소프트웨어 정의망 실험을 위한 하이브리드 방식의 테스트베드 구축

장수지, 윤승현, 김성환, 임 혁*

광주과학기술원

sujijang@gm.gist.ac.kr, {seunghyunyoon, sunghwankim, hlim}@gist.ac.kr

Constructing Hybrid-Type Testbed for Software-Defined Networking Experiments

Suji Jang, Seunghyun Yoon, Sunghwan Kim, Hyuk Lim*

Gwangju Institute of Science and Technology (GIST)

요 약

네트워크의 제어 평면을 데이터 전달 평면과 분리하여 중앙의 제어기를 통해 네트워크를 유연하고 민첩하게 관리할 수 있는 software-defined networking (SDN) 기술은, 네트워크 모니터링 및 중앙 집중식 제어 등의 장점으로 인해 활발하게 사용되고 있다. 이에 따라 SDN의 연구 및 실험 수행을 위한 테스트 환경 구축이 필요하다. 기존의 SDN 실험 환경은 대부분 에뮬레이션 기반의 Mininet 또는 상용 SDN 하드웨어 스위치로 구성한 하드웨어 테스트베드로 구성되었다. 하지만 Mininet은 에뮬레이션 환경으로 실제 네트워킹 환경과의 성능 차이가 있다는 단점이 있고, 하드웨어 테스트베드는 구현에 비용이 많이 들고 확장성이 없다는 단점이 있다. 본 논문에서는 각각의 단점을 상호보완하기 위해 상용 SDN 하드웨어 스위치와 가상화 기술 기반의 소프트웨어 스위치 및 컨테이너를 활용한 하이브리드 방식의 테스트베드 구조를 제안한다.

Ⅰ. 서 론

소프트웨어 정의 네트워킹(software-defined networking, SDN)은 네트워크 시스템의 트래픽 전송을 수행하는 데이터 평면과 트래픽 경로를 지정하는 제어 평면을 분리한 구조로 트래픽 전달 동작을 프로그래밍할 수 있는 기술이다 [1]. 제어 평면의 SDN 제어기는 트래픽 전달을 위한 경로를 설정하여 데이터 평면의 스위치로 OpenFlow [2] 프로토콜을 이용하여 전달하고, 이러한 중앙 집중화된 구조는 동적인 트래픽 흐름 제어 [3], 유연한 네트워크 리소스 관리 및 트래픽 QoS(Quality of Service) 만족과 같은 많은 네트워크 최적화 문제 해결이 가능하다.

SDN 실험을 위한 기존 테스트베드 환경으로는 에뮬레이션 기반의 Mininet [4]과 하드웨어 기반의 테스트베드 [5]가 있다. Mininet은 가상 테스트베드와 개발 환경을 제공하는 SDN 에뮬레이터로, 가상 시스템과 물리적 네트워크 인터페이스 연결을 제공하는 소프트웨어 기반 OvS(Open vSwitch)와 단일 호스트 상에서 여러 개의 독립된 리눅스 시스템들을 실행할 수 있는 Linux container로 구성된다. Mininet은 적은 비용으로 실험을 수행할 수 있다는 장점이 있지만, 시뮬레이션 된 환경과 실제 환경 간의 성능 차이가 있고 규모의 확장성이 제한되어 있다는 단점이 있다. 또한, [5]에서 제안된 OpenFlow 지원 상용 스위치로 구성된 하드웨어 기반의 테스트베드는 실제 네트워크와 유사한 환경에서 실험을 할 수 있다는 장점이 있지만, 테스트베드 구성에 많은 비용이 필요하고 대규모의 네트워크를 구축하기 어렵다.

본 논문에서는 [5]에서 제안한 하드웨어 기반 테스트베드를 확장하여, 상용 SDN 하드웨어 스위치, OvS 소프트웨어 스위치와 가상화 플랫폼인 Docker container를 함께 활용한 하이브리드 테스트베드 구조를 제안한다. 제안한 테스트베드는 실제와 유사한 환경에서 네트워크 실험 및 성능 측정을 할 수 있으며, 가상화 기술을 활용하여 대규모의 확장성 있는 토폴로지 생성이 가능한 것을 확인하였다.

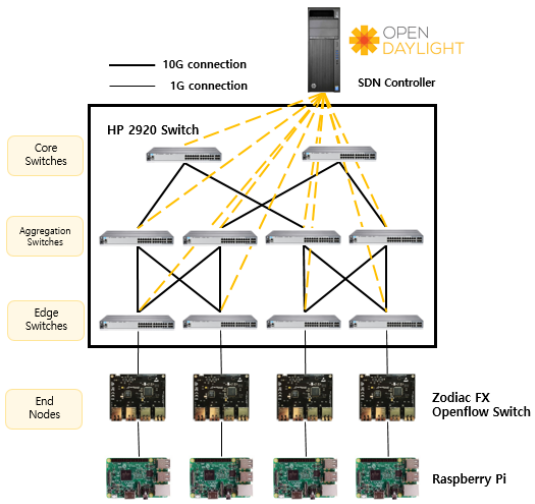

그림 1.하드웨어 테스트베드 fat-tree 토폴로지

Ⅱ. SDN 실험을 위한 기존 테스트베드 환경

Mininet은 SDN 에뮬레이터로 물리적 네트워크 연결 없이 복잡한 토폴로지 실험 수행이 가능하고 새로운 기능을 테스트할 수 있지만, 하드웨어 스위치보다 성능이 낮은 소프트웨어 스위치에 의해 전달되므로 100Mb/sec 이하의 느린 링크를 사용해야 한다. 또한, 일반적으로 에뮬레이션 시스템이 확장됨에 따라 속도가 느려지는 경향이 있다 [5].

하드웨어 SDN 테스트베드는 SDN을 지원하는 실제 상용 스위치를 활용하여 구축할 수 있다. 그림 1은 OpenFlow 1.3을 지원하는 HP 2920 스위치 10개로 구축한 테스트베드 환경을 보여준다. Fat-tree 구조는 데이터센터에서 보편적으로 활용되는 네트워크 구조로 네트워크에 대규모의 트래픽이 흐를 때 병목현상을 방지하기 위한 core 스위치, aggregation 스

위치, edge 스위치의 3개의 층으로 구성된다. HP 스위치에는 Zodiac FX OpenFlow 스위치와 호스트로 사용되는 Raspberry Pi가 연결되어 있고, OpenDaylight을 SDN 제어기로 사용한다. 하드웨어 테스트베드는 구조의 유연성이 없어 토폴로지 확장에 비용이 많이 들고 대규모 토폴로지 생성에 제약이 있다.

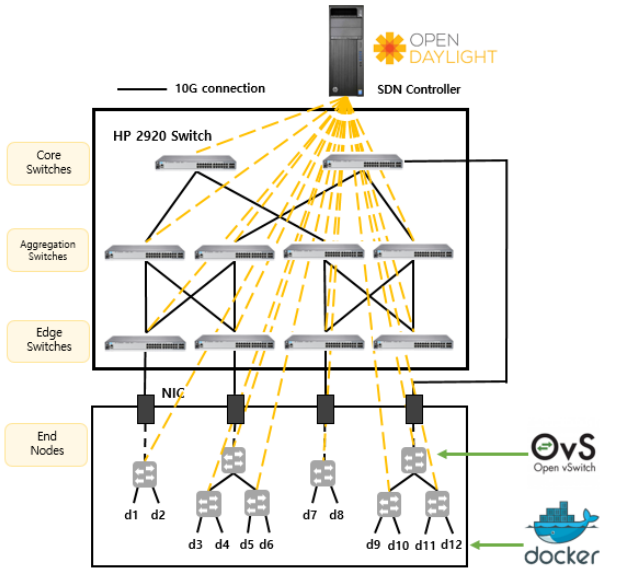

그림 2. 구축한 소프트웨어 기반의 테스트베드 fat-tree 토폴로지

III. 확장성을 고려한 하이브리드 방식의 테스트베드

본 논문은 Mininet과 하드웨어 테스트베드 각각의 단점을 상호보완하기 위한 하이브리드 방식의 테스트베드를 제안한다. 그림 2는 구축한 소프트웨어 기반의 테스트베드 fat-tree 토폴로지이다. 제안한 테스트베드는 기존 테스트베드 [4]의 하드웨어 스위치(Zodiac FX)와 호스트(Raspberry Pi)를 가상 스위치인 OvS와 가상화 플랫폼인 Docker container로 대체하여 HP 스위치에 연결한 구조이다. Docker container를 사용하여 단일 환경에서 여러 호스트의 생성이 가능하며, OvS를 사용하여 여러 소프트웨어 브리지를 생성할 수 있다. OvS와 Docker container를 연결하여 다양한 구조의 대규모 토폴로지 생성이 가능해져 네트워크의 확장성이 개선되었다. 또한, 네트워크의 core 및 aggregation 스위치에도 가상환경 토폴로지의 연결이 가능하고, 독립된 토폴로지를 다른 Ethernet device로 매핑시켜 각각 다른 하드웨어 스위치로 연결이 가능하다. 제안한 하이브리드 테스트베드에서는 호스트 및 스위치에 필요한 리소스(e.g., Memory, CPU or I/O)가 격리되므로 성능 충실도가 보장되어 과부하로 인한 지연을 방지할 수 있다.

IV. 데이터 전송 실험 결과

제안한 하이브리드 테스트베드의 성능을 평가하기 위하여 Mininet 기반 가상환경 테스트베드와 비교 실험하였다. 각각의 테스트베드에서 두 개의 컨테이너 기반 호스트 사이에 스위치가 1개로 연결된 경우와 7개로 연결된 경우를 나누어 통신 왕복 지연 시간인 Round Trip Time (RTT)를 측정하였다. 그림 3은 제안된 테스트베드와 가상환경 테스트베드에서 컨테이너 간의 통신 시간 그래프이다. 호스트가 1홉 거리에 있는 경우는 제안한 하이브리드 테스트베드와 가상환경 기반의 테스트베드의 RTT가 비슷하지만, 7홉이 떨어져 있는 경우 가상환경 테스트베드에서는 스위치의 전송 지연이 잘 반영되지 않아, 1홉의 RTT와 큰 차이가 없다. 반면, 제안한 테스트베드에서는 홉 수가 늘어날수록 스위치의 전송지연이 반영되어 RTT가 증가한 것을 알 수 있다. 이 결과는 제안한 테스트베드가 가상환경 테스트베드보다 실제 네트워크 환경을 잘 반영하여 보다 정확한 시간 지연 결과를 얻을 수 있다는 것을 보여준다.

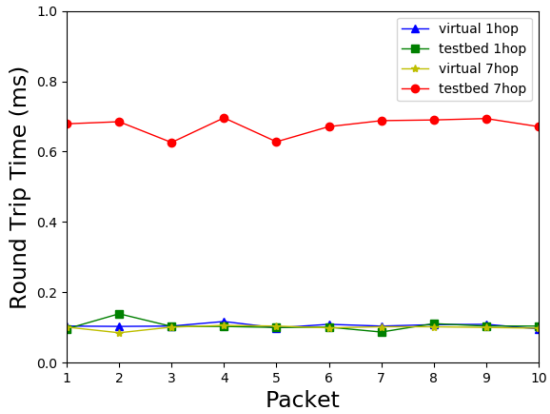

그림 3 가상환경 테스트베드와 제안한 테스트베드의 성능 비교

V. 결론

본 논문에서는 SDN 기술 구현을 위한 가상 에뮬레이터인 Mininet과 하드웨어 기반의 테스트베드의 단점을 상호보완한 하이브리드 방식의 테스트베드 구조를 제안하였다. Mininet기반 가상환경 테스트베드와 비교하여 토폴로지 구조에 따른 왕복 지연 시간 측정을 통해, 제안한 하이브리드 기반의 테스트베드가 실제 환경과 유사한 실험 결과를 얻을 수 있다는 것을 확인하였다.

ACKNOWLEDGMENT

이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2017-0-00421, 새로운 보안 위협에 대처하기 위한 사이버 보안 방어 순환 기술).

참 고 문 헌

[1] S. Agarwal *et al.*, "Traffic Engineering in Software Defined Networks", *IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[2] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Communication Review*, 2008.

[3] S. Yoon *et al.*, "Fast Controller Switching for Fault-Tolerant Cyber Physical Systems on Software-Defined Networks," *IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2017.

[4] R. De Oliveira *et al.*, "Using Mininet for Emulation and Prototyping Software-Defined Networks", *IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2014.

[5] S. Kim *et al.*, "Secure Collecting, Optimizing, and Deploying of Firewall Rules in Software-Defined Networks", *IEEE Access*, 2019.

[6] J. Ortiz *et al.*, "Evaluation of Performance and Scalability of Mininet in Scenarios with Large Data Centers", *IEEE Ecuador Technical Chapters Meeting (ETCM)*, 2016.