DPI 를 이용한 SDN 트래픽 매니지먼트 시스템

정세연 O, 이도영, 최준묵, 홍원기

포항공과대학교 컴퓨터공학과

{jsy0906, dylee90, juk909090, jwkhong} @postech.ac.kr

요 약

Software-Defined Networking (SDN)은 네트워크 분야에서 주목받는 연구 분야의 하나로써 그 기본 개념 및 목적은 네트워크를 컨트롤 평면과 데이터 평면으로 분리하여 관리 및 운용을 유연하게 만드는 것이다. 컨트롤 평면에 중앙집중화된 형태로 위치하는 컨트롤러는 스위치에 플로우 룰을 설치하고 관리하는 기능 등을 갖는데 이를 통해 효과적으로 네트워크를 관리 및 운용 할 수 있다. 현재까지 공개된 SDN 컨트롤러 중 ONOS 는 SDN 을 구성하는 여러 컴포넌트를 계층화하고 모듈 단위의 API 를 제공함으로써 손쉽게 어플리케이션의 개발 및 실행을 가능하게 한다. 하지만 이러한 편의성과 Programmability 에도 불구하고 현재 ONOS 의 기능을 충분히 활용하는 어플리케이션은 많지 않은 실정이다. 따라서 본 논문에서는 ONOS 가 제공하는 기능들을 활용하여 SDN 트래픽 관리 도구로 동작하는 Firewall 과 Bandwidth Manager 어플리케이션을 제안한다. 두 어플리케이션은 네트워크로 유입되는 플로우의 통과 여부와 Data rate 등을 결정하는 역할을 수행한다.

1. 서론

오늘날 우리는 인터넷이 없는 삶은 상상 할 수 없는 환경에서 살아가고 있다. 네트워크는 등장 이후 극적인 발전을 이루어 오늘과 같은 인터넷 시대를 만들어냈고, 덕분에 수 많은 사람들은 네트워크를 통해 다양한 정보를 얻고 SNS, Email 등의 서비스를 이용하며 살아가고 있다. 하지만 이런 발전에도 불구하고 네트워크 분야는 여전히 해결해야 할 문제들이 많이 남아있는 영역이기도 하다. 특히 네트워크는 이를 구성하는 수 많은 종류의 네트워크 기기들로 이루어져 있는데, 각 기기들이 가지는 인터페이스와 이를 제공하는 회사가 다양하다는 특징이 있다. 이는 다시 말하면 네트워크 설정을 변경하기 위해서는 그 네트워크를 구성하는 모든 기기들의 인터페이스에 맞게 각 기기들의 설정을 변경해야 한다는 것을 의미한다. 때문에 실제로 새로운 서비스 도입이나 네트워크 설정 변경을 목적으로 즉각적으로 네트워크에 변경사항을 적용하는 것은 불가능에 가깝다.

따라서 이러한 문제를 해결하기 위해 많은 노력들이 있어왔는데 그 중에서도 Software-Defined Networking (SDN)은 탁월한 해결방안 중의 하나로 여겨지고 있다. SDN 의 기본 개념은 네트워크를 컨트롤 평면과 데이터 평면으로 분리하여 관리 및 운용을 유연하게 만드는 것이다. 스위치와 라우터로 이루어진 데이터 평면은 패킷을 전달하는 역할을 하는데 이때 어느 경로로 패킷을 전달할 것인지는 컨트롤 평면의 플로우룰 설정에 따른다. 한편 컨트롤 평면은 데이터 평면을 관리 및 제어하는 역할을 갖는데 중앙 집중화된 컨트롤러가 컨트롤평면에 위치하여 다수의 스위치 또는 라우터를 관리한다. 이를 위해 컨트롤러는 자신이 관리하는 네트워크의 데이터 평면, 즉 스위치들과 끊임없이 통신하여 네트워크 상태를 파악해야 한다. 컨트롤러와 스위치들 간 통신을 위해서는 두 요소 간에 표준화된 통신 프로토콜이 요구되었고 그 결과 OpenFlow 프로토콜이 등장하였다. OpenFlow 표준을 따르는 스위치와 컨트롤러 사이에서 새로운 패킷이 스위치에 유입될 경우, 스위치는 패킷의 헤더 일부를 포함한 Packet-In 메시지를 컨트롤러로 전달하고 컨트롤러는 수신한 헤더 정보와 네트워크 토폴로지 정보를 이용하여 패킷의 라우팅 경로를 결정한다. 이후 컨트롤러는 패킷의 진행 경로 상의 스위치들에 OpenFlow 메시지를 통해 플로우룰을 설치한다. 이러한 과정을 통해 SDN 에 특정 플로우의 경로가 결정되고 해당 플로우는 이 경로를 통해 목적지까지 전달된다.

SDN 이 주목 받기 시작하면서 현재까지 수 많은 OpenFlow 기반 오픈소스 SDN 컨트롤러가 개발되었다. 그 중 OpenDaylight [1]는 다중 Southbound 프로토콜 플러그인들과 다양한 서비스 및 어플리케이션들을 제공한다. 이를 통해 어플리케이션 개발자와 연구자들이 네트워크 디바이스들간의 통신보다 SDN API 들에 더 집중할수록 돕고 있다. Ryu [2]는 Python 으로 제작된 컴포넌트 기반의 SDN 프레임워크로써, 체계적으로 정의된 API 와 소프트웨어 컴포넌트들을 제공하여 개발자들이 쉽게 네트워크 관리 어플리케이션을 개발할 수 있는 환경을 제공한다. Floodlight [3]은 Apache-licensed 의 Java 기반 컨트롤러로 Big Switch Networks 에 소속된 많은 개발자들에 의해 개발되어 공개되었다. Floodlight 는 컨트롤러를 쉽게 확장하고 개발할 수 있도록 모듈 로딩

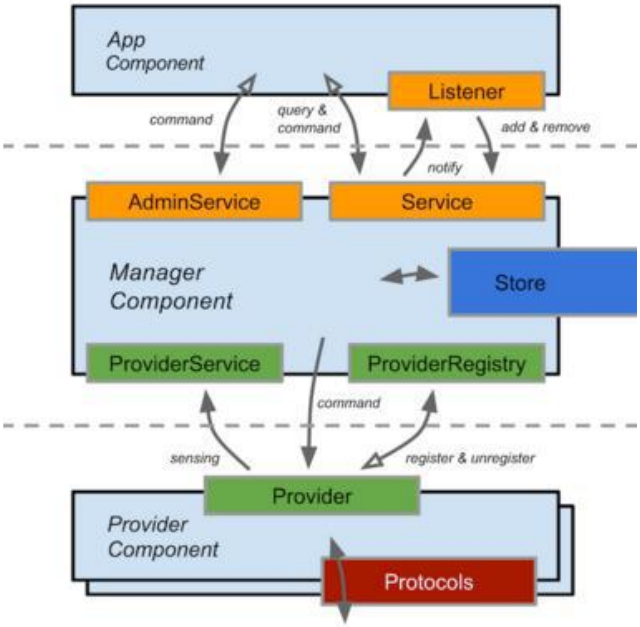
시스템을 제공한다는 특징이 있다. ONOS[4]는 성능, 확장성, 가용성을 중시하는 분산형 SDN 컨트롤러이며 편의성 있는 Web GUI 와 SDN 컴포넌트 단위의 계층 구조를 통해 어플리케이션을 개발하고 컨트롤러에 탑재할 수 있는 기능들을 제공한다. 하지만 ONOS 는 상대적으로 최근에 공개되었고 컨트롤러의 기능 자체에 대한 개발이 우선시 되어왔기 때문에 ONOS 기반 어플리케이션들은 많지 않은 실정이다. 따라서 본 논문에서는 ONOS 가 제공하는 기능을 활용하여 SDN 트래픽 관리 도구로써 동작하는 Firewall 과 Bandwidth Manager 어플리케이션을 제안한다. 이들은 데이터 평면에 존재하는 플로우를 TCP/UDP 포트번호 또는 외부 DPI 프로그램을 통해 분석하여 해당 트래픽을 특정 어플리케이션 및 프로토콜로 분류하며, 이러한 정보를 통해 네트워크의 보안을 강화하고 어플리케이션에 따라 트래픽의 Data rate 을 동적으로 할당하는 등 네트워크를 보다 효율적이고 유연하게 관리하는 것을 목적으로 한다.

본 논문의 이후 구성은 다음과 같다. 우선 2 장에서는 SDN 의 관련 연구 및 배경에 대해 설명하고, 3 장에서는 이 논문에서 제안하는 어플리케이션인 Firewall 과 Bandwidth Manager 의 구체적인 구조 및 구현에 대해 서술한다. 4 장에서는 두 어플리케이션의 실제 검증 결과에 대해 소개하고 마지막 5 장에서는 결론 및 향후 연구 계획에 대해 서술한다.

2. 관련 연구 및 배경

SDN 의 등장 이후, SDN 의 특성을 이용하여 네트워크의 보안성을 강화하기위한 연구가 많이 진행되어왔다. FlowGuard [5]는 SDN 기반의 Firewall 로서 패킷 또는 플로우 단위의 세밀한 수준에서 트래픽을 제어한다. 또한, 이미 존재하는 플로우룰과 FlowGuard 에 의해 새롭게 설치된 플로우룰의 충돌 가능성을 고려함으로써 오픈소스 컨트롤러에 내장된 기본적인 Firewall 보다 더 향상된 기능을 제공한다. 하지만 FlowGuard 는 해당 트래픽을 발생시키는 어플리케이션 또는 프로토콜을 식별할 수 없다는 단점이 있다. 이를 극복하기 위한 노력 중의 하나로 Application-aware SDN 을 구축하기 위해 M Jarschel [6]등은 DPI 를 이용하여 YouTube 트래픽을 분류하고, 해당 트래픽을 발생시키는 호스트에서 버퍼 기반의 네트워크 최적화 기법을 적용시켰다. 이를 통해 네트워크 자원을 효율적으로 사용하는 것이 가능해졌지만 오직 YouTube 트래픽만을 고려한다는 한계가 존재한다. 한편 Zafar Qazi [7]등은 Application-awareness 시스템을 SDN 에 접목한 Atlas 프레임워크를 제안했다. 이는 별도의 DPI 를 이용하지 않고 각 스위치의 버퍼에 쌓인 패킷들을 Machine Learning 기법을 이용하여 특정 어플리케이션의 트래픽으로 분류한다. 하지만 이는 식별된 어플리케이션 트래픽의 Data rate 을 제어할 수 없고 오직 Drop 여부만을 결정 할 수 있다는 한계가 있다.

본 논문에서 제안하는 시스템은 OpenFlow 기반의 SDN 에서 트래픽 분류에 따른 Firewall 기능 및 동적 Data rate 할당 기능을 제공하며, ONOS 컨트롤러에서 동작하는 어플리케이션 형태로 구현되었다. 제안하는 어플리케이션은 ONOS 의 구현 관점의 계층 [그림.1]에서 최상위 계층인 App Component 에서 동작한다. 어플리케이션은 Packet-In 메시지 및 플로우 관련 통계 값들을 하위 계층으로부터 받아오고, 이를 이용하여 특정 네트워크 정책을 결정한다. 그리고 이러한 정책을 네트워크에 적용하기 위한 요청을 하위 계층으로 전달한다. ONOS 는 SDN 을 관리하기 위해 다양한 기능들을 수행하는데 이에 대한 Programmability 를 지원하기 위해 하위 계층에 다양한 Manager 와 Provider 가 존재한다.



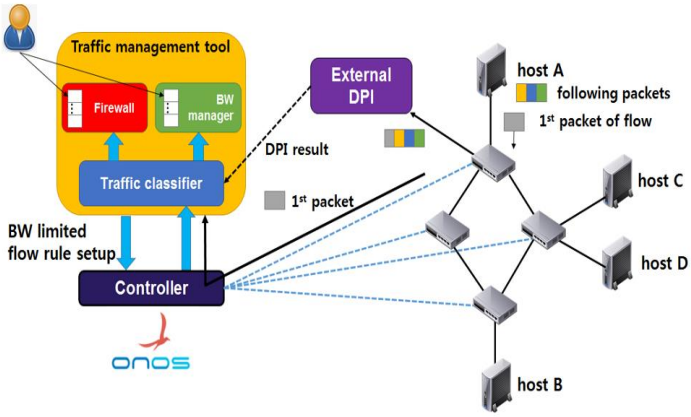
[그림.1] ONOS 구조

3. 시스템 구조 및 구현

본 논문에서 제안하는 시스템은 [그림.2]처럼 다음 세 가지 요소로 구성된다. (1) 트래픽 관리 도구, (2) 호스트와 OpenFlow 프로토콜을 지원하는 스위치들로 구성된 SDN 토폴로지, (3) 외부 DPI 프로그램. 첫 번째 구성요소인 트래픽 관리 도구는 ONOS 컨트롤러 위에서 동작하며, 두번째 구성요소인 OpenFlow 지원 스위치들은 ONOS 컨트롤러에 연결되어 있다. 외부 DPI 프로그램은 Edge 스위치에 새로운 플로우가 유입될 경우 이에 대한 DPI 를 실시하고 분석된 결과(어플리케이션 타입)를 컨트롤러로 전달한다. 이를 통해 컨트롤러는 해당 플로우에 어느 정도의 Data rate 을 할당할지 설정하거나 플로우 자체의 유입을 차단할 수 있다. 이러한 결정은 네트워크 관리자가 ONOS Web GUI 를 통해 트래픽 관리

도구에 설정한 정책을 따른다.

본 연구에서는 (2)에 해당하는 SDN 환경을 구성하기 위해서 Mininet [8] 네트워크 에뮬레이터를 활용해서 가상의 SDN 네트워크를 구성하였다. 그리고 OpenFlow 1.3 에서 지원하는 Meter 테이블 기능을 이용하기 위해 해당 기능들이 구현된 CpqD/ofsoftswitch13 [9]을 OpenFlow 소프트웨어 스위치로 설치하였다. 또한 (3) 외부 DPI 프로그램으로는 오픈 소스 DPI 라이브러리인 nDPI 를 사용하였다. nDPI 는 테스트베드에서 Stand-alone 으로 동작하며 높은 정확도로 200 개 이상의 어플리케이션 트래픽을 분류할 수 있다[10]. nDPI 는 Edge 스위치들 위에서 동작하며 유입된 플로우의 DPI 결과를 REST API 를 통해 ONOS 컨트롤러로 전달한다.

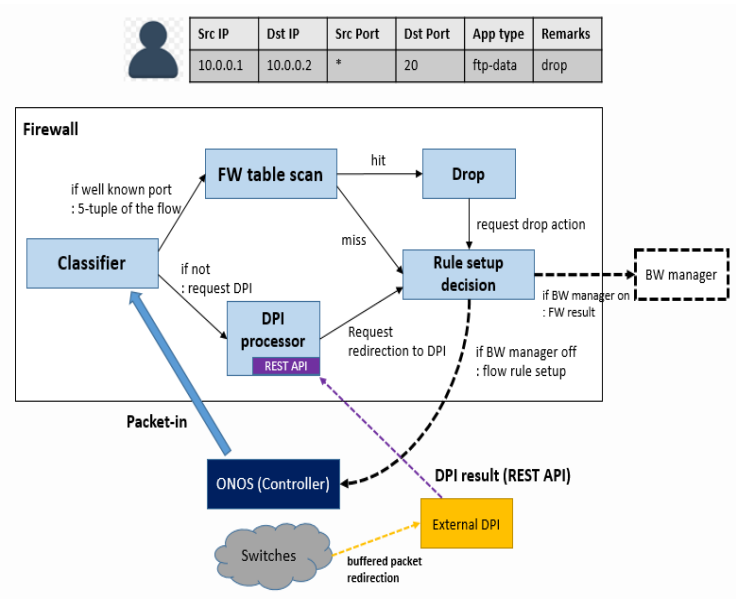


[그림.2] 시스템 전체 구조

A. Firewall

일반적으로 OpenFlow 기반 Reactive 방식의 SDN 에서 새로운 플로우가 유입될 때, 스위치에는 라우팅을 위한 플로우 룰이 설치되어 있지 않다. 이러한 경우에 해당 플로우의 Layer 2 부터 Layer 4 까지의 헤더 정보가 Packet-In 메시지를 통해 컨트롤러로 전달되고, 컨트롤러는 수신한 메시지를 분석하여 경로 설정에 필요한 IP 주소와 TCP/UDP 포트번호 등의 정보를 추출한다. 본 논문에서 제안하는 [그림.3]의 Firewall 은 이러한 정보를 활용하여 TCP/UDP 포트번호 기반 트래픽 분류를 수행, 해당 플로우의 어플리케이션 타입을 식별한다. 만약 해당 플로우가 포트번호 기반으로 분류가 가능할 경우, 플로우의 5-tuple (Source/Destination IP, Source/Destination 포트번호, 프로토콜) 정보를 추출하여 매칭되는 Firewall 룰이 있는지 확인하고 룰의 정책에 따라 해당 플로우를 처리한다. 하지만 동적포트할당(Dynamic Port Allocation) 등의 이유로 포트번호만으로 분류되지 않는 플로우의 경우에는 컨트롤러의 REST API 를 통해 외부 DPI 프로그램으로부터 해당 플로우의 DPI 결과(어플리케이션 타입)를 전달 받아 분류한다. 보다 상세히 설명하면, [그림 2.]를 기준으로 Sender 호스트로부터 발생하는 모든 플로우는 DPI 를 통과하게 되며 분석 중에 새로운 플로우가 식별될 경우 타임스탬프와 해당 플로우의 5-tuple 및 어플리케이션 타입 정보를 컨트롤러로 전달한다(REST API). 컨트롤러는 외부 DPI 프로그램으로부터 전달받은 플로우에 대한 DPI 결과와 스위치로부터 Packet-In 메시지 형태로 전달 받은 플로우 정보를 비교하여 동일한 플로우로 판단되는 경우, 해당 플로우의 5-tuple 과 DPI 결과를 바탕으로 매칭되는 Firewall 룰을 찾아 적용한다. 이와 같은 경우, 외부 DPI 의 플로우 분석 및 REST API 를 통한 결과 전달에 걸리는 시간과 스위치로부터의 Packet-In 메시지의 도착 사이에 시간 차가 존재할 수 있기 때문에 먼저 받은 DPI 결과들을 캐시하거나 DPI 결과가 도착할 때까지 플로우의 처리를 지연시키는 등의 동기적 처리 방식이 요구된다.

Firewall 룰은 관리자가 네트워크를 보호하기 위해 미리 설정하는 것으로 가정하는데, 이 룰들의 명세는 <IP 주소, 포트 번호, 프로토콜(어플리케이션), Action>으로 구성된다. 만약 특정 플로우를 Drop 하는 룰이 정의되어 있다면 Firewall 은 컨트롤러에게 해당 플로우를 Drop 할 것을 요청하고, 컨트롤러는 플로우가 유입된 스위치에 해당 플로우를 Drop 시키는 플로우룰을 설치한다. 결과적으로 네트워크 관리자가 원하지 않은 플로우들은 더 이상 네트워크로 진입하지 못하고 Edge 스위치에서 Drop 된다. 반면, 네트워크 진입이 허용된 플로우의 경우 (Firewall 의 Forward 액션), Bandwidth Manager 로 전달되어 Data rate 제어 및 이후의 플로우 처리 과정이 진행된다.

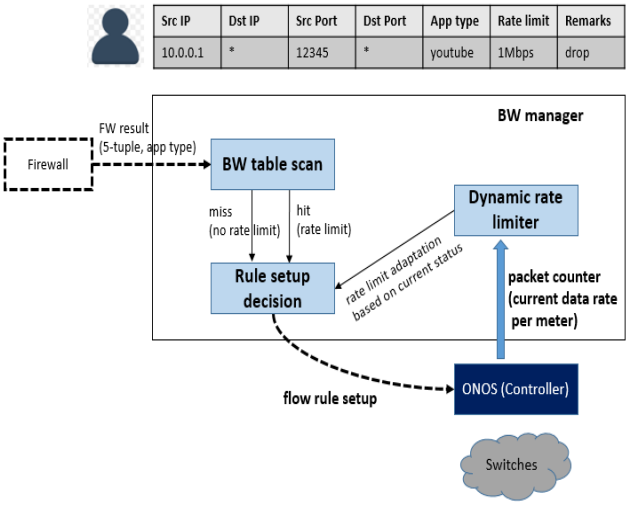


[그림.3] Firewall 설계

B. Bandwidth Manager

Bandwidth Manager 는 [그림.4]와 같이 네트워크 관리자로부터 5-tuple 정보, 어플리케이션 타입 및 Date rate 한도를 Bandwidth 룰로 전달 받는다. 이

룰들은 [그림.5]와 같이 ONOS Web UI 를 통해 네트워크 관리자로부터 입력 받아 테이블 형태로 저장된다. Firewall 을 통과한 플로우는 Bandwidth Manager 에서 미리 설정된 Bandwidth 룰과 매칭되는지 확인된다. 매칭되는 룰이 존재할 경우 Bandwidth Manager 는 해당 플로우의 라우팅을 위한 플로우룰의 설치과정에서 매칭 액션의 한 형태로 OpenFlow 1.3 에서부터 제공되는 Meter band 를 적용한다. 이를 통해 해당 플로우에 적용되는 Data rate 의 한계치를 지정할 수 있다. 컨트롤러는 스위치에 대한 주기적인 OpenFlow 폴링을 통해 플로우룰에 설정된 Meter band 와 관련된 Meter 통계값 (OFPMP_METER)을 Dynamic Rate Limiter 로 전달한다. Meter 통계값은 특정 플로우룰에 적용된 Meter band 의 영향을 받은 패킷 개수와 Byte 정보를 제공한다. Dynamic Rate Limiter 는 설치된 플로우룰의 Meter band 값을 플로우의 상태에 따라 동적으로 조절한다. 예를 들어, 일반적인 플로우룰 패킷 통계값 (OFPMP_FLOW)을 통해 계산된 평균 패킷 처리량과 해당 플로우룰에 설정된 Meter band 에 따라 Drop 된 패킷의 양을 비교했을 때, 일정시간 동안 빠르게 차이가 좁혀지는 경우 설정된 Meter band 의 Data rate 이 플로우의 QoS 를 지나치게 감소시킨다고 판단한다. 따라서 Dynamic Rate Limiter 는 기존에 설정된 Meter band 의 Data rate 을 OFPT_METER_MOD 메시지를 통해 증가시키고 이에 대한 정보를 컨트롤러로 전달해 네트워크 관리자에게 알린다. 이와 같은 과정들은 OpenFlow 1.3 에 소개된 Meter 테이블 기능을 적극 활용하여 이루어진다.

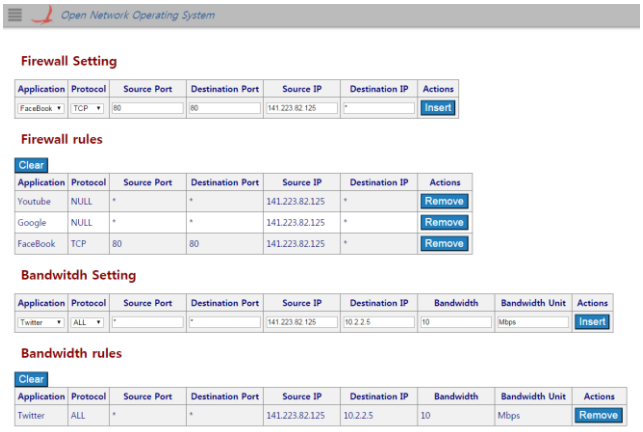


[그림.4] Bandwidth Manager 설계

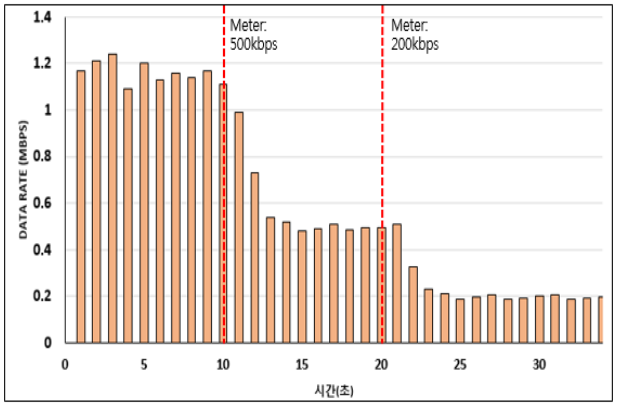
4. 시스템 검증

구현된 프로토타입의 성능평가를 위해 간단한 테스트 베드에서 Bandwidth Manager 의 Meter band 적용에 관한 실험을 진행하였다. 테스트베드는 Mininet 위에 구축된 Sender, Receiver 호스트가 OpenFlow 1.3 을 지원하는 소프트웨어 스위치를 토해 연결되어 있고, 스위치에는 nDPI 가 Stand-alone 으로 동작하여 Sender 와 Receiver 간에 트래픽을 분석한다. 또한 스위치는 제안된 어플리케이션이 동작하고있는 ONOS 컨트롤러와 연결되어 Bandwidth Manager 의 Data rate 정책을 포함하는 플로우룰이 설치된다. Sender 와 Receiver 사이에 FTP 세션을 만들고 Sender 에서 대용량 파일을 전송할 때 네트워크 관리자가 전송 시작 후 10 초 시점에서 Bandwidth Manager 를 통해 해당 세션의 Meter band 를 0.5 Mbps 로 제한하고 20 초 시점에서는 0.2 Mbps 로 설정하는 시나리오를 수행한다. 시나리오의 수행결과는 [그림.6]과 같다. 파일 전송이 시작되면 컨트롤러는 DPI 의 분석결과를 토대로 FTP 플로우를 식별하여 Sender 와 Receiver 사이의 스위치에 플로우룰을 설정한다. 전송 시작부터 10 초 간은 Data rate 의 제한이 없으므로 해당 플로우룰에 적용되는 Meter band 없이 대략 1.2Mbps 의 속도로 전송된다. 10 초 시점에 관리자가 Bandwidth Manager 를 통해 Sender 와 Receiver 사이의 FTP 플로우에 0.5 Mbps 의 Meter 를 적용하면, Receiver 에서는 대략 3 초 후에 0.5 Mbps 의 Data rate 을 측정할 수 있다. 마찬가지로 20 초 시점에서도 0.2 Mbps 의 Meter 가 적용될 때 변화되는 Data rate 을 확인할 수 있다. Meter 의 적용과 감지되는 Data rate 의 시간은 네트워크의 복잡도나 모니터링 프로그램에 따라 달라질 수 있다.

현재 OpenFlow Meter 의 구현은 설정된 Data rate 를 초과하는 패킷을 강제로 Drop 시키거나 IP 헤더의 DSCP 필드에 매칭하여 Drop 의 Precedence 를 조정하는 방식을 지원하고 있다 [11]. 본 논문에서는 강제 Drop 방식을 사용했는데, TCP 기반의 FTP 실험에서 Meter band 를 초과하는 Data rate 이 감지되는 시점 직후에 패킷 Drop 에 따른 Data rate 이 감소했다가 다시 Meter band 값 수준으로 증가하는 패턴을 확인할 수 있다.



[그림.5] 어플리케이션 Web UI



[그림.6] Meter band 적용에 따른 Data rate

5. 결론 및 향후 연구

본 논문에서는 SDN 에서 트래픽 관리를 위해 DPI 를 이용하는 ONOS 어플리케이션을 제안하였다. OpenFlow 기반 SDN 에서 네트워크를 구성하는 스위치가 새로운 플로우의 패킷을 수신할 경우 이에 대한 헤더 정보를 Packet-In 메시지를 통해 컨트롤러에게 전달하고, 컨트롤러는 해당 정보를 통해 패킷이 어느 경로를 통해 전달되어야 하는지를 결정한다. 본 논문에서는 이러한 특징을 바탕으로 Firewall 을 구현하였는데, 플로우의 헤더 정보를 관리자에 의해 미리 설정된 Firewall 룰과 비교하여 해당 플로우를 Drop 하여 Edge 스위치에서 차단할 것인지 또는 라우팅 경로를 설정하여 목적지로 전달 할 것인지를 결정한다. 컨트롤러가 새롭게 유입되는 패킷의 헤더 정보를 확인하는 것은 OpenFlow 기반 스위치와 컨트롤러간에 표준을 따르는 것이기 때문에 Firewall 은 이러한 정보를 손쉽게 제공받을 수 있다. OpenFlow 기반 SDN 의 기본적인 동작 방식에서는 컨트롤러가 전달받은 패킷을 통해 해당 플로우의 라우팅 경로를 결정하고 경로상의 스위치에 적합한 플로우룰을 설치하는 등의 수동적인 역할을 하는 것에 비해, 본 연구에서는 Firewall 을 동작 시킴으로써 해당 플로우의 네트워크 진입 여부를 네트워크 관리자가 제어하게 되어 네트워크의 보안성을 높일 수 있게 된다. 이와 유사하게, Bandwidth Manager 는 플로우의 헤더 정보와 Meter band 를 이용하여 특정 어플리케이션의 트래픽 단위로 Data rate 을 제어할 수 있다. 플로우의 헤더 정보만으로는 트래픽 분류가 어려울 경우 외부 DPI 프로그램의 트래픽 분석 결과를 활용할 수 있다.

본 논문에서 제안한 시스템의 주요기능이 검증을 통해 정상적으로 동작하는 것은 확인되었지만 아직까지는 프로토타입 수준이기 때문에 향후 개선되어야 할 여지가 있다. 특히 보다 크고 다양한 어플리케이션 트래픽이 발생하는 네트워크에서 관리자가 취할 수 있는 여러 시나리오를 통해 제안한 방식이 잘 동작하는지 확장성 측면에서 검증할 필요가 있다. 또한, Bandwidth Manager 에서 이용하는 Meter band 가 OpenFlow 1.3 버전 이상에서 지원되는데, 실제 환경에서 잘 동작하기 위해 해당 기능을 구현하는 소프트웨어 또는 하드웨어 스위치가 필수적이다.

6. 감사의 글

이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-15-1009, ICBMS 플랫폼 간 정보 모델 연동 및 서비스 매쉬업을 위한 스마트 중재 기술 개발)

7. 참고 문헌

[1] Medved, Jan, et al. "Opendaylight: Towards a model-driven sdn controller architecture." *2014 IEEE 15th International Symposium on*. IEEE, 2014.
[2] Ryu. http://osrg.github.io/ryu/.
[3] Floodlight. http://www.projectfloodlight.org/floodlight/.
[4] Berde, Pankaj, et al. "ONOS: towards an open, distributed SDN OS." *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014.
[5] Hu, Hongxin, et al. "FLOWGUARD: building robust firewalls for software-defined networks." *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014.
[6] Jarschel, Michael, et al. "Sdn-based application-aware networking on the example of youtube video streaming." *Software Defined Networks (EWSDN), 2013 Second European Workshop on*. IEEE, 2013.
[7] Qazi, Zafar Ayyub, et al. "Application-awareness in SDN." *ACM SIGCOMM Computer Communication Review*. Vol. 43. No. 4. ACM, 2013.
[8] Mininet. http://mininet.org.
[9] Ofsoftswitch13 cpqd. https://github.com/CPqD/ofsoftswitch13.
[10] Deri, Luca, et al. "nDPI: Open-source high-speed deep packet inspection."*Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*. IEEE, 2014.
[11] Mohan, Purnima Murali, Dinil Mon Divakaran, and Mohan Gurusamy. "Performance study of TCP flows with QoS-supported OpenFlow in data center networks." *Networks (ICON), 2013 19th IEEE International Conference on*. IEEE, 2013.