http://dx.doi.org/10.7840/kics.2016.41.11.1598

SmartX-mini Playground 상의 IoT-Cloud 서비스에 대한 SDN 기반 모니터링 데이터 수집 경로 설정

윤 희 범◆, 김 승 룡*, 김 종 원°

SDN-Based Collection-path Steering for IoT-Cloud Service Monitoring Data over SmartX-mini Playground

Heebum Yoon◆, Seungryong Kim*, JongWon Kim°

요 약

부상하는 IoT-Cloud 서비스들을 효율적으로 지원하기 위해서는 IoT-Cloud 간의 안정적이고 지속적인 데이터 전송이 필수적이다. 본 논문에서는 초융합형 SmartX Box에 기반한 SmartX-mini Playground 상에서 IoT-Cloud 서비스를 위한 다양한 모니터링 데이터를 보다 안정되게 수집/전송함에 있어서 오픈소스 메시징 시스템인 Kafka를 활용하고 ONOS(Open Network Operation System) SDN 제어기에 기반한 SDN 응용을 적용함으로써 실제적인 IoT-SDN-Cloud 환경에서의 안정적인 IoT-Cloud 서비스에 대응하는 모니터링 데이터 전송을 위한 유연한 수집 경로 설정이 가능함을 확인한다.

Key Words : IoT, SDN, Cloud, Kafka, Monitoring

ABSTRACT

Safe transmitting monitoring data is essential for supporting IoT-Cloud services efficiently. In this paper, we find ways to configure data path flexibly in SDN based for IoT-Cloud services utilizing SmartX-mini Playground. To do this, we use ONOS(Open Network Operating System) SDN Controller, ONOS NBI Applications made from us to check flexible and safe data path configuration for IoT-Cloud monitoring data transmitting in real IoT-SDN-Cloud environments.

Ⅰ. 서 론

최근 부상하는 사물인터넷(IoT: Internet of Things) 관련 서비스들은 분산된 IoT 단말들로부터 수집된 대량의 데이터를 저장/처리하기 위해 클라우드 인프라와 연계되고 있다[1]. 데이터 그 자체만으로는 가치를 온전히 살릴 수 없기 때문에 사물 인터넷 시대에는 빠르게 생성되는 많은 양의 데이터를 안전하게 전송하고 실시간으로 수집, 관리, 분석 할 필요성이 있다[2]. 즉, 다수의 IoT 기기와 클라우드간의 안정적인 데이터 전

※ 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-16-2012, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발) 그리고 2016년 미래창조과학부의 재원으로 SW융합기술고도화 사업의 지원을 받아 수행된 연구임 (S0182-16-1001)

◆ First Author : Department of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, hbyoon@nm.gist.ac.kr, 학생회원

° Corresponding Author : Department of Electrical Engineering Computer Science, Gwangju Institute of Science and Technology, jongwon@nm.gist.ac.kr, 종신회원

* Department of Electrical Engineering Computer Science, Gwangju Institute of Science and Technology, srkim@nm.gist.ac.kr, 학생회원

논문번호 : KICS2016-08-187, Received August 5, 2016; Revised October 31, 2016; Accepted October 31, 2016

달과 이에 대한 지속적인 모니터링이 필요하다. 그러나 전통적인 방식의 네트워크 인프라에서 IoT-Cloud 서비스를 도입할 경우 실무자들은 네트워크 장비를 구매하고, 물리적으로 연결하고, 필요한 서비스를 설정하고, 기존 인프라와의 호환성을 점검하고 안정화 작업도 거쳐야 한다. 이런 과정은 서비스를 추가할 때마다 반복된다. 사용자 특성이나 사업 환경이 시시각각 달라질 수 있는 IoT 시대에 구축 절차와 물리적 시간 소요가 거듭 된다면 안정적인 서비스 제공에 문제를 야기할 수 있다. 가령 IoT-Cloud 서비스를 이용할 때 문제가 발생할 경우 해당 문제가 물리적인 네트워크 장비 설정에서 발생한 것인지, IoT-Cloud서비스 내에서 발생한 문제인지 파악하기가 어렵다. 또 물리적인 네트워크 장비의 문제일 경우 정확하게 어떠한 장비에서 발생한 문제인지, 장비 설정을 바꾼 후 네트워크에 어떠한 영향을 미칠지 알 수 없을뿐더러, 원활한 서비스를 제공 할 수 없다.

소프트웨어-정의-네트워킹(SDN: Software-Defined Networking)을 이용하면 IoT-Cloud간 여러 전송 경로들에 대한 손쉬운 관제가 가능하다. 관리자는 SDN 제어기를 통해 다수의 네트워크 장치들에 대한 트래픽 관리, 경로설정 등을 한 번에 할 수 있어 IoT-Cloud 서비스를 효율적으로 지원할 수 있기 때문이다. 이를 위해 SmartX-mini Playground 공용 개발 환경을 활용한 IoT-Cloud 자원 모니터링 서비스를 통해 IoT-SDN-Cloud 서비스를 실증한다. IoT-Cloud 자원 모니터링 서비스는 Apache Kafka 기반으로 다량의 데이터를 신속하게 전달하기에 적합한 서비스이지만, 물리적인 경로 상에서 발생하는 문제에 대한 대응에는 취약점을 갖고 있다. 때문에 해당 서비스에 ONOS SDN 제어기에 기반한 SDN 응용을 활용해 유연한 데이터 경로 설정을 적용함으로써 SDN 네트워크를 통한 보다 더 유연하고 안정적인 IoT-Cloud 서비스 관리/제어 가능성을 확인할 수 있다.

이어진 2절에서는 관련 기술 및 배경에 대해 기술하고 3절에서는 SmartX-mini Playground의 구축 및 개발에 대해 설명한다. 4절에서는 SDN 기반의 모니터링 데이터 수집 경로 설정을 위한 응용 프로그램에 대해 소개하고, 5절에서는 SDN 기반의 응용프로그램을 SmartX-mini Playground 상에 적용한 결과에 대해 기술하며 마지막으로 6장에서는 결론을 제시한다.

Ⅱ. 관련 기술 및 배경

2.1 ONOS(Open Networking Operating System) SDN 제어기

SDN(Software-defined Networking)이란 네트워크 제어 기능(control plane)이 물리적 네트워크와 분리되어 있는 네트워크 구조를 말한다. 하드웨어와 분리된 제어 기능은 제어기(controller)로 불리는 스위치나 라우터가 아닌 별도의 장치에서 구현된다. 이 제어 기능은 SDN의 두 가지 요소와 상호작용하는데 하나는 응용이고 다른 하나는 하드웨어에 구현된 추상화 계층이다. 제어 기능을 통해서 응용은 네트워크의 다양한 정보를 얻을 수 있고, 반대로 네트워크 역시 응용 요구 사항 등의 정보를 얻을 수 있다.

이러한 핵심적인 역할 때문에, 제어 기능은 종종 네트워크 운영체제(Network OS)라 호칭되고 있다. 하드웨어와 소프트웨어를 분리하여, 소프트웨어는 자연스럽게 네트워크 전체를 관장하게 되는 중앙 집중형으로 구성되고, 전체 네트워크가 중앙의 제어 기능을 통해 단일화하여 통제 받음으로써, 효율적인 통신이 이루어지고, 일괄적인 정책 적용을 통해 빠르면서도 안전한 네트워크 운용이 이루어지게 된다. Openflow는 SDN을 위한 인터페이스 표준 기술로 정의되며, ONF(Open Networking Foundation)[3]를 중심으로 연구/개발된 가장 대표적인 SDN을 위한 프로토콜이다. OpenFlow는 제어 평면과 데이터 평면을 분리하고, 이들을 통신 할 수 있도록 하는 통신 규약이다. OpenFlow enabled 장비는 기존과 다르게 SDN 제어기에 의한 제어가 가능하고 고정적이던 기존의 네트워크에서 벗어나 정책이나 사용 목적에 따라 관리 변경이 가능하므로 유연하게 활용할 수 있는 장점이 있

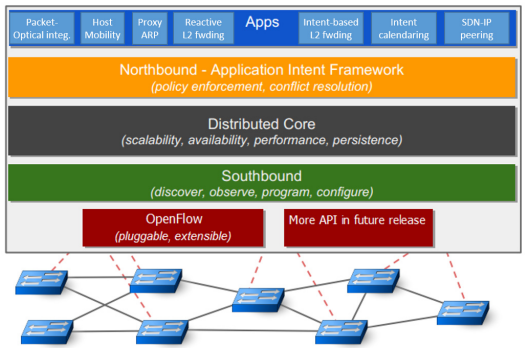

그림 1. ONOS(Open Networking Operating System) 구조
Fig. 1. ONOS(Open Networking Operating System) Architecture

다[4].

ONOS(Open Network Operating System)[5]는 ON.Lab에서 개발하였으며, 제어기는 Floodlight에 기반이며 분산형 SDN 제어기 중 가장 대표적이다. 중앙집중형식의 SDN 구조는 단일 SDN 제어기가 중앙에서 SDN을 구성하는 모든 네트워크 장치들을 제어/관리해 제어 평면의 병목 현상문제를 야기 할 수 있다. 이를 극복하기 위한 방안으로 분산형 SDN 제어기가 나왔고, 중앙 집중식 SDN 제어기에서 야기되는 병목 현상 문제를 완화 할 수 있으므로, 최근 각광 받고 있다. 그림 1은 ONOS의 구조를 개념적으로 나타낸다. 그림 1에서 볼 수 있듯이, 최 하단에 위치한 네트워크 스위칭 장비들을 분산형 구조를 갖는 ONOS 제어기가 제어하는 구조이다.

2.2 Docker, Apache Kafka

Docker[6]는 Linux 기반의 Container 기술을 활용한 오픈소스 소프트웨어로서, 가상 머신과 유사한 특징을 갖지만 보다 경량화된 형태로 활용 가능하다. IoT-Cloud 환경에서는 다수의 기기에 필요한 소프트웨어의 배포 및 관리가 요구되므로 가상화 기술의 도입이 필연적인데, Docker Container는 별도의 게스트 OS없이 호스트의 커널을 공유하는 까닭에, 한정적인 자원을 갖고 있는 IoT 환경에서도 유용하게 활용 가능하다. Docker Hub라는 공용 저장소를 활용하여 손쉽게 원하는 이미지를 주고 받을 수 있는 장점도 갖고 있다.

Apache Kafka[7]는 대용량의 실시간 로그처리에 특화되어 설계된 메시징 시스템으로써 publish-subscribe 모델을 기반으로 동작하며 크게 producer, consumer, broker로 구성된다. 또한 분산 시스템을 기본으로 설계되었기 때문에, 기존 메시징 시스템에 비해 분산 및 복제 구성을 손쉽게 할 수 있고 파일 시스템에 메시지를 저장하기 때문에 별도의 설정을 하지 않아도 데이터의 영속성이 보장된다. Producer에서 전달한 데이터는 정해진 topic에 따라 분류되며 여러 대의 broker로 구성될 경우 topic은 몇 개의 partitions으로 구성되며 각 partitions들은 여러 개의 사본을 가질 수 있다. 때문에 일부 broker에 장애가 생기더라도 다른 broker에서 원하는 topic에서 필요한 데이터를 전달 받을 수 있다.

기존 범용 메시징 시스템(ActiveMQ, RabbitMQ)과 비교하였을 때 다양한 기능을 지원하지는 않지만, 수많은 IoT 기기에서 생성되는 데이터에 대응하여 많은 양의 데이터를 안정적으로 빠르게 전달하는 측면에서는 매우 우수하다.

2.3 모니터링 데이터 수집

데이터 모니터링은 여러 분야에서 효과적으로 활용된다. 서비스 제공 측면에서는 모니터링을 통해 문제를 발견하고 이를 해결함으로써 서비스의 질적 개선이 가능하며, 인프라 관리에 있어서도 자원의 낭비 또는 장비의 이상을 파악하고 대처를 위해 필요하다. 다양한 종류의 자원에 걸쳐 서비스를 운용하는 IoT-Cloud 환경에서도 데이터 모니터링은 중요하다. 자원에 대한 모니터링을 통해 보다 적은 비용으로도 안정적인 IoT-Cloud 서비스 제공이 가능해지기 때문이다.

이러한 모니터링 시스템은 주로 컴퓨팅, 네트워킹, 스토리지 등의 자원에 대한 데이터를 주기적으로 수집하는 기능을 포함한다. 이뿐 아니라 특정 임계값에 따른 알람 및 로그 데이터에 대한 분석 또한 시스템을 구성하는 중요한 기능이다. 이러한 정보는 일반적으로 대시보드 형태로 구성되어 사용자가 인식 가능하게끔 시각화된다.

본 실험에서도 위와 같은 기능을 포함하는 모니터링 시스템의 구현을 목표로 하며, 데이터의 수집 측면에서는 다양한 기기에 대응하기 위하여 SNMP 프로토콜을 활용한다. 시스템 내의 다양한 메세지의 안정적이고 빠른 전달을 위해 pub/sub 형태로 구성된 Apache Kafka를 적용한다. 수집된 데이터에 대해서는 비동기 처리가 가능한 node.js를 사용하였으며, 데이터의 시각화는 Grafana를 통해 처리하였다.

2.4 SmartX-mini Playground의 참조모델 클라우드 테스트베드: SmartX Playground

IoT와 Cloud가 접목되는 IoT-Cloud 서비스를 효율적으로 체험하려면, 계산/저장/네트워킹 자원을 융합하여 단일 Box로 통합한 초융합형(hyper-converged) SmartX Box 들을 기반으로 소위 SmartX Playground로 불리는 실증형 공용개발환경을 구축하여 활용하면 효과적이다[8]. 차세대 기술로 손꼽히고 있는 IoT, Cloud, SDN의 기술들을 실증할 때, 시뮬레이션 기반의 실험보다 실제 인프라 상에서 실증 실험을 하는 것이 더 효과적이고 직관적일 수 있다. 실제 실험 환경을 구성하기 위해 국내에서는 2012년~2013년 내에 국내5개와 국외 7개 사이트에 한국형 융합 지원박스(서버)로써 SmartX Box를 그림2와 같이 설계하였고, SmartX Playground라는 이름의 미래인터넷 테스트베드 운용을 시작하였다[9]. Playground는 의미 그대로

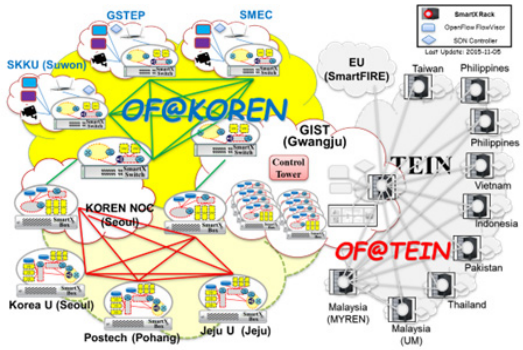

그림 2. SmartX Playground 구성
Fig. 2. SmartX Playground Environments

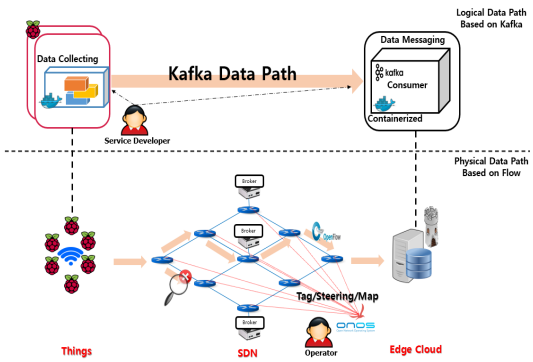

그림 3. IoT-SDN-Cloud 환경에서의 논리적/물리적 전송 경로
Fig. 3. Logical/Physical Data Paths on IoT-SDN-Cloud

사용자들이 하고 싶은 다양한 실증을 자유롭게 할 수 있는 환경을 제공하는 테스트베드를 의미한다. SmartX Box는 계산/저장/네트워킹 자원을 하나의 박스 단위로 확보해 다양한 서비스를 실증할 수 있는 초융합형 자원 박스로 정의한다.

SmartX-mini Playground는 SmartX Playground의 참조 모델로써, IoT-SDN-Cloud 환경을 구성하고 실험을 실증하기 위한 단일 사이트내의 테스트베드이다. IoT 기기를 다루는 경우에 기존의 고가 서버 자원을 사용하는 경우 자원의 낭비로 이어질 수 있기 때문에 보다 저렴한 비용으로 구축할 수 있는 환경이 필요하다. 이에 SmartX-mini Playground는 이러한 요구사항을 충족시키기 위해 상대적으로 저렴한 하드웨어를 활용하여 구축하였다.

Ⅲ. SmartX-mini Playground를 활용한 SDN 기반 경로 구축

IoT-Cloud 자원 모니터링 서비스를 지원하기 위해서는 IoT-Cloud 간 안정적인 데이터 전송이 필수적이다. 이를 지원하기 위해 SDN 기술을 도입함으로써 기존의 네트워크에서 보다 유기적으로 네트워크를 운영/제어 할 수 있다. 그러나 단지 SDN을 이용함으로써 IoT-Cloud 서비스의 온전한 연결은 보장받기 어려울 수 있다. 물리적인 네트워크 환경 위에서 돌아가는 IoT-Cloud 서비스 자체에 문제가 생길 경우 SDN은 단지 물리적인 네트워크 환경에 대해서만 관여할 뿐, 이를 알아차릴 수 없기 때문이다. 즉 IoT-SDN-Cloud 서비스를 온전히 지원하기 위해서는 그림 3과 같이 실제 데이터의 물리적인 전송 경로와 논리적인 데이터 전송 경로의 두 가지 데이터 전송 경로가 모두 안전히 확보될 필요가 있다. 데이터 전송 경로가 모두 안정적이고 유연하도록 하는 요구사항으로는 다음과 같다.

1) 물리적인 전송 경로가 어떠한 내/외부 원인으로 문제가 발생할 경우, SDN 제어기를 이용한 우회경로를 통한 데이터의 전송
2) 물리적인 전송 경로의 문제 혹은 경로 변경에 따른 시간동안의 데이터 손실 최소화

위 두 가지 요구사항을 만족하면, IoT-Cloud 서비스를 지원함에 있어 어느 정도 안정된 데이터 전송 경로를 확보할 수 있다. 1)의 요구 사항을 만족하기 위해 실제 물리 네트워크 토폴로지의 구성할 때, 최소 2개 이상의 목적지로 향하는 경로를 구성해야 한다. SDN 제어기로 관리되는 네트워크 토폴로지에서는 제어기를 통해 스위치를 제어할 수 있기 때문에 처음 환경을 구성한 뒤에는 다시 스위치 장비에 대해 일일이 관리할 필요가 없다.

2)의 요구사항을 충족시키기 위해서는 IoT-Cloud 서비스의 데이터 손실을 막기 위한 메시징 시스템이 필요하다. 기존의 시스템들이 메시지를 메모리와 같은 일시적인 저장 공간에 보관하는 것과 달리 Apache Kafka는 메시지를 파일 시스템에 저장함으로써 데이터의 손실을 줄일 수 있으므로, 물리적인 전송 경로상의 문제에 의한 단절에도 데이터를 안정적으로 보관할 수 있는 버퍼로써 동작할 수 있는 장점을 갖는다.

따라서 위 두 가지 요구사항을 만족시키기 위해 본 논문에서는 Apache Kafka를 이용한 메시지 시스템을 활용한 논리적인 데이터 전송경로를 만들고 이를 SDN 제어기를 통해 물리적인 데이터 전송경로를 관리함으로써 보다 안정적이고 유연한 데이터 전송이 가능하도록 한다.

3.1 SmartX-mini Playground 구성 및 개발

IoT-Cloud 서비스 실증을 위한 공용개발환경을 제공하는 SmartX-mini Playground는 IoT 종단(end)에 적합한 장치인 µBox, 소규모로 꾸며진 Cloud를 대표하는 장치인 µCloud Box, IoT Gateway를 대표하는 IoT-Actuator, OpenFlow SDN을 지원하는 스위치, 그리고 전체 공용개발환경을 관제하기 위한 DevOps (개발운영병행체제) Tower를 포함한다[10]. SmartX-mini Playground는 소규모의 저렴한 장비들을 이용해 IoT-SDN-Cloud 환경에서의 다양한 실험을 실증하기 위한 테스트베드로써 사용된 하드웨어는 표 1 와 같다. 전체적인 테스트베드 구성은 그림 4와 같다. µBox 하드웨어는 Raspberry Pi 2를 사용하고, IoT Actuator 및 DevOps Tower의 하드웨어로는 Intel 사의 소형 PC인 NUC을 활용한다. 또한 µCloud Box는 Intel ONP Server를 이용, SDN 스위치는 Mikrotik 사의 스위치를 사용한다. µBox는 13대, IoT Actuator는 3대, 9대의 switch, 한 대의 µCloud Box를 이용한다. 상기한 그림 4를 보면, SmartX-mini Playground의 네트워크 토폴로지 구성을 볼 수 있다. 최소한 2개 이상의 물리적인 전송 경로를 만들어 두어, 물리적인 전송 경로의 문제가 발생했을 시, 우회 경로로 데이터

표 1. SmartX-mini Playground 하드웨어 사양
Table 1. SmartX-mini Playground Hardware

Device	name	brand	aim	OS
µBox	Raspberry Pi2	Raspberry Pi Foundation	IoT device	hypriot OS
IoT Actuator	NUC	Intel	IoT Gateway	Ubuntu 14.0.4 LTS
DevOps Tower	NUC	Intel	DevOps, SDN Controller	Ubuntu 14.0.4 LTS
µCloud Box	ONP Server	Intel	µCloud	Ubuntu 14.0.4 LTS
Switch	CRS 125-24G-1s-2HnD	Mikrotik	OpenFlow switch	Router OS MIPSBE 6.34
Switch	CRS 109-8G-1s-2HnD	Mikrotik	OpenFlow switch	Router OS MIPSBE 6.34
Switch	RB 750GL	Mikrotik	OpenFlow switch	Router OS MIPSBE 6.34

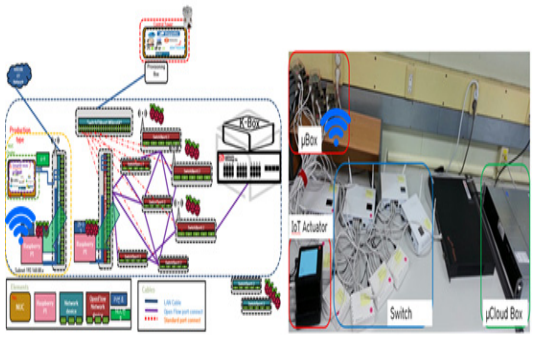

그림 4. SmartX-mini Playground 구성
Fig. 4. SmrtX-mini Playground environments

전송이 이루어지도록 구성한다. µBox의 경우 스위치와 무선/유선으로 연결한다. 본 실험환경에서는 무선 µBox는 4대, 유선 µBox는 9대로 연결한다. 무선의 경우µBox의 동글을 설치하여 스위치와 연결한다.

3.2 IoT-Cloud를 위한 자원 모니터링 서비스 설계 및 구현

다수의 IoT 기기들을 제어하기 위해서는 기본적으로 모든 IoT 기기들의 상태를 알아야 할 필요가 있기에 이러한 환경을 효율적으로 다루기 위한 지속적이고 신뢰할 수 있는 운영 데이터 수집이 요구된다. 따라서 SmartX-mini Playground에서도 IoT-Cloud를 위한 자원 모니터링 서비스를 지원함으로써, 환경의 운영에 필요한 데이터를 모니터링하

고 분석할 필요가 있다. 이 때, 상대적으로 자원이 부족한 IoT 기기의 특성상 자원의 효율적인 활용과 실시간 처리 역시 다뤄져야 할 문제이다. 이를 해결하기 위해 하이퍼바이저(Hypervisor)에 비해 적은 자원을 요구하는 컨테이너(Container) 기술 중 Docker를 활용하여 다수의 IoT 기기에 원하는 서비스를 손쉽게 배포하고 운영하고자 한다.

그림 5는 SmartX-mini Playground에서 해당 서비스를 지원하기 위하여 µBox, IoT Actuator, µCloud Box에 설치된 소프트웨어 구성을 나타낸다. µBox들은 Apache Flume을 통해 주기적으로 µBox에서 동작하는 SNMP 데몬으로부터 필요한 자원 상태 정보를 수집한다. 수집된 정보는 종류에 따라 Kafka Broker 내의 topic에 대응되므로, 이를 Kafka producer API를 활용해 IoT actuator에 위치한 Kafka broker로 전달한다[11]. 각각의 µBox로부터 수집하는 자원 상태 정보는 Raspberry Pi2의 CPU, Memory, I/O 등에 대한 16개의 항목을 포함한다. 이 때, 각 broker들은 각각의 topic에 대해 1개 이상의 partition을 갖고 있으며,

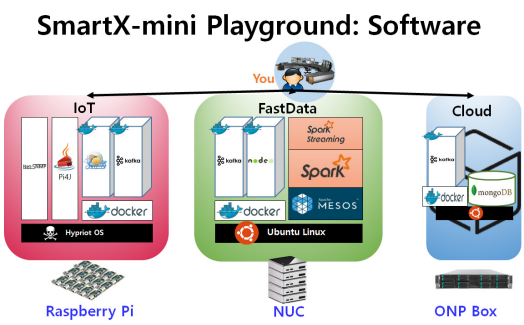

그림 5. SmartX-mini Playground 소프트웨어 구성
Fig. 5. SmartX-mini Playground Software

손실을 방지하기 위하여 각 partition은 2개의 사본이 존재한다. 각 partition에 저장된 정보들은 파일 시스템 내에 저장되므로 일시적인 장애로 인해 발생한 지연에 대응가능하다. 그러나 물리적인 전달 경로에서 발생한 문제가 논리적인 경로에서 감당하기 힘들 정도로 지속되면 Kafka의 구조적인 특징으로도 안정적인 메시지 전달을 보장할 수 없다. 이 때, 그림 3과 같이 ONOS(Open Network Operating System) SDN 제어기에 기반한 SDN 응용을 적용하여 물리적인 경로를 보완하도록 하면 보다 유연한 데이터 전송이 가능하게 된다. SDN 제어기는 우회 경로를 통해 데이터 전송이 이뤄지도록 스위치들을 조정할 수 있으며 Kafka는 데이터를 파일 시스템에 저장하면서 전달하기 때문에 플로우의 단기적 단절이나 경로 변경에 따른 데이터의 손실을 막을 수 있는 것이다.

Ⅳ. SmartX-mini Playground를 위한 SDN 기반 응용 설계 및 구현

SmartX-mini Playground에서 운영되는 IoT-Cloud 서비스의 데이터 경로를 안정적이고 유연하게 확보하기 위해서 ONOS(Open Networking Operating System) SDN 제어기와 SDN에 기반 한 응용을 이용한다. ONOS SDN 제어기는 IoT-Actuator와 같은 Intel NUC Box에 설치하며, 사용한 ONOS의 버전은 1.7.0 버전을 사용한다. SDN 기반 응용 소프트웨어는 SmartX-mini Playground의 µBox, IoT Actuator, µCloud Box에 대한 패킷을 모니터링을 통해 Host-to-Host forwarding rule을 수집하고, Intent를 활용해 Box들에 대한 Intent를 자동적으로 설치한다. 이를 통해 µBox에서 IoT Actuator 내의 Ka람 broker로, Kafka broker에서 다시 µCloud Box 쪽으로 메시지가 전달되는 경로를 설정 할 수 있다.

4.1 ONOS SDN 기반 응용 소프트웨어 설계 및 기능 명세

ONOS SDN 기반 응용 시스템은 .oar(ONOS App aRchive) 파일 형태로 제공 되고, OAR 파일은 app.xml, features.xml 및 bundle artifacts를 포함하고 있는 JAR 파일이다. ONOS SDN 기반 응용은 크게 세 가지 종류가 있다. Bundle 응용은 API를 제공하지 않고 자체적으로 모든 기능을 수행하고, 서비스 Interface가 정의된 응용은 다른 응용에서 접근 가능한 API를 제공하고 네트워크 환경뿐만 아니라 다른 구성요소들과도 상호작용을 한다. 서비스 Component 형태로 활성화 가능한 응용은 Activate/deactivate 등의 메소드들을 가진 singleton 클래스다.

그림 6은 SmartX-mini Playground 환경에서 IoT-Cloud 서비스를 지원하기 위한 ONOS SDN 응용의 구조와 알고리즘을 나타낸다. 패킷을 모니터링하기 위해 Data Model을 정의하고, 이 Data Model을 여러 응용에서 사용하기 위해 서비스 Interface를 정의한다. Data Model은 출발지와 목적지의 MAC 주소와 스위치에서 해당 forwarding rule을 요청한 횟수를 정의한다. 서비스 Interface는 스위치 식별자를 가지고 앞서 정의한 Data Model을 리스트화해서 스위치별 패킷 모니터링이 가능하도록 한다. Bundle 응용은 전체 응용이 활성화 되었을 때 실행할 사용자 정의 로직을 정한다. 즉 제어기에서 수신 받은 제어 패킷을 어떻게 처리할지를 결정한다. 토폴로지 정보를 바탕으로 ONOS SDN 응용은 가장 먼저 자동으로 Intent를 이용해 플로우를 설치한다. Intent는 토폴로지와 µBox, IoT-Actuator, µCloud Box를 기반으로 'single-to-multiple', 'multiple-to-single' Intent를 이용한다. 그림 6의 ONOS SDN 응용 알고리즘과 같이 Intent 설치와 각 스위치 별 플로우 트래픽을 계산한

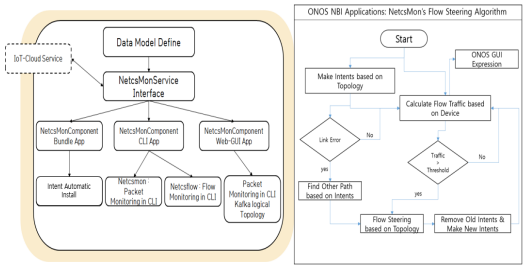

그림 6. ONOS SDN 응용 구조 및 플로우 처리 알고리즘
Fig. 6. ONOS SDN Application Architecture & Flow processing Algorithm

다. 트래픽 계산은 각 스위치의 설치된 모든 플로우 트래픽을 계산한다. Intent를 설치함으로써 두 호스트 사이의 경로 연결성을 보장할 수 있다. 즉, µBox에서 IoT-Actuator, IoT-Actuator에서 µCloud Box로 이동하는 Kafka 데이터 흐름을 물리적으로 보장하게 된다. 만일 물리적인 장애(링크 단절, 스위치 고장)가 발생했을 경우, 미리 설치된 Intent를 기반으로 다른 경로를 찾는다. 설치한 Intent는 스위치의 포트를 기반으로 플로우를 설치하기 때문에, 해당 포트가 이상이 생겼을 경우, 스위치의 다른 포트를 통해 패킷을 전송할 수 있다.

ONOS SDN 응용은 트래픽을 기반으로 플로우를 제어한다. 각 스위치별 트래픽을 계산 한 뒤, 같은 레이어의 링크를 비교해 트래픽을 분산시킨다. 다수의 µBox들이 하나의 스위치에 무선 또는 유선으로 연결되어있고, 이 스위치는 2개의 링크를 통해 패킷을 다음 스위치로 전송한다. 트래픽이 미리 정의한 threshold 값보다 낮을 경우, 하나의 링크로 패킷을 전송한다. 만일 트래픽이 threshold 값보다 높다면, 이를 탐지한 후 기존의 설치된 Intents를 지우고 두 개의 링크를 통해 패킷을 전송할 수 있도록 Intents를 자동으로 재설치 한다. 이를 통해 트래픽을 분산시키고 물리적인 장애가 생겼을 경우엔 다른 링크로 패킷을 전송해 안정적으로 데이터를 전송할 수 있도록 한다.

CLI 응용은 CLI 입력을 통해 IoT-Cloud 서비스의 패킷을 모니터링하고, 스위치에서 IoT-Cloud 서비스의 대한 플로우를 확인 할 수 있다. "netcsmon"이라는 명령어를 ONOS CLI에 입력하고 스위치 ID를 매개변수로 넘기면 해당 스위치ID를 가진 모든 forwarding rule 정보를 나열한다. 또한 "netcsflow" 명령어를 입력하고 스위치 ID를 매개 변수로 넘기면 마찬가지로 해당 스위치 ID를 가지면서, IoT-Cloud 서비스를 지원하는 µBox, IoT Actuator, µCloud Box들에 대한 플로우 정보를 나열한다. Web-GUI 응용은 CLI 응용과 마찬가지로 패킷을 모니터링하고 그 결과를 ONOS Web GUI에 보여 준다. 또한 IoT-Cloud 서비스를 대상으로 하는 논리적인 Kafka 데이터 경로를 물리적 네트워크 토폴로지와 함께 보여준다. 논리적인 Kafka 데이터 경로는 Cytoscape[12]에서 제공하는 html을 바탕으로 µBox, IoT Actuator, µCloud Box의 Kafka 메시징을 토폴로지로 나타낸다.

4.2 ONOS SDN 기반 응용 제한

본 논문에서 구현한 SDN 응용은 몇 가지 제한점이 있다. SDN 응용은 SmartX-mini Playground 테스트베드에서 동작하는 다수의 µBox, IoT Actuator, µCloud Box에 대한 MAC 주소를 사전에 알고 있다고 가정한다. 사실 IoT-Cloud 서비스는 다양한 서비스를 다수의 IoT 기기에서 동작하고, SDN 응용은 이러한 서비스를 구분하면서 필요한 만큼의 플로우를 내려 네트워크 트래픽을 관리하고 데이터를 효율적으로 전달해야한다. 이를 위한 방법 중 하나는 IoT-Cloud 서비스와 SDN 제어기가 서로 사전 정보를 넘겨받는 방식이다. 본 논문에서는 이 부분은 제외한 나머지 기능에 대해 실험을 실증한다. 본 테스트베드는 IoT-Cloud 환경 대상으로 SDN 제어기를 통해 IoT-Cloud 서비스의 데이터를 안정적으로 확보하는 것이다. 다만, 본 테스트베드에서 IoT 네트워크 통신을 위해 무선을 사용하지만, 실제 무선 통신 신호에 관한 전송부분은 고려하지 않는다.

IoT 서비스를 효과적으로 제공하기 위해 본 테스트베드와 무선 측면에서 LTE-A 시스템 기반의 네트워크[13]를 결합해 활용한다면, 더 효율적인 IoT-Cloud 서비스를 제공할 수 있다.

V. SmartX-mini Playground를 위한 SDN 기반 경로 설정과 검증

앞서 설명한 SmartX-mini Playground와 ONOS SDN 제어기를 활용해 IoT-SDN-Cloud 환경을 구축하였다. 그림 7은 IoT-Cloud 자원 모니터링 서비스가 돌아가는 상황에서 결과를 보여준다. 그림 7의 1번과 2번은 ONOS의 Web GUI와 Intent 화면이다. ONOS SDN 제어기를 실행한 후, 개발한 SDN 응용을 실행

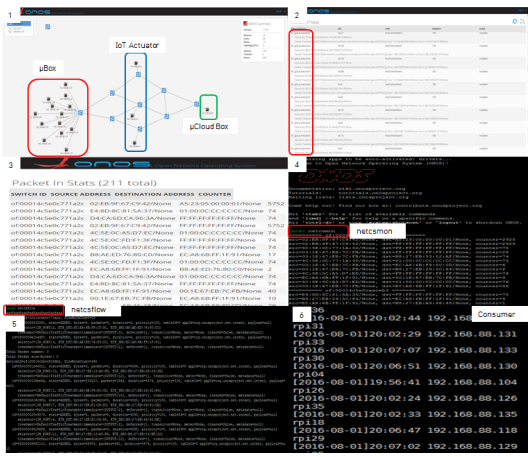

그림 7. ONOS SDN 응용 기능 명세
Fig. 7. ONOS SDN Application's function in SmartX-mini Playground

한 후로, Intent를 자동으로 설치한다. 총 13대의 µBox, 3대의 IoT Actuator 마지막 1대의 µCloud Box를 대상으로 총 47개의 Intent를 설치한다.

그림 8은 경로에 따른 트래픽과 플로우의 흐름을 나타낸다. 그림 8의 1번은 SmartX-mini Playground에서 SDN으로 제어하고 있는 네트워크 토폴로지의 전체 트래픽을 보여주고, 2번은 초기에 자동으로 설치된 Intent 관련 트래픽을 보여주고 있다. 그림 8의 1번을 보면 현재까지 표시된 부분으로 Kafka 관련 트래픽이 가장 많이 지나 간걸 알 수 있다. 그림 8의 3번은 설치된 Intent 트래픽 중 표시된 IoT Actuator와 관련된 트래픽을 표시하고 있다.

실험은 임의로 트래픽이 지나가는 곳 중 하나의 링크를 제거했을 때, 그에 따른 경로 변화와 데이터 손실의 유무를 체크한다. 그림 8의 4번을 보면 표시된 링크 하나를 제거한 후에 표시된 Intent 트래픽이다. 5번은 3번과 마찬가지로 표시된 IoT Actuator로 가는 트래픽을 보여준다. 3번과 비교하였을 때, 원래 µBox들이 IoT Actuator로 가는 경로가 끊어져 대체 경로로 이동하는 것을 볼 수 있다. 6번은 ONOS Web GUI에서 Kafka를 통해 이동하는 논리적 데이터의 토폴로지를 나타낸다.

그림 9는 IoT-Cloud 자원 모니터링 서비스로, 그림 8의 µCloud Box에서 Grafana[14]라는 모니터링 도구를 이용하여 µBox들의 자원을 모니터링하고 있는 것을 보인다. 총 14대의 µBox에 대해 CPU, 메모리, 디스크 사용량이 시시각각 어떻게 변하는 지를 확인한다. 이를 위한 각 자원 정보는 Kafka를 통해 전달되며, Grafana는 수집된 정보를 바탕으로 매 5초마다 화

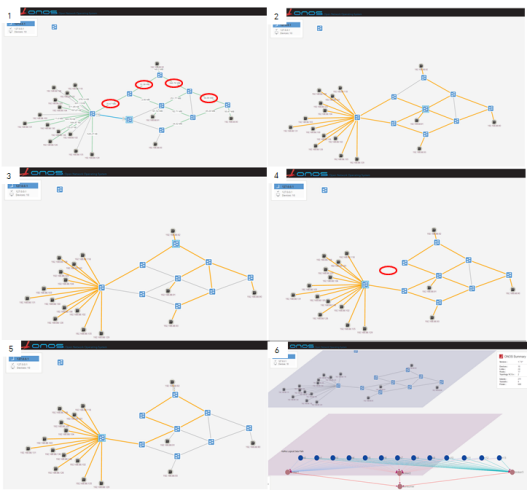

그림 8. SmartX-mini Playground SDN 경로 검증
Fig. 8. SmartX-mini Playground SDN Path

그림 9. µBox 운영 데이터 모니터링
Fig. 9. µBox's Resource Operation Data Visibility

면을 갱신하여 새로운 정보를 보여준다. 만약 이를 위한 전달 경로에 문제가 발생할 경우 Grafana에서는 정상적인 입력을 받지 못한 까닭에 차트 일부에서 단절이 나타날 것이다. 그림 9의 시점은 그림 8의 3번의 물리적인 링크가 단절되어 SDN이 우회 경로로 대체해 4번처럼 우회경로를 변경했을 때의 상황이다. 그림 9를 보면 그림 8의 3번에서 4번처럼 물리적인 링크가 단절되어 우회경로로 패킷을 전송하더라도, 차트에서는 정상적으로 매 주기 마다 수집한 데이터를 나타내는 것을 확인할 수 있다.

표 2는 µBox에서 2대의 IoT Actuator, µCloud Box로 ICMP 패킷을 보내는 시간을 그림8의 3번 상황과 4번 상황에서 각각 10번 시도해 평균치를 측정한 것이다. 그림 8의 3번에서 4번으로 임의의 링크가 단절되었을 때, 순간 ICMP 패킷의 속도가 평균 0.156 ms 만큼의 시간이 지연되었다. µBox에서 IoT-Actuator (IP: 192.1688.92)로 가는 시간이 그림 8의 3번보다

표 2. µBox 평균 Ping 테스트 결과
Table 2. µBox Average Ping Test Result

µBox (Fig. 8. number 3)	IoT Actuator 192.168.88.92	IoT Actuator 192.168.88.91	µCloud Box 192.168.88.90
192.168.88.104	1.929 ms	0.732 ms	0.917 ms
192.168.88.103	0.921 ms	0.711 ms	0.694 ms
192.168.88.135	0.913 ms	1.016 ms	0.841 ms
192.168.88.129	0.839 ms	0.901 ms	0.862 ms

µBox (Fig. 8. number 4)	IoT Actuator 192.168.88.92	IoT Actuator 192.168.88.91	µCloud Box 192.168.88.90
192.168.88.104	1.132 ms	0.945 ms	0.929 ms
192.168.88.103	1.077 ms	0.916 ms	1.012 ms
192.168.88.135	1.108 ms	0.972 ms	0.953 ms
192.168.88.129	1.022 ms	0.901 ms	0.803 ms

그림 8의 4번에서 평균 0.156 늘어난 것을 볼 수 있다. 다만 이 시간에서의 지연은 Kafka의 저장 능력으로 충분히 허용되어 결과적으로 모니터링 데이터의 손실 없이 그림 9에서 결과 값이 잘 출력되는 것을 알 수 있다.

Ⅵ. 결 론

기존의 네트워크에서 보다 더 안정적인 IoT-Cloud 서비스를 위해, SDN 네트워크를 적용한 SmartX-mini Playground 환경을 통해 IoT-Cloud를 위한 자원 모니터링 서비스의 원활한 데이터 경로 설정 가능성을 확인하였다. 구현한 ONOS SDN 응용은 인식되는 IoT 장치의 패킷을 모니터링하고, IoT-Cloud 서비스를 위한 종단간 Intent를 자동적으로 설치하도록 적용하였다. 따라서 기존의 경로의 문제가 생길 경우, 우회 경로를 통해 안정적인 데이터의 전송을 보장할 수 있음을 확인하였다. 다만, 이를 위해선 해당하는 IoT-Cloud 서비스가 어느 장치에서 운영되는 지에 대한 사전 정보가 필요하다. 따라서 향후에는 IoT-Cloud 서비스와 ONOS SDN 제어기 간의 통신을 통해 서비스를 수행하는 장치를 인식하고 서비스 별 트래픽 관리, 모니터링이 가능하도록 구현할 계획이다. 또한 SmartX-mini DevOps Tower를 통해 종단 단말들 자체에 발생하는 문제점을 자동으로 복구하도록 연결하면, 보다 더 안정적인 IoT-Cloud 서비스 실증이 가능할 것이다. IoT-Cloud 서비스를 진행하는데 있어 SmartX-mini Playground 테스트베드를 활용하면 효과적이다. 본 테스트베드를 활용해 모니터링 데이터를 수집하는 하나의 예로써 연구실 내의 서버실 온도를 체크하고 제어하는 서비스를 실행하고 있다. µBox들을 서버실내에 배치한 후, 서버들의 온도를 센서를 이용해 데이터를 확보하고 이를 µCloud Box로 전달해 서버실의 온도를 제어한다.

References

[1] K. S. Yeo, M. C. Chain, T. C. W. Ng, and D. A. Tuan, "Internet of things: Trends, challenges and applications," in *Proc. IEEE ISIC,* Dec. 2014.

[2] Y. Kim and S. Shin, "Design secure IoT (Internet of things) environments with SDN and NFV," *KICS Inf. and Commun. Mag.*, pp. 27-33, vol. 32, no. 7, Jul. 2015.

[3] ONF, https://www.opennetworking.org/about/onf-overview

[4] N. McKeown, et. al., "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOM Computer Commun. Rev.*, pp. 69-74, Apr. 2008.

[5] ONOS, http://onosproject.org/

[6] Docker, https://www.docker.com/

[7] Kafka, http://kafka.apache.org/

[8] B. R. Cha and J. W. Kim, "Construction and operation of integrated testbeds for software-defined infrastructure," *OSIA S&TR J.,* vol. 29, no. 01, Mar. 2016.

[9] J. W. Kim, et. al., "OF@TEIN: An OpenFlow-enabled SDN testbed over international smartX rack sites," in *Proc. APAN-Networking Research Workshop*, vol. 36, Aug. 2013.

[10] H. Yoon and J. W. Kim, "SDN based data routing to support services utilizing SmartX-mini playground systems," in *Proc. KICS 2015 Fall Conf.,* pp. 117-118, Seoul, Korea, Nov. 2015.

[11] S. Kim and J. W. Kim, "Enabling container-based operational visibility for SmartX-mini playground," in *Proc. KICS 2015 Fall Conf.,* pp. 133-134, Seoul, Korea, Nov. 2015.

[12] Cytoscape, http://js.cytoscape.org/

[13] S.-S. Yoo, S.-H. Lee, J. Shin, S.-M. Oh, and J.-H. Kim, "Performance evaluation of LTE-A random access procedure for IoT service," *J. KICS, v*ol. 41, no. 08, Aug. 2016.

[14] Grafana, http://grafana.org/

윤 희 범 (Heebum Yoon)

2015년 8월 : 아주대학교 컴퓨터공학과 졸업

2015년 9월~현재 : 광주과학기술원 전기전자컴퓨터 공학부 석사과정

<관심분야> SDN, IoT-Cloud

김 승 룡 (Seungryong Kim)

2015년 2월 : 전남대학교 전자컴퓨터공학부 졸업
2015년 3월~현재 : 광주과학기술원 전기전자컴퓨터 공학부 석사과정
<관심분야> IoT-Cloud Visibility

김 종 원 (JongWon Kim)

1994년 : 서울대학교 제어계측공학과 박사
2001년~현재 : 광주과학기술원 전기전자컴퓨터 공학부 교수
2008년~현재 : 광주과학기술원 슈퍼컴퓨팅센터 센터장
<관심분야> Networked Computing System focusing on “Dynamic & Resource-aware Composition of Media-centric Services employing Programmable and Virtualized Computing/Networking Resources”.