# 고확장성과 고가용성을 지원하는 Transport SDN 기술 동향

윤빈영, 김태홍, 김영화, 양선희
한국전자통신연구원

## 요 약

본고에서는 Transport SDN의 고확장성과 네트워크 장애를 신속하게 복구하기 위한 고가용성을 제공하는 SDN 컨트롤러 구조와 데이터 플레인 장애 복구 기술 동향을 소개한다. SDN 컨트롤러 기술에 대해서는 최근 많은 관심을 받고 있는 오픈소스 프레임워크 기반의 SDN 컨트롤러를 중심으로 확장성 및 고가용성 기술 동향을 살펴본다. 그리고 데이터 플레인에서 노드 및 경로 장애 발생 시 신속하게 경로를 복구하기 위한 보호기술과 복구기술 동향을 소개한다.[1]

## Ⅰ. 서 론

인프라의 구조적 유연성과 개방성을 제공할 수 있는 SDN은 중앙집중적인 제어 계층과 개방형 API에 기반한 프로그래머빌리티를 지원함으로써 차세대 네트워킹 기술로써 각광받고 있다. 2011년 SDN 기술에 대한 표준 제정과 연구 촉진을 목적으로 비영리단체인 오픈 네트워킹 파운데이션 (Open Networking Foundation, ONF)이 설립되었으며, 미국, 일본, 한국 등을 포함한 전세계 산업체, 학계, 연구 기관에서 다양한 종류의 SDN 컨트롤러 기술이 개발되었다. 그 동안에 개발된 주요 SDN 컨트로롤러에는 Floodlight [1], NOX/POX [2], Beacon [3], Ryu [4], IRIS [5] 등이 포함된다.

최근 SDN 컨트롤러에 집중적으로 유입되는 트래픽으로 인하여 발생되는 성능저하와 안정성 문제를 해결하기 위해서 분산 SDN 컨트롤러 구조가 연구되고 있다. 즉, 논리적으로는 중앙집중 제어방식의 컨트롤러라는 패러다임을 유지하면서도 물리적으로는 복 수개의 컨트롤러 서버에 분산하여 배치한다. 따라서, 일부 컨트롤러 서버에 장애가 발생하더라도 다른 컨트롤러 서버로 신속하게 대치하거나 트래픽을 분배하여 컨트롤러 가용성과 고확장성을 제공한다. 분산 컨트롤러의 필요성과 가능성에 대한 연구는 2010년 HyperFlow [6]에서 처음 제안되었으며, 이후 Elasticon [7], ONOS [8] 등에서 분산 컨트롤러 구조에 대한 제안과 성능 검증이 이루어졌다. 또한, 분산 컨트롤러 환경에서 컨트롤러간 상태 정보 불일치로 발생되는 네트워크 동작 오류 및 성능 저하 문제점을 분석한 연구 [9]가 발표되었다. 이러한 연구 결과를 반영한 분산 컨트롤러 구조는 로드 밸런싱, 고장 감내 기능, 컨트롤러 간 분산 데이터의 일관성 (Consistency) 등이 고려되어 한다.

컨트롤러 측면의 고확장성 및 가용성 기술과 함께 Transport SDN 네트워크(데이터 플레인) 장애로 인한 서비스 중단을 최소화하기 위해서 전달망 보호기술(protection)과 전달망 복구기술(restoration)과 같은 네트워크 생존성 (network survivability) 기술이 연구되고 있다. 보호기술과 복구기술의 구분은 워킹 경로에 장애가 발생하기 이전에 백업 경로가 사전에 준비되어 있는지 여부에 의하여 구분할 수 있다[10][11]. 즉, 장애를 대비하여 백업 경로 설정이 완료된 경우를 보호기술로 정의하고 그렇지 않은 경우를 복구기술로 정의한다. ITU-T와 IETF에서는 APS 프로토콜을 이용하여 Peer-to-peer 노드 간 보호 절체를 수행하는 보호기술 표준 문서를 권고하고 있다[12][13][14]. 최근에는 APS (Automatic Protection Switching) 프로토콜을 사용하는 대신 오픈플로우 (OpenFlow) 프로토콜을 이용하여 SDN 컨트롤러 개입 없이 자동으로 보호 절체되는 SDN 보호기술들이 연구되고 있다[15]-[18]. 또한, SDN 컨트롤러에 내장된 토폴로지 정보, 네트워크 자원 정보, PCE (Path Computation Element)를 사용하여 장애 발생 시, 신속하게 백업 경로를 계산하고 설정하여 장애를 복구하는 SDN 보호기술이 연구되었다[19][20].

본 고에서는 최근 많은 관심과 집중을 받고 있는 오픈소스 프레임워크인 OpenDaylight과 ONOS를 중심으로 분산 컨트롤러 구조와 컨트롤러 간 상태정보 공유 방식 관점에서 SDN 컨트롤러의 고확장성과 가용성 지원 기술 동향에 대하여 살펴본다. 또한, 데이터 평면의 고가용성을 위해서 오픈플로우의 플

1 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (B0101-15-233, 스마트 네트워킹 핵심 기술 개발)

로우 테이블을 이용한 세그먼트 보호기술과 그룹 테이블을 이용한 보호기술을 소개한다. 그리고 광통신 패러미터를 가변적으로 선택하여 광 네트워크 자원을 효율적으로 상용할 수 있는 EON (Elastic Optical Network)에서 동작하는 SDN 복구기술 방법과 GMPLS 복구기술 방법의 특징을 살펴보고 성능을 비교, 분석한다.

## Ⅱ. 컨트롤러 중심의 확장성 및 가용성 지원 동향

SDN 기술 개발 초기에는 오픈플로우 프로토콜의 표준화 단체인 오픈 네트워킹 파운데이션의 주도하에 Floodlight, NOX/POX, Ryu 등 다양한 종류의 SDN컨트롤러가 개발되었으나, 최근에는 오픈소스 소프트웨어 플랫폼인 ODL (OpenDaylight) 및 ONOS (Open Network Operating System)를 중심으로 시장이 재편되고 있다[21][22]. 두 오픈소스 플랫폼 모두 분산 컨트롤러 기반의 확장성 및 가용성 지원을 목표로 하고 있으며, 본 장에서는 각 플랫폼의 특징 및 개발방향에 대하여 살펴보도록 한다.

### 1. ODL

ODL은 2013년 4월 리눅스 파운데이션 (Linux Foundation)에 의해 공식 출범한 오픈소스 프레임워크 프로젝트이다. 시스코를 비롯하여 IBM, 브로케이드, 빅스위치네트웍스, 주니퍼네트웍스, VMWARE, 마이크로소프트 등 글로벌 IT 업체 대부분이 참여한 범 개방형 네트워크 연합체로써, 오픈소스 기반의 표준 SDN 프레임워크 개발 및 어플리케이션, 툴, 서비스 등을 포함하는 SDN 생태계 전반을 구축하는 것을 목표로 한다. ODL의 첫 번째 버전인 Hydrogen은 2014년 2월, 두 번째 버전인 Helium은 2015년 3월, 세 번째 버전인 Lithium은 2015년 6월에 발표되었다.

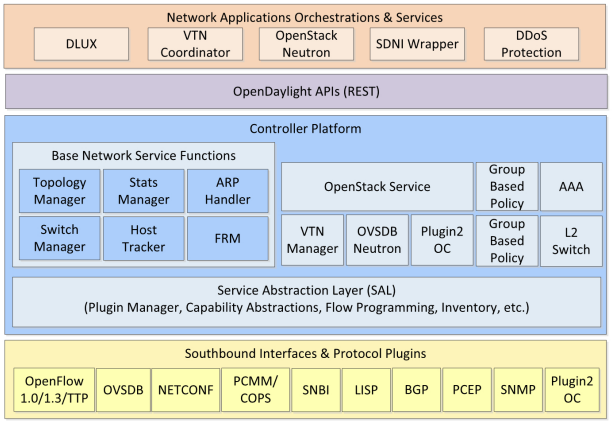


그림 1. OpenDaylight (ODL) 구조

ODL은 동적으로 장착 가능한 모듈 형태의 컨트롤러 플랫폼으로, 자바 가상 머신 (JVM) 상에서 동작하므로 하드웨어 및 운영체제와 독립적으로 동작한다. ODL은 응용 계층을 위해 OSGI 프레임워크 및 노스바운드 (Northbound) API를 지원하며, 응용 계층에서는 컨트롤러를 이용하여 네트워크 정보를 수집하고 네트워크 전반에 대한 관리/제어용 비즈니스 로직 또는 알고리즘을 적용할 수 있다.

ODL의 컨트롤러 플랫폼은 필요에 따라 동적으로 장착 가능하도록 각 네트워킹 기능들을 모듈 형태로 포함하고 있으며, 기본적으로 제공되는 네트워크 서비스는 토폴로지 관리 기능, 스위치 관리 기능, 호스트 트래커, 상태 관리자 등이다. 응용 요구사항에 따라 클라우드와의 연동을 위한 오픈스택 (OpenStack) 서비스, 멀티테넌트 가상 네트워크를 위한 VTN (Virtual Tenant Networking) 서비스, 식별자/위치자 분리 프로토콜 (Locator/ID Separation Protocol, LISP) 서비스 등을 자유로이 확장할 수 있다.

사우스바운드 (Southbound) 인터페이스에서는 오픈플로우 1.0/1.3, BGP-LS 등 다중 프로토콜 등을 지원하며, 오픈플로우를 지원하는 장비와 지원하지 않는 장비의 제어기능을 모두 지원함으로써 서로 다른 장비간의 통합 제어가 가능하다. 또한, 각각의 네트워크 프로토콜은 서비스 추상화 계층 (Service Abstraction Layer, SAL)과 동적으로 연결되며, 컨트롤러와 네트워크 장치 사이의 통신 방식과 독립적으로 서비스 추상화 계층에서 요청 서비스에 대한 처리방법을 결정할 수 있다.

2015년 3월에 발표된 Helium 버전에서는 컨트롤러의 고가용성과 클러스터링, 보안 중심의 개선이 이루어졌으며, 패킷 케이블 멀티미디어 (Packet Cable Multimedia), 서비스 기능 체이닝 (Service Function Chaining) 등의 프로토콜이 추가되었다 [23].

고가용성이란 중단 없는 서비스 제공을 의미하며, 컨트롤러 클러스터를 이용하여 운영 중인 서버의 장애 발생 또는 서버 업그레이드 시에 다른 서버가 작업을 대신하도록 함으로써 끊김 없는 서비스의 제공을 할 수 있다. ODL에서는 고가용성의 지원을 위하여 Akka 라이브러리 [24] 기반의 클러스터링 구현 및 Raft 알고리즘 [25]을 이용한 분산처리 작업을 수행한다.

Akka 라이브러리는 대규모 분산 시스템에서의 확장성과 고장 감내 기능을 제공하는 것을 목표로, 타입세이프 (Typesafe)라는 회사에서 제공하는 액터 모델 (Actor model) 기반의 플랫폼으로써 자바와 스칼라 개발 환경에서의 라이브러리를 제공한다. 액터 모델이란 기존의 분산/병렬 시스템을 위하여 개발된 모델로써, 액터는 자신만의 메시지큐를 가지며 다른 액터로 메

시지를 주고 받고 해당 메시지에 대응하는 동작을 수행할 수 있는 객체이다. 액터 간의 통신은 고속처리를 위하여 비동기 방식의 동시처리 (Concurrent processing) 및 병렬처리 (Parallel processing) 방식을 지원하며, 기존 방식과 달리 데드락이나 기아상태 (Starvation) 등 분산 환경에서 발생하는 문제에도 자유로울 수 있도록 설계되었다. 또한, 하나의 액터 객체가 처리할 수 있는 데이터의 단위가 클 경우에는 재귀적인 방식으로 작은 단위의 데이터를 갖는 액터로 분할함으로써 데이터 처리 성능을 향상시킬 수 있다.

ODL의 클러스터링 모델에서는 기존의 데이터 저장소를 샤드 (shard) 형태로 분할하고, 클러스터 내에 존재하는 서버들이 분산 형태로 샤드를 관리할 수 있도록 설계하였다. ODL의 Helium 버전에서 기본으로 제공하는 샤드는 크게 디폴트 (default), 토폴로지 (topology), 인벤토리 (inventory), 토스터 (toaster)로 구성되며, 설정을 통해 각각의 샤드를 복제하는 서버 리스트를 지정할 수 있다. 이 때, 샤드는 Akka 라이브러리의 액터 객체 형태로 구현되며, 액터 간 메시지 교환을 통하여 분산 처리 및 데이터의 복제 과정이 이루어진다[23].

ODL에서는 서버 이중화 (Active-standby) 모델을 통한 컨트롤러의 고가용성을 지원한다. 즉, 운영 서버의 상태 정보를 대기 서버들에 복제한 후 운영 서버의 장애 등이 발견되었을 때 대기 서버 중 하나를 운영 서버로 전환함으로써, 지속적인 서비스를 제공하게 된다. 이 때, 클러스터를 구성하는 서버 중 운용 서버 역할을 하는 리더의 선정은 미국 스탠포드대에서 개발한 Raft 알고리즘을 이용한다. Raft 알고리즘에서는 분산 방식의 리더 선출 과정을 거치게 되고, 리더는 주기적으로 하트비트 (Heartbeat) 메시지를 전송하여 자신의 상태 정보를 알린다. ODL에서는 리더로 선정된 서버가 토폴로지 변경 등 자신의 샤드 정보가 변경될 때마다 샤드 트랜잭션을 통하여 대기 서버들의 샤드 정보 및 내부 메모리 공간인 저널 (journal)에 복제할 수 있도록 한다. 만약, 리더로 선정된 서버에 장애가 발생하면, Raft 알고리즘을 통하여 새로운 리더를 선정할 수 있으며, 리더로 선정된 서버는 저널에 저장되어 있는 정보를 복구함으로써 기존의 서버와 동일한 상태정보를 바탕으로 지속적인 서비스를 제공할 수 있게 된다[26][27].

현재의 ODL Helium 버전에서는 클러스터링을 이용한 컨트롤러 가용성만을 제공하고 있으며, 향후 분산 서버를 모두 운용 서버를 활용하는 컨트롤러 확장성의 지원 여부에 관심을 가질 필요가 있다. 또한, 이는 컨트롤러의 확장성과 가용성을 모두 지원하고 있는 ONOS와의 차이점 중 하나라고 할 수 있다.

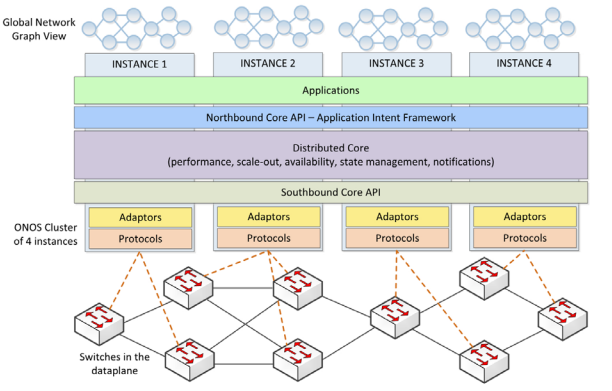


그림 2. ONOS 구조

## 2. ONOS

ONOS는 통신사업자를 위해 개발된 네트워크 운영체제로써, 캐리어 그레이드 (carrier-grade)의 고확장성 및 고가용성을 제공하는 것을 목표로 한다. ONOS의 로고에는 자유 및 개방을 상징하는 새 그림이 포함되어 있으며, 이에 맞추어 릴리즈되는 버전의 이름은 알파벳순을 따르는 새의 이름으로 선정된다. 첫 번째 오픈소스 버전인 Avocet은 2014년 12월, 두 번째 오픈소스 버전인 Blackbird는 2015년 4월에 발표되었으며, 세 번째 버전인 Cardinal은 2015년 6월에 발표되었다.

ON.Lab [28]은 ONOS 개발을 주도하고 있는 비영리단체로써, AT&T, NTT, SK Telecom 등의 통신사업자와 Ciena, Cisco, Ericsson, Huawei, Intel, NEC 등의 네트워크 장비 벤더가 참여하고 있으며, 한국의 ETRI, KISTI 등도 공동연구에 참여하고 있다. 또한, Nick McKeown, Scott Shenker 등 SDN을 발명한 미국 스탠포드대와 UC버클리대 연구자들의 그룹인 오픈 네트워킹 연구 센터 (Open Networking Research Center, ONRC)와 긴밀한 협력관계를 유지하고 있다.

ONOS는 Java 기반의 Floodlight 컨트롤러 [1]를 바탕으로 개발되었으며, 〈그림 2〉에서 보는 바와 같이 논리적으로는 중앙집중형 컨트롤러 기능을 제공하면서도, 물리적으로는 분산 컨트롤러 구조를 이용하여 컨트롤러의 고확장성과 고가용성을 제공한다. 이는 네트워크 규모와 트래픽 측면에서 폭증하고 있는 네트워크 제어 요구사항을 수용하기 위함이며, ONOS에서는 네트워크 환경에 따라 컨트롤러 클러스터의 인스턴스를 추가, 삭제할 수 있다. 또한, 특정 인스턴스에 집중되어 있는 트래픽 부하를 새로운 컨트롤러 인스턴스로 분산시킬 수 있다.

ONOS는 기존의 단일 SDN 컨트롤러와 마찬가지로, 디바이스 및 링크 관리, 네트워크 토폴로지 관리, 종단 간 호스트 사이의 경로 계산 등의 서비스를 제공할 수 있으며, 응용 레벨에서의 네트워크 제어를 위해 매치/액션 (Match/Action) 등의 플로

우 룰 (Flow Rule) 전송뿐만 아니라 정책 기반의 Intent를 이용하여 사용자의 의도에 따라 네트워크를 제어할 수 있다. 즉, 응용 계층에서는 네트워크 대역폭, 링크 타입과 같은 네트워크 자원의 제어와 트래픽 분할, 패킷 헤더 수정, 전송 포트 변경과 같은 명령을 Intent의 형식으로 전달할 수 있으며, 응용레벨 Intent 프레임워크에서는 Intent의 컴파일을 통하여 사용자의 의도에 맞는 플로우룰을 설정할 수 있다.

또한, ONOS의 분산 컨트롤러 구조의 가장 중요한 특징은 개별 컨트롤러 인스턴스가 관리하는 스위치 디바이스와 별개로 각 인스턴스의 응용에서 전체 토폴로지에 대한 정보를 공유할 수 있다는 점이다. 이는 각 컨트롤러 인스턴스가 관리하는 네트워크 일부에 대한 정보만으로는 네트워크 전체 정보를 필요로 하는 경로 계산 등의 서비스를 할 수 없다는 점에서, 분산 컨트롤러 시스템 구축에 필수적인 기능 중 하나라 할 수 있다. ONOS에서는 분산 데이터 저장소의 일관성 수준에 따라 엄격한 수준의 일관성을 요구하는 강한 일관성 (Strong consistency)와 상대적으로 낮은 수준을 요구하는 결과적 일관성 (Eventual consistency)으로 구분하며 스위치 디바이스, 링크, 호스트 등의 정보를 저장하는 분산 저장소는 결과적 일관성 유지를 위하여 가십 프로토콜 (Gossip protocol) 기반의 정보 복제 기법을 이용한다. 또한, 각 컨트롤러 인스턴스와 스위치 간의 관계 정보 (Mastership)는 Hazelcast의 분산 자료 구조를 이용하여 강한 일관성을 유지한다. 또한, 스위치 연결, 포트 연결, 링크 다운과 같은 네트워크 토폴로지 이벤트는 높은 정확도 및 신속한 처리를 위하여 각 컨트롤러 인스턴스의 메모리에 캐시되며, 가십 프로토콜에서는 각 컨트롤러 인스턴스에서 발생된 이벤트 정보를 인스턴스와 스위치 디바이스와의 관계정보와 시퀀스 번호의 조합인 타임스탬프를 이용하여 타 컨트롤러 인스턴스에게 브로드캐스트한다. 토폴로지 이벤트 정보를 수신한 컨트롤러 인스턴스들은 타임스탬프를 바탕으로 이미 수신한 정보인지 여부를 판단한 후, 메모리에 반영함으로써 전체 토폴로지 정보를 유지할 수 있게 된다.

〈그림 3〉은 2015년 4월 ON.Lab에서 발표한 ONOS 분산 컨트롤러의 성능을 분석한 자료 [29]의 일부로써, 컨트롤러 클러스터 내의 컨트롤러 인스턴스 수에 따른 플로우 처리량을 측정한 결과이다. 하나의 컨트롤러 인스턴스가 초당 약 50만 플로우만을 처리할 수 있는 반면, 7개의 컨트롤러 인스턴를 이용할 경우에는 최대 초당 300만 플로우를 처리할 수 있다. 위의 그림은 플로우를 생성하는 인스턴스와 설치하는 인스턴스가 동일할 때의 결과이며, 자료 [29]에서 보다 상세한 분석 결과를 확인할 수 있다. 또한, 이와 같은 결과는 ONOS 분산 컨트롤러의 확장성을 뒷받침할 수 있는 근거로써, 현재 고가용성만을 지원하는 ODL과의 차별화된 특징이라 할 수 있다.

## Ⅲ. 데이터 평면 보호 및 복구기술

본 장에서는 최근 연구되고 있는 SDN 기반의 전달망 보호기술과 복구기술을 살펴본다.

### 1. SDN 보호기술

SDN 보호기술은 SDN 컨트롤러 개입 없이 데이터 평면에서 자동적으로 장애를 복구하는 방법이다. 즉, 장애가 발생되면, 오픈플로우 스위치가 플로우 테이블에 보호 경로를 새롭게 추가 및 변경하기 위해서 SDN 컨트롤러와 통신하지 않고 워킹 경로를 보호경로로 자동으로 절체한다. 보호기술에는 이웃 스위치 사이의 링크 단위로 우회 경로를 제공하는 세그먼트 보호기술과 여러 노드를 통과하는 경로를 절체하는 기술이 연구되고 있다. 본 장에서는 각각의 기술 개발에 대해서 대표적인 보호기술 개발 사례를 소개한다.

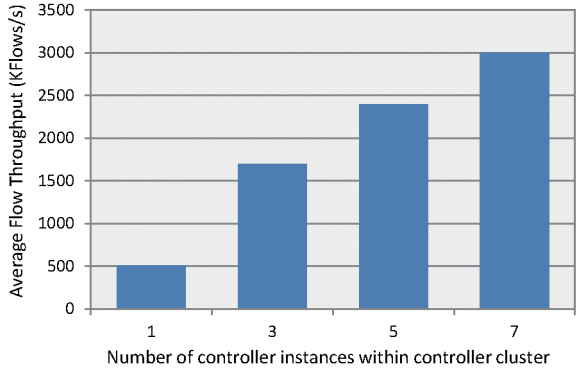


그림 3. ONOS 컨트롤러 크기별 플로우 처리량 (출처: ONOS whitepaper [29])

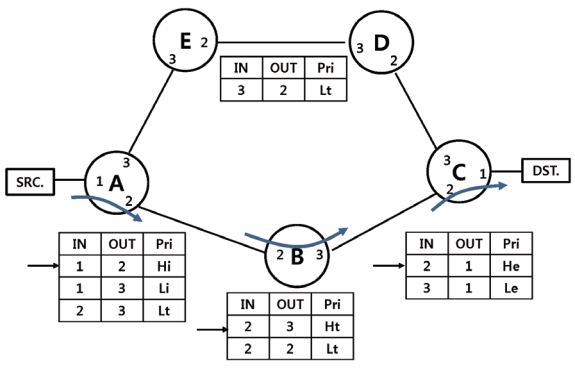


그림 4. 정상 상태의 트래픽 경로

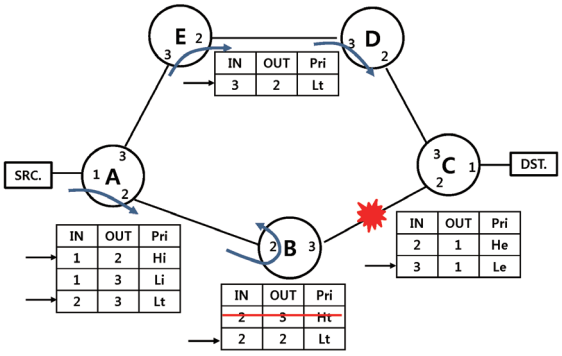


그림 5. 장애 상태의 트래픽 경로

### 가. 오픈플로우 세그먼트 보호기술

오픈플로우의 플로우 테이블을 이용하여 빠르게 장애를 복구시킬 수 있는 세그먼트 보호기술이 연구 [17] 되었으며, 본 절에서는 〈그림 4〉와 같이 5개의 스위치로 구성된 링 네트워크의 예를 통하여 보호 절체 동작과 성능을 분석한다.

보호 절체 경로는 워킹 경로와 백업 경로로 구분되며, 우선 순위에 따라서 워킹 경로는 높은 엔트리(H)를 부여하고 백업 경로는 낮은 엔트리(L)를 부여한다. 또한, 네트워크 상의 스위치 위치에 따라서 ingress(i), egress(e), transit(t)으로 엔트리를 구분한다. 예로써, 스위치 A의 테이블에서 Hi는 ingress에 위치하고 높은 우선 순위 레벨을 갖는 워킹 경로 엔트리임을 표시한다. 반면에, 스위치 C의 테이블에서 Le는 egress에 위치하고 낮은 우선 순위를 갖는 백업 경로임을 표시한다. 만약, 링크 B와 C 사이에 장애가 발생하면, 스위치 B에 있는 워킹 엔트리가 삭제되므로, 〈그림 5〉와 같이 패킷이 백업 경로인 A-B-A-E-D-C로 경유하게 된다. 따라서, ingress 스위치 포워딩 테이블에는 한 개의 워킹 엔트리와 두 개의 백업 엔트리를 갖는다. 워킹 엔트리(1;2;Hi)는 워킹 경로를 따라서 소스 호스트에서 목적지 호스트로 패킷을 전달하는 엔트리를 표시하며, 백업 엔트리(1;3;Li)와 (2,3,Lt)는 스위치 A와 B 사이의 링크와 스위치 B와 C사이의 링크에 각각 장애가 발생되는 경우, 백업 경로로 패킷을 전달하기 위해서 사용되는 엔트리를 표시한다.

각 세그먼트 구간의 장애를 감시하기 위해서 인접 노드 간에 OAM (Operation, Administration, Maintenance) 패킷을 주기적으로 교환한다. 만약, 일정 기간 동안 OAM 패킷이 수신되지 않으면, 장애로 판단하여 보호 절체를 시작한다. 먼저, 보호 절체 동안 사용이 종료된 워킹 경로로의 패킷 전달을 차단하기 위해서 플로우 테이블에서 해당 엔트리를 삭제(혹은 비활성화)한다. 만약, 백업 경로에 일정 시간 동안 패킷이 전송되지 않으면 idle timeout에 의하여 플로우 테이블에서 백업 엔트리가 삭제될 수 있다. 이러한 문제를 해결하기 위해서 컨트롤러와 호스트 사이에 제어 메시지를 사용하지 않고 renew packet을 주기적으로 전달하여 문제를 해결한다. Renew 패킷은 네트워크 내부에서만 사용되는 패킷이므로 최종 호스트로 전달되지 않도록 egress 스위치에서 폐기된다.

5개의 이더넷 스위치와 각 스위치 당 8개의 호스트로 구성된 네트워크를 구성하여 성능을 측정한 결과, 평균 switch-over 시간이 약 33ms로 측정되었다. 대부분의 switch-over 시간은 링크 장애를 탐지하는데 소요되는 시간으로 관찰되었다. 각 스위치에 등록된 엔트리 개수와 recovery 시간과의 상관 관계를 분석한 결과, 500개 이상의 엔트리를 초과하는 경우에는 동일한 시간이 소요되는 것으로 관찰되었다. 모든 호스트들 간 연결이 가능하도록 구성하는 경우, 각 스위치의 플로우 테이블에는 2308개의 엔트리를 갖는다. 또한, 노드와 호스트 개수가 증가하면, 플로우 테이블의 엔트리 개수가 크게 증가하므로 확장성 문제가 발생할 수 있다. 현재 FPGA를 이용한 오픈플로우 스위치는 64,000개 정도의 활성화된(active) 엔트리들을 수용할 수 있으므로 수십 개의 스위치와 호스트로 구성된 네트워크에도 적용될 수 있다.

### 나. 그룹 테이블을 이용한 보호기술

세그먼트 단위의 보호 절체와 달리 여러 개의 노드를 경유하는 워킹 경로를 보호 경로로 절체하는 보호기술은 오픈플로우의 그룹 테이블 사용하여  구현될 수 있다. 본 절에서는 그룹 테이블

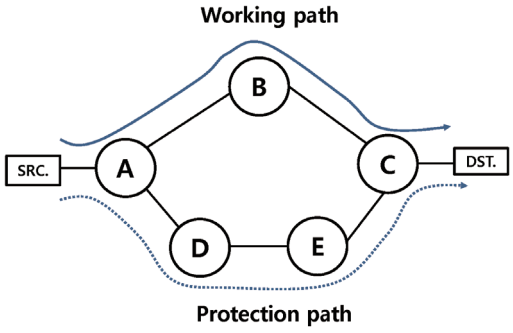


그림 6. 네트워크 구성 및 트래픽 경로

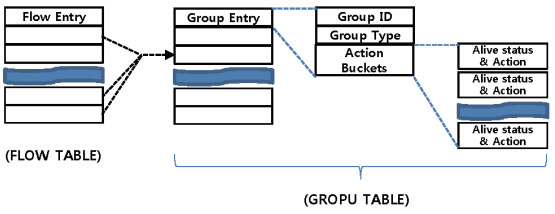


그림 7. 그룹 테이블 구성도

을 이용한 경로 단위의 1:1 보호 절체 기술을 소개한다 [18].

그룹 테이블을 이용한 보호 절체를 위해서 〈ABC〉 노드를 경유하는 워킹 경로에 대해서 〈ADEC〉를 경유하는 보호 경로가 설정된다. 스위치 A는 보호 절체를 위하여 그림 7과 같은 형식의 그룹 테이블을 가지며, 출력 포트 B와 D를 위한 두 개의 Action buckets을 갖는다. 스위치 B와 C는 워킹 경로를 위한 플로우 테이블 엔트리를 가지며, 스위치 D, E, C는 보호 경로를 위한 플로우 엔트리를 갖는다. 스위치 B, D, E는 그룹 테이블을 갖지 않으며, 스위치 C는 워킹 경로와 보호 경로를 위한 두 개의 플로우 테이블 엔트리를 갖는다. 그룹 테이블 엔트리는 그룹 타입(fast failover type)과 복수 개의 action bucket으로 구성된다. 따라서, 스위치 A에서 플로우 엔트리에 매칭되는 패킷을 수신하면, 그룹 ID가 참조되어 해당 그룹 테이블로 전달된다. 전달된 패킷은 alive 상태의 action bucket에 포함된 명령어에 의하여 처리된다.

만약, 장애가 발생되면, BFD (Bidirectional Forwarding Detection)와 같은 OAM 패킷에 의하여 장애가 탐지된다. 이때, 스위치 A는 그룹 엔트리의 Alive status를 변경하여 특정한 action bucket이 동작하지 않도록 한다. 이러한 원리에 의하여, 스위치 A에서 출력 포트 B와 관련된 action은 수행되지 않고 출력 D와 관련된 action이 수행되어 패킷이 워킹 경로에서 보호 경로로 전달된다.

## 2. SDN 복구기술

복구기술은 네트워크 자원을 효율적으로 사용할 수 있는 장점을 가지나, 장애 발생 이후에 백업 경로에 대한 경로 계산 및 프로비전닝이 요구되므로 보호기술에 비하여 장애 복구 시간이 길어지는 단점을 갖는다. 본 절에서는 차세대 광네트워크 기술인 Elastic optical network (EON)에서 GMPLS와 SDN 기반의 복구 기술을 구분하고 성능을 비교한다[19]. EON은 12.5GHz 단위의 주파수 슬라이스(slice)를 가변으로 할당하여 광 링크(Optical link) 상에 주파수 사용 효율을 향상시켜 데이터 전송 효율을 증가시킨다. EON은 일정한 운영시간이 경과하면, 가변 주파수 대역을 갖는 광 경로 (lightpath) 설정과 해제가 반복되어 주파수 스펙트럼 상에 fragmentation이 발생하므로 주기적인 복구 과정이 요구된다. 또한, EON은 작은 주파수 슬라이스 단위로 잘게 쪼개어 사용되므로 기존의 광 네트워크 기술에 비하여 광 선로 당 파장의 개수가 크게 증가할 뿐만 아니라, 가변 주파수 대역을 설정 및 해제하기 위한 configuration 시간이 상대적으로 길기 때문에 빠른 복구가 요구된다. 다음은 EON에 적용할 수 있는 4 가지 복구 기술을 구분하고 성능을 비교한다.

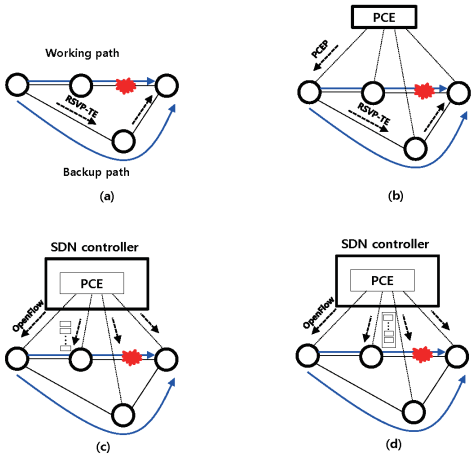


그림 8. 복구기술 종류별 백업 경로 설정 방법

### 가. GMPLS 방식

링크에 장애가 발생하면, 장애 링크에 직접 연결된 노드가 이를 탐지하여 OSPF-TE 와 RSVP-TE프로토콜을 사용하여 장애 정보를 전파한다. 링크 장애를 통보 받은 소스 노드는 RSVP-TE 프로토콜을 사용하여 다음과 같은 절차에 의하여 복구 과정을 수행한다.

- 워킹 경로에 할당된 네트워크 자원 해제
- 장애 경로를 우회하는 백업 경로 계산
- 계산된 백업 경로를 따라서, 연결 설정 요구
- 목적지로 부터 백업 경로 설정 확인 수신

GMPLS 복구 방식의 연결 설정 및 해제는 목적지 노드, 중간 노드, 그리고 소스 노드에 대해서 순차적 방식으로 설정 되므로 SDN 복구 방식에 비하여 복구 시간이 길다.

### 나. GMPLS/PCE 방식

GMPLS와 PCE (Path Computation Element)가 결합되어 네트워크를 제어하는 경우, 순수한 GMPLS 경우와 달리 소스 노드 외부에 위치한 PCE가 경로를 계산한다. 즉, 소스 노드가 워킹 경로 장애를 탐지하면, PCEP (Path Computation Element Protocol)을 이용하여 PCE에 백업 경로 계산을 요청한다. PCE는 장애 지점를 우회하는 백업 경로를 계산하여 소스 노드로 회신하며, 나머지 절차는 GMPLS와 유사하게 RSVP-TE 메시지에 의하여 워킹 경로 해제 및 백업 경로가 설정된다. 이 방법은 PCE에 백업 경로 계산을 요구하는 추가적인 PCEP 통신 지연이 발생하며, PCE가 복수 개의 노드로 부터 동시에 PCE 계산을 요청받는 경우 집중되는 연산의 과부하로 인하여 순수한 GMPLS 방식에 비하여 전체적인 복구 시간이 지연 될 수 있다.

### 다. SDN 방식

SDN 기반의 네트워크 제어에서는 SDN 컨트롤러가 경로

계산 기능과 모든 노드에 직접 경로 설정기능을 수행한다. GMPLS 방식과 비교하며, SDN 컨트롤러는 백업 경로를 계산하여 백업 경로를 통과하는 모든 노드들에게 네트워크 자원 예약을 지시하는 제어 메시지를 병렬로 전달한다. 그러므로 백업 경로 노드에 따라서 순차적 시그널링 방식으로 네트워크 자원을 예약하는 GMPLS 방식에 비하여 빠르게 장애가 복구된다. SDN 컨트롤러가 노드로부터 오픈플로우 포트 상태 메시지에 의하여 장애를 통보 받으면, 다음과 같은 절차에 의하여 복구 과정을 수행한다.

- 장애에 의하여 서비스 제공이 어려운 경로들을 수집하고 TED안에 해당 경로들이 사용하는 자원들을 해제한다
- 백업 경로를 계산하고 TED 정보를 업데이트 한다
- 백업 경로를 네트워크에 설정하기 위해서 백업 경로가 통과하는 모든 노드들에게 오픈플로우 플로우 모드 변경 메시지를 전달하여 백업 경로를 설정한다.

SDN 제어 방식은 SDN 컨트롤러와 노드 사이의 제어 메시지 (OpenFlow 메시지) 전달 방법에 따라서 여러 개의 제어 메시지를 동시에 전달하는 bundle 방식 (SDN-bundle)과 한 번에 한 개의 제어 메시지만을 전달하는 개별 방식 (SDN-individual)으로 구분된다.

### 라. 성능 분석

성능 분석을 위해서 25개 노드에 55개 양방향 링크로 연결되는 네트워크을 구성하고 256 개의 주파수 슬라이스 (12.5GHz 단위)를 갖는 시뮬레이션 모델을 구성하였다. 그리고 장애가 탐지되어 경로 설정이 요청되는 시점부터 시그널링에 의하여 연결 설정이 종료되는 시간을 측정하였다. 측정 결과, SDN 제어 방식은 90ms 미만이 소요되는 반면에, GMPLS의 경우에는 약 280ms 정도 소요되었다. 이때, 경로 계산을 위한 소요 시간과 노드 설정 시간을 각각 10ms와 50ms로 가정하였다. 따라서, GMPLS 제어를 사용하는 경우에 비하여 SDN을 사용하는 경우 프로비전닝 시간이 약 3배 정도 빠르다고 할 수 있다. 또한, 순차적인 시그널링 방식인 GMPLS 방식은 경로를 통과하는 노드 개수가 증가할수록 프로비전닝 시간이 증가하므로 네트워크 규모가 크면 클수록 SDN 방식에 비하여 복구 시간이 지연된다.

## Ⅳ. 결 론

본 고에서는 OpenDaylight과 ONOS 오픈소스 프레임워크에서의 컨트롤러 고확장성과 고가용성을 지원 방안에 대하여 살펴보았다. 두 프레임워크 모두 논리적으로는 중앙집중형 컨트롤러를 지향하면서도 분산 클러스터 방식의 컨트롤러 확장을 통하여 컨트롤러의 가용성과 확장성을 지원하였다. 특히 ONOS에서는 트래픽 로드의 분배를 통하여 시간당 처리할 수 있는 트래픽 수용능력을 향상시킴으로써 높은 수준의 성능을 요구하는 통신사업자망에 적용할 수 있는 가능성을 높였다고 할 수 있다.

또한, 네트워크 장애가 발생되는 경우, 신속하게 복구하기 위한 recovery 기술로서 SDN 기반의 보호기술과 복구기술이 소개 되었다. SDN 보호기술인 OpenFlow 플로우 테이블을 이용한 세그먼트 단위의 보호 절체와 OpenFlow 그룹 테이블을 이용한 보호 절체 모두 50ms 이내의 빠른 보호 절체 성능을 제공한다. 그리고 EON에서 동작하는 SDN 기반의 복구 방식과 기존 GMPLS 복구 방식의 성능 비교 결과, SDN 복구 방식이 기존의 GMPLS 복구 방식에 비하여 복구 시간이 크게 향상되었다.

## 참 고 문 헌


[1] Floodlight Project, “www.projectfloodlight.org/floodlight”

[2] NOX OpenFlow controller, “www.noxrepo.org”

[3] Beacon controller, “https://openflow.standford.edu/display/Beacon/Home”

[4] Ryu controller, “osrg.github.io/ryu”

[5] IRIS: The Recursive SDN OpenFlow Controller by ETRI, “openiris.etri.re.kr”

[6] A. Tootoonchian, and Y. Ganjali, HyperFlow: A Distributed Control Plane for Openflow, INM/WREN’10, San Jose, CA, April, 2010.

[7] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, Towards an Elastic Distributed SDN Controller, HotSDN’13, Hong Kong, China, August 12-16, 2013..

[8] P. Berde, M. Gerola, J. Hart, et al, “ONOS: towards an open, distributed SDN OS,” HotSDN’14, Chicago, Illinois, August 17-22, 2014.

[9] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, “Logically Centralized? State Distribution Trade-offs in Software Defined Networks,” HotSDN’12, Helsinki, Finland, August 13-17, 2012.

[10] G.805, “Generic functional architecture of transport networks,”2000.

[11] E. Mannie and D. Papadimitriou, “Recovery (Protec-

tion and Restoration) Terminology for Generalized Multi-Protocol Label Switching (GMPLS)," IETF RFC4427, 2006.
[12] ITU-T Rec. G.873.1, "Optical Transport Network(OTN): Linear Protection," 2014.
[13] ITU-T Rec. G.8131/Y.1382, "Linear Protection Switching for MPLS Transport Profile (MPLS-TP)," 2014.
[14] J. Ryoo, T. Cheung, D. King, A. Farrel, H. Hoovoort, "MPLS-TP Linear Protection for ITU-T and IETF," IEEE Communications Magazine, vol. 52, issue: 12, pp. 16-21, Dec., 2014.
[15] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takacs, and P. Skoldstrom, "Scalable fault management for OpenFlow," ICC 2012, Ottawa, Canada, June, 2012.
[16] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P.Castoldi, "Effective flow protection in OpenFlow rings," OFC/NFOEC, Anaheim, USA, Mar. 2013.
[17] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "OpenFlow-Based Segment Protection in Ethernet Networks," IEEE/OSA Journal of Optical Communications, vol. 5, issue. 9, pp. 1066-1075, Sep., 2013.
[18] S. Staessens, S. Sharma, D. Colle, M. Pickavet, P. Demeester, "OpenFlow: Meeting Carrier-Grade Requirements" LANMAN 2011, NC, USA, Oct. 13-14, 2011.
[19] A. Giorgetti, F. Paolucci, F. Cugini, and P. Castoldi, "Dynamic Restoration with GMPLS and SDN Control Plane in Elastic Optical Networks," IEEE/OSA Journal of Optical Communications and Networking, vol. 7, issue 2, pp. A174-A182, Feb. 2015.
[20] A. Giorgetti, F. Paolucci, F. Cugini, and P. Castoldi, "Fast restoration in SDN-based flexible optical networks," OFC2014, San Francisco, USA, March 11-13, 2014.
[21] 이범철, 양선희, 이병선, "SDN/NFV/Cloud 동향,"전자통신동향분석, 30권 1호, pp.87-93, 2015.
[22] 백은경, "소프트웨어 기반 NFV 상용화와 오픈 소스 기반 개발 동향," TTA Journal, vol. 157, pp. 54-59, 2015.
[23] OpenDaylight project, "www.opendaylight.org"
[24] Akka, "www.akka.io"
[25] Raft algorithm, "https://raftconsensus.github.io"
[26] Moiz Raja, "odl-mdsal-clustering," OpenDaylight Silicon Valley Meetup, April, 2015.
[27] Colin Dixon, "Consistency Trade-offs for SDN Controllers," OpenDaylight Summit, 2014.
[28] ON.LAB, "www.onlab.us"
[29] ONOS whitepaper, "Raising the bar on SDN control plane performance and scalability," ON.Lab, April, 2015.

## 약 력

윤 빈 영

1986년 중앙대학교 전자공학과 공학사
1991년 중앙대학교 전자공학과 공학석사
2004년 충남대학교 전자공학과 공학박사
1991년~현재 한국전자통신연구원 책임연구원
관심분야: Transport SDN, 광가입자망(PON) 기술, 네트워크 인터페이스 기술

김 태 홍

2005년 아주대학교 정보및컴퓨터공학 학사
2007년 KAIST 정보통신공학 석사
2012년 KAIST 전산학 박사
2012년~2014년 삼성전자 책임연구원 (종합기술원/DMC연구소)
2014년~현재 한국전자통신연구원 선임연구원
관심분야: 무선 센서 네트워크, ZigBee, 소프트웨어 정의 네트워크

김 영 화

1987년 전남대학교 이학사
1997년 충남대학교 이학석사
2005년 충남대학교 이학박사
1988년~현재 한국전자통신연구원 책임연구원, SDN기술연구실장
관심분야: 통신 네트워크 및 시스템, 미래인터넷, 소프트웨어 정의 네트워크

양 선 희

1984년 경북대학교 전자공학과 공학사
1986년 한국과학기술원 전기 및 전자공학과 공학석사
2004년~2005년 미국 UC Davis 방문연구원
1988년~현재 한국전자통신연구원 책임연구원, 방송통신융합미래기술연구부장, SDN기술연구실장, 스마트네트워크연구부장
관심분야: 프로그래머블 네트워크, 인프라 가상화, 네트워킹 프로토콜 기술