The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 19, No. 4, pp.13-19, Aug. 31, 2019. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2019.19.4.13
JIIBC 2019-4-3

SDN Controller 배포를 이용한 산업 분산형 네트워크 관리 기법

Method for Industrial Distributed Network Management using SDN Controller Deployment

박도권*, 정종필**

Do Gun Park*, Jongpil Jeong**

요 약 SDN은 통신에서의 트래픽 문제를 해결하기 위해 가장 활발하게 연구되고 있는 주제 중 하나이다. SDN은 진보된 API를 통해 네트워크 자원을 가상화하여 하나의 물리적 네트워크에서 다중 네트워크를 구현한다. Network Function Virtualized(NFV)는 네트워크 기능을 소프트웨어 인스턴트, 가상화 기술을 사용하는 하드웨어에서 VNF으로 배포한다. 이러한 기능은 IP, 라우터등을 가상화하여 네트워크 관리를 쉽게하고 성능을 향상시킨다. 본 논문은 가상화된 산업 네트워크에서 SDN 배포를 통해 SDN의 분산 컨트롤러 효과를 제공하고 트래픽을 제어하는 방식을 제안하였다. SDN 배포는 제안된 기법을 사용시 트래픽 관리를 보다 효율적으로 할 수 있으리라 기대한다.

Abstract SDN is one of the most actively researched topics to solve traffic problems in communication. SDN implements multiple networks in a single physical network by virtualizing network resources through an advanced API. Network Function Virtualized (NFV) distributes network functions from hardware using software instant, virtualization technology to VNF. These features make network management easier and improve performance by virtualizing IP, routers, and so on. In this paper, we propose a method to control the traffic and provide the distributed controller effect of SDN through SDN distribution in the virtualized industrial network. It is expected that SDN distribution will be able to manage traffic more efficiently when using the proposed scheme.

Key Words : SDN, NFV Distributed network

Ⅰ. 서 론

산업 네트워크는 현재 가장 급격하게 성장하고 있는 중요한 네트워크 중 하나이다. 이미 기존의 네트워크의 반응속도와 트래픽 해결능력으로는 산업 네트워크의 발전에 따라 요구하는 반응속도와 데이터 전송량을 따라가는 것이 쉽지 않다. 특히나 클라우드 서비스를 통한 데이터수집, 산업내 발생하는 엄청난 양의 데이터 등은 새로운 형태의 네트워크를 요구하게 되었다.

SDN(Software Define Network)는 이러한 네트워

*준회원, 성균관대학교 스마트팩토리융합학과 석사과정
**정회원, 성균관대학교 스마트팩토리융합학과(교신저자)
접수일자 2019년 7월 2일, 수정완료 2019년 7월 24일
게재확정일자 2019년 8월 2일

Received: 2 July, 2019 / Revised: 24 July, 2019 /
Accepted: 2 August, 2019
Corresponding Author: jpjeong@skku.edu
Department of Smart Factory Convergence, Sungkyunkwan University, Korea

크에 대한 요구를 충족시키기 위해 제안되었다. SDN 기술은 API를 활용하여 네트워크 구성 요소를 기존의 프로그래밍을 통해 관리하고, 기존에 구축된 네트워크에 가상화를 통해 적용할 수 있다.[1] 특히 제어계층(Control plane)과 데이터계층(Data plane)을 분리함으로써 유지보수와 확장에 용이하도록 구축할 수 있다. 이는 SDN Controller에 의해 모든 패킷의 경로가 결정되기 때문에 기존의 네트워크 라우팅 프로토콜보다 정밀한 트래픽 관리를 가능하게 한다. 이러한 SDN의 장점을 이용하여 기존의 산업과 네트워크에 접합하기 위한 여러 연구가 이루어지고 있다.[2,3]

기존의 네트워크에서는 각 스위치 노드가 스스로 routing을 통해 패킷을 routing하였다. 소규모 네트워크 구축에는 유리할 수 있으나 대규모 네트워크 제어에선 스위치 노드에 부하를 준다. SDN은 Control plane에선 SDN Controller를 통해 provisioning하여 데이터 트래픽의 경로를 설정함으로 스위치 노드의 부하를 감소시키고 트래픽을 효율적으로 관리할 수 있다.[4]

현재 산업현장에서 SDN 구축 방식은 SDN 컨트롤러가 클라우드에서 제어하는 중앙 집중형 구조이다. 이러한 SDN 컨트롤러는 관리하기 용이하고 비용이 저렴하나 모든 경로를 SDN 컨트롤러가 관리함으로 컨트롤러의 부하를 야기한다. 따라서 컨트롤러의 부하를 막기 위해 vSDN(Virtualized SDN)가 제시되었다.[5] 하지만 모든 네트워크의 자원을 공유해야하도록 NFV(Network Function Virtualization)을 통해 구축해야하며 일반적인 트래픽 전송만을 고려하여 설계되었기 때문에 산업 네트워크에서는 사용할 수 없다.

SDN 컨트롤러 여러 대를 설치하는 분산형 SDN 구조는 주로 통신관련 업체에서 사용하는 구조이다. P. Kevin은 Distributed SDN Control plane인 DISCO를 제안하였다[6]. 분산형 SDN 구조는 여러 대의 컨트롤러가 트래픽을 분담하여 처리함으로 중앙집중형 보다 효과가 좋지만 설치하는데 비용이 많이 드는 단점이 있다.

따라서 본 논문에서는 클라우드 상의 SDN Controller의 처리에 과부하가 걸릴 경우 Openflow를 기반으로 한 가상화된 네트워크를 가진 산업 네트워크의 하드웨어에 임시 SDN Controller(tSDN) 배포를 하여 load balancing을 통해 효율적인 트래픽 관리를 할 수 있는 기법을 제시한다. tSDN의 운용 방식과 그 배포에 대한 조건, 그리고 트래픽 관리에 대해 설명하고 이를 mininet으로 시뮬레이팅하여 그 성능 평가를 하였다.

이 논문의 나머지 부분은 다음과 같이 구성되어 있다. 섹션 2에서 본 연구에 필요한 SDN과 그 Openflow, NFV에 대한 전반적인 설명이 간략하게 소개된다. 3 장에서는 제안된 tSDN 컨트롤 관리기법을 설명한다. 4 장에서는 제안된 방법의 효율성을 기술한다. 마지막으로 5 절에서 결론을 도출한다.

II. 관련 연구

SDN은 개방형 Openflow를 통해 네트워크 트래픽 전달 동작을 소프트웨어 기반 컨트롤러에서 제어/관리하는 접근방식이다. 기존의 라우터 방식과는 달리 트래픽경로와 데이터 전송을 수행하는 plane이 분리되어 있으므로 네트워크의 요구사항에 맞춰 유연하게 네트워크를 관리할 수 있다. SDN 은 3개의 plane으로 구성되어 있으며, Infrastructure plane, Control plane, Management plane이다. Infrastructure에서는 Control plane의 프로토콜 정보를 저장하고 트래픽을 실제로 전송하는 역할을 수행한다. Control plane에서는 이웃하는 switch의 정보를 찾고 경로를 설정해주는 작업을 수행한다. Management plane에서는 이러한 일련의 상황을 모니터링, 평가하는데 이용한다. SDN Controller에서는 이러한 일련의 작업을 수행하기 위하여 통신 Control plane에 위치하여 트래픽의 최적의 경로를 검색하고 제어한다.

1. SDN

SDN 개념은 네트워크 장비의 데이터 전달 평면에서의 라우팅 기능을 컨트롤러가 전담한다. 이는 라우터에서 이루어지는 라우터 결정이라는 기존의 네트워크 관리 방식을 바꾼다. 실시간으로 자료를 수집하고 네트워크를 변경해야 하는 산업 네트워크에서 SDN은 매우 중요한 역할을 할 수 있다. 일반적인 가상화 네트워크용 SDN 컨트롤러 배포는 플러드 라이트, OpenDaylight, Ryu, POX, ONOS 및 Trema 등의 오픈 소스 구현이다. NFV 패러다임을 사용하여 SDN 컨트롤러 기능을 가상화하는 것은 로드 밸런싱, 라우팅 및 포워딩, 방화벽 및 트래픽 엔지니어링을 비롯한 네트워크 기능의 사용을 위한 정교한 접근 방식으로 간주된다. 또한 이 기법은 재해 또는 실패 조건에서 하드웨어 유지 일시 중지 및 복구 시간 개선과 같은 보충적인 이점을 제공한다. 가상화 된 SDN 컨트롤러[7]는 하드웨어 유지가 필요할 때 (하드웨어 유지 일시 중지), 가상화 SDN 컨트롤러의 상태에 대한 스냅

샷 및 백업을 하나의 서버에서 공유 할 수 있는 경우 데이터 센터 클라우드 내의 물리적 서버 간에 즉각적으로 이동할 수 있다. 데이터 센터를 클라우드의 다른 서버에 연결하여 장애 발생 후 빠른 재구성이 가능하도록 한다.

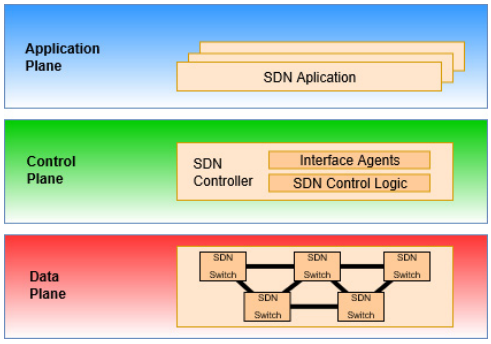

그림 1. SDN
Fig. 1. SDN Architecture

2. NFV

일반적으로 NFV는 데이터 센터 또는 분산 컴퓨팅 플랫폼의 전용 하드웨어에 소프트웨어 인스턴스로 배포된다. NFV는 정적 및 동적 네트워크 인프라에서 모든 Control plane 기능 또는 Data plane 패킷 처리에 적합한 가상화 기술이다.[8] 그럼에도 불구하고 이 작업은 IP 기능의 가상화, 특히 모든 리소스의 과부하를 피하기 위해 여러 리소스에 작업 부하를 분산하기 위해로드 균형 조정을 수행하는 트래픽 로드 관리에 중점을 둔다. 몇몇 로드 밸런싱을 통한 플로우 목표 처리량 및 대역폭 전체를 활용한 최적화 트래픽 전송 지연 및 응답 시간을 최소화하는 기법들이 제안되었다.[9,10] 자원 절약과 관련하여 로드 밸런싱은 중앙 집중식 결정 기반 또는 분산 결정 기반[11]이 될 수 있다. 중앙 집중식 결정은 중앙 집중식 의사 결정과 분산 된 결정은 처리 지연과 확장 된 완료 시간 때문에 매우 효율적인 방법이 아니다. 중앙 집중식 결정은 로컬 컨트롤러의 모든 로드 정보를 수집하고 로드 밸런싱 요청을 로컬 오버로드 된 컨트롤러로 보낸다. Distributed decision[12] 모든 컨트롤러가 명령을 보내지 않고 로컬에서 로드 밸런싱을 수행 할 수 있도록 한다. 중앙 집중식 결정의 처리 지연과 분산 결정의 로드 균형 조정 완료 시간이 길어지기 때문에 두 전략의 가용성과 확장성이 저하된다.

3. OpenFlow

OpenFlow는 네트워크 컨트롤러가 스위치 네트워크에서 네트워크 패킷의 경로를 결정할 수있게 한다. 컨트롤러는 스위치와 구별된다. 이렇게 제어를 전달과 분리하면 액세스 제어 목록 (ACL) 및 라우팅 프로토콜을 사용하는 것으로 정교한 트래픽 관리가 가능하다. 또 OpenFlow를 사용하면 서로 다른 공급 업체의 스위치를 하나의 개방형 프로토콜을 사용하여 원격으로 관리할 수 있다.

OpenFlow는 패킷 일치 규칙 및 동작을 추가, 수정 및 제거하여 레이어3 스위치의 패킷 전달 테이블을 원격 관리 할 수 있다. 이렇게 하면 라우팅 결정을 주기적으로 또는 임시로 수행할 수 있다. 구성 가능한 수명을 가진 규칙과 동작으로 변환된 다음 스위치의 플로우 테이블에 배치되어 해당 규칙이 지속 되는 동안 유선 속도로 스위치에 일치 패킷을 실제로 전달한다. 스위치와 일치하지 않는 패킷은 컨트롤러로 전달 될 수 있다. 그런 다음 컨트롤러는 하나 이상의 스위치에서 기존 플로우 테이블 규칙을 수정하거나 새 규칙을 배포하여 스위치와 컨트롤러 사이에서 트래픽의 구조적 흐름을 방지하도록 결정할 수 있다. 심지어 헤더 대신 패킷 전체를 전달하도록 스위치에 지시했다면 트래픽 자체를 전달하기로 결정할 수도 있다.

OpenFlow 프로토콜은 TCP (Transmission Control Protocol) 위에 계층화되어 있으며 TLS (Transport Layer Security) 의 사용을 규정한다. 컨트롤러는 TCP 포트 6653에서 연결을 설정하려는 스위치를 청취해야합니다. 이전 버전의 OpenFlow 프로토콜은 비공식적으로 포트 6633을 사용했다.[13.14]일부 네트워크 제어 플레인 구현은 프로토콜을 사용하여 네트워크 전달 요소를 관리한다.[15] OpenFlow는 주로 보안 채널에서 스위치와 컨트롤러 사이에서 사용된다.

Ⅲ. tSDN 제안 기법

1. 시스템 구조

tSDN은 NFV서비스를 통해 동적 네트워크 인프라에서 처리된다. NFV가 가진 강력한 VNF 비스는 각 VTN 당 서로다른 VNT 토폴로지를 가질 수 있으며 동일한 물리적 인프라를 공존할 수 있다. 이를 통해 독립적인 tSDN의 배포가 가능하다. 이를 위해서는 모든 네트워크 상의 하드웨어는 VNF를 구축한 상태여야 한다. SDN Controller에서는 끊임없이 트래픽관리와 로드 밸런싱

을 하고 있다. 산업 네트워크 새로운 프로세스 추가 등으로 인해 예상되지 못한 트래픽이 발생하여 SDN Controller가 과부하가 발생하게 된다고 판단되었을 때 SDN Controller는 SDN Orchestrator에 tSDN 배포요청을 할 수 있다. SDN Orchestrator는 SDN Controller request내 트래픽 정보를 통해 가장 최적화된 tSDN 배치 위치를 확인한다. 이러한 절차가 끝난 후, SDN Orchestrator는 tSDN을 배치하여 트래픽 처리를 분담하게 한다. tSDN은 SDN Controller와 본질적으로 같으나, 임시로 배포된 것이며 클라우드에 위치하는 게 아닌 스위치 및 하드웨어에 위치한다. 그러므로SDN Orchestrator는 투명성을 위해 tSDN의 라이프 사이클을 감독한다.

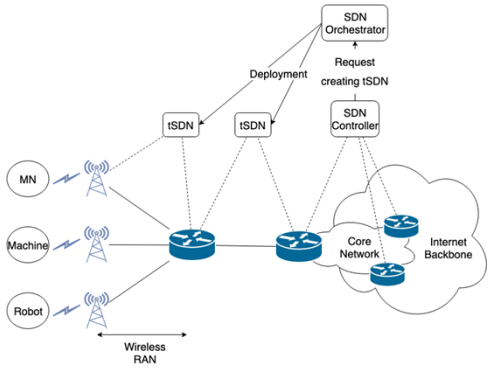

그림 2. tSDN 아키텍처
Fig. 2. tSDN Architecture

2. 시스템 구조

NFV는 동적 네트워크 인프라에서 VNF 및 관련 서비스를 효율적으로 처리한다. tSDN을 VNF로 사용할 때 기본 네트워크를 통과하는 트래픽을 공유하기 위해 동일한 작업에 대해 동일한 VNF를 더 추가할 수 있는 개방성이 있다. 불균형 트래픽로드가 증가하는 경우 tSDN 사용 네트워크에서 로드 균형 조정을 위해 tSDN을 복수 배포할 있다. 보조 tSDN 컨트롤러의 필요성이 결정되고 정확하게 작동하고 트래픽 부하를 공유하는 원래 tSDN과 완전히 동일한 구성으로 생성된 tSDN 컨트롤러의 사본이 필요하다. 본 절에서는 네트워크에서 트래픽 부하 분산을 위한 tSDN의 배포절차와 그 필요 구성에 대해 설명한다.

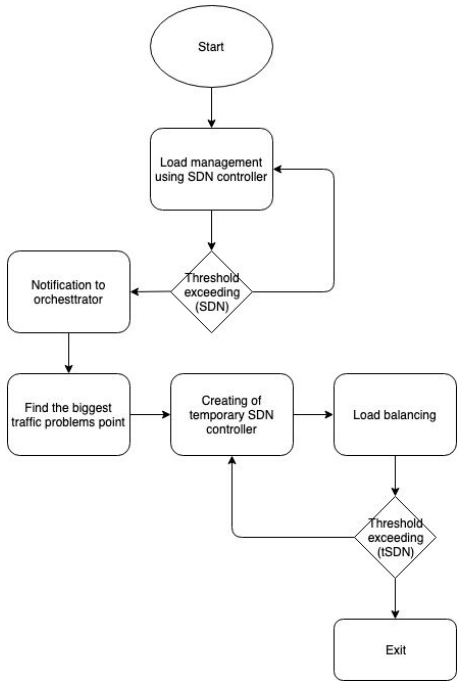

그림 3. tSDN 배포 절차
Fig. 3. tSDN Deployment Procedure

tSDN을 배포하는 것은 트래픽 내 하드웨어의 자원을 소모하는 일임으로 이는 신중하게 이루어져야 한다. 그러므로 SDN은 스스로 과부하가 일어날 것인지를 평가하여야 한다. 내부의 계산에 의해 역치값을 넘어섰다고 판단될 경우에는 Orchestrator는 tSDN을 배포해야 한다.

새 tSDN 컨트롤러를 프로비저닝하기 위해 SDN Controller에서 요청이 들어오면, Orchestrator가 필요한 SDN 컨트롤러 배포 (예 : OpenDaylight, ONOS, POX 또는 플러드 라이트 등)를 지정한다. 그런 다음 SDN Orchestrator는 트래픽 문제가 발생하는 스위치에 VM 세팅 요청을 보낸다. 스위치에서는 tSDN을 위한 사전 작업을 수행하고 이를 Orchestrator에 대답한다. SDN을 할당하기위한 Cloud Controller가 orchestrator에 요청을 받으면 tSDN Controller를 위한 자원 할당을 Orchestrator에 응답한다. 사전작업이 끝나면 Orchestrator는 Switch에 tSDN을 전송하고 이를 SDN Controller에 응답한다. 생성 후, tSDN 컨트롤러는 지정된 IP 주소를 사용하여 VTN을 구성하도록 요청한다. IP주소는 Orchestrator에서 설정한다. 전체 프로세스가 성공적으로 완료되면 tSDN이 기능을 시작하고 주기적으로 Orchestrator에 tSDN의 정보를 전달한다.

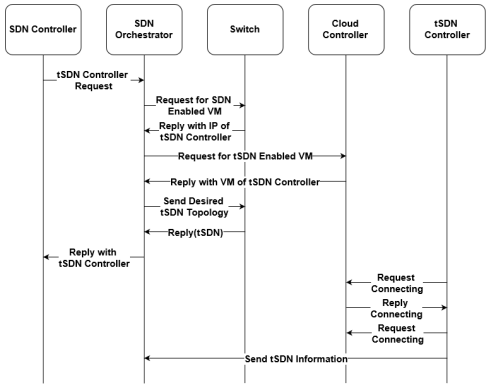

그림 4. tSDN 배포 플로우 차트
Fig. 4. tSDN Deployment FlowChart

Ⅳ. 실험 및 결과

1. Mininet

Mininet은 네트워크 에뮬레이터 또는 네트워크 에뮬레이션 오케스트레이션 시스템이다. Linux 커널에서 호스트, 스위치, 라우터 및 링크 모음을 실행한다. 경량 가상화를 사용하여 단일 시스템을 완벽한 네트워크처럼 보이게 하고 동일한 커널, 시스템 및 사용자 코드를 실행한다.[] Mininet은 실제 머신과 같이 동작하기 때문에, 임의의 프로그램을 연결하여 실행할 수 있다. 실행하는 프로그램은 실제 이더넷처럼 보이는 패킷을 보낼 수 있습니다 인터페이스, 주어진 링크 속도와 지연. 패킷은 실제 이더넷 스위치, 라우터 또는 미들 박스처럼 일정량의 대기열로 처리된다. 그러므로 SDN을 활용한 네트워크를 시뮬레이팅 하는데 있어 가장 강력한 기능을 가지고 있다.

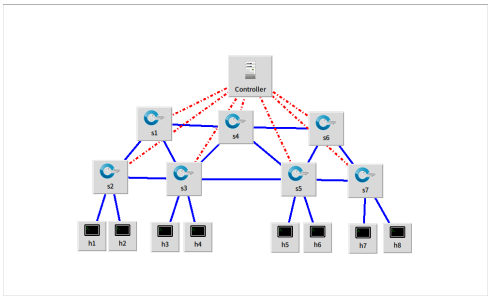

그림 5. Mininet
Fig. 5. Mininet

이 에뮬레이팅은 Intel i7 3.2GHz CPU, 32GB RAM, Ubuntu 18.04에서 실험되었다. VMs는 VirtualBOX 6.0.4에서 실행되었다. 패킷을 분석하기 위해 우리는 Wireshark를 사용하였다. SDN Controller는 OpenDaylight Contorller를 사용하였다.

2. 트래픽 제어

SDN 컨트롤러의 경우 링크 용량의 70퍼센트가 넘어가게 되면 오버로드되어 과부하가 발생되게 된다. 이럴경우 tSDN을 배치하여 tSDN으로 처리를 유도하게끔한다. 오버로드를 확인할 수 있는 식은 다음과 같다.

$$L_{Trans} = \frac{L_c - L_{thr}}{L_{thr}} \quad (1)$$

특정 switch A에서의 트래픽 비율은 다음과 같다.

$$R = \frac{\sum_{i=1}^{n} T_i}{T_M} \quad (2)$$

T_M은 A의 트래픽 허용 최대량, $\sum_{i=1}^{n} T_i$ 은 A를 점유하는 트래픽이다. R이 역치 값을 넘었을 경우 tSDN은 트래픽을 가장 해결하기 용이한 path로 load balancing 할 수 있다. 그때 비용을 계산하는 식은 다음과 같다.

$$C = \sum_{i=1}^{n} C_0 L_0 + \sum_{i=1}^{n} C_i L_i + \sum_{i=1}^{m} C_i \quad (3)$$

$\sum_{i=1}^{n} C_0 L_0$은 path를 사용할 때의 기본적인 코스트, $\sum_{i=1}^{n} C_i L_i$은 path i를 사용할 때의 코스트의 합을 뜻한다. $\sum_{i=1}^{m} C_i L_i$는 스위치를 통과할 때 필요한 코스트를 뜻한다. 최적의 path로 tSDN은 rerouting한다. 이를 통해 tSDN은 과부하를 막을 수 있다.

3. 성능평가

이번 실험에서는 먼저 단일 원격 tSDN 컨트롤러 인 OpenDaylight 컨트롤러 베릴륨 배포판을 배포하고 Mininet 에뮬레이터의 토폴로지와 연결했다. 처음에는 연결된 모든 호스트, 연결된 스위치, IP 주소, MAC 주소 및 포트 매핑 등에 대한 정보와 같은 연결 정보를 얻는다. 링크에 대한 통계는 정기적으로 수집되므로 트래픽로드가 임계 값 한계에 도달 할 때마다 알림을받는다. 전송

된 트래픽이 링크 용량보다 70 % 높을 때, tSDN은 동일한 토폴로지에 배포된다. 이 단계에서 오버 트래픽 스위치에 대한 네트워크 토폴로지는 tSDN 컨트롤러로 제어됩니다. 두 호스트 간의 경로 / 경로 가용성 정보는 Dijkstra를 사용하여 로드 균형 조정을 수행해야하는 Fat-Tree 토폴로지의 단 하나의 섹션에 대한 최단 경로 검색을 제한하는 방식으로 얻어진다. 전송 된 데이터의 관점에서 두 호스트 사이의 모든 경로에 대한 링크의 총 비용을 계산하는 요청이 이루어집니다. 패킷 흐름은 현재 시간에서 링크의 최소 전송 비용을 고려하여 형성되고 최적의 경로가 결정되고 정적 흐름이 다른 컨트롤러 및 주어진 최적 경로에 있는 모든 스위치로 이동한다. 소스 IP, 출발지 MAC, 목적지 IP, 목적지 MAC, 인 포트 및 아웃 포트와 같은 실질적인 정보는 모든 플로우에 제공됩니다. 프로그램은 주기적으로 이 정보를 주기적으로 업데이트하여 동적으로 만듭니다. Wireshark는 컨트롤러가 실행 중이고 Mininet에서 생성 된 토폴로지에 연결될 때 호스트 간의 연결을 포착하고 분석하는 데 사용되었다.

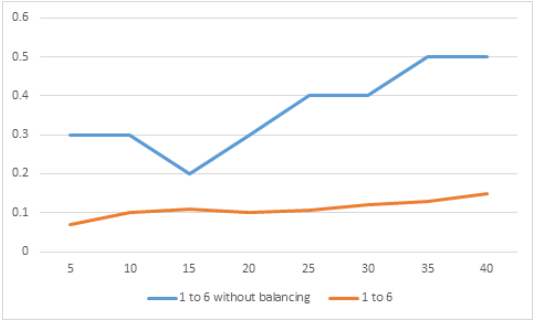

그림 6. Ping Test
Fig. 6. Ping Test

그림 6은 Host1에서 Host6까지의 로드 균형 조정 전후의 iPerf 핑의 개선을 보여준다. 이 그림은 제안 된로드 균형 조정 계획으로 인해 왕복 시간 (RTT)이 현저히 감소되었음을 명확하게 보여준다. 40 개의 패킷이있는 경우, 로드 균형 조정 시스템, 즉 기존 SDN을 사용하지 않는 구성표의 평균 핑 시간은 0.5 초다. 그러나 같은 수의 패킷 (로드)에 대해 제안 된 구성표는 평균 핑 시간을 0.15 초로 줄였다. 이는 50 % 이상의 개선이었습니다. 이렇게 감소한 핑 시간은 제안 된로드 밸런싱이 로드를 분산 시켰기 때문이다.

V. 결론

본 논문에서는 산업 네트워크에 사용되기 위해 제안된 임시 SDN Controller 배포를 통한 분산형 SDN 제어 방법을 제안하였다. 기존의 수평적 NFV 구조가 아닌 일반 네트워크형태의 계층적 형태의 사용을 위해 Controller 배치를 제안하였다. mininet을 통한 평가로 제안 기법이 성능을 상승 시킨 것을 확인할 수 있었다. 다만 SDN Contoller의 배치 및 파기에 사용되는 역치 값의 정밀한 측정은 실험을 통해 다시 얻을 필요가 있다.

References

[1] IEEE “1903.1-2017 - IEEE Standard for Content Delivery Protocols of Next Generation Service Overlay Network”, IEEE, 25 May 2018. DOI:https://dx.doi.org/10.1109/IEEESTD.2018.8365911

[2] Woo-Seok Yang, Jung-Ho Kim, Jae-oh Lee, “A Management for IMS Network Using SDN and SNMP", Journal of the Korea Academia-Industrial cooperation Society, 18(4) pp. 694 - 699. 2017. DOI:https://dx.doi.org/10.5762/KAIS.2017.18.4.694

[3] Junhyuk Park, Wonyong Yoon, “SDN Device-Centered Integration Architecture for Loosely-Coupled LTE/WiFi Networks", The Journal of Korean Institute of Information Technology, 17(6) pp. 63 - 71. 2019. DOI:http://dx.doi.org/10.14801/jkiit.2019.17.6.63

[4] Diego Kreutz, “Software-Defined Networking: A Comprehensive Survey", Proceedings of the IEEE, 103(1), pp. 14 - 76 January. 2015. DOI:https://dx.doi.org/10.1109/JPROC.2014.2371999

[5] Sikandar Ejaz, “Traffic Load Balancing Using Software Defined Networking (SDN) Controller as Virtualized Network Function,” IEEE Access, 7, pp. 46646-46658, April, 2019. DOI:https://dx.doi.org/10.1109/ACCESS.2019.2909356

[6] P. Kevin, B. Mathieu, L. Jeremie, “DISCO: Distributed SDN controllers in a multi-domain environment”, IEEE Network Operations and Management Symposium, May. 2014. DOI:https://dx.doi.org/10.1109/NOMS.2014.6838273

[7] Adiran Lara, Anisha Kolasani Byrav Ramamurthy, “Network Innovation using OpenFlow: A Survey”, IEEE Communications Surveys & Tutorials, 16(2), pp 493-512, 2014. DOI:https://dx.doi.org/10.1109/SURV.2013.081313.00105

[8] Zichuan Xu, Weifa Liang, Meiltian Huang, Mike Jia, Song Guo, Alex Galis, “Efficient NFV-Enabled Multicasting in SDNs”, IEEE Transactions on

Communications. 67(3), pp 2052-2070, November. 2018.
DOI:https://dx.doi.org/10.1109/TCOMM.2018.2881438

[9] A. A. Neghabi, N. J. Navimipour, M. Hosseinzadeh, and A. Rezaee, "Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature", IEEE Access, 6, pp 14159-14178, 2018.
DOI:https://dx.doi.org/10.1109/ACCESS.2018.2805842

[10] S. K. Askar, "Adaptive load balancing scheme for data center networks using software defined network", Science Journal of University of Zakho, 4(2), pp 275-286, 2016.

[11] J. Yu, Y. Wang, K. Pei, S. Zhang, and J. Li, "A load balancing mechanism for multiple sdn controllers based on load informing strategy", in Network Operations and Management Symposium (APNOMS), 2016 18th Asia- Pacific. IEEE, pp. 1-4, 2016,
DOI:https://dx.doi.org/10.1109/APNOMS.2016.7737283

[12] Y. Zhou, M. Zhu, L. Xiao, L. Ruan, W. Duan, D. Li, R. Liu, and M. Zhu, "A load balancing strategy of sdn controller based on distributed decision", 2014 IEEE 13th International Conference on. Trust, Security and Privacy in Computing and Communications (Trust-Com), pp. 851-856, 2014,
DOI:https://dx.doi.org/10.1109/TrustCom.2014.112

[13] "OpenFlow Switch Errata v1.0.2-rc1" (PDF). Open Networking Foundation. October. 2013.

[14] "Service Name and Transport Protocol Port Number Registry". IANA.

[15] Koponen, Teemu, "Onix: A Distributed Control Platform for Large-scale Production Networks". USENIX. October 2010.

저 자 소 개

박 도 권(준회원)

- 2018년 9월 ~ : 성균관대학교 스마트팩토리융합학과 석사과정 재학중
- 주관심분야 : 스마트팩토리, 네트워크, CNN 등

정 종 필(정회원)

- 2008년 ~ 2009년 : 성균관대학교 컨버전스연구소 연구교수
- 2010년 ~ 현재 : 성균관대학교 정보통신대학 겸 산학협력단 교수
- 2015년 ~ 현재 : 전자부품연구원 IoT 융합연구센터 전문연구위원
- 2016년 ~ 현재 : 성균관대학교 스마트팩토리융합학과 사업총괄책임자
- 주관심분야 : 스마트팩토리, 모바일융합컴퓨팅, 센서 네트워크, 차량 모바일 네트워크, 네트워크 보안, IT융합, 인터렉션 사이언스, 스마트 헬스케어, IoT/M2M, 웨어러블 컴퓨팅 등

※ 이 논문은 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터지원사업의 연구결과로 수행되었음 (IITP-2019-2018-08-01417).