편집위원 : 김동기, 임용재 (방송통신위원회)

SDN 표준 참조구조 기반의 개방형 인터페이스, 추상화 기술 및 컨트롤러 언어 분석

신명기, 남기혁, 최윤철, 김형준
한국전자통신연구원

요 약

본 고에서는 최근 활발히 연구되고 있는 SDN (Software-Defined Networking) 기술과 관련하여 다양한 SDN의 정의와 개념을 포괄하는 표준 참조 구조 (Reference Architecture)를 제안하고, 이를 기반으로 대표적인 표준화 대상인 개방형 인터페이스, SDN 스위치 추상화와 컨트롤러 응용 언어 기술에 대해 논의한다.

Ⅰ. 서 론

최근 기존 네트워크를 개방화, 네트워크 가상화, 프로그램화 기능등을 통해 고객의 요구에 따라 보다 유연하게 제어, 설정, 관리를 제공해 주는 새로운 네트워킹 개념으로 SDN 기술이 국내외로 크게 논의 중에 있다. SDN의 초기 실현은 스탠포드에서 미래인터넷 프로그램의 일환으로 개발되기 시작한 오픈플로우 개념에서 출발한다. 오픈플로우는 라우터나 스위치와 같은 네트워크 장치에 종속적인 형태로 제공되던 제어 기능을 논리적으로 중앙 집중적인 형태로 분리하여, 표준 인터페이스를 통해 통신하는 구조를 토대로 개발된 기술이다. 현재 오픈플로우는 SDN 개념으로 확장하여, Google, Facebook, Verizon, Cisco 등과 같은 업체 중심으로 새롭게 결성된 표준화 기구인 ONF(Open Networking Foundation)를 통해 현실 적용에 필요한 표준 규격과 기술을 개발하고 있으며, 기존 인터넷기술표준을 제정하는 IETF에서도 주니퍼와 시스코를 중심으로 종래의 오픈플로우 기술과는 다른 I2RS (Interface to the Routing System)이라는 새로운 기술을 제안하여 기존 레거시 장비에서도 오픈플로우와 유사하게 내부 라우팅 제어나 관리를 위한 외부 인터페이스를 제공하도록 하는 표준을 개발중에 있다.

SDN 개념은 처음 학술적인 측면에서 논의가 시작하여 실제 상용화를 위해 현재는 오픈플로우-기반 SDN, I2RS-기반 SDN 등 다양한 모델이 제시되고 있으며, ITU-T, IETF, ONF 등의 표준화기구에 따라 조금씩 다른 관점을 가지고 논의되고 있다. 현재 논의 되고 있는 SDN 개념을 정리하여 공통적인 특징을 살펴보면, 대략 다음과 같은 세가지 주요 특징으로 수렴될 수 있다.

① 제어기능과 데이터 전송의 분리
② 동적인 네트워크 프로그래밍과 중앙집중방식의 설정과 관리가 가능한 제어 방식의 지원
③ 하부 인프라 네트워크 및 ICT 가상화를 통한 다중 가상 네트워크의 지원

〈그림 1〉은 이를 정리하여 기존 네트워킹 구조와 SDN의 네트워킹 구조를 비교 기술하여 도식화한 것이다.

우리나라에서는 스마트인터넷을 미래의 인터넷을 실현하기 위한 첫번째 네트워크라고 정의하고 표준화 및 다양한 기술개발 과제 등을 추진하고 있다. 이때 SDN은 이를 실현하기 위한 주요 기술로서 "네트워크 장비의 기능을 정의할 수 있는 API를 제공하여 소프트웨어적으로 네트워크 경로 및 제어 등을 동적으로 프로그램할 수 있도록 함으로서 다양한 융복합 서비스를 위한 최적화된 네트워킹 환경을 구현할 수 있도록 하는 기술"로 SDN을 보다 구체적으로 정의하고 있다.

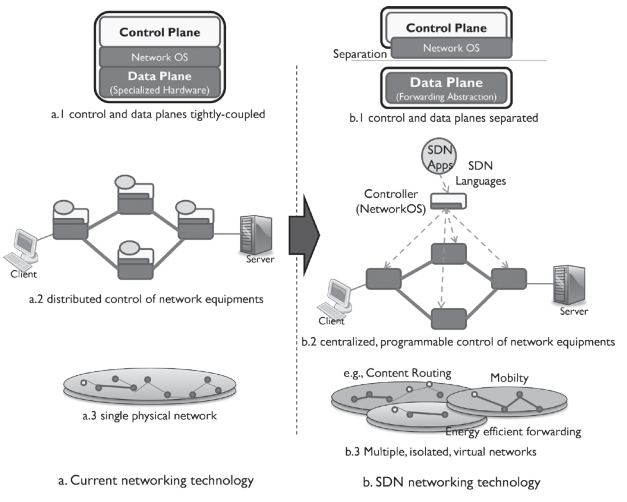


그림 1. 기존 네트워킹 구조와 SDN 구조 비교

Ⅱ. 다양한 SDN모델 수용을 위한 표준 참조구조 및 개방형 인터페이스를 위한 요구사항 분석

SDN은 단순화된 표준 기반의 기술 및 생태계를 포괄하고 있다. 이를 기반으로 SDN의 생태계를 이루는 각 통신사업자, 장비 제조업체, 소프트웨어 개발업체들은 자사의 전략에 따른 다양한 SDN 모델을 발표하고 있으며, 이때의 시사점은 SDN 솔루션 기술이 전에 없던 아주 새로운 기술이라기 보다는 종래의 자사가 보유한 기술을 기반으로 솔루션을 발표하고 제안하고 있다는 점에 주목할 필요가 있다. 따라서 SDN 상용화 초기에는 다양한 SDN 모델 및 플랫폼들이 제안될 가능성이 있으며, 시장의 경쟁에 따라 2~3개의 모델로 정착될 가능성이 있다고 보여진다. 본 장에서는 이러한 다양한 SDN 구조 논의에 따라 우선 공통적인 요소들을 추출하고 이를 개념화 할수 있는 표준 참조 모델을 제안하며, 이를 구성하기 위한 요구사항 들을 하나씩 분석하여 필요한 부분을 맞춤화된 형태로 빌딩블록화 할수 있도록 하는 접근방식을 선택했다.

현재 국내외로 가장 크게 이슈화되고 있는 부분은 SDN 표준 참고구조를 정의하고 이를 기반으로 표준화해야 하는 기술 및 인터페이스를 정의하는 작업으로, 표준화기구인 ITU-T, ONF, IETF 등에서 현재 초기 논의 중에 있다. 본 고에서는 최근 ETRI가 국내 TTA 에 제안하여 표준화가 완료된 SDN 표준 참조구조를 기반으로 개방형 인터페이스, SDN 디바이스 추상화, SDN 프로그래밍 언어 요소 등을 기술한다. 〈그림 2〉는 이를 도식화하여 개념화한 구조를 나타낸다[1].

본 장에서는 SDN네트워크를 위한 개방형 인터페이스를 세가지로 분류하고 각각의 요구사항을 분석한다.

SDN을 위해 필요한 개방형 인터페이스로는 크게 사우스바운드(Southbound), 노스바운드 (Northbound), 이스트웨스트바운드 (East-westbound) 인터페이스로 분류되며, 현재 논의되고 있는 각 인터페이스의 표준 후보와 각각의 요구사항 등은 다음과 같다.

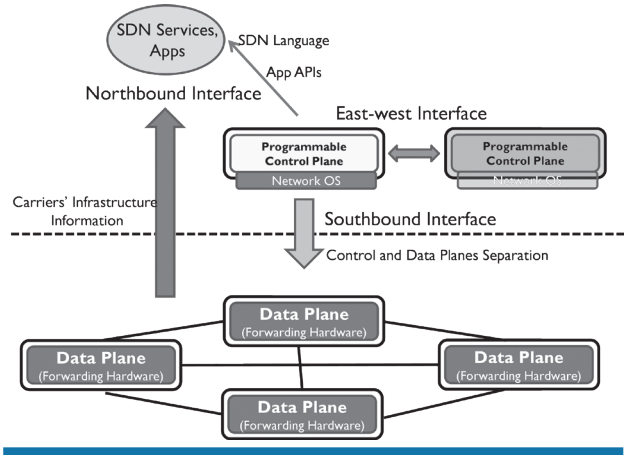


그림 2.SDN 표준 참조 구조

1. 사우스바운드 인터페이스

- 표준기술 후보 : 오픈플로우, I2RS, MPLS-TE, L2/L3 VPN 등
- 기술적 요구사항 :

① 프로그램화와 빠른 재설정 (programmability and quick re-configurability): 사우스바운드 인터페이스는 제어 기능에서의 유연성을 지원해야 하며, 네트워크 상의 새로운 제어 기법들을 쉽게 적용 가능해야 한다. 사우스바운드 인터페이스는 제어 부분의 프로그램화와 빠른 재설정을 위해 사용된다.

② 자원의 공유 (sharing of resources): 사우스바운드 인터페이스는 물리 자원의 특성에 대한 가상화를 지원해야 하며, 컨트롤러는 해당 인터페이스를 통해 자원에 접근하고 다른 사용자와 해당 자원을 공유한다.

③ 트래픽 분리 (traffic isolation): 사우스바운드 인터페이스는 다중의 가상 네트워크 간의 보안, 성능 분리 기능을 포함하는 안전한 트래픽 분리 기능을 제공해야 한다.

④ 네트워크 추상화 (network abstraction): 사우스바운드 인터페이스는 물리 자원의 특성에 대한 복잡도를 줄이기 위해 물리 자원의 특성을 추상화하고, 해당 자원의 제어를 위한 단순화된 하이레벨 인터페이스를 제공해야 한다.

2. 노스바운드 인터페이스

- 표준기술 후보 : ALTO, SNMP, NetConf 등
- 기술적 요구사항 :

① 라우팅 관련 인터페이스: 노스바운드 인터페이스는 네트워크 인프라로부터 컨트롤러 혹은 SDN 사용자에게 토폴로지 발견, 트래픽 엔지니어링, 지연, 지터, QoS 등 된 라우팅 관련한 정보들을 제공해야 한다.

② 관리 기능 관련 인터페이스: 노스바운드 인터페이스는 네트워크 인프라로부터 컨트롤러 혹은 SDN 사용자에게 자원, 에너지, 모니터링, 관리, 계정 관련한 정보들을 제공해야 한다.

③ 정책 관련 인터페이스: 노스바운드 인터페이스는 네트워크 인프라로부터 컨트롤로 혹은 SDN 사용자에게 접근제어, 보안등 정책 관련한 정보들을 제공해야 한다.

3. 이스트웨스트바운드 인터페이스

- 표준기술 후보 : SDNi, BGP 등
- 기술적 요구사항 : 이스트웨스트바운드 인터페이스는 다중 컨트롤러 간의 인터페이스를 정의하며, 서로다른 도메인간 혹은 도메인내의 컨트롤로 동기화 등에 대한 요구사항들을 반영하여 정의되어야 한다. 확장성 (Scalability), 호환성 (Interoperability) 등의 이슈들을 포함한다.

현재 SDN의 구조는 각 산업체 마다 이해 관계가 달라 위와 같은 표준 참조구조 이상의 공통된 표준 구조를 정의하기는 쉽지 않을것으로 예상된다. 따라서 본 고에서 정의한 바와 같이 기본 참조 구조만 정의하고 보다 상세한 프로토콜 구조 혹은 플랫폼 등은 상용화 환경 및 전략에 따라 비즈니스 맞춤형을 지원하는 확장성 있는 구조로 발전할 것으로 예상된다. 대표적인 SDN의 실현 모델로는 〈그림 3〉에서 도식한 바와같이 a) 오픈플로우-기반 SDN, b) 컴파일로-기반 SDN, c) 브로커-기반 SDN, d) 하이브리드/E2E SDN 등으로 분류 가능하다.

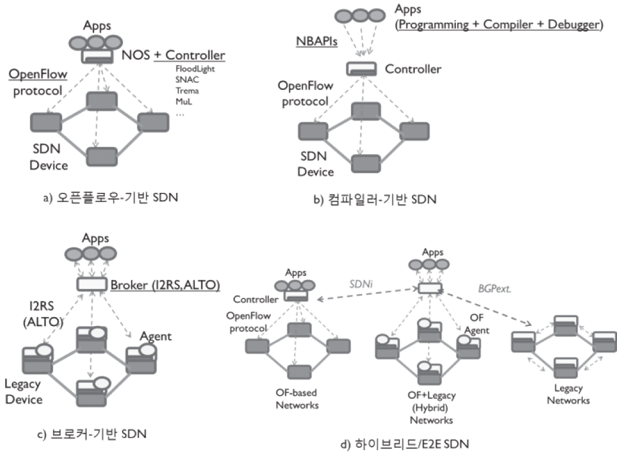


그림 3. 다양한 SDN 실현모델

Ⅲ. SDN 스위치 추상화 기술

본 장에서는 SDN 스위치 추상화의 필요성과 현재 표준관점의 논의된 내용에 대해 살펴본다. 하드웨어 기술은 무어의 법칙에 따라 급격하게 발달 하였지만 네트워크 제어 부분은 매우 느리게 발전하였다. 네트워크 설정, 관리하는 부분은 네트워크 서비스의 한 부분으로 들어가기 보다는 운영에 포함되었다. 네트워크 기술이 발달함에 따라 서비스 속도나 운영에 대한 요구사항은 점점 증가하여 네트워크 설정은 매우 복잡해지고 디버깅은 매우 어렵고 힘든 작업이 되었다. 이러한 현상이 일부 네트워크 환경에서만 나타나는 것이 아니라, 대규모 네트워크 환경이나 데이터 센터, 서비스 제공업자들에게도 나타나고 있다. 이러한 문제의 해결책으로 제시 되고 있는 것 중에 하나가 SDN 기술이다.

오픈플로우에서는 사용자의 요구사항을 프로토콜, 구조등에 반영하고 있다. 그러나 다양한 사용자의 요구사항을 충족시켜 줄 만큼 충분하지는 않다. 특히 ASICs, NPUs 로 구현되는 스위치 하드웨어 포워딩부분은 테이블을 기록하는 과정에서 정보가 손실 되거나 state 정보가 넘쳐나거나 하는 문제가 발생할 수 있다. 특히 실시간으로 기능을 매핑하는 과정은 운영하는 과정중에 시간 지연이나 오류와 같은 문제가 발생할 수 있다. 이러한 문제를 해결하기 위해 어떠한 연구가 진행되고 있는지 살펴본다.

1. 오픈플로우 테이블 타입 분류 및 변수

테이블 타입을 정의하고 분류하는 작업을 통해 얻을 수 있는 이익은 현재 도메인 상에서 동작하는 네트워크를 지원하고 추후에 확장하는데 유용 할 수 있다. 또한 이렇게 분명하게 분류를 하면 TCAM 으로 구현하는데 효과적이다.

〈그림 4〉와 같이 EM(Exact Match), LPM(Longest Prefix Match), MM(Masked Match), RM(Rang Match), REM(Regular Expression Match) 테이블 타입을 지원한다. 각각이 지원하는 하위 테이블 타입을 확인해 보면 현재의 네트워크뿐만이 아니라 추상화 가능한 형태도 지원하여 확장에 유용하게 되어있다. 그리고 테이블 변수에는 테이블 사이즈 변수와 키 사이즈 변수, 대역폭, 성능과 관련 된 테이블 검색 지연시간이 포함된다.

Table Type	Sub Table Type	Use Case
Exact Match (EM)	Indexed Exact Match (Direct Lookup)	VLAN
	Fixed Length Data Associated	MAC
	Fixed Length Set Membership Query	
	Variable Length Data Associated Variable Length Set Membership Query	Pattern matching
Longest Prefix Match (LPM)	Bit based	IP
	Word based	NDN/CCN
Masked Match (MM)	Fixed Length	ACL
Rang Match (RM)	Fixed Length	TCP port
Regular Expression Match (REM)	Variable Length	DPI/ L7 Content Based Routing

그림 4.오픈플로우 테이블 타입 분류

2. 오픈플로우 테이블 패턴

테이블 패턴 내부에서 동일 레벨 간의 테이블 간의 연결은 링크드 리스트로 연결되며 서로간의 의존성은 없다. 스위치상에

서는 L2, L3 테이블이 처음 레벨로 표현 가능하다. 각각의 테이블 모두 동일 구조는 갖는다. 테이블 간의 연결은 2가지가 필요하다. 우선 동일 레벨 간의 연결이 필요하고, 다음 레벨을 연결하는 링크가 필요하다. 그리고 병렬로 여러 테이블과의 매치가 발 생하므로 우선 순위 필드(table priority field)가 테이블 구조에 포함되어 있다고 가정하였다. 테이블 패턴을 사용한 예는 〈그림 5〉와 같고 순서에 대한 설명은 다음과 같다.

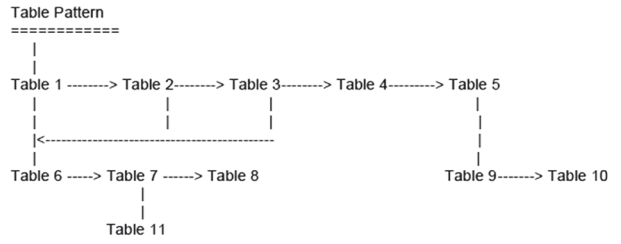


그림 5. 오픈플로우 테이블 패턴 예제

① 파이프 라인으로 동일 레벨의 테이블이 1-5까지 연결 되어 있다. 서로 간의 의존 관계는 없다.

② 테이블 1, 2, 3 에서 매치가 발생하면 다음 레벨의 테이블 6, 7, 8을 확인하여 매치 여부를 검사한다. 패러럴 레벨의 의존 관계는 없다.

③ 테이블 4에서 매치가 발생하면 파이프 라인의 끝 부분이기 여기서 종류된다.

④ 테이블 5에서 매치가 발생하면 다음 단계인 테이블 9, 10을 확인한다.

3. 오픈플로우 플로우 테이블 Metadata

64bit로 non-packet header 데이터를 테이블 간에 주고 받기 위해 설계되어 오픈플로우 1.1에서 추가가 되었다. 오픈플로우 1.2에서는 〈그림 6〉과 같이 extensible matching 으로 변경되었다. 테이블 특징이나 action 에 대한 정의한 특정한 구조에 대해서는 재사용이 가능하다.

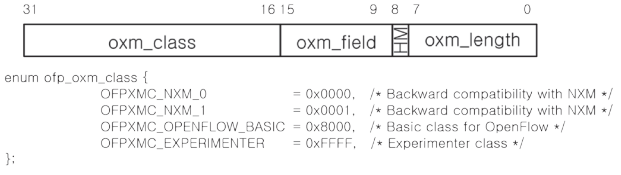


그림 6. OXM(OpenFlow Extensible Match) 필드

4. 오픈플로우 Group Table

오픈플로우는 멀티플 엔트리를 갖는 그룹 테이블을 갖고 있고 각각의 엔트리는 타입, 한 개 혹은 복수개의 action buckets, 그룹 id,카운터 값을 갖는다. 여기서 말하는 action buckets 은 action sets을 가지고 있는 것을 지칭하며 그룹의 action 또한 파이프 라인의 마지막 단계에서 수행을 한다. 그룹의 타입에는 필수 요소로 indirect 와 all 이 있고 옵션으로 select 와 fast failover를 갖는다. Indirect 는 하나의 버킷에 하나의 action 을 담는 형태로 멀티 테이블의 플로우 엔트리와 유사하다. All 은 multicat와 같이 동일한 패킷이 여러 번 복사가 되어야 하는 곳에 사용된다. Select는 multipath와 같이 여러 개 중에서 임의의 개수를 선택하여야 할 때 사용된다.

그룹 테이블을 사용하므로써 얻을 수 있는 이득은 중복되는 action list를 제거 할 수 있고, 컨트롤러와의 연결 횟수를 줄일 수 있게 된다. 그리고 특정 그룹에 장애가 발생하는 경우 빠르게 대처가 가능해 진다.

5. Freescale 오픈플로우 데이터패스

〈그림 7〉은 데이터패스 테이블을 가변으로 정의하여 사용한 예제를 보여준다. 〈그림 6〉과 같이 컨트롤러에 방화벽/NAT 모듈, 라우팅 모듈, ARP 모듈,QoS 모듈이 올라가 있다. 방화벽/NAT 모듈은 세션 정보를 관리하고 테이블에 없는 패킷이 들어오는 경우 방화벽/NAT 모듈로 보낸다. 그리고 방화벽/NAT를 적용하여 플로우를 생성시킨다. 라우팅 모듈은 PBR(Policy Based Routing)을 적용하여 PBR ACL 리스트를 업데이트 하고, PBR ACL 리스트에 일치하면 라우팅 테이블로 전달하여 패킷을 내보낸다. ARP 모듈은 ARP 캐쉬 테이블을 관리하고 테이블에 없는 경우 컨트롤러에 보내고 매치가 되면 패킷을 전달한다. QoS 모듈은 나가는 패킷을 분류하여 Queue에 나누어 저장하고 QoS 정책에 의해 패킷을 내보낸다.

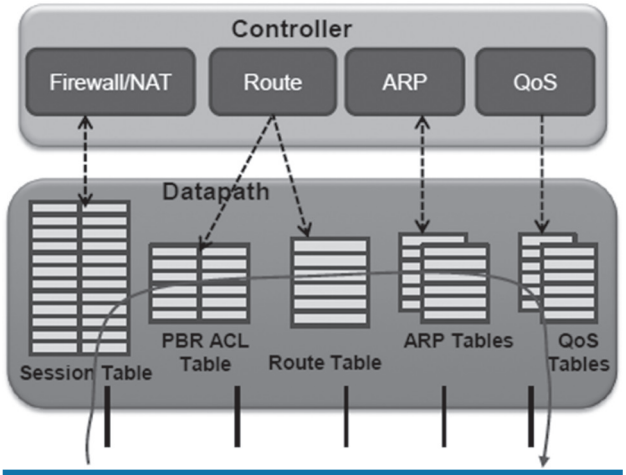


그림 7. Freescale 오픈플로우 데이터패스

Ⅳ. SDN 컨트롤러 언어

현재 SDN 컨트롤러를 통해 실행되는 응용은 특정 컨트롤러에서 지원하는 인터페이스와 언어로 작성하는 것이 대부분이다. 이처럼 컨트롤러의 확장 모듈 형태로 동작할 수도 있지만, SDN을 통해 구현하고자 하는 응용이 복잡하고, 이에 대한 사용자의 의존도가 높아질수록, SDN 응용 제작에 필요한 전용 언어의 필요성이 높아질 것이다. 이 장에서는 앞서 소개한 SDN 구조에서 응용을 쉽고 간결하게 작성하게 해주는 전용 언어를 살펴본다 [2].

1. FML (Flow-based Management Language)

FML은 엔터프라이즈 네트워크에 적용할 정책 (Policy)을 쉽게 작성하기 위해 개발된 논리 기반 언어로서, 기존 라우터와 스위치에서 제공하는 인터페이스보다 쉬우면서도 견고하게 정책을 구현할 수 있다. SDN/OpenFlow를 대상으로 거의 최초로 등장한 언어다 [3].

논리에 기반한 정형적인 언어의 형태를 띠고 있지만, if-then 형태의 문장을 단순히 나열하는 문법 구조를 제시함으로써, 논리에 익숙하지 않은 일반 관리자도 쉽게 정의할 수 있다는 점이 특징이다. 반면, 쉬운 문법으로 인해 언어 처리 과정에서 여러 개의 규칙이 서로 상충되거나 중복되는 현상이 쉽게 발생할 수 있는데, 이에 대해 별도의 알고리즘으로 처리하고 있다.

FML은 C++와 Python 코드로 10,000라인 정도의 NOX 기반 모듈로 구현됐고, 현재 SNAC 컨트롤러의 Policy Manager에서 사용하고 있다 [8].

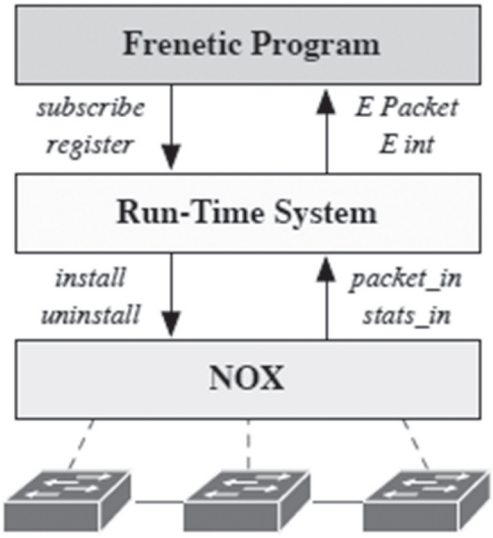


그림 8. Frenetic 구조[3]

2. Frenetic

프린스턴 대학 연구팀에서는 NOX 컨트롤러의 응용을 보다 쉽고 효율적으로 작성할 수 있도록, 모듈화와 단일 티어 추상화, race condition 처리 등의 기능을 제공하는 Frenetic 이란 언어를 개발했다 [4].

Frenetic은 DATALOG와 같은 선언적인 DB 쿼리 언어의 영향을 받았으며, 2011년 제1회 ONS를 통해 SDN 커뮤니티에 알려졌다. 단순히 코드 작성의 효율성을 높이는데 그치지 않고, 정형 의미론과 컴파일 메커니즘을 보완하여 NCore라는 언어도 제안된 바 있으며, 향후 Nettle 런타임과 통합될 것으로 예상된다 [5].

3. Nettle

예일 대학에서 FRP(Functional Reactive Programming) 방식의 Haskell 기반 언어로서, 기존의 라우터 제어를 위한 기능을 SDN/OpenFlow에 맞게 발전시킨 언어 및 런타임이다 [6]. Nettle에서는 OpenFlow 컨트롤러와 스위치가 서로 주고 받는 이벤트와 메시지를 스트림으로 추상화하고, 이산적인 이벤트 발생뿐만 아니라, 연속적인 시간에 따른 값의 변화도 언어에서 다룰 수 있다는 점이 특징이다.

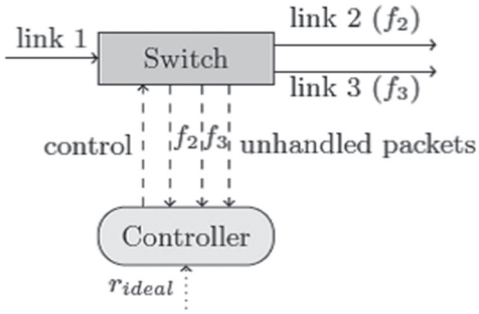


그림 9. 연속 개념 적용 예[5]

Nettle에도 참여한 매릴랜드 대학의 Nick Feamster 교수 연구팀에서는 Procera라는 또 다른 언어도 제안한 바 있으며, Frenetic과 Nettle과 유사하지만, 보다 고수준으로 정책을 정의할 수 있으며, Lithium이라는 컨트롤러를 기반으로 동작하고, 이벤트 히스토리를 다룰 수 있다는 점이 특징이다.

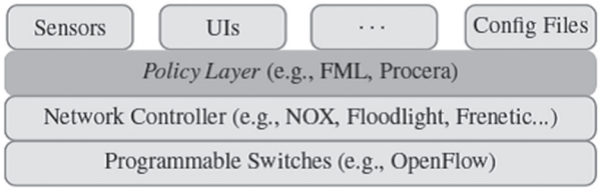


그림 10. Procera 구조[7]

4. REST 기반 API

앞서 소개한 사례처럼, 전용 언어를 정의하고, 이를 구현하는 런타임을 컨트롤러에서 지원함으로써, SDN 응용 제작의 신뢰성과 효율성을 높일 수도 있지만, 또 다른 접근 방법으로 기존 컨트롤러에서 제공하는 주요 기능을 REST 방식의 Open API 형태로 제공함으로써, 사용자가 원하는 프로그래밍 언어로 응용을 개발하도록 지원할 수도 있다.

NOX나 Trema를 비롯한 대부분 컨트롤러에서 REST 인터페이스만 추가하면 쉽게 지원할 수도 있지만, 본격적인 REST API로는 Big Switch Networks의 Floodlight가 대표적이다[3].

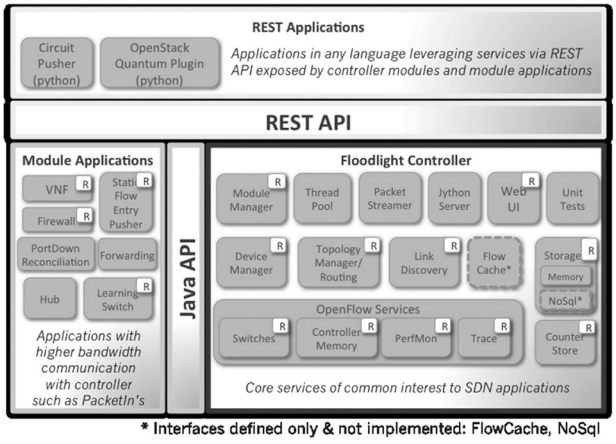


그림 11. Floodlight의 REST API[3]

V. 결론

본 고에서는 다양한 SDN 실현 모델을 논의하고, 이를 수렴하는 표준 기반의 SDN 참조 구조를 살펴보았다. 또한 이를 기반으로 표준화가 필요한 개방형 인터페이스 요구사항, 오픈플로우 스위치 추상화 기술, 컨트롤러 프로그래밍 언어 기술에 대해 논의하였다.

현재 SDN은 단순화된 표준 기반의 기술 및 생태계를 포괄하고 있다. 현재 오픈플로우는 SDN 실현을 위한 대표적인 기술이지만, 오픈플로우 프로토콜만 가지고는 다양한 SDN의 요구사항을 만족하는 비즈니스 솔루션을 실현하기에는 어려울 것으로 보인다. 따라서 SDN의 각 플레이어들은 각자의 다양한 전략적 모델을 가지고 있으며, 기존 자사가 보유한 기술을 기반으로 SDN 솔루션을 제안하고 있는 상황이다. 대표적인 사례로는 오픈플로우-기반SDN, I2RS-기반 SDN, Carrier-SDN, E2E SDN, 하이브리드SDN, Cell/모바일SDN 등이 있다.

끝으로 국내의 표준화 추진 전략을 살펴보면, 우리나라의 경우 ITU-T, IETF, ONF 등의 각각의 표준화기구에 다음과 같은 차별화된 전략을 가지고 관련 표준 개발을 추진하는 것이 바람직할것으로 보인다.

- ITU-T : 통신사업자 중심의 표준화, 유무선 엑세스망에서의 SDN도입에 초점
- IETF : 제조업체 중심의 표준화, 데이터 전송 하드웨어 보다는 제어 소프트웨어 및 개방형 인터페이스 표준화에 초점을 맞추는 것이 바람직함 (다양한 기존의 RFC표준들의 재활용에 초점)
- ONF : 오픈플로우 기반, 서비스사업자 중심, 데이터센터/클라우드 관련 표준 기술에 초점

Acknowledgement

※본 연구는 방송통신위원회의 지원을 받는 방송통신표준기술력향상사업의 연구 결과로 수행되었음

참 고 문 헌

[1] 신명기, 소프트웨어 정의 네트워크(SDN)를 위한 개방형 인터페이스 요구 사항, TTA 단체표준 TTAK.KO-01.0182, 2012.

[2] 남기혁, 신명기, 김형준, SDN/OpenFlow 전용언어 및 신뢰성 검증 방법 연구동향, 한국통신학회지, 2012.11.

[3] Floodlight, http://www.openflowhub.org/display/floodlightcontroller/Floodlight+REST+API

[4] Nate Foster, Rob Harrison, Michael J. Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, David Walker. Frenetic: a network programming language. In ICFP, September 2011

[5] Frenetic-language, http://frenetic-lang.org

[6] Andreas Voellmy, Paul Hudak. Nettle: Functional reactive programming of OpenFlow networks. In PADL, January 2011

[7] Andreas Voellmy, Hyojoon Kim and Nick Feamster, Procera: a language for high-level reactive network control. In SIGCOMM HotSDN, August 2012

[8] SNAC http://www.openflowhub.org/display/Snac/SNAC+Home

[9] David Meyer, Curt Beckmann,Forwarding Abstractions Working Group Charter,2012.07
[10] Haoyu Song, Table Type Classification and Parameters, ONF FAWG meeting,2013.01
[11] Abhijit Kumbhare, Table Pattern Proposal Extending, ONF FAWG meeting,2012.08
[12] Colin Dixon, Improving OpenFlow metadata,ONF FAWG meeting,2013.01
[13] OpenFlow Switch Specification, 2012.09.
[14] CurtBeckmann, Freescale_IP_forwarding_Example,ONF FAWG meeting,2012.09

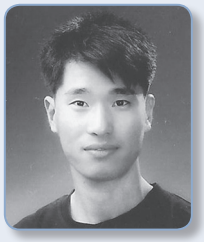
약 력

신 명 기

2003년 충남대학교 컴퓨터공학과 공학박사
1994년~현재 한국전자통신연구원 책임연구원
2008년~현재 한국기술연합대학원대학교(UST) 겸임교수
2004년~2005년 미국NIST (National Institute of Standards and Technology) 초빙연구원
관심분야: 미래인터넷, 네트워크 가상화, SDN, OpenFlow

남 기 혁

2002년 고려대학교 컴퓨터학과 이학사
2004년 고려대학교 컴퓨터학과 공학석사
2004년~현재 한국전자통신연구원 선임연구원
관심분야: SDN, OpenFlow, 가상화, 정형 기법



최 윤 철

2007년 충남대학교 전자전파정보통신공학과 학사
2010년 충남대학교 전자전파정보통신공학과 석사
2012년~현재 한국전자통신연구원 연구원
관심분야: SDN, OpenFlow

김 형 준

2007년 충남대학교 공학박사
2007년~2008년 University of Virginia 초빙연구원
1988년~현재 한국전자통신연구원 표준연구센터 팀장/책임연구원
2008년~현재 한국기술연합대학원대학교(UST) 겸임교수
관심분야: 미래인터넷, RFID/USN, M2M