

K-ONE 기술 문서 #11

ONOS 컨트롤러의 OpenFlow 모니터링 시스템 설계 및 구현

Document No. K-ONE #11

Version 1.0

Date 2016-04-30

Author(s) 김우중

■ 문서의 연혁

버전	날짜	작성자	비고
초안 - 0.1	2016. 03. 01	김우중	
1.0	2016. 04. 29	김우중	3장추가
1.1	2016. 04. 30	김우중	교정

본 문서는 2015년도 정부(미래창조과학부)의 재원으로 정보통신
기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-15-2012, 글로벌
SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)

This work was supported by Institute for Information &
communications Technology Promotion(IITP) grant funded by the
Korea government(MSIP) (No. B0190-15-2012, Global SDN/NFV
OpenSource Software Core Module/Function Development)

기술문서 요약

본 기술문서에서는 현재 널리 연구/개발되고 있는 SDN 컨트롤러에 하나인 ONOS(Open Network Operating System)에 대한 개념을 살펴보고, ONOS 컨트롤러와 OpenFlow 스위칭 장비 사이의 OpenFlow 메시지를 모니터링 할 수 있는 시스템(OFMon)을 디자인 및 구현사항을 서술하였다. 기존에 본 연구진이 구현한 OFMon의 경우, ONOS의 디바이스 서브시스템 상에 구현한 구조로써 초기 버전을 디자인 하였다. 초기 버전에는 네트워크 관리자 및 사용자가 ONOS에서 제공하는 CLI(Command Line Interface) 및 GUI(Graphical User Interface)를 통해 현재 ONOS 컨트롤러와 OpenFlow 스위칭 장비들 사이에 오가는 OpenFlow 메시지를 확인할 수 있다. 이뿐만 아니라, 실시간으로 송/수신하는 OpenFlow 메시지를 모니터링 하기 위해, ONOS 로깅 시스템인 Karaf 로그 파일에 이를 실시간으로 출력하도록 구현하였다.

그러나 ONOS의 경우 오픈 소스 프로젝트(open-source project)이기에, 타 개발자들과의 협업을 위한 구현이 반드시 필요하다. 그러나 디바이스 서브시스템의 경우, ONOS의 중추적인 요소이기 때문에, 이를 함부로 수정하거나 삭제 및 추가적인 코드를 작성하면, 이른바 더러운 코드 이슈(dirty-code issue)가 발생한다. 따라서 본 연구진에서는 기존에 본 연구진이 개발한 OFMon을 상기의 이슈가 발생하지 않도록 새로운 서브시스템 형태로 구현하였다. 그리고 초기의 OFMon에서 제공하는 기능들을 모두 구현하여, 네트워크 관리자 및 사용자가 손쉽게 OpenFlow 메시지를 모니터링할 수 있도록 하였다.

Contents

K-ONE #11. ONOS 컨트롤러의 OpenFlow 모니터링 시스템 설계 및 구현

1. 개요	6
2. 기존에 제안된 ONOS 컨트롤러 상 OpenFlow 모니터링 시스템	11
2.1. 디자인	11
2.2. 문제점	14
3. 새로운 ONOS 컨트롤러의 OpenFlow 모니터링 시스템 설계/구현	15
3.1. 디자인	15
3.2. 수행 결과 - ONOS 로깅 시스템 상의 OFMon 출력 결과	17
3.3. 수행 결과 - GUI 기반 어플리케이션 상의 OFMon 출력 결과	18
4. 결론	19

그림 목차

그림 1. SDN의 구조	6
그림 2. OpenFlow 스위치 및 컨트롤러 개념도[2]	7
그림 3. ONOS 컨트롤러의 구조[5]	9
그림 4. ONOS 컨트롤러에 구현된 기능 및 프로토콜[5]	10
그림 5. 본 연구진에서 제안한 기존의 OFMon 디자인[6]	11
그림 6. ONOS의 디바이스 서브시스템 개념도[6]	12
그림 7. OFMon의 초기 디자인[6]	13
그림 8. ONOS 디바이스 서브시스템과 새로운 OFMon 서브시스템의 구조	15
그림 9. ONOS 로깅 시스템에 출력되는 OFMon의 모니터링 결과	17
그림 10. GUI 기반 어플리케이션 상의 OFMon 출력 결과	18

표 목차

표 1. 제안된 SDN 컨트롤러 리스트[3]	8
--------------------------------	---

K-ONE #11. ONOS 컨트롤러의 OpenFlow 모니터링 시스템 설계 및 구현

1. 개요

최근 SDN(Software Defined Networking) 개념을 대규모 네트워크에 적용하여 사용함으로써, 이를 관리 및 운영하는 사업자들의 CAPEX(Capital Expenditure)와 OPEX(Operational Expenditure)을 낮추는 시도가 진행 중이다. 본 개요에서는 OpenFlow[1]를 포함한 SDN의 개념 및 이를 제어/관리할 수 있는 SDN 컨트롤러에 대해 다룬다. 그리고 이들 컨트롤러 중 ONOS에 대해 살펴보고, ONOS 컨트롤러에 OpenFlow 모니터링 시스템에 대한 필요성 및 요구사항을 분석한다.

SDN이란 기존에 장비 제조사에 종속적으로 네트워크를 구성하는 것과는 달리, 어떠한 장비를 사용한다 하더라도 소프트웨어를 통하여 네트워크를 구성 및 유지/관리하는 개념이다. SDN은 네트워크의 관리를 위해 먼저 제어 평면(control plane)과 데이터 평면(data plane)으로 분리한다. 그리고 제어 평면을 오픈 API(Application Programming Interface)를 통해 프로그램된 소프트웨어를 활용하여 데이터 평면을 제어한다.

SDN의 구성은 크게 네트워크 인프라스트럭처(infrastructure) 계층, 제어 계층(control layer), 그리고 어플리케이션 계층으로 구성할 수 있다(그림 1). 먼저 어

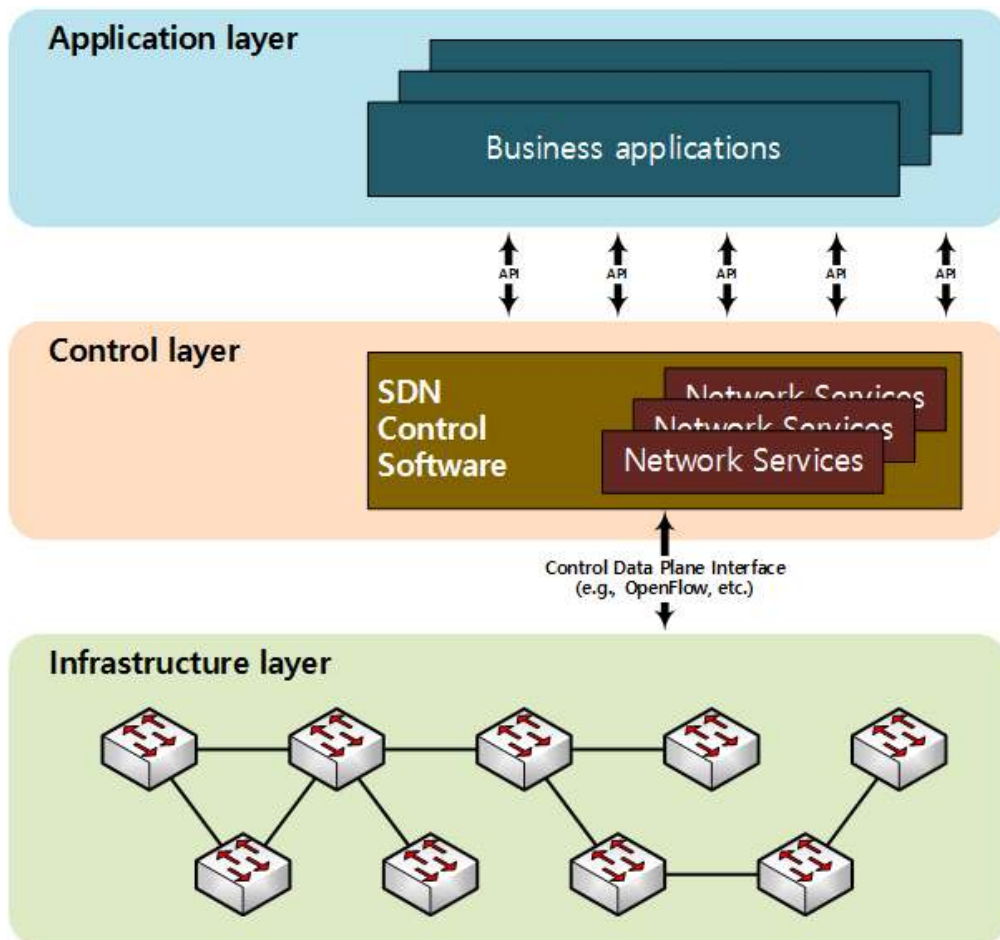


그림 1. SDN의 구조

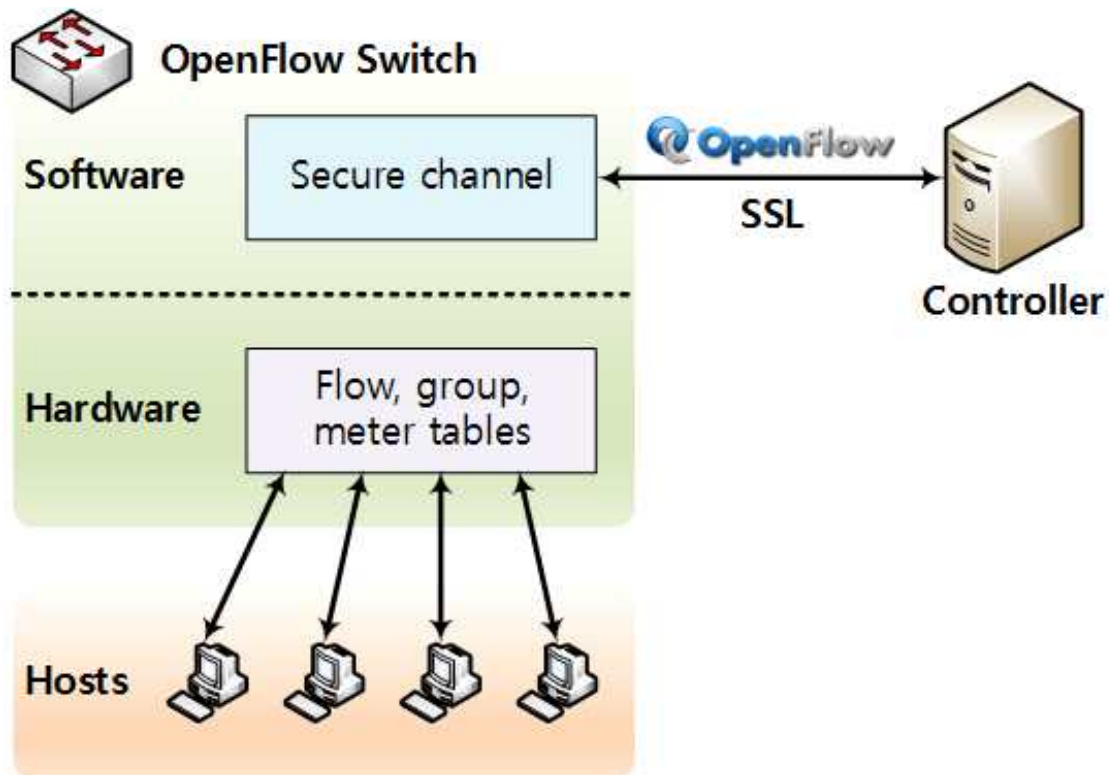


그림 2. OpenFlow 스위치 및 컨트롤러 개념도[2]

플리케이션 계층에 있는 다양한 어플리케이션들(예: 비즈니스 어플리케이션, 관리 어플리케이션, 등)이 존재한다. 제어 계층에서는 SDN을 제어할 수 있는 소프트웨어가 존재한다. 이는 SDN 컨트롤러라고 하며, 컨트롤러 내부에는 네트워크를 제어 및 관리할 수 있는 다양한 기능들이 구현되어 있다. 마지막으로 인프라스트럭처 계층에는 스위치와 같은 다양한 네트워크 장비들이 위치한다. 어플리케이션과 제어 계층 사이에는 API를 통해 통신 및 호출을 할 수 있으며, 제어와 인프라스트럭처 계층 사이에는 OpenFlow, LISP(Location ID Separation Protocol), NetConf 등과 같은 프로토콜/인터페이스를 통해 통신한다.

OpenFlow[1]는 ONF(Open Networking Foundation)[2]를 중심으로 연구/개발된 가장 대표적인 SDN을 위한 프로토콜이다. OpenFlow는 제어 평면과 데이터 평면을 분리하고, 이들을 통신할 수 있도록 하는 통신 규약이다. 그림 2와 같이, OpenFlow의 구조는 크게 컨트롤러와 OpenFlow 스위치로 구성되어 있다. 컨트롤러와 스위치 사이의 통신을 OpenFlow 프로토콜을 통해 수행이 되며, OpenFlow 프로토콜은 보안 채널인 SSL(Secure Socket Layer)을 사용한다. 컨트롤러는 OpenFlow 프로토콜을 통해, 스위치의 정보를 전달 받거나, 스위치에게 데이터 평면과 관련된 명령을 전달할 수 있다. 한편 스위치의 경우, 소프트웨어 계층과 하드웨어 계층으로 분리된다. 소프트웨어 계층은 컨트롤러로부터 전달받은 OpenFlow 메시지 혹은 컨트롤러로 알려줘야 할 정보를 담은 OpenFlow 메시지를 처리하는 부분이다. 반면 하드웨어 계층은 플로우 테이블(flow table), 그룹 테이블

표 1. 제안된 SDN 컨트롤러 리스트[3]

이름	구조	Northbound API	라이선스	언어
Beacon	중앙집중식	ad-hoc	GPLv2	java
DISCO	분산형	REST		java
ElastiCon	분산형	RESTful		java
Fleet	분산형	ad-hoc		
Floodlight	중앙집중식	RESTful	Apache	java
HP VAN SDN	분산형	RESTful		java
HyperFlow	분산형	-		C++
Kandoo	계층형	-		C++, Python
Onix	분산형	NVP	commercial	C, C++, Python
Maestro	중앙집중식	ad-hoc	LGPLv2.1	Java
Meridian	중앙집중식	extensible API		Java
MobileFlow	-	SDMN		
Mul	중앙집중식	multi-level	GPLv2	C
NOX	중앙집중식	ad-hoc	GPLv3	C++
NOX-MT	중앙집중식	ad-hoc	GPLv3	C++
NVP Controller	분산형	-	commercial	
OpenContrail	-	REST	Apache 2.0	C++, Java, Python
OpenDayLight	분산형	REST, RESTCONF	EPL v1.0	Java
ONOS	분산형	RESTful	-	Java
PANE	분산형	PANE	-	
POX	중앙집중식	ad-hoc	GPLv3	Python
ProgrammableFlow	중앙집중식	-		C
Rosemary	중앙집중식	ad-hoc		
RYU	중앙집중식	ad-hoc	Apache 2.0	Python
SMaRtLight	분산형	RESTful		Java
SNAC	중앙집중식	ad-hoc	GPL	C++
Trema	중앙집중식	ad-hoc	GPLv2	C, Ruby
yanc	분산형	filesystem		

블(group table), 그리고 미터 테이블(meter table)을 정의하고, 이를 활용하여 실제 스위치에 연결된 호스트 장비들의 플로우(flow)를 제어, 그룹 단위로 제어, 각 플로우의 성능 제어 등을 수행한다.

OpenFlow를 포함한 다양한 형태의 SDN을 제어 및 관리하기 위해서는 핵심적으로 SDN 컨트롤러가 반드시 필요하다. SDN 개념이 제안된 이래로, 현재 다양한 SDN 컨트롤러들이 제안되고 실제 구현이 되어왔다. 표 1은 현재까지 발표된 다양한 컨트롤러 중 일부를 기술한 내용이다. 총 28가지의 컨트롤러가 기술되어 있으나, 실제로는 이보다 더 많은 컨트롤러가 제안되었으며, 현재에도 지속적으로 새로운 컨트롤러를 논문, 특허 및 오픈소스 프로젝트로써 발표 및 개발되고 있다. 각 SDN 컨트롤러의 구조들을 살펴보면, 크게 중앙 집중식 구조와 계층형 구조를 포함한 분산형 구조로 나눌 수 있다. 중앙 집중식 구조란, 단일 컨트롤러가 중앙에서 SDN을 구성하는 모든 네트워크 장치들을 제어/관리하는 것을 의미한다. 그러나 이와 같은 중앙집중식 방식은 소수의 컨트롤러가 제어 평면의 병목 현상 문제를 야기할 수 있다. 이를 극복하기 위한 방안으로 나온 것이 바로 분산형 SDN 컨트롤러이다. 분산형 컨트롤러를 사용하게 되면 앞서 중앙집중식 SDN 컨트롤러에서 야기되는 병목 현상 문제를 완화할 수 있으므로, 최근 각광받고 있

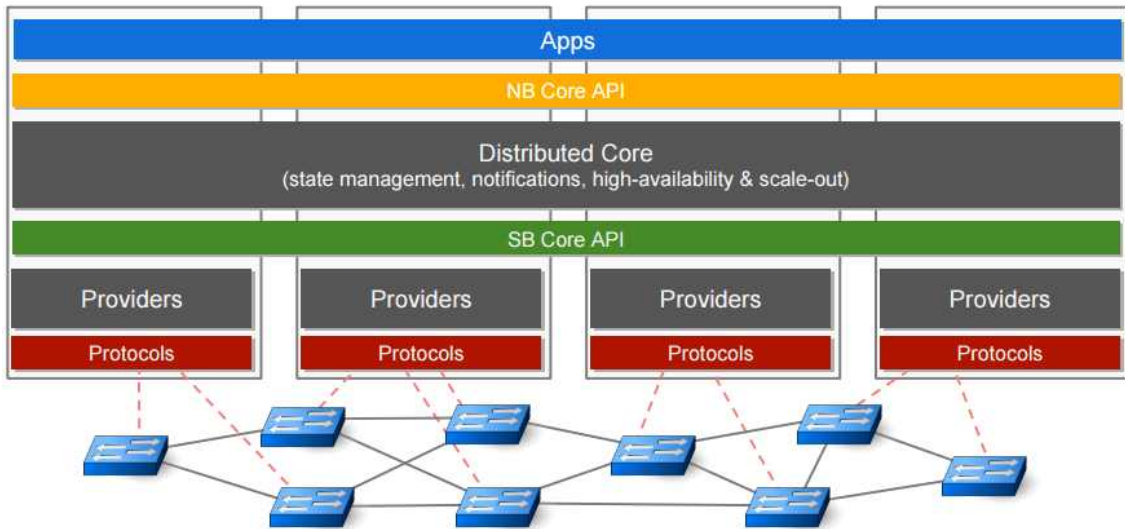


그림 3. ONOS 컨트롤러의 구조[5]

는 기술 중 하나이다.

분산형 SDN 컨트롤러 중 가장 대표적인 컨트롤러는 ONOS[4] 이다. ONOS란 Open Network Operating System의 약자로, 기존의 중앙집중식 SDN 컨트롤러 구조의 문제점을 극복하고, 실제 대규모 네트워크 환경에 적합한 구조로 설계되어 있다. 그림 3은 ONOS의 구조를 개념적으로 나타내는 개념도이다. 그림 3에서 볼 수 있듯이, 최 하단에 위치한 네트워크 스위칭 장비들을 분산형 구조를 갖는 ONOS 컨트롤러가 제어하는 구조이다. ONOS의 계층은 크게 어플리케이션 계층(Apps), 분산 코어 계층(Distributed Core), 프로바이더 계층(Providers), 프로토콜 계층(Protocols)으로 구성된다. 프로토콜 계층은 SDN 기능이 있는 다양한 네트워크 장치들과의 통신을 위한 계층이다. 이를 위해 OpenFlow 이외에도 NetConf 등과 같은 다양한 SDN 프로토콜을 정의하고 있다. 프로바이더 계층은 각 ONOS 컨트롤러마다 독립적으로 동작되는 계층으로, 상위 계층으로부터 요청된 네트워크 명령들을 프로토콜 계층에서 정의한 OpenFlow와 같은 프로토콜을 사용하여 각 네트워크 장치들에게 명령을 지시한다. 반대로, 네트워크 장치들로부터 전달받은 정보들을 상위 계층들로 알려주는 역할을 수행한다. 분산 코어 계층은 SDN 네트워크를 제어 및 관리할 수 있는 다양한 기능들이 정의된 계층이다. 각각의 ONOS 컨트롤러들은 실제로 네트워크상에 독립적으로 구동되고 있으나, 본 계층의 스토어(store)를 통해 분산 코어 계층이 단일 컨트롤러와 같이 구동할 수 있도록 정의되어 있다. 마지막으로 어플리케이션 계층은, 그래픽 유저 인터페이스(GUI: Graphic User Interface) 및 명령 줄 인터페이스(CLI: Command Line Interface)를 통해, 분산 코어 계층에 정의된 네트워크 제어 및 관리 기능들을 수행할 수 있도록 다양한 어플리케이션이 정의된 계층이다. 한편, 프로바이더 계층과 분산 코어 계층, 그리고 분산 코어 계층과 어플리케이션 계층의 통신 및 호출

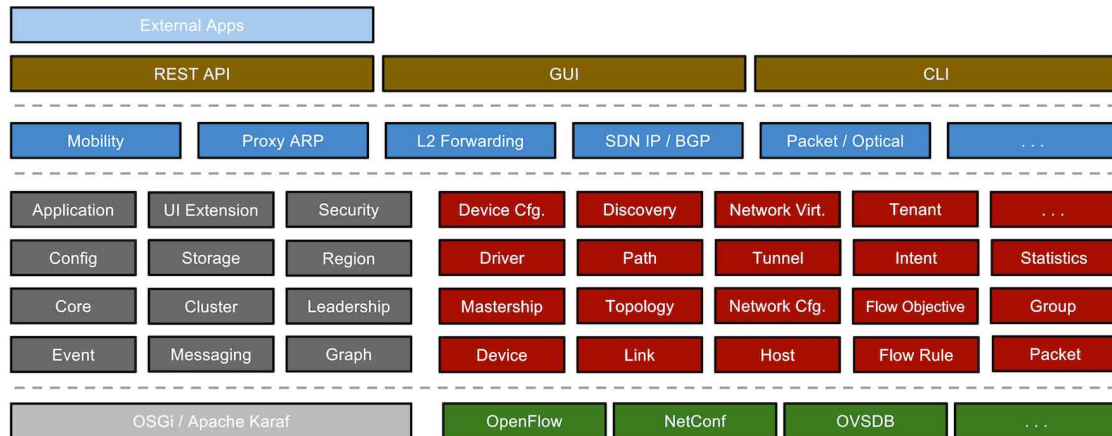


그림 4. ONOS 컨트롤러에 구현된 기능 및 프로토콜[5]

을 위해 SBI(South Bound Interface) 및 NBI(North Bound Interface)가 각각 정의되어 있다. 현재 ONOS상에 구현된 기능들은 그림 4와 같이 되어 있다. 그림 4에 표현된 기능 이외에도, 사용자가 원하는 기능 및 SDN 프로토콜들을 언제든지 소프트웨어로 추가할 수 있으며, 현재 지속적으로 다양한 개발자들이 이를 수행하고 있는 현실이다. 따라서, 향후 ONOS 컨트롤러의 기능 및 영역은 점차 확대 될 것으로 예상된다.

2. 기존에 제안된 ONOS 컨트롤러 상 OpenFlow 모니터링 시스템

2.1. 디자인

기존에 본 연구진이 제안한 ONOS 컨트롤러 상의 OpenFlow 모니터링 시스템 (OFMon)은 그림 5와 같이 디자인이 되어 있다[6]. 먼저 SDN 네트워크에 다수의 호스트 장비들과, 이들을 연결하는 OpenFlow 스위칭 장비들이 존재한다. 이와 같은 환경에서, ONOS 컨트롤러가 존재하여 OpenFlow 스위칭 장비들을 제어 및 관리하는 환경이다. 이와 같은 환경에서, 각 OpenFlow 스위칭 장비들과 ONOS 컨트롤러 사이에 오가는 OpenFlow 메시지들을 모니터링 하기 위해, OFMon을 ONOS 상에 구현하였다. 먼저 프로바이더 계층에는 OFMonitor와 Database를 정의한 후, OFMonitor는 SDN 네트워크로부터 올라오는 OpenFlow 메시지들을 모니터링 하는 모듈이다. 그리고 모니터링 된 결과를 실시간으로 Database에 저장하게 된다. 네트워크 관리자가 각 ONOS 컨트롤러에 실시간으로 올라오는 각각의 OpenFlow 메시지를 가시적으로 보여주기 위해, ONOS의 로깅 시스템(logging system)인 Karaf의 로그 파일에 실시간으로 출력해 준다. 이뿐만 아니라, 추가적으로 ONOS 어플리케이션 계층에 CLI 및 GUI 기반의 어플리케이션을 구현한 후,

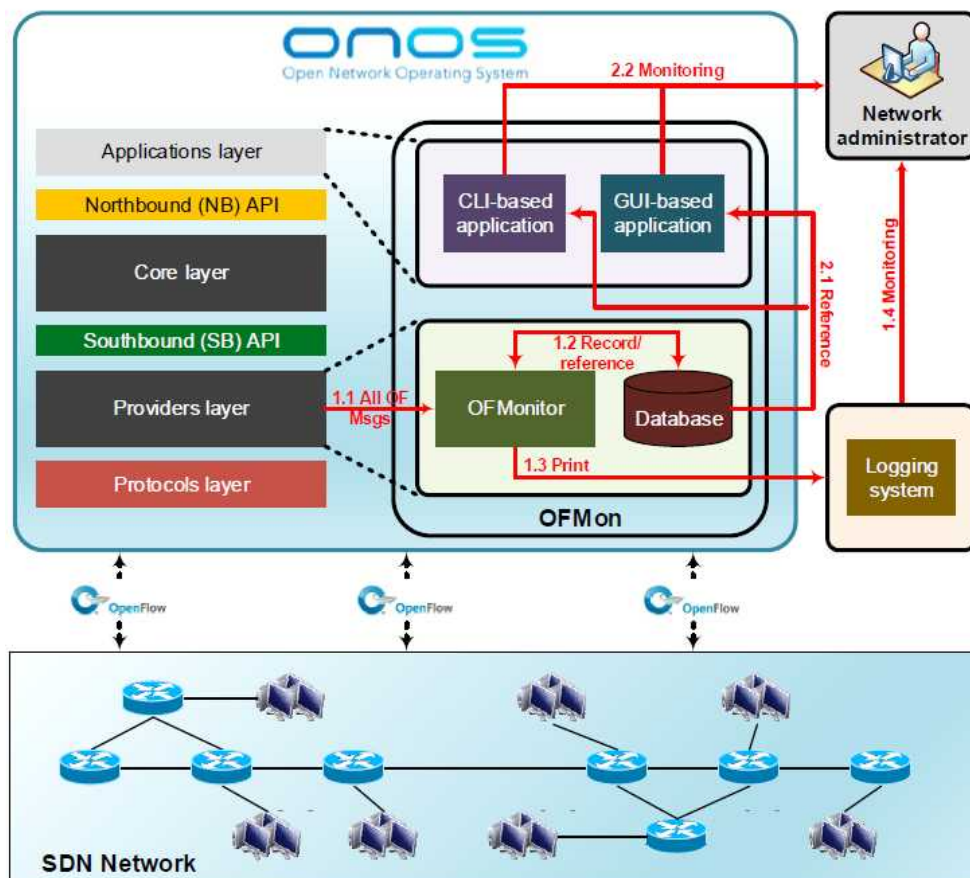


그림 5. 본 연구진에서 제안한 기존의 OFMon 디자인[6]

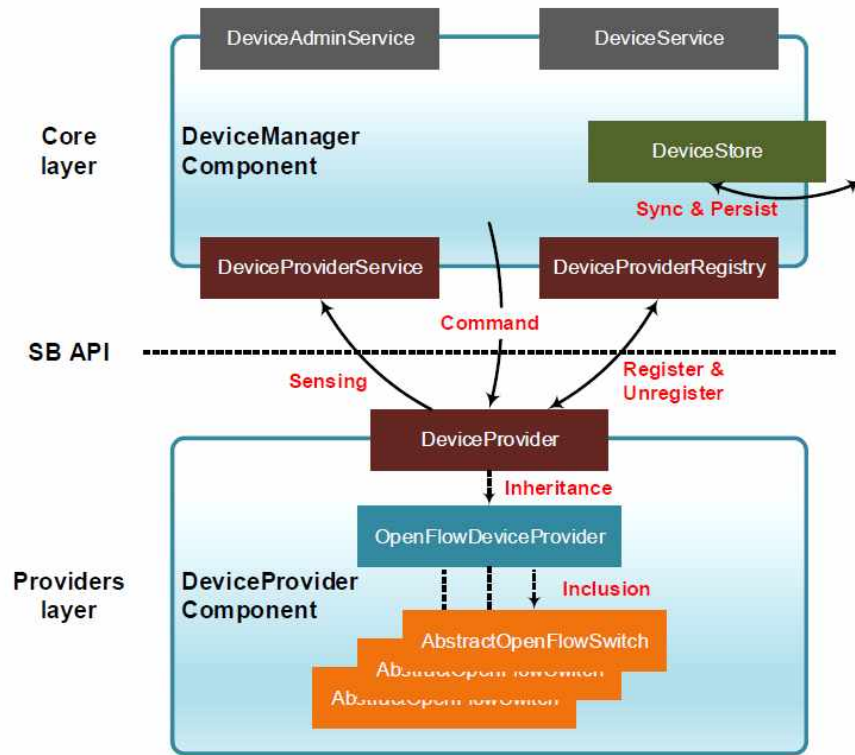


그림 6. ONOS의 디바이스 서브시스템 개념도[6]

각각의 어플리케이션이 분산 코어 계층을 거쳐 모니터링 결과를 출력하도록 디자인했다.

이와 같은 디자인을 실제 ONOS상에 구현하기 위해, 먼저 우리는 디바이스 서브시스템을 분석하였다. 그림 6은 디바이스 서브시스템의 구조를 나타낸다. 먼저 프로바이더 계층에는 SDN 네트워크를 구성하는 OpenFlow 스위치들이 추상화되어 있는 클래스(**AbstractOpenFlowSwitch**)가 존재하고, 이를 디바이스 프로바이더를 상속한 OpenFlow 디바이스 프로바이더 클래스(**OpenFlowDeviceProvider**)가 관리한다. 그리고 분산 코어 계층에서는 디바이스 매니저 클래스에서 디바이스 프로바이더와 통신을 하게 구성이 되어 있다. 이를 위해 ONOS 컨트롤러에는 디바이스 프로바이더 서비스 인터페이스(**DeviceProviderService**)와 디바이스 프로바이더 등록 인터페이스(**DeviceProviderRegistry**)가 정의되어서 사용되고 있다. 그리고 다른 분산 코어 계층에서 다른 ONOS 컨트롤러와의 동기화를 위해 디바이스 스토어(**DeviceStore**)를 정의하여 사용한다. 한편 어플리케이션 계층에서 디바이스 매니저를 사용하기 위해, 두 가지의 인터페이스를 정의하여 사용하고 있다. 첫 번째는 디바이스 서비스 인터페이스(**DeviceService**)이고, 다른 하나는 디바이스 관리자 서비스 인터페이스(**DeviceAdminService**)이다. 전자의 경우, 어떠한 사용자도 사용할 수 있는 서비스들을 정의하고 있는 인터페이스인 반면, 후자의 경우에는 네트워크 관리자와 같은 권한을 갖는 사용자만이 쓸 수 있는 기능들을 정의한 인터페이스이다.

이와 같은 디바이스 서브시스템 상에, 우리는 OpenFlow 메시지를 모니터링 할

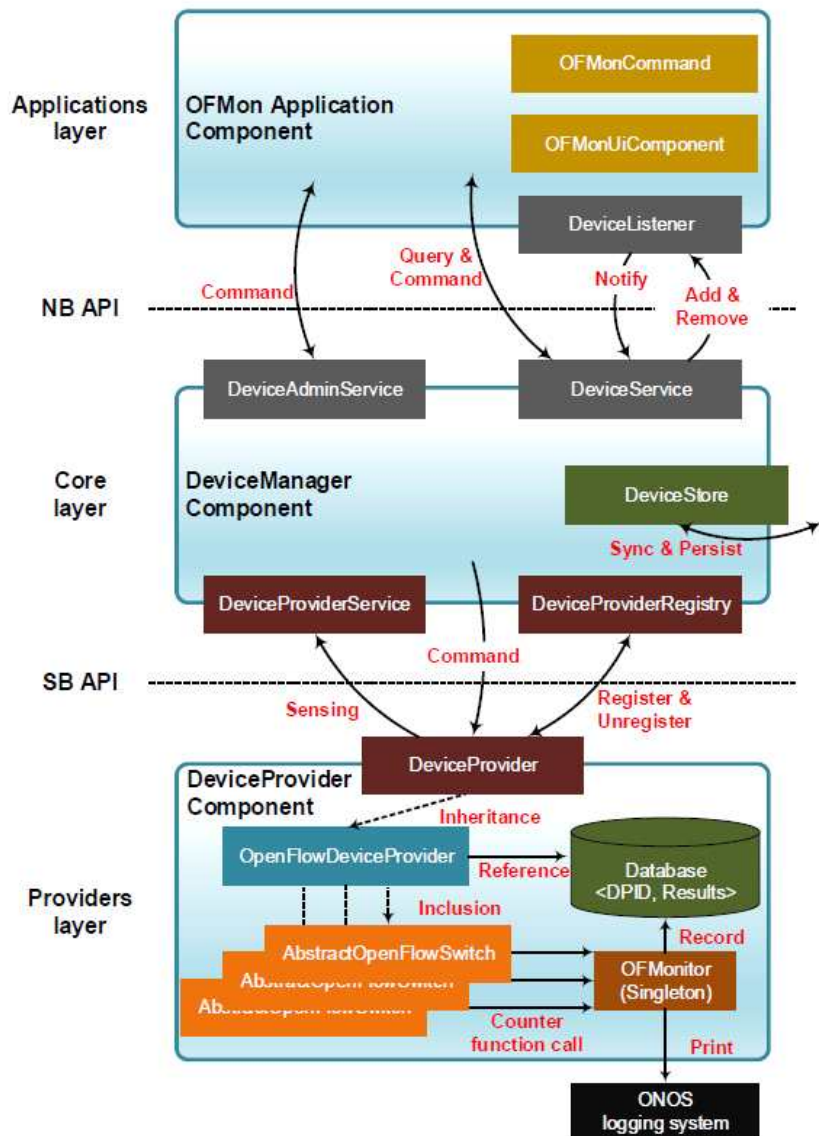


그림 7. OFMon의 초기 디자인[6]

수 있는 OFMon을 구현하였다(그림 7). 먼저, 프로바이더 계층에는 OpenFlow 메시지를 모니터링 할 수 있는 OFMonitor 클래스를 싱글톤(singleton) 패턴을 사용하여 구현하였다. 모니터링을 위해 AbstractOpenFlowSwitch 클래스를 수정하여, 해당 클래스에서 OpenFlow 메시지가 OpenFlowDeviceProvider로 올라가거나, OpenFlow 프로토콜 계층으로 내려가게 되면, OFMonitor에 내장된 카운터 함수를 구동하였다. 이와 같이 모니터링 된 결과는 OFMonitor가 프로바이더 계층에 새로이 정의한 Database에 모두 기록하였다. 한편 분산 코어 계층에서는 다른 ONOS 컨트롤러와 모니터링 결과를 수합하기 위해, 새로운 일관된 맵(consistent map)을 정의하고 해당 자료구조에 모두 저장하였다. 마지막으로 어플리케이션

계층에서는 디바이스 서비스 인터페이스를 통해 일관된 맵에 저장된 정보를 접근하여 출력하였다. 한편 OFMon에서 카운트 하는 OpenFlow 메시지는 ONOS에서 인식할 수 있는 모든 형태의 OpenFlow 메시지를 모니터링 하였다.

2.2. 문제점

위와 같은 ONOS 디바이스 서브시스템을 수정한 OFMon의 초기버전은 가장 큰 문제를 안고 있다. ONOS의 경우 오픈 소스 프로젝트(open-source project)로써, 수많은 개발자들과 협업을 통해 개발되고 있다. 이 때, ONOS상의 디바이스 서브시스템은 단일 개발자가 쉽게 수정하거나 삭제 및 기능 추가를 해서는 안 되는 가장 핵심적인 서브시스템이다. 그러나 OFMon은 이와 같은 핵심적인 서브시스템 상에 수정 및 기능 추가를 수행하여, 더러운 코드 이슈(dirty-code issue)를 야기하였다. 따라서 이러한 이슈를 해결하기 위한 새로운 OFMon의 디자인이 필요하다.

3. 새로운 ONOS 컨트롤러의 OpenFlow 모니터링 시스템 설계/구현

3.1. 디자인

2절에서 언급한 바와 같이, OFMon의 더러운 코드 이슈를 해결하기 위해, 본 연구진은 그림 8과 같이 OFMon을 새로운 서브시스템으로써 디자인하였다. 먼저 프로바이더 계층에서는, 기존에 디바이스 프로바이더에 존재하는 클래스들을 수정한 것과는 달리, 새로운 OFMon 프로바이더 클래스를 생성하였다. 그리고 프로바이더 계층 내에, 기존에 디바이스 프로바이더 계층에 있던 OFMonitor 클래스와 Database를 구현하였다. 기존의 OFMon에서는 AbstractOpenFlowSwitch에 있는 코드를 수정하여, OpenFlow 메시지가 ONOS로 수신되거나, OpenFlow 스위치로 송신되는 함수에 OFMonitor 클래스의 카운터 함수를 호출했다. 그러나 이는 더러운 코드 이슈를 야기하게 되므로, 우리는 이러한 과정을 옵저버(observer) 패턴을 통하여 해결하였다. 현재 AbstractOpenFlowSwitch에는 OpenFlow의 이벤트 리스너를 등록할 수 있도록 ONOS상에 구현이 되어 있다. 이 이벤트 리스너는 OpenFlow 메시지가 ONOS 컨트롤러로 송/수신이 된 경우, 리스너 내부에 있는 handleMessage 함수를 호출하도록 되어있다. 본 연구진은 여기에 착안하여, OFMonitor 클래스 내부에 새로운 OpenFlow 이벤트 리스너를 정의하고, 이를 AbstractOpenFlowSwitch에 등록하였다. 그리고 해당 리스너 안에 handleMessage

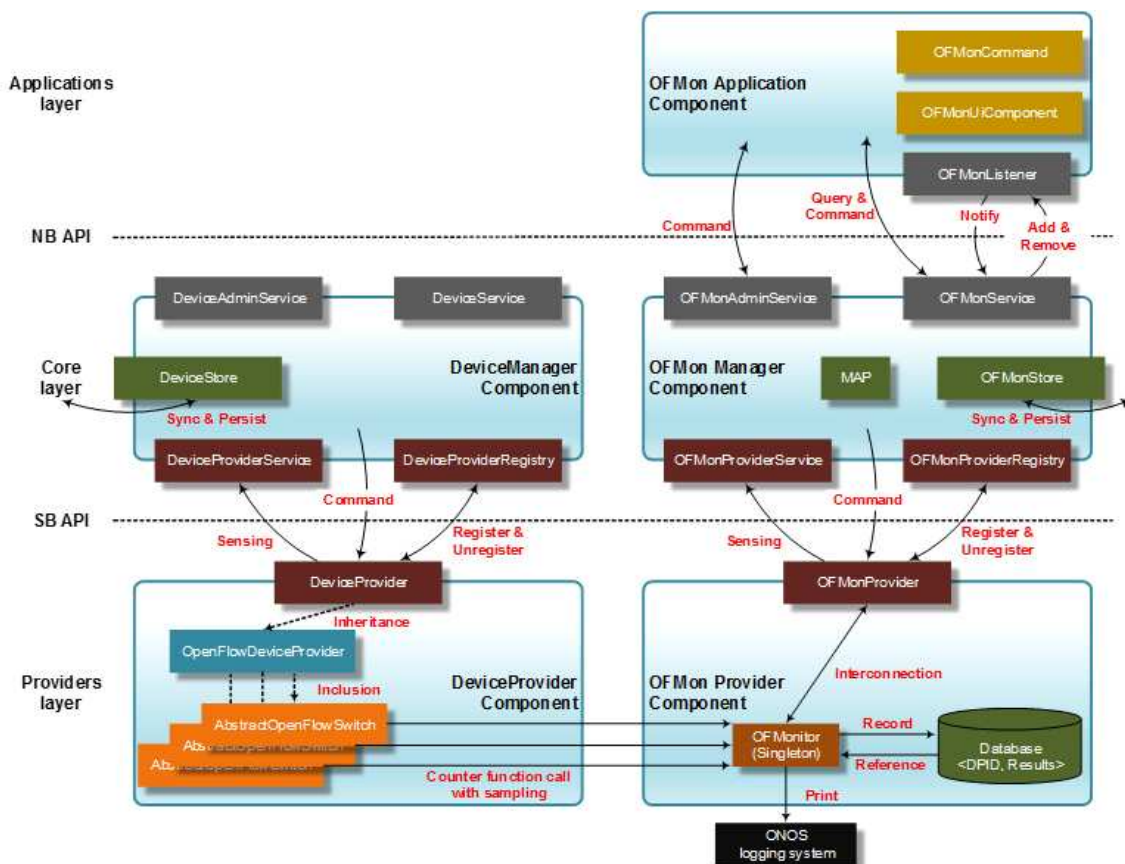


그림 8. ONOS 디바이스 서브시스템과 새로운 OFMon 서브시스템의 구조

함수가 호출 될 경우, OFMonitor 내부에 있는 카운터 함수를 호출하도록 수정하였다. 카운터 함수는 Database에 모니터링 결과를 등록하는 과정까지 포함이 되어 있다. 한편 Database의 구조는, 각각의 OpenFlow 스위칭 장비를 구분할 수 있는 구분자인 DPID를 키(key) 값으로 하며, 각각의 키 값은 각 메시지 및 메시지 별 카운트 결과를 담고 있는 Results 오브젝트를 가지는 맵(Map)형태의 자료 구조로 구성되어 있다. 실시간으로 모니터링 결과를 네트워크 관리자에게 보여주기 위해, 초기 OFMon과 마찬가지로 ONOS 로깅 시스템인 Karaf 로그 파일에 실시간으로 기록하는 함수 역시 정의되었다.

한편, 분산 코어 계층은 새로운 OFMon 매니저 클래스를 정의하였다. OFMon과 OFMon 프로바이더 계층 사이의 통신을 위해 먼저 OFMon 프로바이더 서비스 인터페이스(OFMonProviderService)를 정의하였고, OFMon 프로바이더의 등록을 위해 OFMon 프로바이더 등록 인터페이스(OFMonProviderRegistry)를 정의하였다. 이들 두 인터페이스를 활용하여, OFMon 매니저는 OFMon 프로바이더 계층의 database에 접근을 할 수 있다. 한편, 초기 OFMon과는 달리, 여러 ONOS 컨트롤러로부터 수집된 모니터링 정보를 디바이스 매니저에 저장하는 것이 아닌, OFMon 매니저에 일관된 맵(consistent map)과 OFMon 스토어(OFMonStore)를 새로이 구현하였다. 그리고 어플리케이션 계층이 통합된 정보를 호출하기 위해서, OFMon 서비스 인터페이스(OFMonService)와 OFMon 관리자 서비스 인터페이스(OFMonAdminService)를 새로이 구현하였다.

마지막으로, 어플리케이션 계층에서는 OFMon 초기 어플리케이션과 마찬가지로 GUI/CLI 기반의 어플리케이션이 구현되었다. 초기 OFMon에서는 디바이스 매니저에 직접 리스너를 등록하여, 디바이스 매니저 내부에 있는 맵 정보를 직접 호출했다. 그러나 새로운 OFMon에서는 OFMon 매니저 클래스가 이를 소유하고 있기 때문에, OFMon 매니저와 리스너 및 OFMon 서비스 인터페이스를 통해 이를 접근 및 호출하도록 수정하였다.

상기의 디자인 결과를 종합하여 보았을 때, 새로운 OFMon은 기존의 OFMon에서 야기하는 더러운 코드 이슈를 완전히 해결하였다. 왜냐하면 ONOS에서 핵심적인 디바이스 서브시스템의 어떠한 코드도 고치지 않았기 때문이다. 상기에서 언급한바와 같이, 이러한 문제의 해결을 위해 현재 널리 사용되고 있는 디자인 패턴 중 하나인 옵저버 패턴을 사용하였다. 옵저버 패턴을 활용하여, 리스너를 디바이스 서브시스템에 등록하고, 이를 통해 필요한 정보들을 새로운 서브시스템으로 가져오도록 새로이 구현하였다.

3.2. 수행 결과 - ONOS 로깅 시스템 상의 OFMon 출력 결과

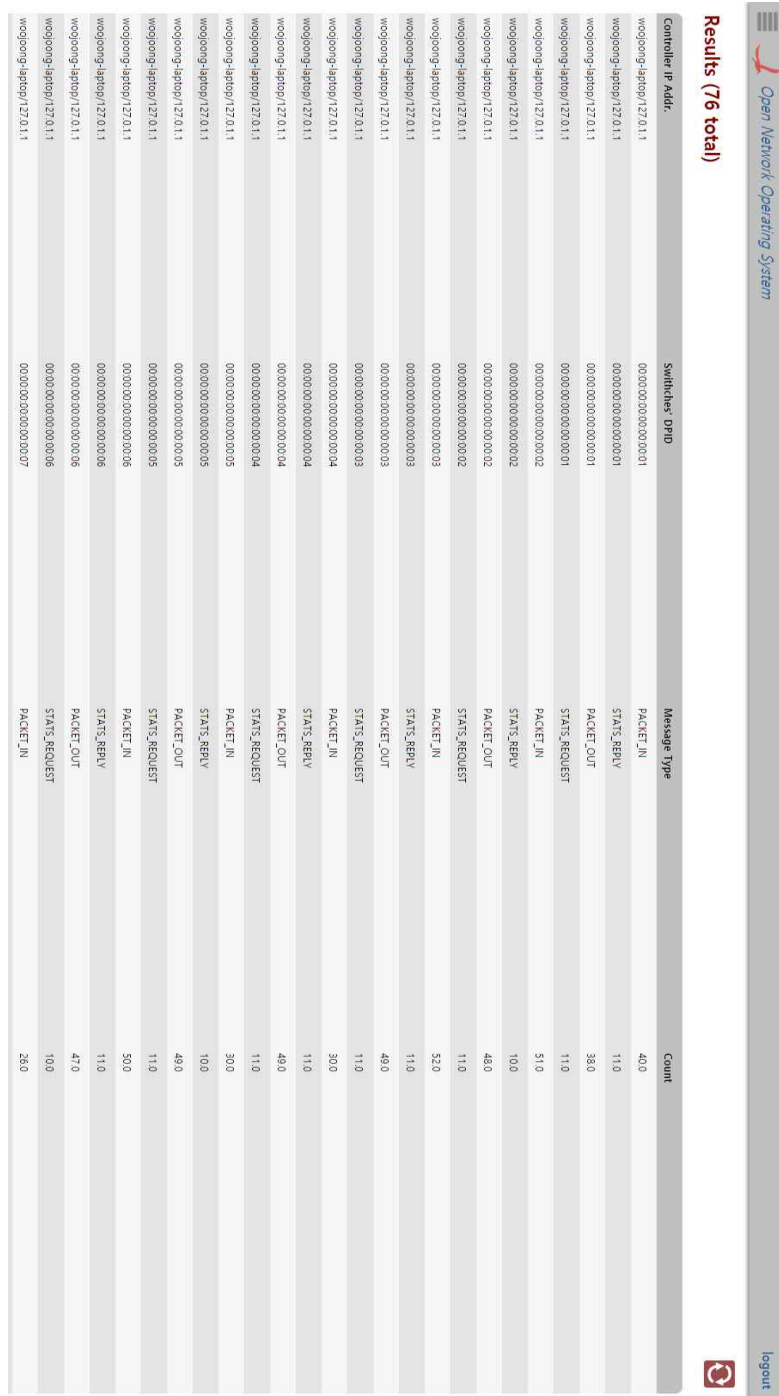
먼저 그림 9는 ONOS 로깅 시스템에서 출력된 결과를 표시하고 있다. 해당 결과에서 볼 수 있듯이, OFMon을 통해 현재 ONOS로 들어오는 OpenFlow 메시지(incoming OF message)와 ONOS에서 OpenFlow 스위칭 장비로 송신하는 OpenFlow 메시지(outgoing OF message)를 실시간으로 모두 출력할 수 있도록 구현이 되어 있다. 그리고 ONOS 로깅 시스템은 각각의 ONOS 컨트롤러마다 보는 시스템이므로, 각각의 ONOS 컨트롤러 로깅 시스템에 접속하여 확인을 할 수 있다.

```
| 160 - org.onosproject.onos-of-ctl - 1.3.0 | Incoming OF message event: PACKET_IN msg
| 159 - org.onosproject.onos-of-api - 1.3.0 | Outgoing OF message event: PACKET_OUT msg
| 160 - org.onosproject.onos-of-ctl - 1.3.0 | Incoming OF message event: PACKET_IN msg
| 159 - org.onosproject.onos-of-api - 1.3.0 | Outgoing OF message event: PACKET_OUT msg
| 159 - org.onosproject.onos-of-api - 1.3.0 | Outgoing OF message event: FLOW_MOD msg
| 160 - org.onosproject.onos-of-ctl - 1.3.0 | Incoming OF message event: PACKET_IN msg
| 159 - org.onosproject.onos-of-api - 1.3.0 | Outgoing OF message event: PACKET_OUT msg
| 160 - org.onosproject.onos-of-ctl - 1.3.0 | Incoming OF message event: BARRIER_REPLY msg
| 159 - org.onosproject.onos-of-api - 1.3.0 | Outgoing OF message event: FLOW_MOD msg
| 160 - org.onosproject.onos-of-ctl - 1.3.0 | Incoming OF message event: BARRIER_REPLY msg
| 159 - org.onosproject.onos-of-api - 1.3.0 | Outgoing OF message event: FLOW_MOD msg
| 160 - org.onosproject.onos-of-ctl - 1.3.0 | Incoming OF message event: BARRIER_REPLY msg
| 159 - org.onosproject.onos-of-api - 1.3.0 | Outgoing OF message event: STATS_REQUEST msg
| 160 - org.onosproject.onos-of-ctl - 1.3.0 | Incoming OF message event: STATS_REPLY msg
| 159 - org.onosproject.onos-of-api - 1.3.0 | Outgoing OF message event: PACKET_OUT msg
```

그림 9. ONOS 로깅 시스템에 출력되는 OFMon의 모니터링 결과

3.3. 수행 결과 - GUI 기반 어플리케이션 상의 OFMon 출력 결과

한편, 그림 10은 로깅 시스템이 아닌 ONOS에서 제공하는 웹 기반의 어플리케이션을 통해 모니터링 한 결과이다. 결과는 먼저 어떤 컨트롤러에서 수집된 결과인지, 어떤 OpenFlow 스위치에서 온 결과인지, 어떤 OpenFlow 메시지에 대한 결과인지, 그리고 각 메시지 당 송/수신된 횟수는 몇 번인지가 모두 표시되어 있다. 그리고 각각의 항목에 대하여 정렬을 할 수 있도록 구현이 되어 있다.



Controller IP Addr.	Switches' DPID	Message Type	Count
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:01	PACKET_IN	400
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:01	STATS_REPLY	110
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:01	PACKET_OUT	380
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:01	STATS_REQUEST	110
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:02	PACKET_IN	510
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:02	STATS_REPLY	100
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:02	PACKET_OUT	480
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:02	STATS_REQUEST	110
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:03	PACKET_IN	520
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:03	STATS_REPLY	110
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:03	PACKET_OUT	490
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:03	STATS_REQUEST	110
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:04	PACKET_IN	300
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:04	STATS_REPLY	110
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:04	PACKET_OUT	490
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:04	STATS_REQUEST	110
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:05	PACKET_IN	300
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:05	STATS_REPLY	100
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:05	PACKET_OUT	490
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:05	STATS_REQUEST	110
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:06	PACKET_IN	500
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:06	STATS_REPLY	110
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:06	PACKET_OUT	470
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:06	STATS_REQUEST	100
wcogong-laptop/127.0.1.1	00:00:00:00:00:00:07	PACKET_IN	260

그림 10. GUI 기반 어플리케이션 상의 OFMon 출력 결과

4. 결론

본 문서에서는 ONOS 상에 필요한 OpenFlow 메시지 모니터링 시스템에 대한 설계 및 구현을 다루었다. OpenFlow 메시지를 모니터링 하는 것은 제어 평면을 최적화 하는 기능을 수행함에 있어 가장 기본적으로 필요한 도구이므로, 본 문서에서 제안한 시스템은 ONOS 상에 반드시 구현되어야 할 필수 도구이다. 이를 바탕으로, 향후 본 연구진에서는 제어 평면을 최적화하기 위한 다양한 기능들을 추가로 구현할 계획이다. 이뿐만 아니라, 현재 구현한 OpenFlow 모니터링 시스템의 성능을 실제 네트워크 환경에서 평가한 후, 이를 바탕으로 성능 최적화를 수행할 계획이다.

References

- [1] Open Networking Foundation, "OpenFlow Switch Specification," *Technical Specification*, Apr. 2015.
- [2] Open Networking Foundation (ONF), [online] Available: <https://www.opennetworking.org>.
- [3] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *IEEE Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015.
- [4] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Toward an Open, Distributed SDN OS," in Proc. *ACM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, Aug. 2014.
- [5] ON. Lab, "ONOS Architecture Guide," *Technical Document*, Jan. 2015.
- [6] W. Kim, J. Li, J. W. -K. Hong, and Y. -J. Suh, "OFMon: OpenFlow Monitoring System in ONOS Controllers," To Appear in *IEEE NetSoft Workshop on Open-Source Software Networking (OSSN)*, Jun. 2016.

K-ONE 기술 문서

- K-ONE 컨소시엄의 확인과 허가 없이 이 문서를 무단 수정하여 배포하는 것을 금지합니다.
- 이 문서의 기술적인 내용은 프로젝트의 진행과 함께 별도의 예고 없이 변경될 수 있습니다.
- 본 문서와 관련된 문의 사항은 아래의 정보를 참조하시길 바랍니다.
(Homepage: <http://opennetworking.kr/projects/k-one-collaboration-project/wiki>, E-mail: k1@opennetworking.kr)

작성기관: K-ONE Consortium
작성년월: 2016/04