 높다. 비록 Controller-driven의 토폴로지 탐색 기능사용으로 많은 트래픽이 발생하는 제한점을 확인했지만 추상화 토폴로지 구현으로 해결방안을 제시했다. 뿐만 아니라 표준프로토콜 사용으로 SDN 컨트롤러 재사용

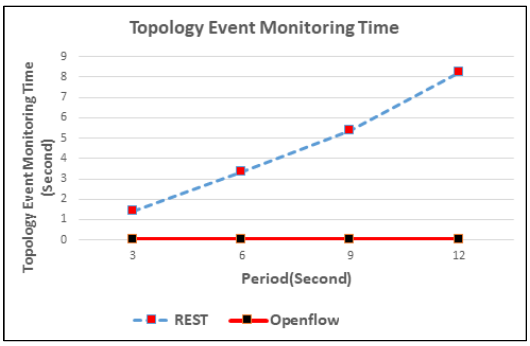

그림 11. REST API 와 오픈플로 기반 오케스트레이션 프로토콜을 이용한 토폴로지 이벤트 모니터링 시간 실험 결과
Fig. 11. Topology Event Monitoring Time experiment result using REST API and Openflow-based Orchestration Protocol.

이 가능하여 호환성을 가진 다양한 오케스트레이션 환경 구축이 가능함을 입증했다.

Ⅳ. 결 론

본 논문에서는 SDN 표준 오픈플로 프로토콜 기반 오케스트레이션 프로토콜(Openflow-based Orchestration Protocol)에 대한 구현, 실험 및 검증을 진행했다. 실험 결과를 통해 REST의 폴링 방식과 비교하여 이벤트를 최소 51배 빠르게 처리했다. 특히 폴링 주기와 무관하게 평균 27ms(95% 신뢰구간 23ms-30ms) 시간으로 이벤트를 처리해 안정적인 토폴로지 이벤트 모니터링 성능을 보여주었다. 비록 토폴로지 탐색 실험에서 5.5배 많은 트래픽을 소모하여 제한점을 확인했지만, 다이나믹 매핑을 사용한 추상화 토폴로지 구성 기능을 통해 트래픽 발생량 차이를 3.2배로 줄여 트래픽 발생량 문제를 완화했다.

본 논문에서 제시된 오픈플로 기반 오케스트레이션 프로토콜(Openflow-based Orchestration Protocol) 장점을 요약하면 다음과 같다.

첫째, 표준 프로토콜을 재사용했기 때문에 다른 프로토콜과 비교하여 오케스트레이션 프로토콜 제작에 있어 편리성을 갖고 시장 확산 시간을 단축시킨다. 둘째, 오픈플로 프로토콜을 활용한 통신 방식이기 때문에 SDN 컨트롤러를 추가적인 구현 없이 오케스트레이터로 재사용 할 수 있다. 셋째, 표준화된 프로토콜 사용으로 이기종 SDN 컨트롤러 및 네트워크 간의 상호 운용성을 가진다. 넷째, Controller-driven 토폴로지 탐색 기능사용으로 신뢰성 높은 도메인 관리 기능과 이벤트 기반 토폴로지 갱신 기능사용으로 토폴로지 변화에 대한 즉각적인 처리로 오케스트레이션 프로토콜로써의 성능이 보장된다. 다섯째, 다이나믹 매핑 메커니즘이 적용된 추상화 기법을 통해 트래픽이 감소할 뿐만 아니라 탐색 시간이 감소하는 효과를 얻었다. 마지막으로 이미 개발된 오픈소스 기반의 SDN 컨트롤러를 사용하여 가상화된 토폴로지 구성과 다양한 규모의 네트워크에 대한 관리의 효율성을 가진다.

표준 오픈플로 프로토콜 기반 오케스트레이션 프로토콜은 네트워크 환경 변화에 대한 유용성, 이질적 서비스간의 호환성, 가상화된 토폴로지 관리 신뢰성 등 시장의 요구 사항을 수용할 수 있는 오케스트레이션 프로토콜로써 나아가야하는 방향성을 제시했다.

비록 본 연구에서는 토폴로지 탐색 및 토폴로지 이벤트 모니터링에 대한 제한된 메트릭 분석 연구에 한정되었지만 추후에 다른 메트릭과 종단간 흐름을 프로비저닝(Provisioning) 하기 위한 연구가 진행 되어야 한다. 또한 오픈플로 오케스트레이션 프로토콜의 장점인 빠른 이벤트 기반 토폴로지 갱신 기능의 성능과 Controller-driven 의 신뢰성 높은 도메인 관리 기능을 유지하고 데이터 트래픽 발생량 해결을 위한 오픈플로 오케스트레이션 프로토콜 개선이 추후 연구로 필요하다.

References

[1] J. S. Choi and X Li, "Hierarchical distributed topology discovery protocol for multi-domain SDN networks," *IEEE Commun. Lett.*, vol. 21, pp. 773-776, 2017.

[2] A. Mayoral, et al., "Control orchestration protocol: Unified transport api for distributed cloud and network orchestration," *J. Optical Commun. and Netw.*, vol. 9, no. 2, pp. 216-222, 2017.

[3] S. Choi, S. Kang, and Y. Lee, "Design and evaluation of a PCEP-based topology discovery protocol for stateful PCE," *Optical Switching and Netw.*, vol. 26, no. 11, pp. 39-47, 2017.

[4] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. Dissertation, University of California, Irvine, 2000.

[5] Z. Haojun, "Design and implementation of an OpenFlow-based orchestration protocol for multiple SDN controllers," M.S. Thesis, Dept. Computer Science, Hanyang Univ., Seoul, Korea, 2018.

[6] X. Li, "Design and evaluation of BGP-LS based hierarchical distributed topology discovery protocol for multi-domain SDN networks," M.S. Thesis, Dept. Computer Science, Hanyang Univ., Seoul, Korea, 2017.

[7] A. Wundsam(2018), Retrived Jun. 2020, from https://github.com/floodlight/loxigen/wiki/OpenFlowJ-Loxi

[8] Netty Framework, Retrived Jun. 2020, from https://netty.io

[9] Openflow, Retrived Jun. 2020, from https://www.opennetworking.org/wp-content/uploads/

2014/10/openflow-spec-v1.3.2.pdf
[10] Mininet, Retrived Jun. 2020, from http://mininet.org/
[11] ONOS, Retrived Jun. 2020, from https://wiki.onosproject.org/display/ONOS/Southbound:+Protocol,+Providers,+Drivers
[12] V. Bhuvaneswaran, et al., "*Benchmarking methodology for software-defined networking (SDN) controller performanceents*," RFC 845, Oct. 2018.

이 길 호 (Gil-ho Lee)

2017년 2월 : 삼육대학교 컴퓨터학부 학사
2018년 2월~현재 : 한양대학교 컴퓨터 소프트웨어 학과 석사과정

<관심분야> SDN, NFV, 가상화, Cloud, AI, Smart Home, Smart Factory, Big Data

권 원 식 (Won-sic Kwon)

2020년 2월 : 경북대학교 컴퓨터학부 졸업
2020년 3월~현재 : 한양대학교 컴퓨터 소프트웨어공학 박사과정

<관심분야> SDN, NFV, Cloud, AI, Smart Home, Smart Factory, Big Data

천 세 준 (Se-joon Chun)

2015년 2월 : 한양대학교 공과대학 컴퓨터공학부 졸업
2017년 3월~현재: 한양대학교 컴퓨터 소프트웨어 박사 과정

<관심분야> SDN, NFV, Network Hypervisor, 가상화, 제어 관리 프레임워크
[ORCID:0000-0002-4711-2427]

최 진 식 (Jin-seek Choi)

1985년 2월 : 서강대학교 전자공학과 학사
1987년 2월 : 한국과학기술원 네트워크 석사
1995년 2월 : 한국과학기술원 네트워크 박사
2004년~현재 : 한양대학교 컴퓨터소프트웨어학부 교수

<관심분야> Network control and management framework, energy management framework for smart-Grid, software-defined networking, mobile IP, carrier Ethernet, switching and Routing
[ORCID:0000-0003-1554-3879]