5G 시대의 SDN/NFV

2019. 10.

안 종 석
james@jslab.kr
JS Lab

1

SDN/NFV for 5G
james@jslab.kr

2

SDN/NFV for 5G

james@jslab.kr

I. 개요
II. 소프트웨어 정의
III. 가상화와 클라우드 서비스
IV. SDN 개요
V. NFV
VI. 오버레이 / 언더레이
VII. SDN 관련 기술
VIII. 클라우드 네트워크
IX. 텔레콤 환경을 위한 SDN/NFV
X. 관리

❖ 부록: OpenFlow
❖ 실습교재 (별도)

JS Lab

3

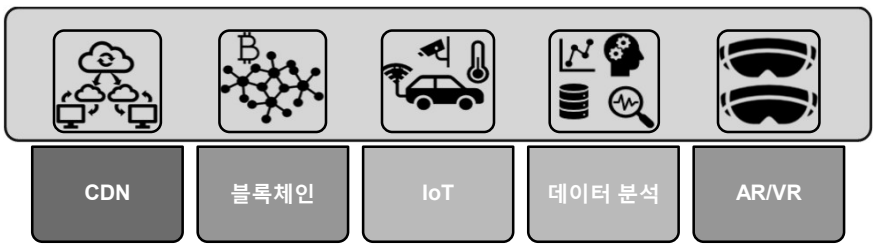
SDN/NFV for 5G

james@jslab.kr

I. 개요

❖ **5G 활용 기술 분야**

- **하드웨어**
- **인프라 소프트웨어** (오픈소스)
- **CDN, 블록체인, IoT, 데이터 분석** (ML/AI), **AR/VR**
- **서비스 애플리케이션**

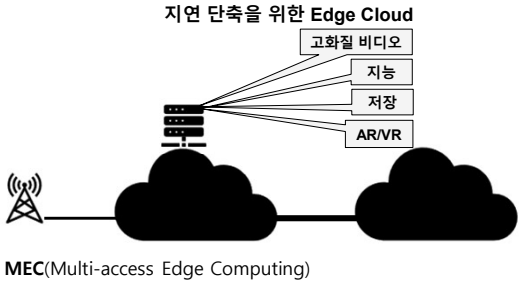


JS Lab

4

I. 개요

- ❖ **기존 시장의 기술/비즈니스 5G로 확장 중** (단말/서버)
- ❖ **B2B 시장 기회** (Vertical Market 별 환경 수용)
- ❖ **지연 단축을 위한 Edge Cloud Computing 적용기술 확대**



MEC(Multi-access Edge Computing)

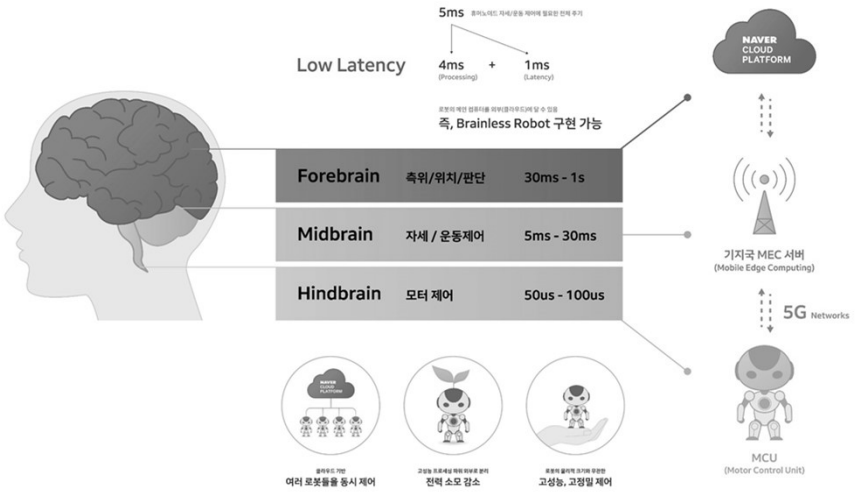
SDN/NFV for 5G
james@jslab.kr

JS Lab

5

I. 개요

- ❖ **5G 브레인리스 로봇 기술은 로봇의 뇌를 클라우드 에서 수행**



출처: https://www.naverlabs.com/storyDetail/110?fbclid=IwAR3CjFDSmOw1JwHkvdgLHteORbACfdEmiFM5Hwm5aUgphJewd5RAqPNp6ZE

SDN/NFV for 5G
james@jslab.kr

JS Lab

6

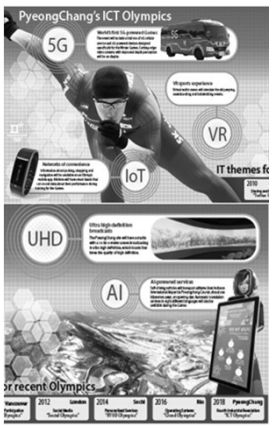
SDN/NFV for 5G

james@jslab.kr

I. 개요

❖ **통신사의 서비스**

- **KT 평창 동계올림픽**
- **SKT 5G 클러스터**



출처: https://news.joins.com/pyeongchang2018/Daily/Article/22144541 출처: https://www.sktinsight.com/116852

JS Lab

7

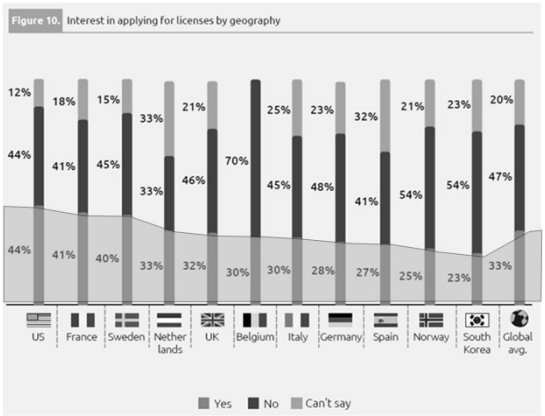
SDN/NFV for 5G

james@jslab.kr

I. 개요

❖ **통신사 B2B 비즈니스 모델 변화:** 일본(총무성), 미국(FCC), 독일(BNetzA), 영국(Ofcom), 등은 기업용 Private 5G 주파수를 할당

❖ **기업들의 유연한 5G 이용 고려:** 직접 Private 5G망 구축 또는 통신사의 Licensed 주파수 기반 Private 5G 서비스 가능



기업들이 지역별 라이선스에 관심

출처: https://www.capgemini.com/gb-en/wp-content/uploads/sites/3/2019/06/Report-5G-in-industrial-operations.pdf

JS Lab

8

I. 개요

❖ **독일 BNetzA(Bundesnetzagentur) 연방통신청:** 5G를 제조/공장/병원 등 산업계에서 활용 가능한 100MHz(3.7-3.8GHz)의 대역을 일반 기업용으로 할당 (DT의 5G 주파수 총대역폭: 130MHz)

- **독일의 5G 상용화 시기: 2020년**
- **이통사용 5G 주파수 경매:**
  - ✓ 주파수 대역 (대역폭): 2GHz (120MHz), 3.4-3.7GHz (300MHz)
  - ✓ 경매: 2019년 6월 경매 완료, 4개 이통사(Deutsche Telekom, Vodafone, Telefonica, 1&1 Drillisch)에게 총 420MHz를 €6.5Bn에 판매
- **일반 기업용 Local 5G 주파수와 할당 시기:**
  - ✓ 주파수 대역 (대역폭): 3.7-3.8GHz (100MHz)
  - ✓ 2019년 하반기 신청 시작 (경매가 아닌 신청/검토/할당 방식임)

폭스바겐은 2020년에 Local 5G 주파수(3.7-3.8GHz)를 이용하여 122개의 자기 공장에 Private 5G망을 자체 구축하겠다고 올 4월에 발표했다. Bosch와 Siemens는 이미 자체 테스트용 Private 5G망을 구축했고 3.7-3.8GHz 스펙트럼이 상용화되면 이 주파수로 전환할 예정이다. Audi는 작년에 Ericsson과 공장내 5G trial망 장비 계약을 했고, 상용/최종 망 구축 비용은 $100,000에 이른다고 알려져 있다. – Netmanias -

I. 개요

❖ **군(軍)을 위한 5G 네트워크의 강점** By Jamie Carter June 13, 2019 (TechRadar)

- **미국의 중국 5G 무기화 대응** (5G: why is it a geopolitical issue?)
- **항공, 전력망, 댐, 커넥티드카 등의 인프라** (5G as critical infrastructure)
- **초음속 무기 방어 시스템** (5G and hypersonic weapons)
- **스마트 기지** (5G and the 'smart' military base)
- **전투망** (5G and the 'battle network')
- **전투복** (5G and 'battlefield wearables')
- **드론, 인공지능** (5G, drones and artificial intelligence)

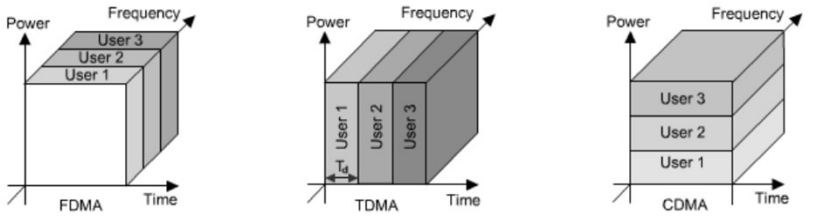


출처: https://www.techradar.com/news/how-the-5g-network-could-benefit-the-military

I. 개요

❖ **무선(Radio):** 스펙트럼의 물리적 가용성을 향상시키는 방법으로 발전

- **주파수 분할: Frequency Division Multiple Access (FDMA)**
- **시간 분할: Time Division Multiple Access (TDMA)**
- **코드 분할: Code Division Multiple Access (CDMA)**

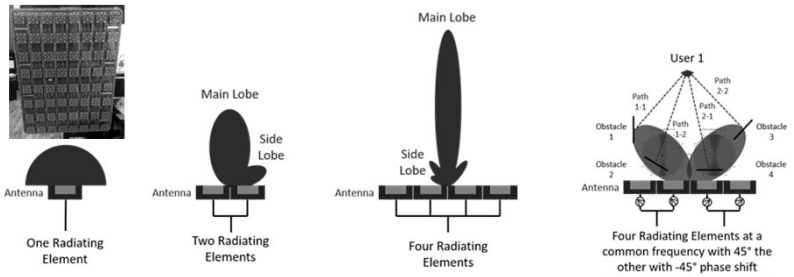


11

I. 개요

❖ **빔포밍** (What is 5G beamforming?)

- **무선 신호의 위상제어(Beam steering):** 위상변화를 사용 수신자 목표로 지향성 무선 신호를 송신 **(**Beam steering is achieved by changing the phase of the input signal on all radiating elements. Phase shifting allows the signal to be targeted at a specific receiver.)
- **Massive MIMO**(Multiple Input and Multiple Output)**:** 일반 주파수를 동시에 여러 방향으로 전송 (동일 주파수 중복 사용)



출처: https://www.metaswitch.com/knowledge-center/reference/what-is-beamforming-beam-steering-and-beam-switching-with-massive-mimo

12

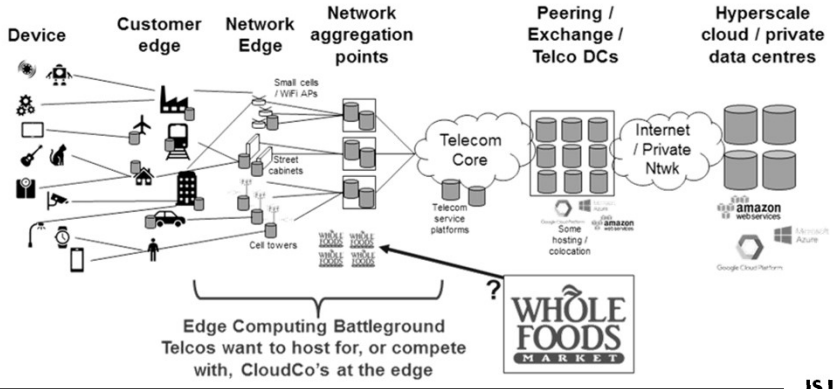
SDN/NFV for 5G

james@jslab.kr

I. 개요

❖ **아마존은 왜 Whole Foods를 인수 했나?**

- **클라우드 고객의 저지연 서비스 요구 증가**
- **클라우드 서비스 업체는 에지에서 Telco와 협력 또는 경쟁**
  (예: AWS, MS Azure, Tencent Cloud 등)



출처: https://disruptivewireless.blogspot.com/2017/06/does-amazons-purchase-of-whole-foods.html

JS Lab

13

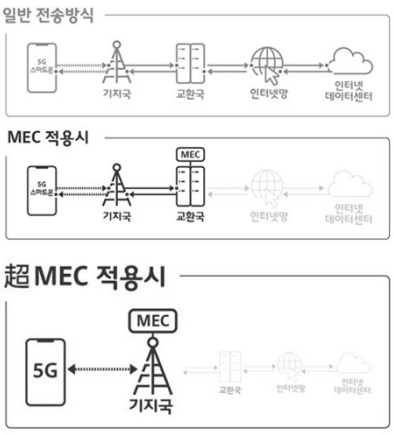
SDN/NFV for 5G

james@jslab.kr

I. 개요

❖ **초(超)엣지 @ SKT**

- **@ 5G Phase 1 (MEC)**
- **@ 5G Phase 2 (초MEC)**



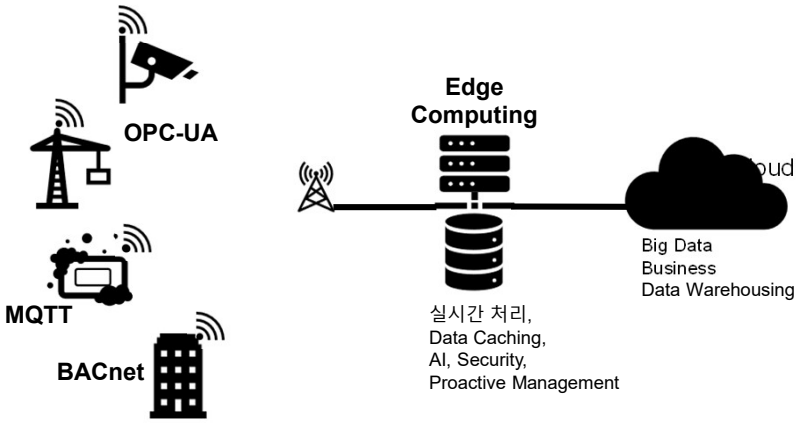
출처: https://www.sktinsight.com/117285

JS Lab

14

I. 개요

- **에지 컴퓨팅 (Edge Computing) :**데이터를 발생하는 사물 옆이나 내장하는 형태의 컴퓨팅
- **B2B 생태계의 기존 사업 보호 경향 영역** (문화 유지와 수익의 균형)



SDN/NFV for 5G
james@jslab.kr

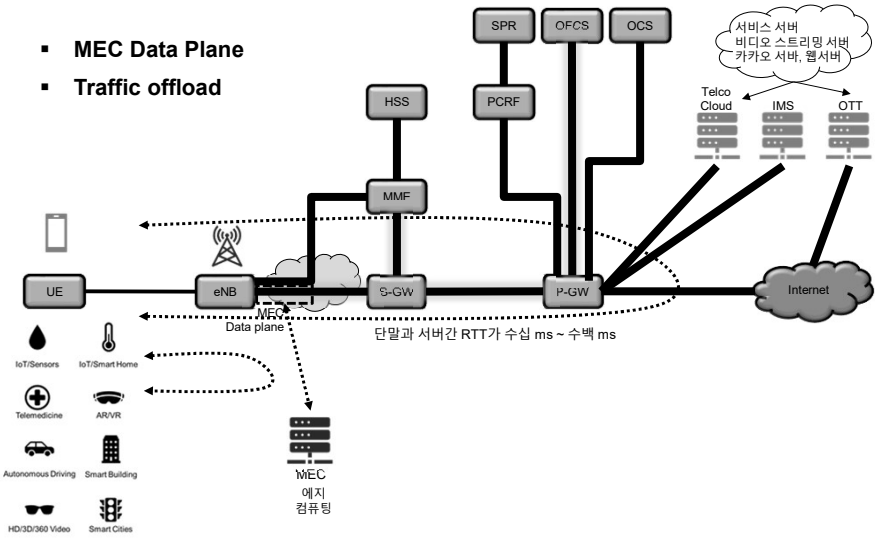
BACnet Building Automation and Control (BAC) networks **OPC** Unified Architecture (**OPC UA**) - Open Platform Communications

JS Lab

15

I. 개요

- **모바일 에지 컴퓨팅** (MEC) **@ 4G**
  - **MEC Data Plane**
  - **Traffic offload**



SDN/NFV for 5G
james@jslab.kr

MEC (Mobile|Mutli-access Edge Computing)

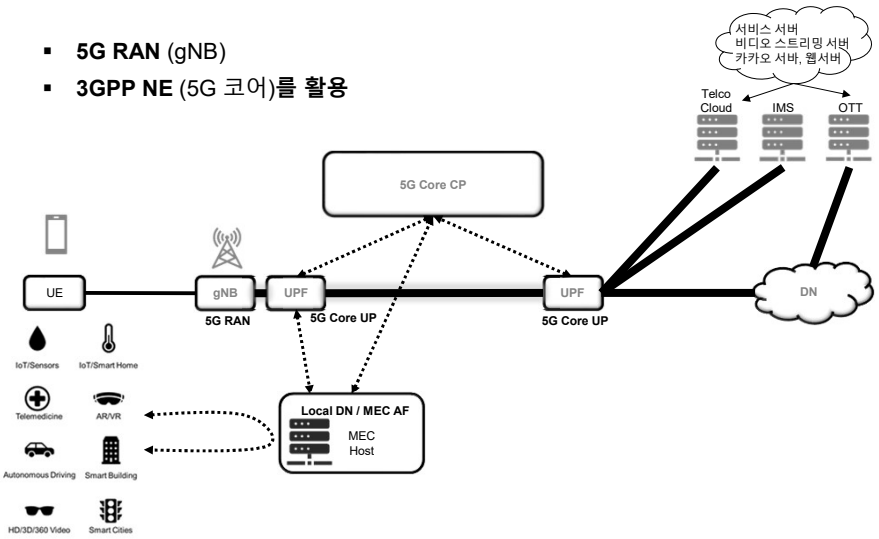
JS Lab

16

I. 개요

❖ **모바일 에지 컴퓨팅** (MEC) **@ 5G**

- **5G RAN** (gNB)
- **3GPP NE** (5G 코어)**를 활용**



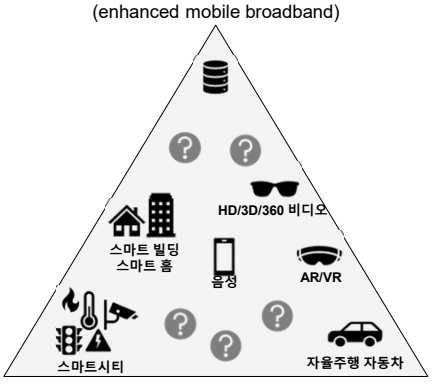
17

I. 개요

❖ **5G 표준의 서비스 시나리오 고려 표준**

- **eMBB**(대역폭 개선), **mMTC**(기기 종류 및 수량 증가), **URLLC**(초저지연)

eMBB: 대역폭 10배 이상 증가
(enhanced mobile broadband)



mMTC: 지원기기 종류/수량 대폭증가
(massive machine-type communication)

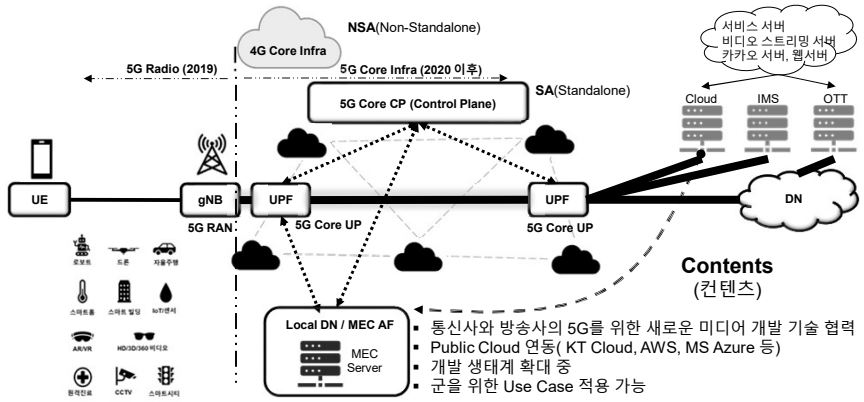
URLLC: 고신뢰 초저지연
(ultra-reliable low latency)

18

I. 개요

❖ **5G와 MEC** (Mobile | Multi-access Edge Computing)

- **에지의 데이터센터 기술 도입:** 국사의 데이터센터화 기지국 확대 고려
- **5G는 4G EPC 코어 공유로 서비스 시작:** 5G 코어 적용 확대 중
- **MEC는 Eco-system 확대 영역:** API 제공 및 B2B 등의 모델 확대



RAN(Radio Access Network) **MEC**(Mobile | Multi-access Edge Computing) **AF**(Application Function) **UPF**(User Plane Function)

JS Lab

SDN/NFV for 5G

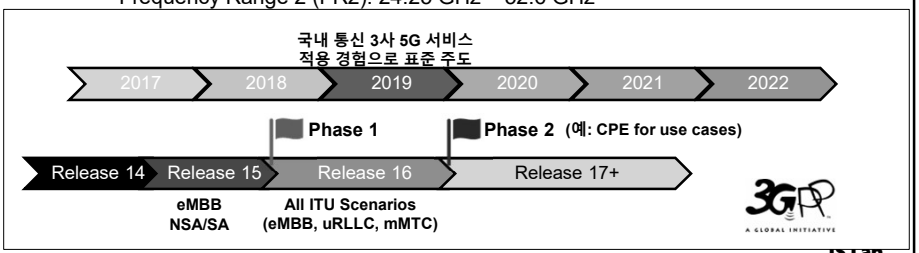
james@jslab.kr

19

I. 개요

❖ **5G 표준 Roadmap**

- **5G 표준은 '3GPP' 세계 통신 표준 단체에서 규정**
- **Phase 1** (3GPP Rel. 15, 2018년 6월)
- **Phase 2** (3GPP Rel. 16, 2019년 12월 이후 freeze 예상)
- **3GPP Rel. 17은 5G 개선** (2020년 시작)
- **국내 통신 3사 5G 서비스 시작** (2019년)
- **표준 적용은 대개 18개월정도 예상**
- **3GPP는 5G Radio 주파수를 2 부분으로 진행중**
  - Frequency Range 1 (FR1): 450 MHz – 7.125 GHz
  - Frequency Range 2 (FR2): 24.25 GHz – 52.6 GHz



3GPP(3rd Generation Partnership Project) **eMBB**(대역폭 개선), **mMTC**(기기 종류 및 수량 증가), **URLLC**(초저지연)

JS Lab

SDN/NFV for 5G

james@jslab.kr

20

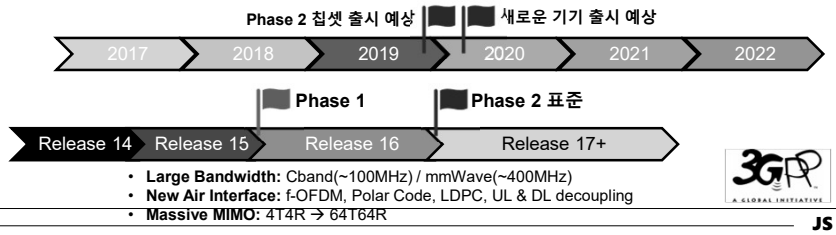
I. 개요

❖ **5G 표준과 Market의 Radio 환경 변화/발전**

- **Phase 1** (Chipset, Device, Operator)
- **Phase 2** (Next Chipset, Next Device, Full Scale Commercial Service)

주파수	Sub 6GHz		Above 6GHz		
Operator	<3GHz	3~5 GHz	6~24 GHz	24~30 GHz	30~40 GHz
SKT		3.6~3.7 GHz (100MHz)		28.1~29.0 GHz (800MHz)	
KT		3.5~3.6 GHz (100MHz)		26.5~27.3 GHz (800MHz)	
LGU+		3.42~3.5 GHz (80MHz)		27.3~28.1 GHz (800MHz)	

5G NR (100MHz) 1.5 Gbps
4G LTE (145MHz) 1.2 Gbps
} 배터리, latency 고려 동시지원 가능



- **Large Bandwidth:** Cband(~100MHz) / mmWave(~400MHz)
- **New Air Interface:** f-OFDM, Polar Code, LDPC, UL & DL decoupling
- **Massive MIMO:** 4T4R → 64T64R

3GPP A GLOBAL INITIATIVE

JS Lab

SDN/NFV for 5G
james@jslab.kr

21

I. 개요

❖ **5G 표준 Phase 2 이후** (Release 17, 2020년 이후)

❖ **무선 등의 서비스 개선과 새로운 기능 추가**

- **NR Light** – NR evolution
- **Small data transfer optimization**
- **Sidelink enhancements** – NR evolution
- **NR above 52.6 GHz (60GHz unlicensed)** – NR evolution
- **Multi SIM Operation**
- **NR multicast broadcast** (예: 재난망)
- **Coverage enhancements**
- **NB-Iot and eMTC enhancememts**
- **IIoT and URLLC enhancements**
- **MIMO enhancements**
- **NR for Non Terrestrial Networks** – New feature (예: 인공위성)
- **Integrated Access and Backhaul Enhancements** – New feature
- **Generic enhancements to NR-U**
- **Power saving enhancement** (예: 10년 이상 배터리 수명 유지)
- **RAN data collection enhancements**
- **Positioning enhancements**

JS Lab

SDN/NFV for 5G
james@jslab.kr

22

I. 개요

❖ **Flow & M2X @ AT&T**

- **기기들을 위한 경로 제공 서비스와 과금**
- **Sample Code 제공**

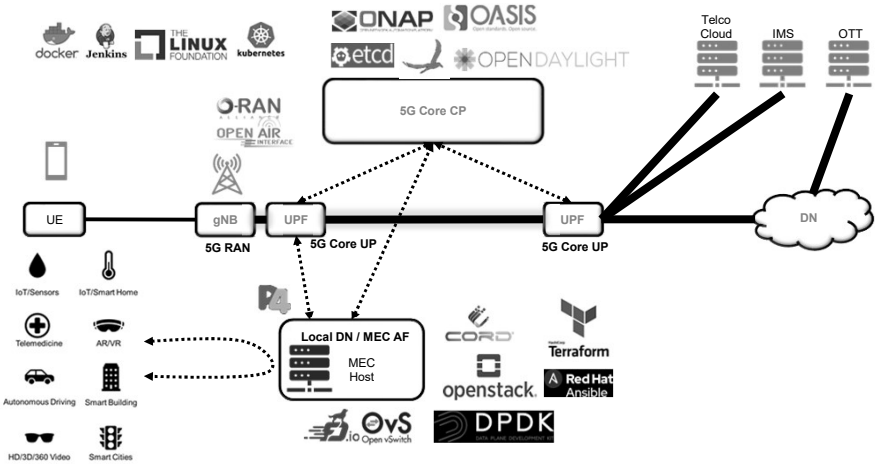
출처: https://flow.att.com/ https://m2x.att.com/

23

I. 개요

❖ **5G를 위한 오픈소스 프로젝트 활성화**

❖ **통신사와 제조사의 개발자 수요 증가와 생태계 확장 노력 중**



24

II. 소프트웨어 정의

❖ DISA(Defense Information Systems Agency)에서 SDN의 역할은 자동화의 시작

Major General Sarah Zabel of the U.S. Defense Information Systems Agency spoke at the Open Networking User Group conference in Mountain View, California, on May 10, 2016. Credit: Stephen Lawson

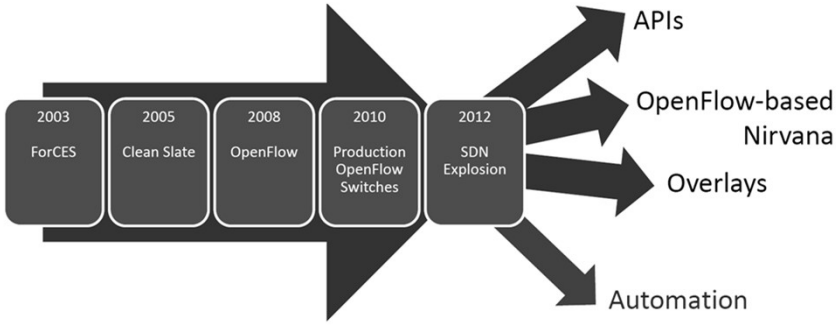
SDN/NFV for 5G

james@jslab.kr

II. 소프트웨어 정의

❖ 소프트웨어 정의 발전

- API (Application Programming Interface)
- Overlays (오버레이)
- OpenFlow-based Nirvana (오픈플로우 기반 너바나)
- Automation (자동화를 위한 소프트웨어 정의 영역 확대)



JS Lab

27

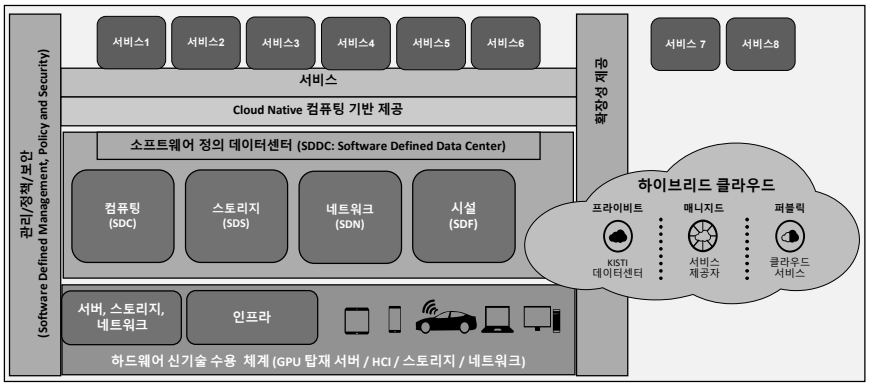
SDN/NFV for 5G

james@jslab.kr

II. 소프트웨어 정의

❖ SDDC 기반 인프라

- Cloud Native등의 기술 발전을 수용하는 SDDC 데이터센터 구축 체계
- 유연한 서비스를 위한 데이터센터 자원의 추상화 관리 (SDDC)
- 베어메탈, Lustre 기반 스토리지 수용

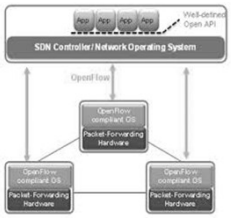


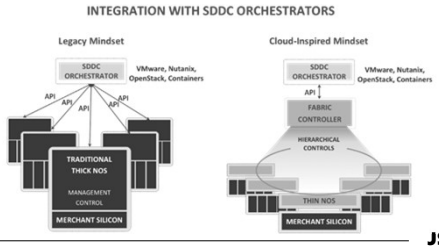
JS Lab

28

II. 소프트웨어 정의

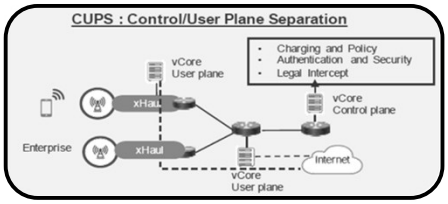
- ❖ Software-Defined Storage
- ❖ Software-Defined Security
- ❖ Software-Defined DataCenter
- ❖ Software-Defined Infrastructure
- ❖ Software-Defined Environment
- ❖ Software-Defined Radio
- ❖ Software-Defined anything(x)

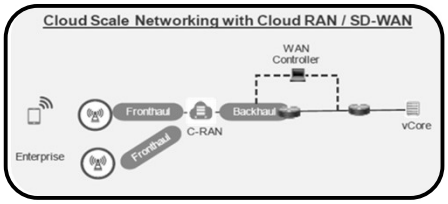


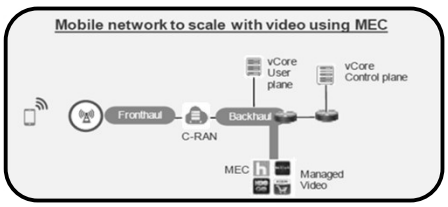


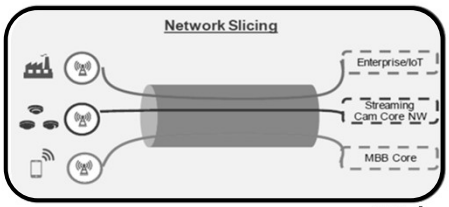
II. 소프트웨어 정의

- ❖ **5G Enabling Technologies:**
  - **CUPS** (제어/사용자 플레인 분리)
  - **Cloud Scale** (클라우드 스케일)
  - **MEC** (모바일 에지 컴퓨팅)
  - **Network Slicing** (네트워크 슬라이싱)



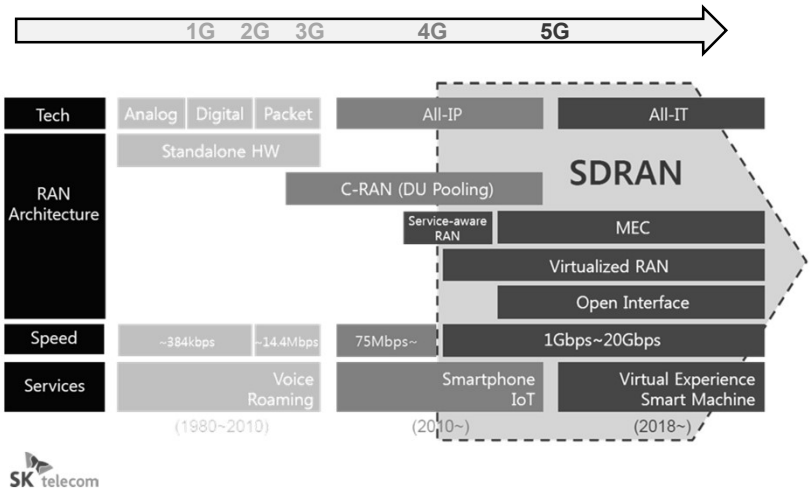






II. 소프트웨어 정의

❖ 5G의 소프트웨어 정의 무선 환경

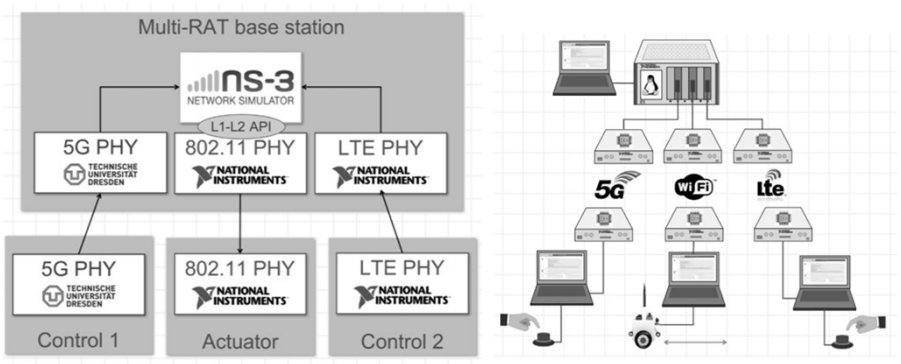


31

II. 소프트웨어 정의

❖ 무선 계층의 추상화

- **Multi-RAT** (Radio Access Technology)
- **SDR** (Software Defined Radio, 소프트웨어 정의 라디오)



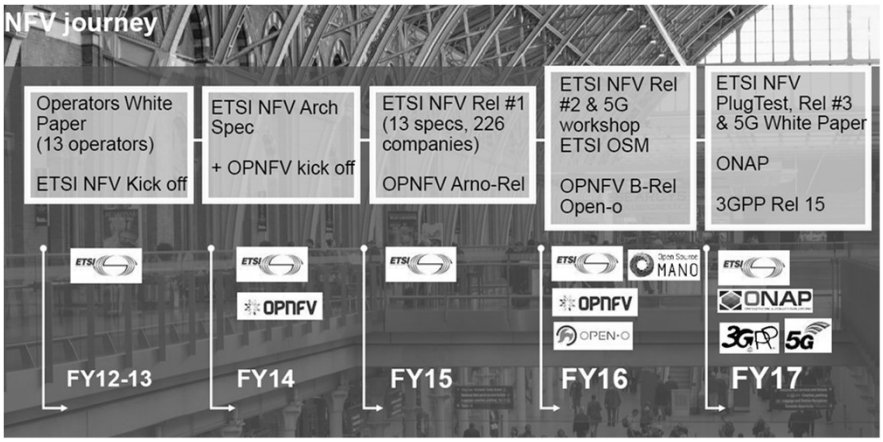
Orchestration and Reconfiguration Control Architecture

출처: https://www.netmanias.com/ko/?m=view&id=oneshot&no=14450

32

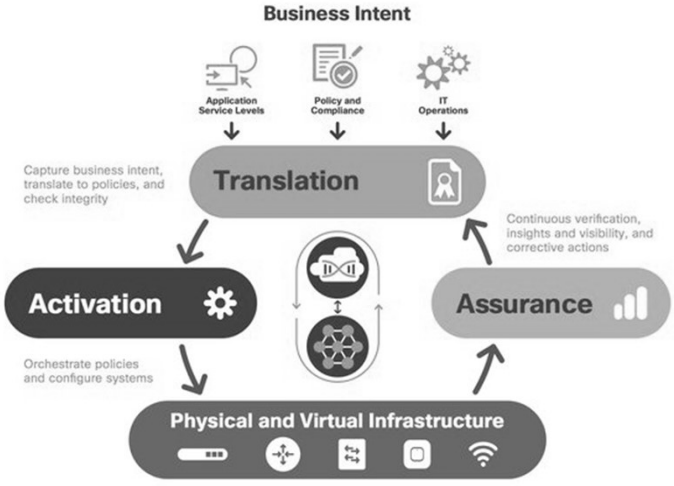
II. 소프트웨어 정의

❖ **운영지원 시스템(OSS)의 미래** (클라우드 네이티브화 진행중)



II. 소프트웨어 정의

❖ **IBN (Intent-Based Networking)**



https://www.sdxcentral.com/networking/sdn/intent-based/definitions/what-is-intent-based-networking/

SDN/NFV for 5G

james@jslab.kr

I. 개요
II. 소프트웨어 정의
III. 가상화와 클라우드 서비스
IV. SDN 개요
V. NFV
VI. 오버레이 / 언더레이
VII. SDN 관련 기술
VIII. 클라우드 네트워크
IX. 텔레콤 환경을 위한 SDN/NFV
X. 관리

❖ 부록: OpenFlow
❖ 실습교재 (별도)

JS Lab

35

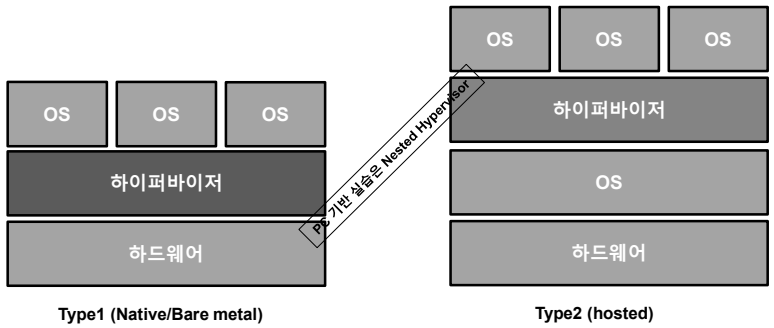
SDN/NFV for 5G

james@jslab.kr

III. 가상화와 클라우드 서비스

❖ 하이퍼바이저(Hypervisor)

① 컴퓨터 안에서 다수의 운영체제(OS)를 동시에 실행하기 위한 논리적인 플랫폼
② VMM(Virtual Machine Monitor/Manager)라고도 불림
③ 네이티브(Native) 하이퍼바이저(Type1)
④ 호스티드(Hosted) 하이퍼바이저(Type 2)

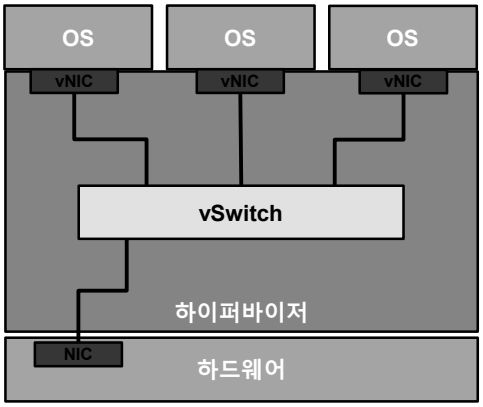


JS Lab

36

III. 가상화와 클라우드 서비스

- ❖ vNIC(Virtual Network Interface Card)
- ❖ vSwitch(Virtual Switch)



37

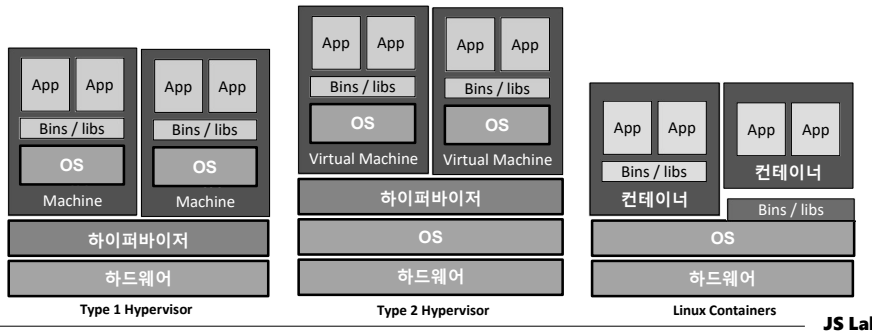
III. 가상화와 클라우드 서비스

- ❖ 컨테이너

① 컨테이너의 어플리케이션을 독립적으로 유지

② 특징

- 가상머신보다 적은 자원 사용
- 스케일링을 가능하게 해주는 관리 도구 포함

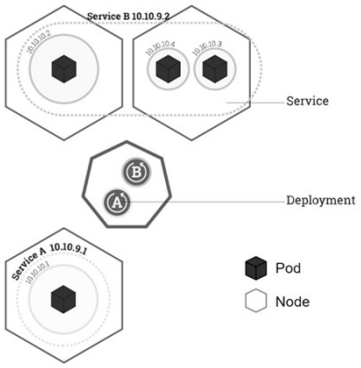


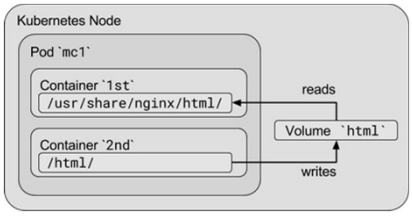
38

III. 가상화와 클라우드 서비스

❖ **Kubernetes Services**

- **관리 단위: Container based Pod**
- **API for Service**



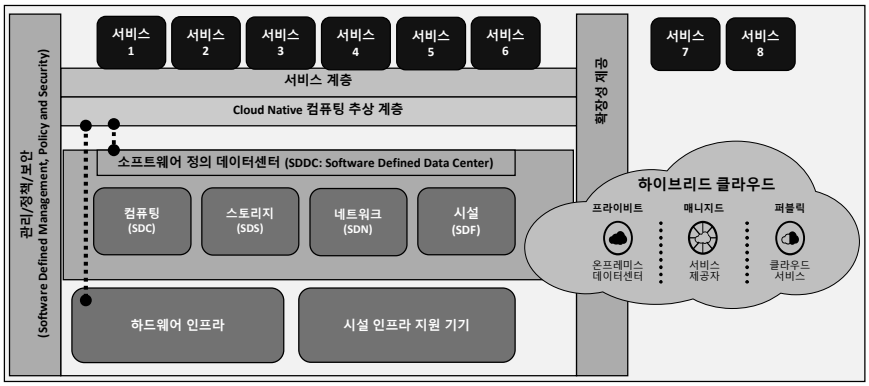


출처: https://kubernetes.io/

III. 가상화와 클라우드 서비스

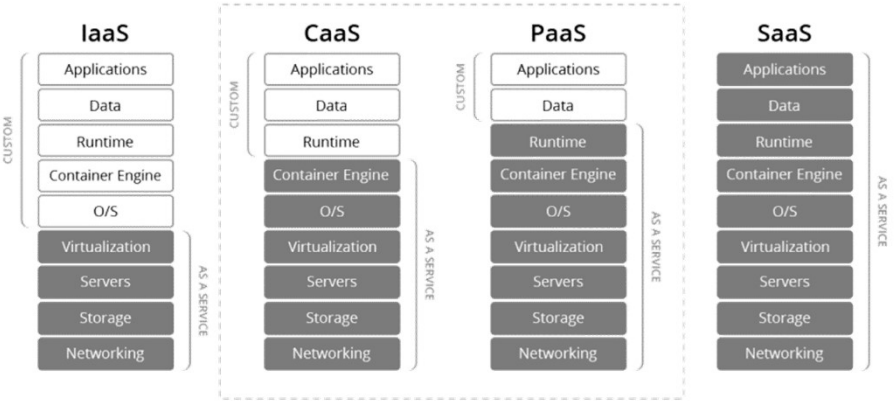
❖ **SDDC 기반 인프라**

- **Cloud Native등의 기술 발전 수용 SDDC 데이터센터 구축 체계**
- **데이터센터 자원의 추상화 관리 (SDDC)**



III. 가상화와 클라우드 서비스

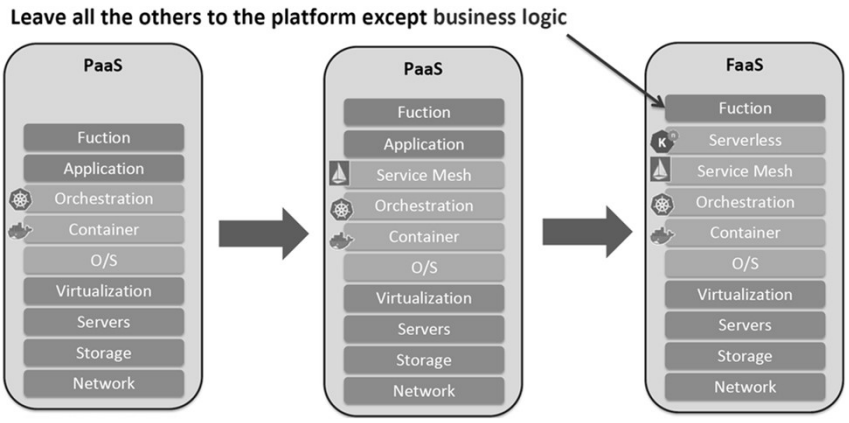
- ❖ 클라우드 서비스의 기술 발전
- ❖ **IaaS (Infrastructure as a Service), SaaS (Software as a Service), and PaaS (Platform as a Service), CaaS (Containers as a Service), FaaS (Functions as a Service)** or Serverless



41

III. 가상화와 클라우드 서비스

- ❖ **K8s/Docker**
- ❖ **Service Mesh**
- ❖ **Serverless**



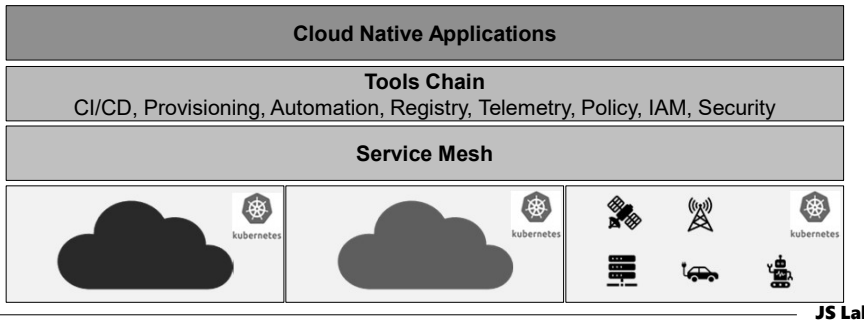
Blog https://zhaohuabing.com, **Github** https://github.com/zhaohuabing

42

III. 가상화와 클라우드 서비스

❖ 멀티 클라우드 Federation w/K8s

- High Availability
- Application Migration
- Policy Enforcement
- Vendor Lock-in Avoidance
- Capacity Overflow



43

III. 가상화와 클라우드 서비스

❖ SDDC 기반 인프라 from Public Cloud

❖ Hybrid Cloud

- IBM Cloud Pak
- AWS Outpost
- MS Azure Stack
- Google Anthos

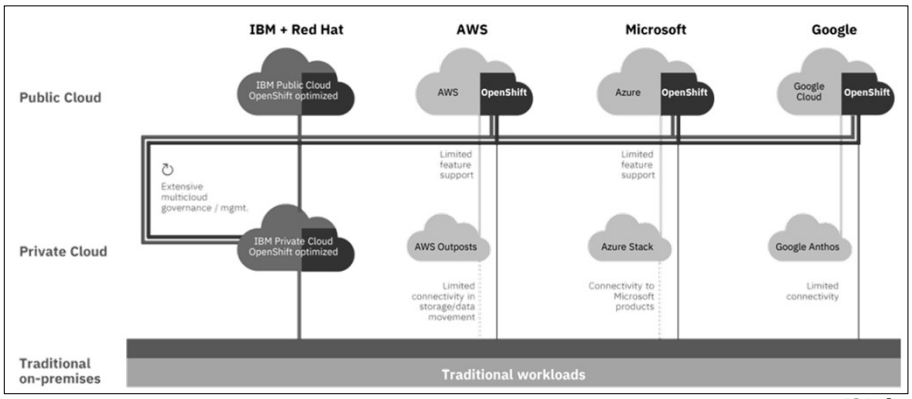
https://www.metricstream.com/technology/grc-cloud.htm

44

III. 가상화와 클라우드 서비스

❖ **IBM Cloud Pak**

- **Hybrid, Multicloud platform**
- **Cloud Pak on OpenShift**
- **Cloud native for all clouds - middleware anywhere**

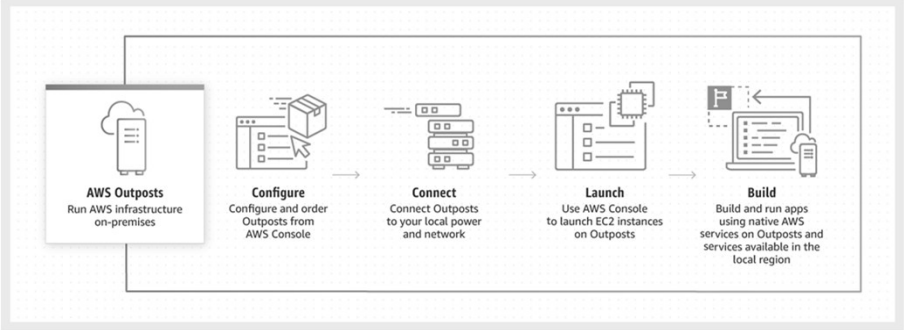


https://www.metricstream.com/technology/grc-cloud.htm

III. 가상화와 클라우드 서비스

❖ **AWS Outpost**

- **AWS 온프레미스 확장**
- **관리 플레인 선택: AWS 네이티브 변형 or VMware Cloud on AWS**
- **완전관리형**

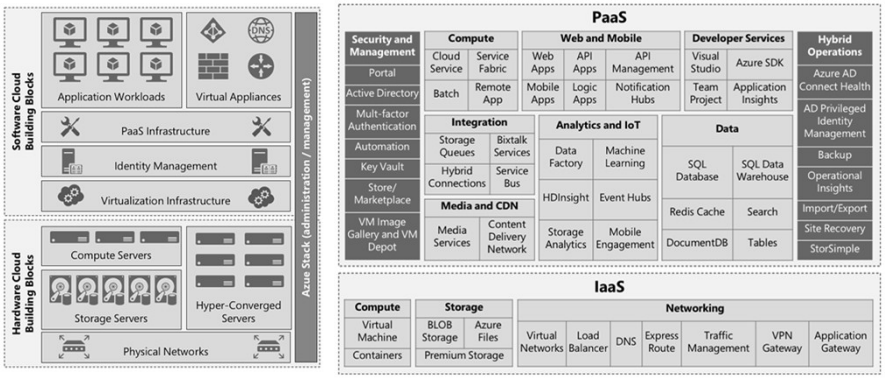


https://aws.amazon.com/ko/outposts/

III. 가상화와 클라우드 서비스

❖ **MS Azure Stack**

- 공용 클라우드 서비스 사용
- 온-프레미스에서 클라우드 서비스 사용
- 온-프레미스에서 가상화된 애플리케이션 실행



https://azure.microsoft.com/ko-kr/overview/azure-stack/ https://argonsys.com/solutions/cloud-building-blocks/

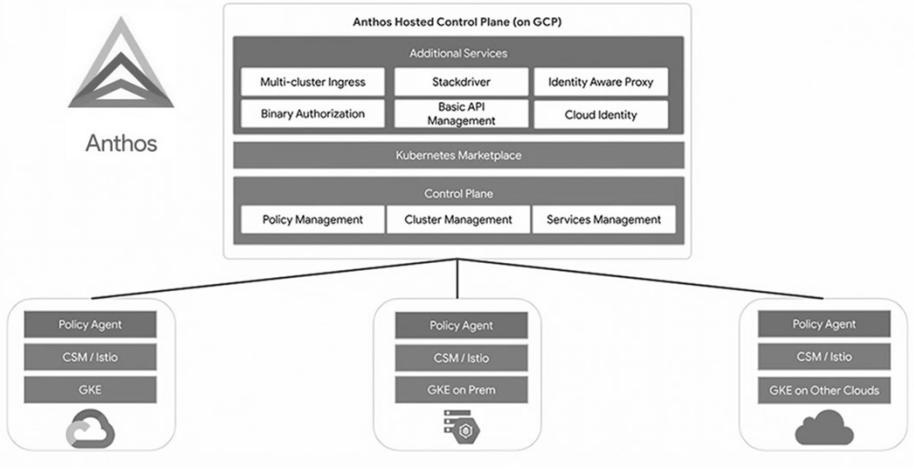
III. 가상화와 클라우드 서비스

❖ **MS Azure Stack**

- **Hybrid Cloud as a Service**

III. 가상화와 클라우드 서비스

❖ **Google Anthos**

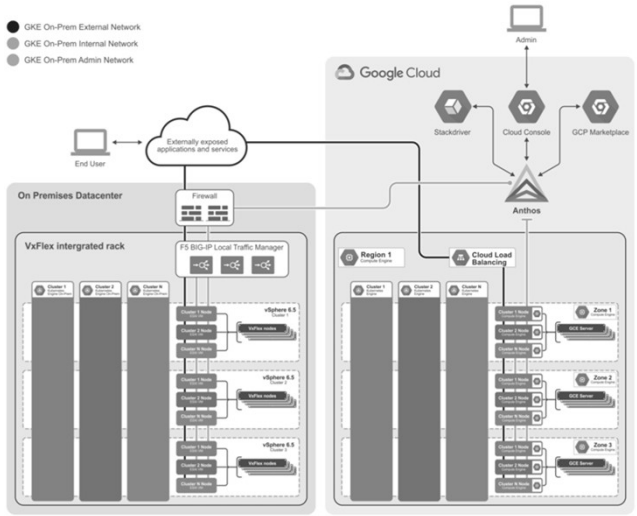


https://www.bespinglobal.com/google-anthos-multi-cloud/ https://blog.aquasec.com/hybrid-cloud-security-google-anthos

49

III. 가상화와 클라우드 서비스

❖ **VxFlex integrated rack for Google Cloud's Anthos**

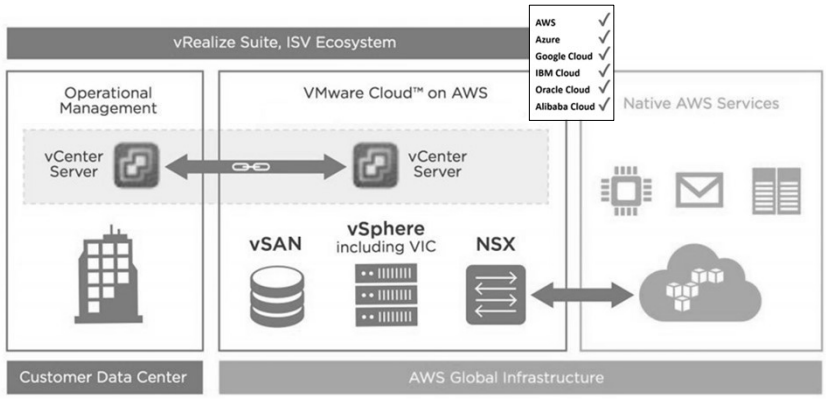


50

III. 가상화와 클라우드 서비스

❖ **VMware Cloud**

- **VMware Cloud on AWS**
- **지원 확장: Azure, Google, IBM, Oracle, Alibaba and KT Cloud**



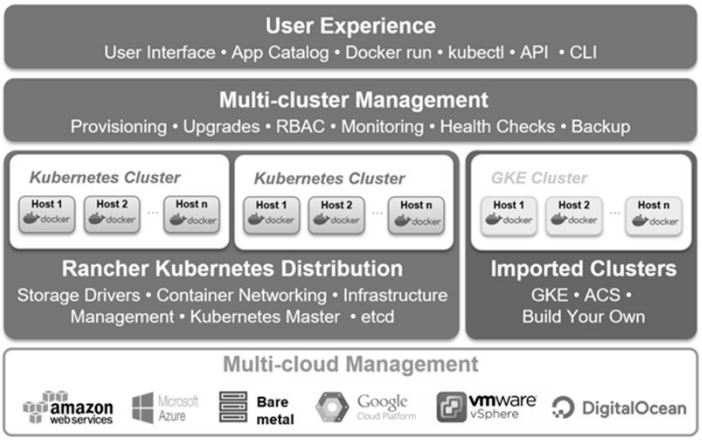
https://www.actualtech.io/getting-hands-dirty-installing-vmware-cloud-aws/

51

III. 가상화와 클라우드 서비스

❖ **Rancher Labs**

- **Rancher 2.x**



https://rancher.com/announcing-rancher-2-0/

52

III. 가상화와 클라우드 서비스

❖ **Rancher Labs**

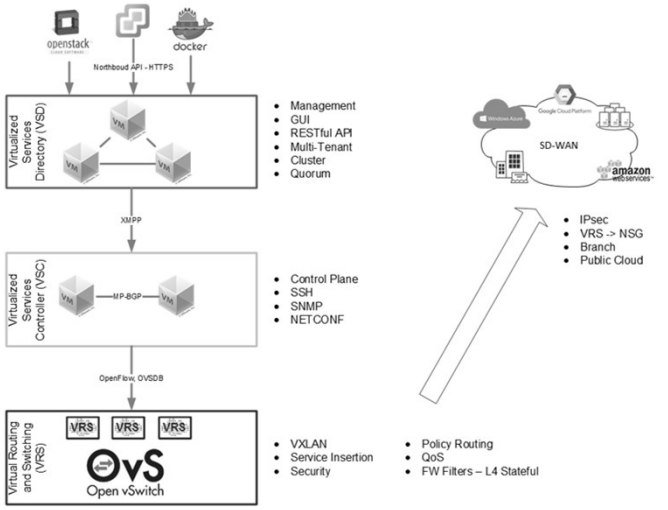
- **Rancher 2.x**

https://rancher.com/announcing-rancher-2-0/

III. 가상화와 클라우드 서비스

❖ **VMware NSX-T 와 KVM 연동**

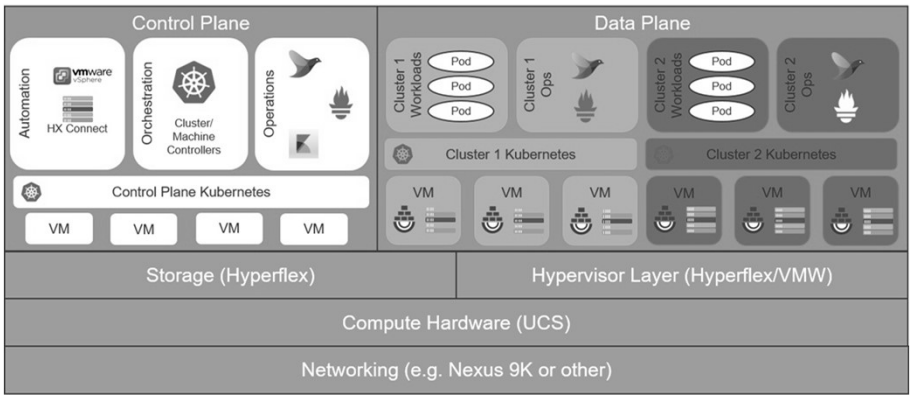
❖ **Kubernetes 연동**



https://blogs.vmware.com/networkvirtualization/files/2017/09/Figure1-Toplogy.png http://www.routetocloud.com/2017/10/introduction-to-nsx-and-kubernetes/

III. 가상화와 클라우드 서비스

- ❖ **Container Platform**
- ❖ **Cisco Container Platform Architecture Overview**



https://www.cisco.com/c/en/us/td/docs/net_mgmt/cisco_container_platform/3-1/Installation_Guide/CCP-Installation-Guide-3-1-0/CCP-Installation-Guide-3-1-0_chapter_00.html

JS Lab

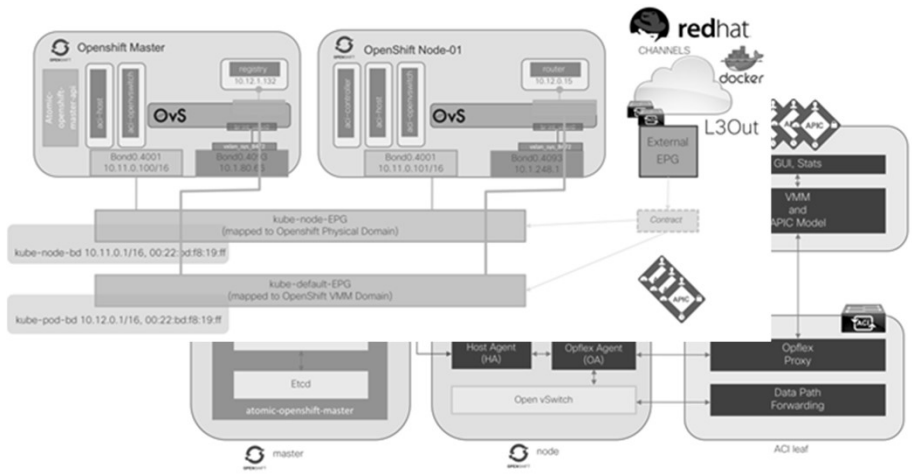
SDN/NFV for 5G

james@jslab.kr

55

III. 가상화와 클라우드 서비스

- ❖ **Cisco Application Policy Infrastructure Controller (APIC)**
- ❖ **General ACI CNI Plugin architecture**



https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/white_papers/Cisco-ACI-CNI-Plugin-for-OpenShift-Architecture-and-Design-Guide.html

JS Lab

SDN/NFV for 5G

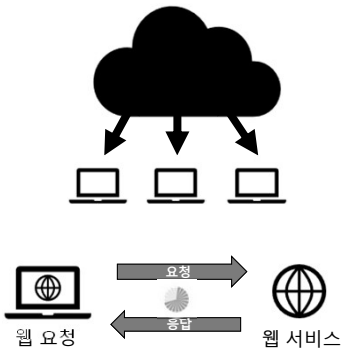
james@jslab.kr

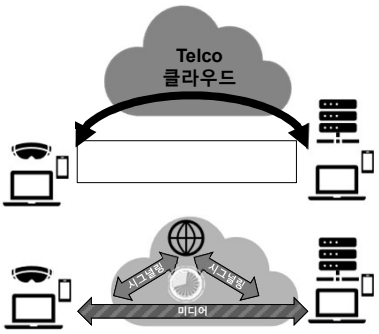
56

III. 가상화와 클라우드 서비스

❖ 5G 코어 인프라 기반 NFV의 클라우드화

- Public Cloud: 소프트웨어 정의 가상 인프라 기반 서비스 (웹서비스)
- Telco Cloud: 언더레이 인프라 기반 클라우드 서비스 (전송경로 제공)
- Telco Cloud는 하드웨어 인프라 환경 고려





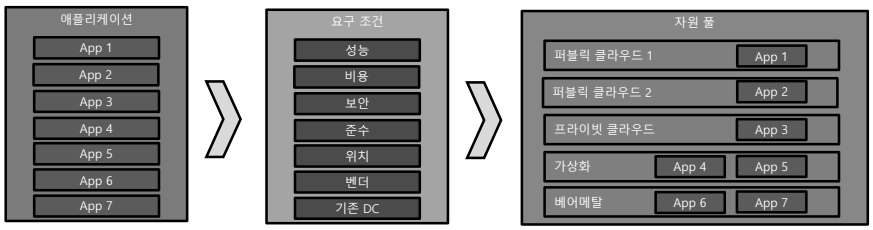
실시간 멀티미디어 애플리케이션
(고화질 스트리밍, 고화질 게임, AR, VR 등을 위한 경로)

SDN/NFV for 5G
james@jslab.kr
JS Lab

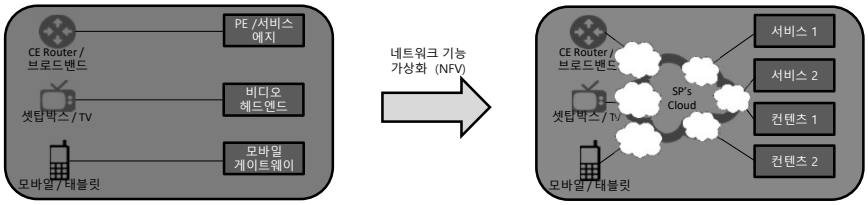
57

III. 가상화와 클라우드 서비스

❖ 엔터프라이즈는 애플리케이션 요구에 적합한 클라우드 자원



❖ 텔코 (Telco)는 네트워크 기능 가상화 기반 데이터센터화

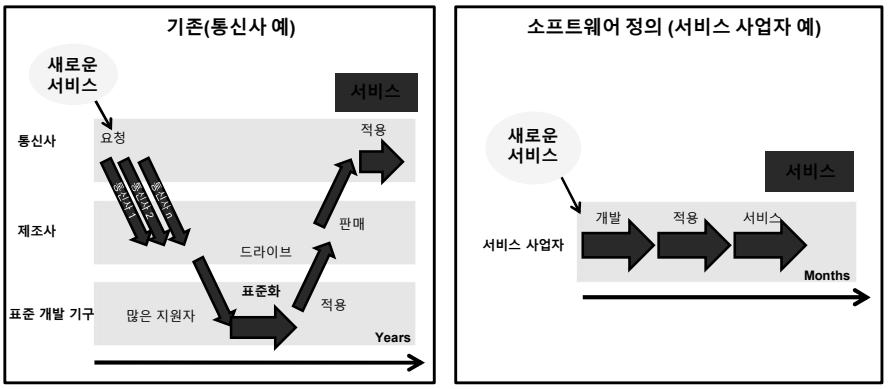


SDN/NFV for 5G
james@jslab.kr
JS Lab

58

III. 가상화와 클라우드 서비스

- ❖ 통신사와 서비스 사업자의 적용 환경
- ❖ 개발능력 미보유 엔터프라이즈는 서비스 제공 가능 오픈소스나 제조사의 SDx 솔루션 필요



III. 가상화와 클라우드 서비스

- ❖ Edge vs Core @ Telco

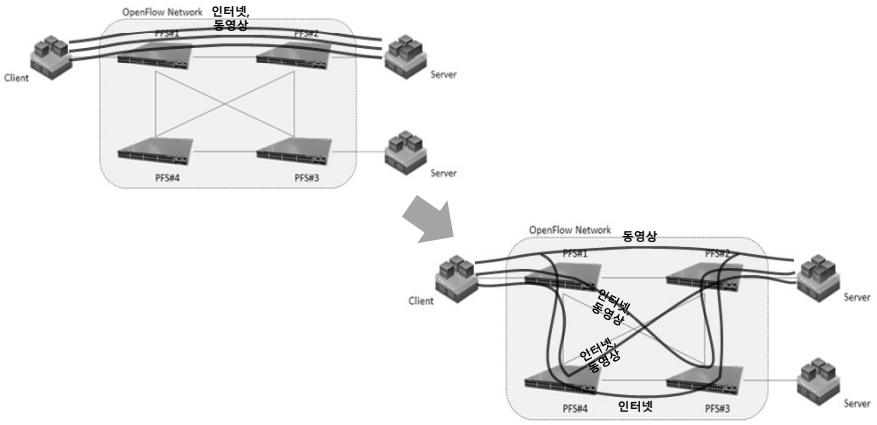
	Edge (에지) 클라우드	중앙 클라우드
App의 위치	노드의 물리적 위치에서 중요한 서비스	비교적 위치와 독립적인 서비스
워크로드의 이동성	워크로드가 노드간 이동	클라우드 노드 장애 이외에는 비교적 고정
워크로드의 역동성	다양한 App들이 다양한 시간에 크게 다른 요구를 함	서비스를 적용하면 대부분의 시간에 안정적인 워크로드
아키텍쳐	다른 형태의 많은 수의 노드와 다양한 용량과 기술	대부분 동일하며 차이가 작음 (예: AWS, OpenStack, Azure 등)
지연	지연과 거리는 종단 사용자들을 위한 주요 역할	대부분 지연에 민감하지 않음
자원 가용성	에지노드는 작고, App을 위한 자원의 가용성을 보장하지 않음	가용성 확보는 중요하며 주요 기능 중 1개

III. 가상화와 클라우드 서비스

❖ **가상 네트워크 구성**

① **실제 네트워크**

② **가상 네트워크**



SDN/NFV for 5G

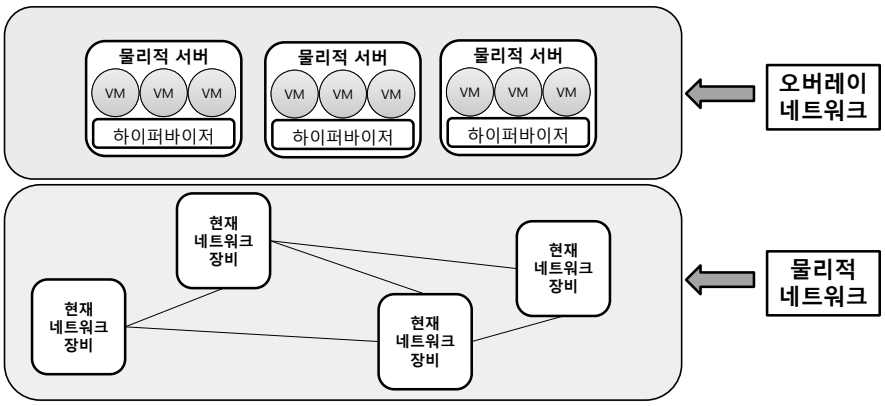
james@jslab.kr

JS Lab

61

III. 가상화와 클라우드 서비스

❖ **오버레이 네트워킹** (터널링 프로토콜 이용 가능)



SDN/NFV for 5G

james@jslab.kr

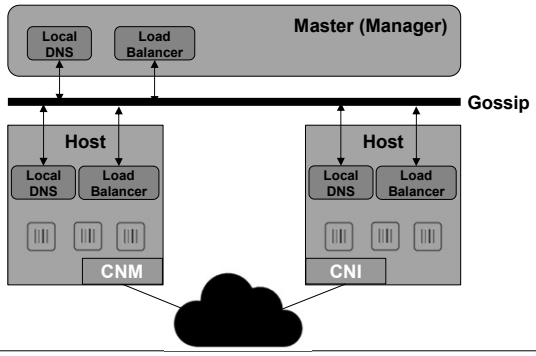
JS Lab

62

III. 가상화와 클라우드 서비스

❖**Container Networking Basics - CNM / CNI**

- **Container Network Model (CNM):** Docker libnetwork에서 사용하며 Cisco Contiv, Kuryr, OVN, Project Calico, Vmware, Vwave에서 사용
- **Container Network Interface (CNI):** CoreOS, K8s, Kurma, rkt, Apache Mesos, Cloud Foundry, Cisco Contiv, Project Calico, Weave, Docker
- **Gossip:** P2P, 일정한 개수의 불특정 노드에 정보를 gossiped, 지정한 오버레이를 통해 다른 노드에 알림

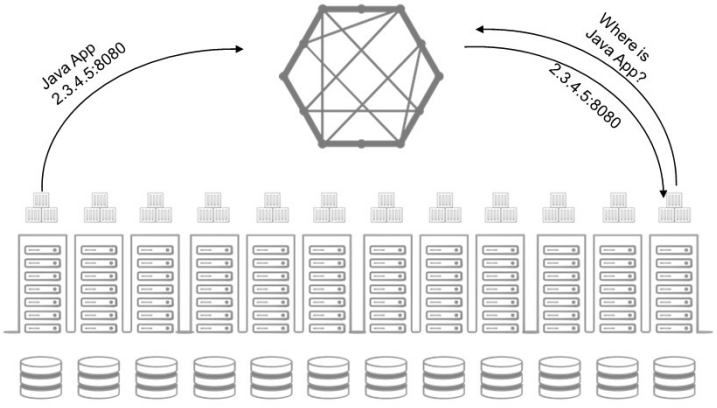


63

III. 가상화와 클라우드 서비스

❖ **서비스 디스커버리 (Service Discovery)**

- **앱들과 인프라가 자동으로 서로의 필요 부분을 찾아 연결**
- **애플리케이션 서비스 트래킹을 지속하여 연결하여 사용**



64

III. 가상화와 클라우드 서비스

❖컨테이너 네트워킹(Container Networking) 종류

- None: 호스트간 연결 없음
- 브릿지(Bridge): L2 브릿지를 사용
- 오버레이(Overlay): 터널링 사용 오버레이로 호스트 간 네트워크 연결
- 언더레이(Underlay): 컨테이너를 물리적 인터페이스에 직접 연결

Docker (예)	오버레이	브릿지 / 포트맵핑	언더레이
Multi-hiost	• Yes	• No (native support)	• No (native support)
SD (Service Discovery)	• 클러스터 간의 글로벌 SD	• 호스트 네트워크 상의 로컬 SD(서비스 디스커버리)	• 호스트 네트워크 상의 로컬 SD(서비스 디스커버리)
로드밸런싱	• 내부 글로벌 VIP 기반 • 내부 글로벌 DNS 기반 • 외부 라우팅 메쉬	• 내부 로컬 DNS 기반	• 내부 로컬 DNS 기반
IP Addressing	• 컨테이너 당 내부 주소체계 • 오버레이당 글로벌 범위	• 컨테이너 당 내부 주소체계 • 브릿지당 로컬 범위	• 물리 네트워크 상의 컨테이너당 외부 주소 체계
암호화	• Yes, 선택	• No	• No
요구사항	• 엔진 1.12 이상 클러스터 스웜(Swarm)모드	• 엔진 1.7 이상	• 호스트 인터페이스에 Promiscuous mode 필요

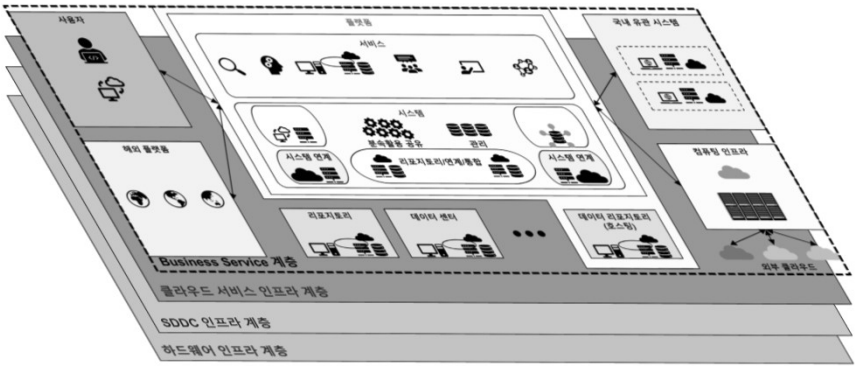
JS Lab

65

III. 가상화와 클라우드 서비스

❖ 클라우드 네이티브 방향의 진화

- Business Service 계층
- 클라우드 서비스 인프라 계층
- SDDC 인프라 계층
- 하드웨어 인프라 계층



JS Lab

66

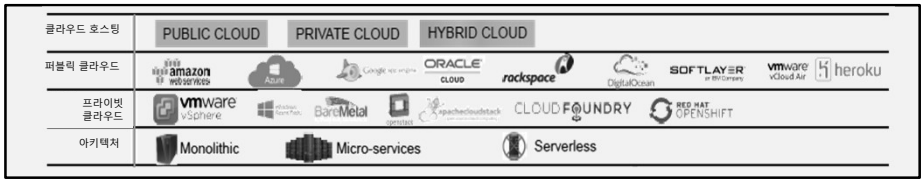
III. 가상화와 클라우드 서비스

❖ 클라우드 환경 컨테이너 네트워킹 연결 방법

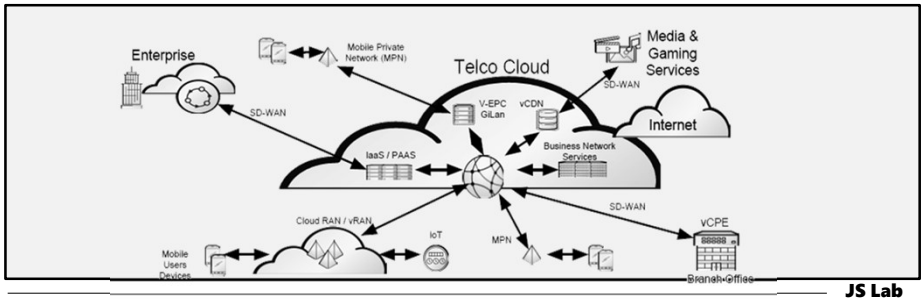
Network Type	Backend	기능
Overlay	VxLAN	• Private 보안 환경에 적합 • Inbterhost 암호화가 아님 • 구현이 쉬우나 대규모 구성 시 성능 이슈 발생 가능
	IPSec	• Interhost 암호화 • 인터넷 경유에 적합
	MPLS	• Linux 4.x 이상에서 구성 가능 • 성능이 Raw 수준에 가깝게 우수하나 구성/관리 어려움
	VLAN	• Private 보안 환경에 적합 • 확장성 한계
Non-overlay	Host-Gateway	• 성능 우수 • 특정 클라우드와 독립적으로 구성
	AWS VPC	• 성능 우수 • 아마존 클라우드 내에서만 구성가능

III. 가상화와 클라우드 서비스

❖ **클라우드:** 서비스 요구에 적합한 클라우드 자원 (네트워크와 함께 가상머신, 컨테이너, 서버리스, 스토리지, 인프라 제공)

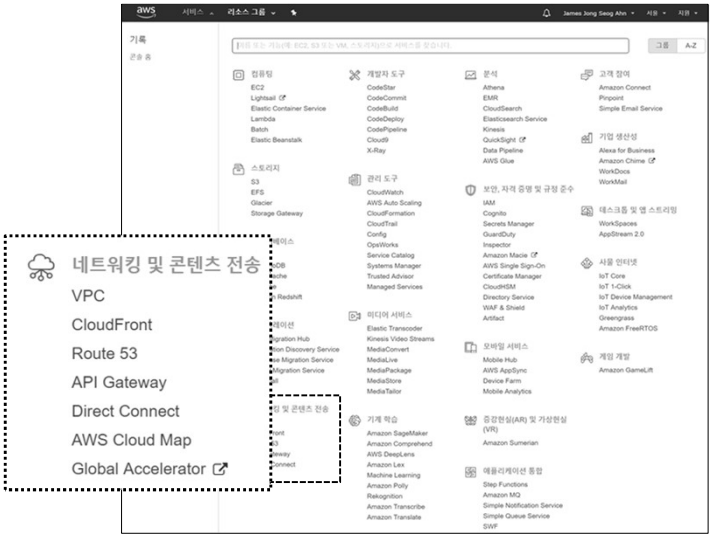


❖ **텔코(Telco) 클라우드:** 국사(Central Office)의 데이터센터화



III. 가상화와 클라우드 서비스

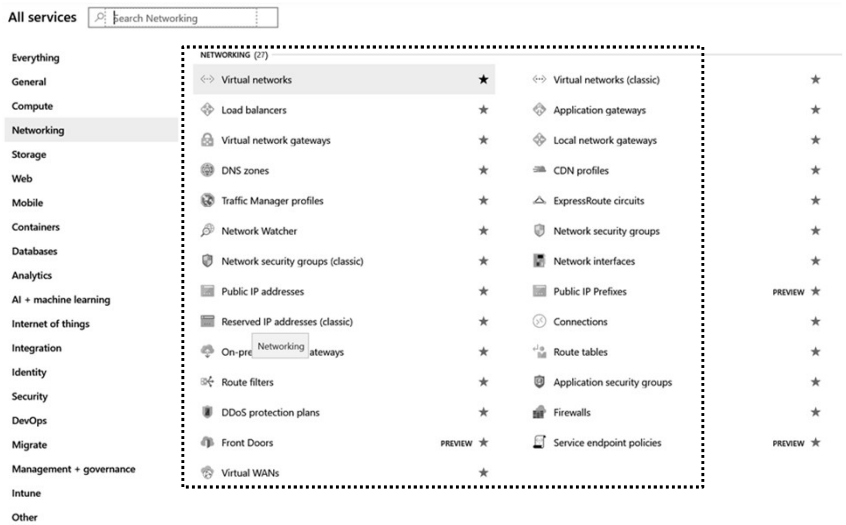
❖ 아마존 AWS



https://ap-northeast-2.console.aws.amazon.com/console/home?region=ap-northeast-2

III. 가상화와 클라우드 서비스

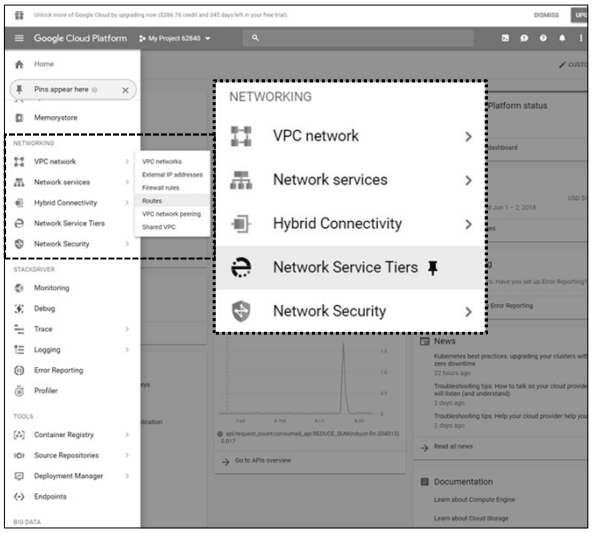
❖ 마이크로소프트 Azure



https://portal.azure.com

III. 가상화와 클라우드 서비스

❖ **Google Cloud Platform**



https://console.cloud.google.com/home/dashboard?project=robust-fin-204013

71

III. 가상화와 클라우드 서비스

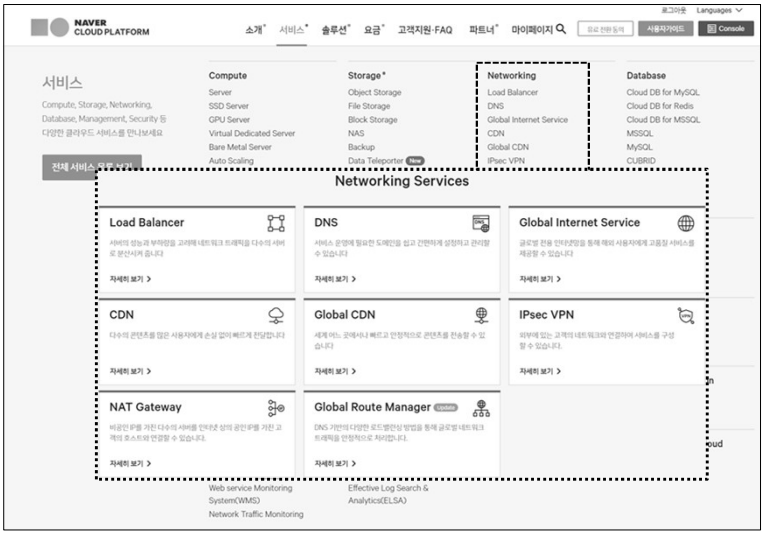
❖ **IBM Cloud**

https://console.bluemix.net/catalog/

72

III. 가상화와 클라우드 서비스

❖ **Naver Cloud Platform**



https://www.ncloud.com/

JS Lab

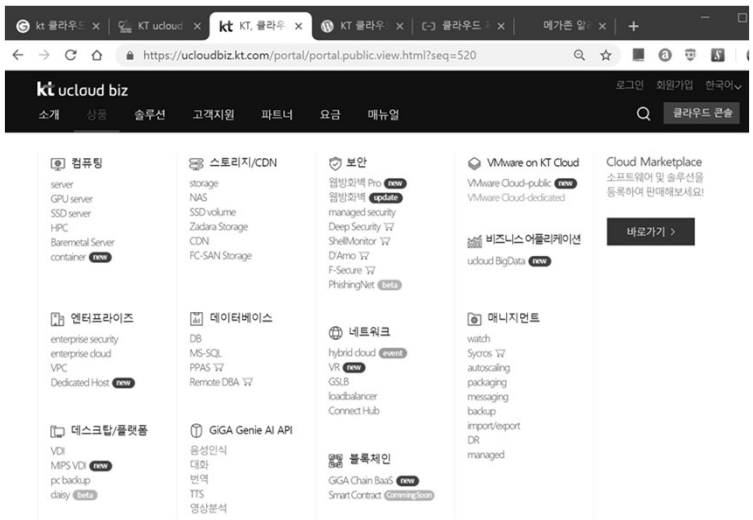
SDN/NFV for 5G

james@jslab.kr

73

III. 가상화와 클라우드 서비스

❖ **KT Cloud**



https://ucloudbiz.kt.com/portal/portal.public.view.html?seq=520

JS Lab

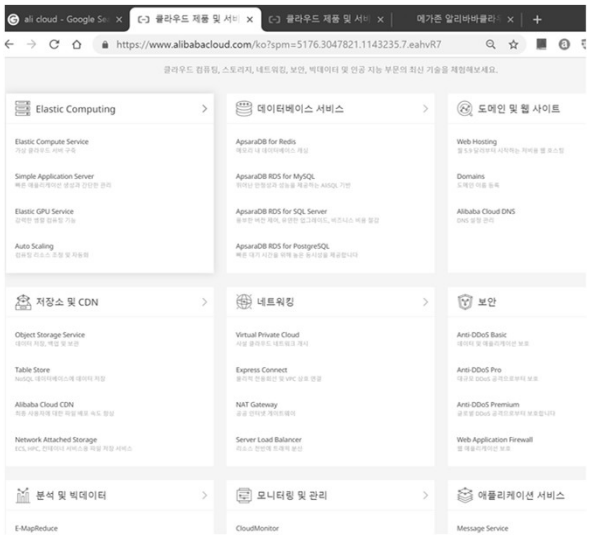
SDN/NFV for 5G

james@jslab.kr

74

III. 가상화와 클라우드 서비스

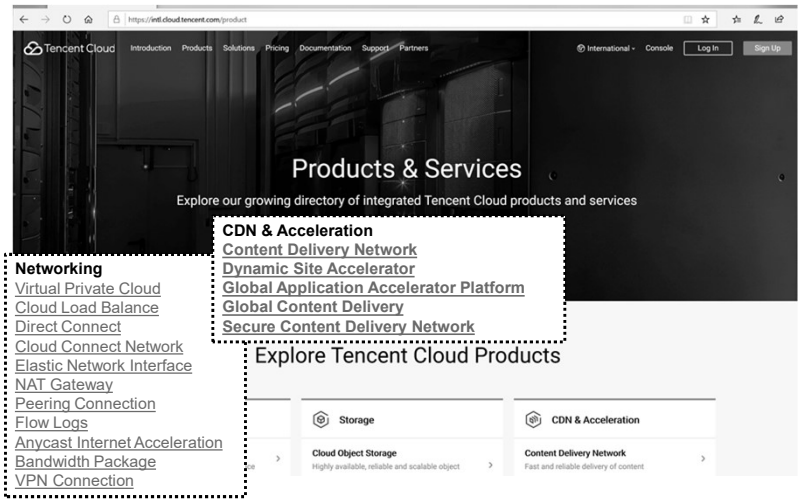
❖ **Alibaba Cloud**



https://www.alibabacloud.com/ko?spm=5176.3047821.1143235.7.eahvR7

III. 가상화와 클라우드 서비스

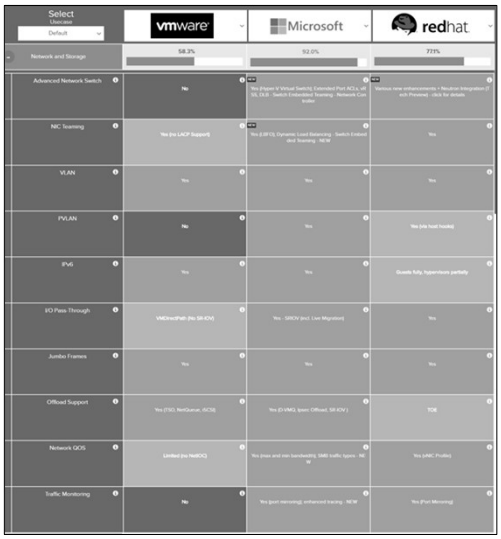
❖ **Tencent Cloud**



https://intl.cloud.tencent.com/

III. 가상화와 클라우드 서비스

❖ 프라이빗 클라우드 비교



https://www.whatmatrix.com/comparison/Virtualization

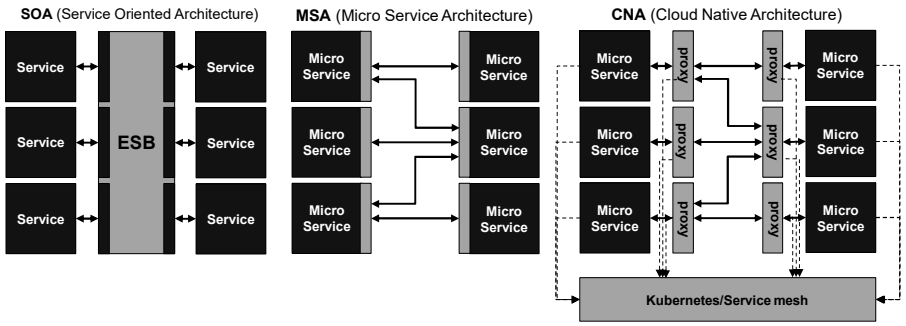
77

III. 가상화와 클라우드 서비스

❖User Plane – Control Plane 분리 : SDN Architecture

❖컨테이너 기반 아키텍처

- **SOA** (Service Oriented Architecture): Smart pipes, dump endpoints
- **MSA** (Micro Service Architecture): Smart endpoints, dump pipes
- **CNA** (Cloud Native Architecture): Infrastructure focused smart platform, business logic focused smart services



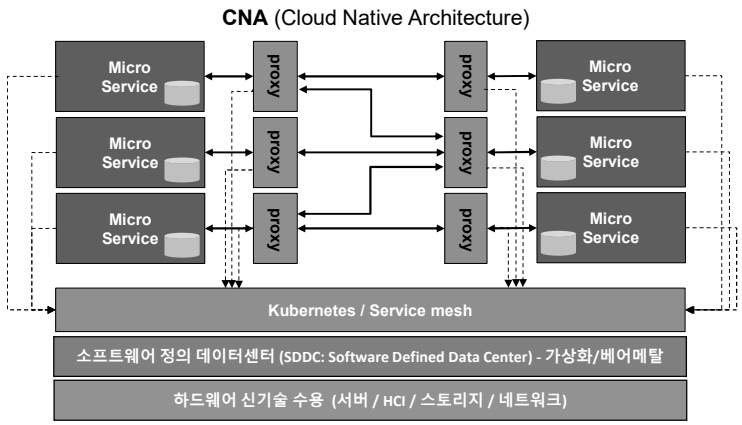
ESB(Enterprise Service Bus)

78

III. 가상화와 클라우드 서비스

❖ 클라우드 네이티브 아키텍처(Cloud Native Architecture)

- MSA(마이크로서비스 기반) 수용 클라우드 네이티브 기술 발전 수용 체계
- 온프레미스(On-premises) 인프라를 위한 SDDC 기반 데이터센터



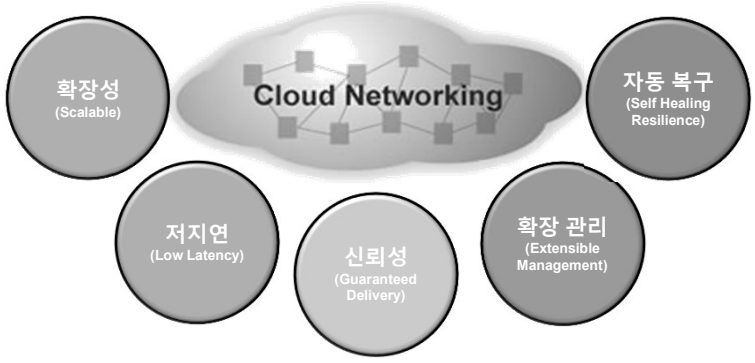
james@jslab.kr

JS Lab

79

III. 가상화와 클라우드 서비스

❖ 클라우드 네트워킹 이슈



SDN/NFV for 5G
james@jslab.kr

JS Lab

80

SDN/NFV for 5G

james@jslab.kr

I. 개요
II. 소프트웨어 정의
III. 가상화와 클라우드 서비스
IV. SDN 개요
V. NFV
VI. 오버레이 / 언더레이
VII. SDN 관련 기술
VIII. 클라우드 네트워크
IX. 텔레콤 환경을 위한 SDN/NFV
X. 관리

❖ 부록: OpenFlow
❖ 실습교재 (별도)

JS Lab

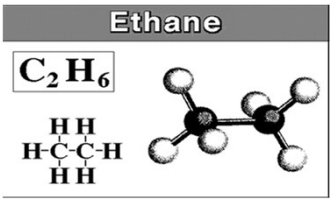
81

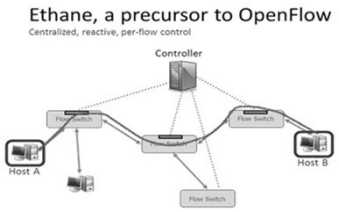
SDN/NFV for 5G

james@jslab.kr

IV. SDN 개요

❖ **Ethane의 시작:** Clean Slate 연구팀, 2006년 가을, 미국 스탠포트대학교






Ethane: Taking Control of the Enterprise


Scott Shenker
UC버클리대학교 교수
니시라 창업자


Nick Mckeown
스탠포트대학교 교수
니시라 창업자


Martin Casado
스탠포드대학교 Lab
니시라 창업자

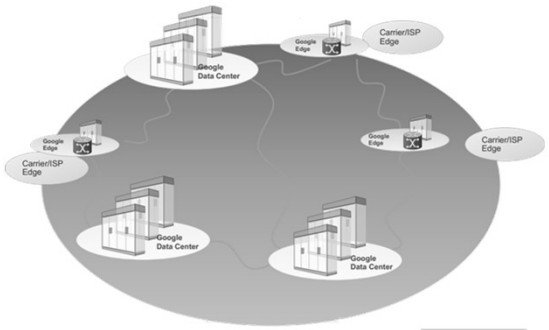
참조. http://cleanslate.stanford.edu/

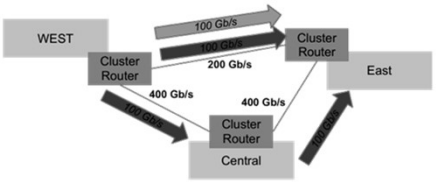
JS Lab

82

IV. SDN 개요

❖ 구글: 2012년 4월 ONS에서 발표



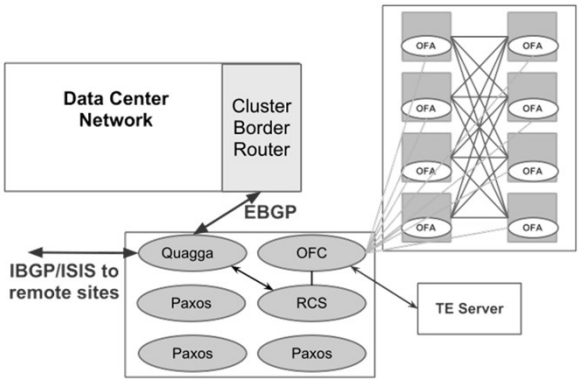


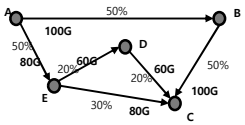
SDN/NFV for 5G
james@jslab.kr
JS Lab

83

IV. SDN 개요

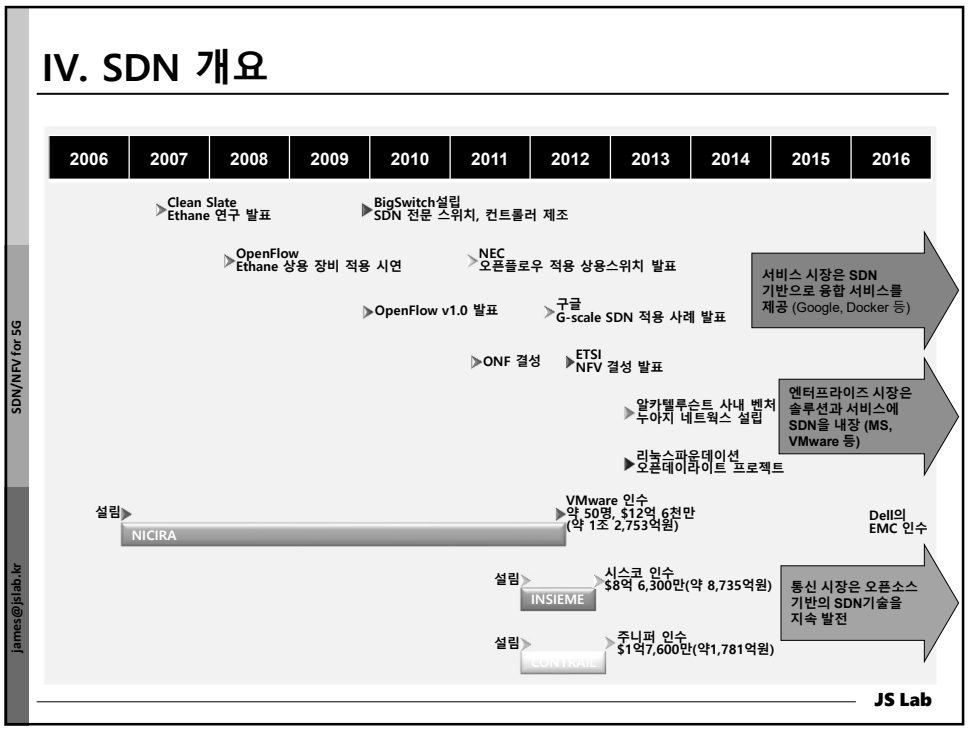
❖ 구글
- 하드웨어 자체 제작
- 오픈소스 소프트웨어 사용
- 단계별 도입



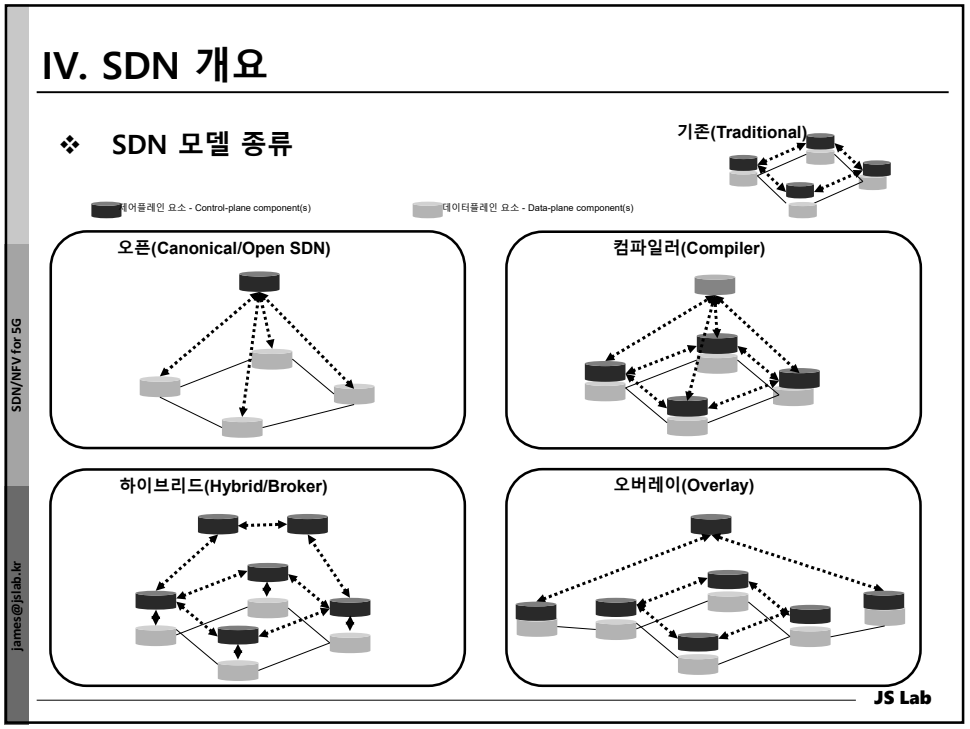


SDN/NFV for 5G
james@jslab.kr
JS Lab

84


IV. SDN 개요
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
Clean Slate
Ethane 연구 발표
BigSwitch설립
SDN 전문 스위치, 컨트롤러 제조
OpenFlow
Ethane 상용 장비 적용 시연
NEC
오픈플로우 적용 상용스위치 발표
서비스 시장은 SDN 기반으로 융합 서비스를 제공 (Google, Docker 등)
OpenFlow v1.0 발표
구글
G-scale SDN 적용 사례 발표
ONF 결성
ETSI
NFV 결성 발표
알카텔루슨트 사내 벤처 누아지 네트웍스 설립
엔터프라이즈 시장은 솔루션과 서비스에 SDN을 내장 (MS, VMware 등)
리눅스파운데이션 오픈데이라이트 프로젝트
설립
NICIRA
VMware 인수
약 50명, $12억 6천만 (약 1조 2,753억원)
Dell의 EMC 인수
설립
INSIEME
시스코 인수
$8억 6,300만(약 8,735억원)
통신 시장은 오픈소스 기반의 SDN기술을 지속 발전
설립
CONTRAIL
주니퍼 인수
$1억7,600만(약1,781억원)
SDN/NFV for 5G
james@jslab.kr
JS Lab

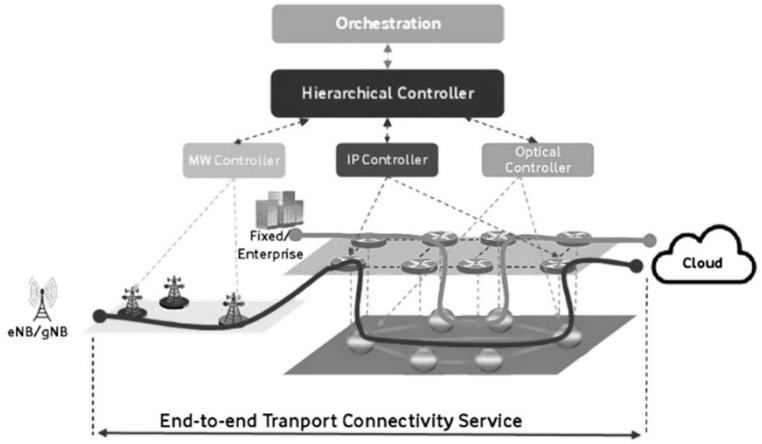
85


IV. SDN 개요
❖ SDN 모델 종류
기존(Traditional)
제어플레인 요소 - Control-plane component(s)
데이터플레인 요소 - Data-plane component(s)
오픈(Canonical/Open SDN)
컴파일러(Compiler)
하이브리드(Hybrid/Broker)
오버레이(Overlay)
SDN/NFV for 5G
james@jslab.kr
JS Lab

86

IV. SDN 개요

❖ SDN 기반 전송계층 아키텍처
- Packet-Optical 멀티레이어 구성의 레이어별 컨트롤러 사용
- 중앙 집중 오케스트레이터

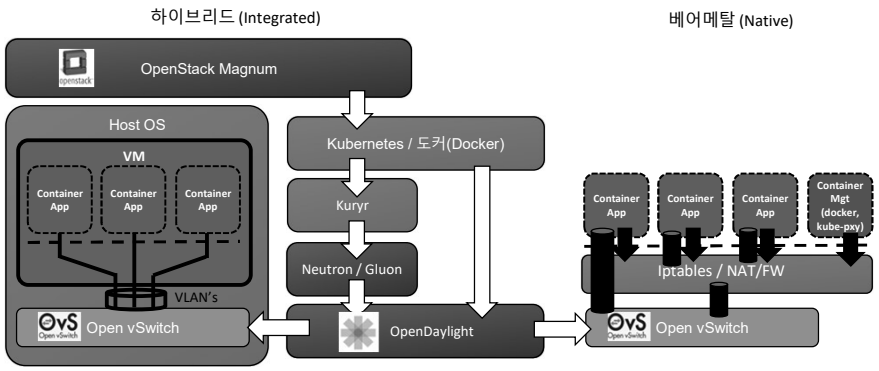


출처: https://metro-haul.eu/2019/05/24/ietf-tools-for-management-and-operation-of-flexible-optical-metro-networks-in-support-of-5g/

87

IV. SDN 개요

❖ ODL의 컨테이너 네트워킹을 위한 2가지 적용 방법
- 하이브리드
- 베어메탈

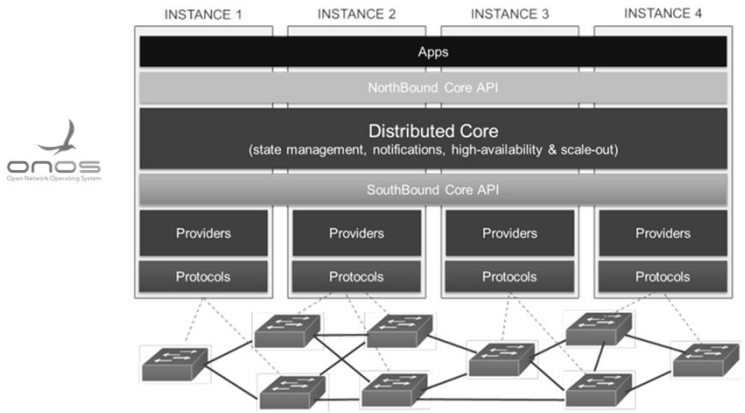


88

IV. SDN 개요

❖ **ONOS : Open Network Operating System**

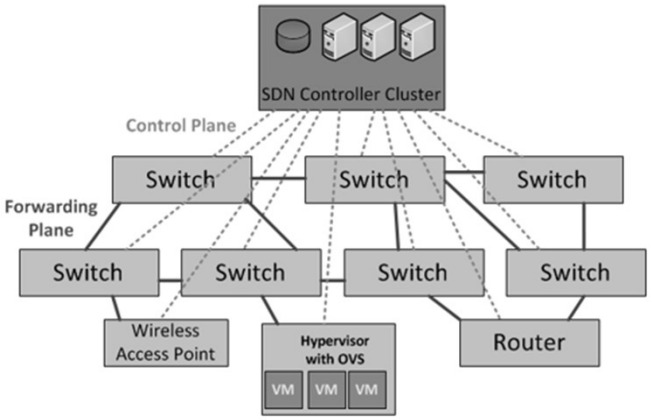
① 통신사 수준의 실제 활용 가능한 개방형 SDN 컨트롤러
② 분산형 구조 기반의 SDN 컨트롤러
③ 3개월마다 주기적으로 출시 (알파벳 순서에 따른 새 이름으로 버전 정보 추가)
④ 국내 참여



89

IV. SDN 개요

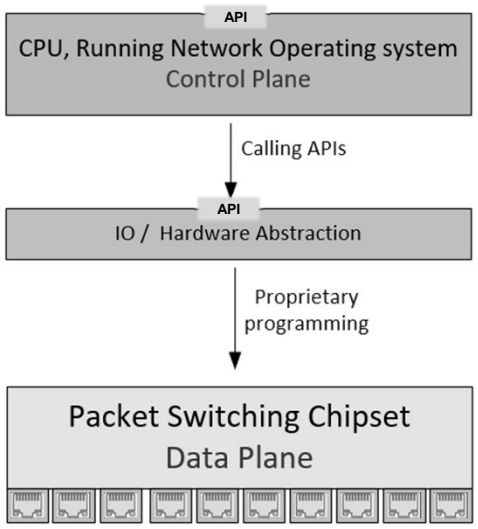
❖ **SDN Controller Cluster**
❖ **Forwarding Plane** (Wired/Wireless/Hypervisor)



90

IV. SDN 개요

❖ **Communication between the Control Plane and the Data Plane**



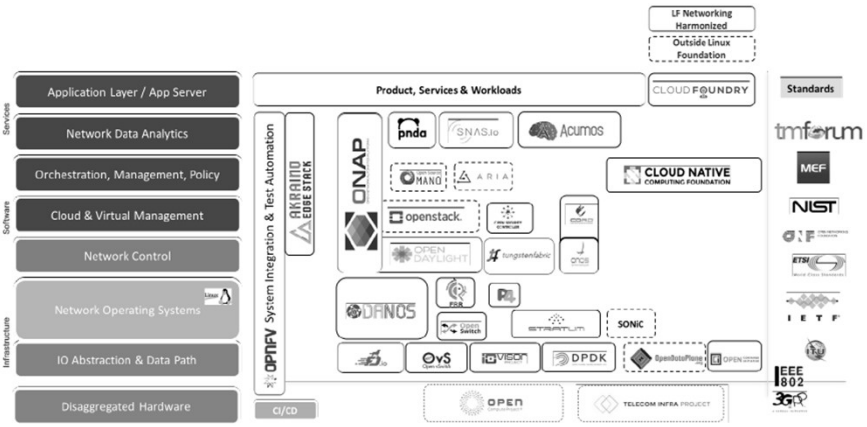
SDN/NFV for 5G
james@jslab.kr
JS Lab

91

IV. SDN 개요

❖ **Linux Foundation의 SDN**

❖ **Open Source and Software Defined Networking Landscape**

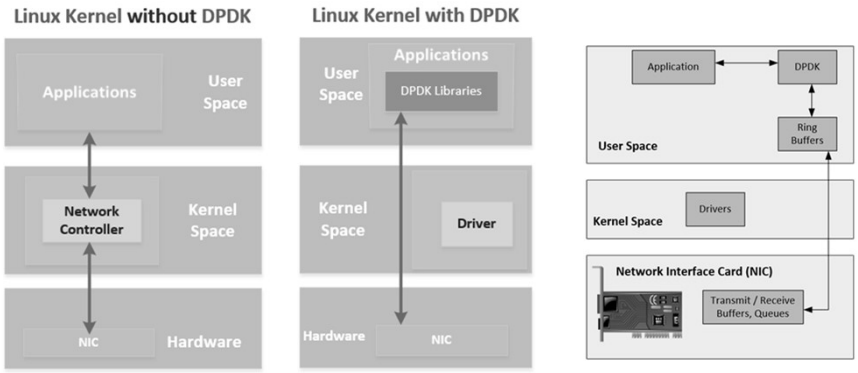


SDN/NFV for 5G
james@jslab.kr
JS Lab

92

IV. SDN 개요

❖ **DPDK Architecture**

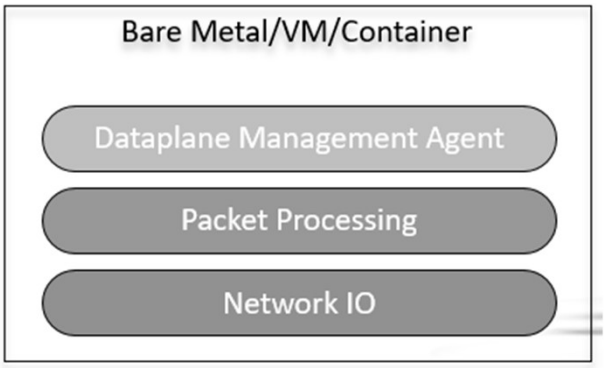


93

IV. SDN 개요

❖ **FD.io Main Components**

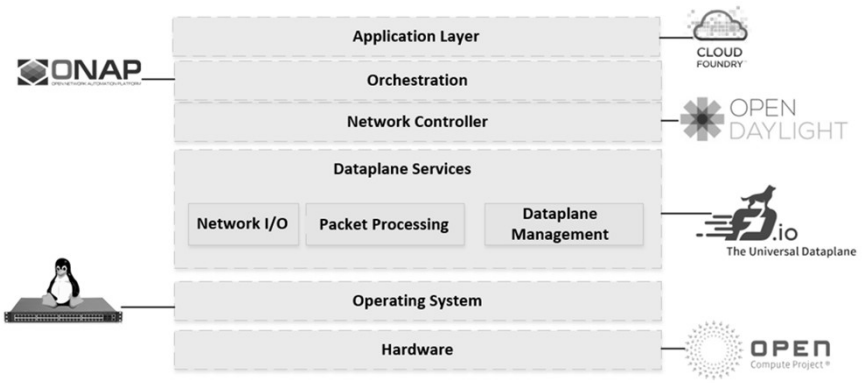
- Data Plane Management Agent
- Packet Processing
- Network IO



94

IV. SDN 개요

❖ **FD.io Communication with Other Networking Subsystems**

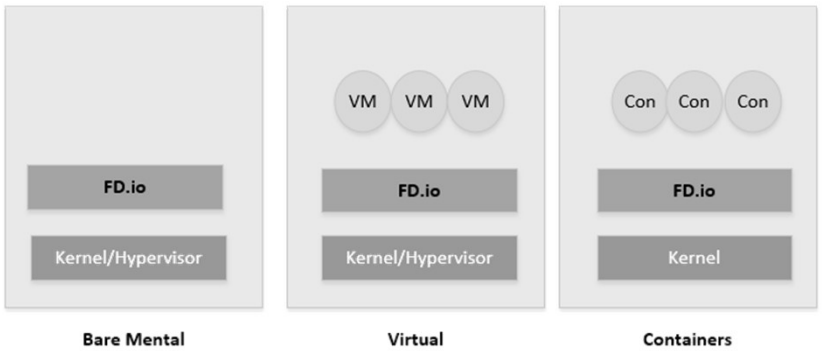


95

IV. SDN 개요

❖ **FD.io can be used in servers to provide data plane functions to:**

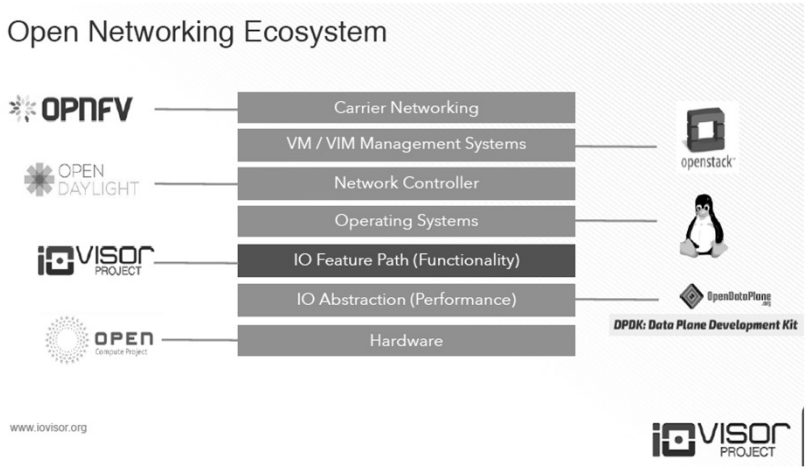
- Bare metal servers directly
- Virtual machines
- Containers.



96

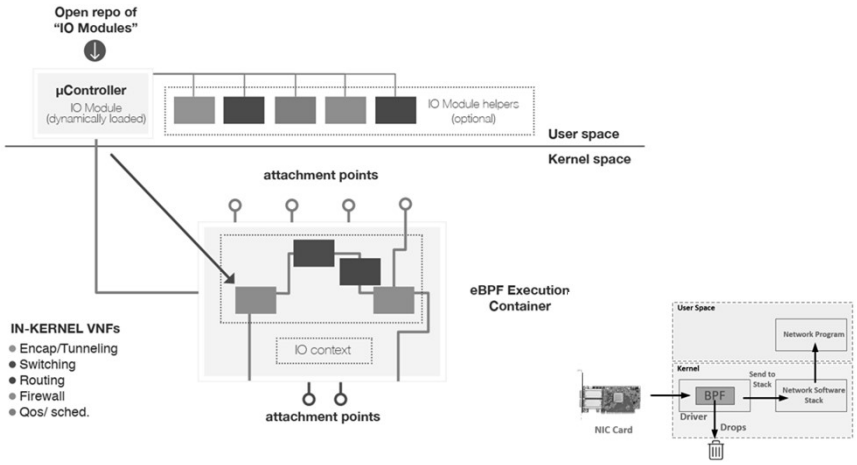
IV. SDN 개요

❖ **The Open Networking Ecosystem**
(by IO Visor Project, retrieved from the IO Visor website)



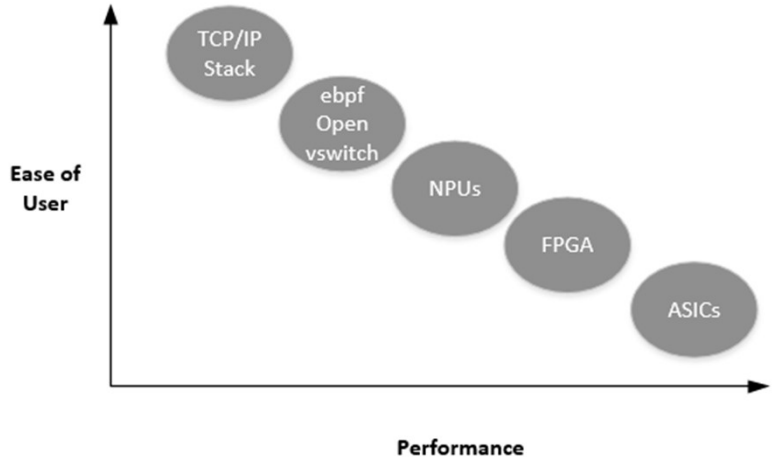
IV. SDN 개요

❖ **eBPF Framework for Networking**
(by IO Visor, retrieved from iovisor.org)



IV. SDN 개요

❖ **Flexibility vs Performance in Different Platforms**

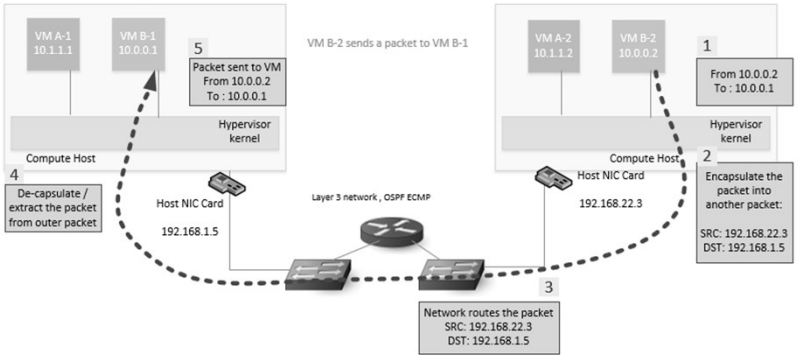


SDN/NFV for 5G
james@jslab.kr
JS Lab

99

IV. SDN 개요

❖ **VMware의 Basic Protocol-Independent Encapsulation and Overlay using VXLAN**

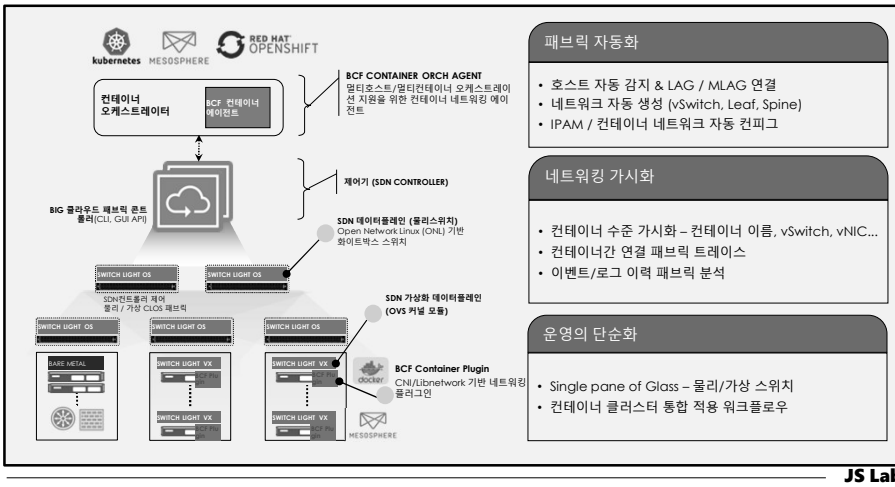


SDN/NFV for 5G
james@jslab.kr
JS Lab

100

IV. SDN 개요

- ❖ BigSwitch/오픈스택(OpenStack)의 뉴트론(Neutron)연동 (예)
- ❖ 분산라우팅, Heat, LBaaS, 방화벽, VM-to-VM 경로/정책 가시화
- ❖ 프라이빗 클라우드를 위한 하드웨어 플랫폼

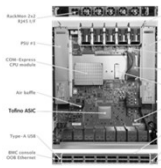


101

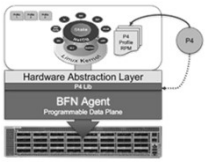
IV. SDN 개요

- ❖ Tofino w/P4

Intel(Baerfoot Networks)

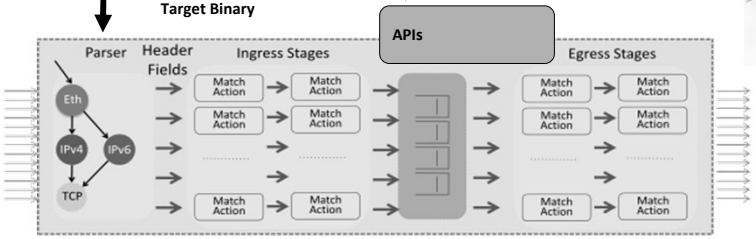


Edgecore



Arista

XILINK



102

IV. SDN 개요

❖ **P4 Language:**

Headers

```
header ethernet_t {
 macAddr_t dstAddr;
 macAddr_t srcAddr;
 bit<16>  etherType;
}
header ipv4_t {
 bit<4>   version;
 bit<4>   ihl;
 bit<8>   diffserv;
 bit<16>  totalLen;
 bit<16>  identification;
 bit<3>   flags;
 bit<13>  fragOffset;
 bit<8>   ttl;
 bit<8>   protocol;
 bit<16>  hdrChecksum;
 ip4Addr_t srcAddr;
 ip4Addr_t dstAddr;
}
```

Parsers

```
parser MyParser(packet_in packet,
  out headers hdr,
  inout metadata meta,
  inout standard_metadata_t std_meta) {
 state start {
   packet.extract(hdr.ethernet);
   transition accept;
  }
}
```

Tables

```
table ipv4_lpm {
  key = {
   hdr.ipv4.dstAddr: lpm;
  }
  actions = {
   ipv4_forward;
   drop;
   NoAction;
  }
  size = 1024;
  default_action = NoAction();
}
```

Actions

```
control MyIngress(inout headers hdr,
          inout metadata meta,
          inout standard_metadata_t std_meta) {
 action swap_mac(inout bit<48> src,
          inout bit<48> dst) {
  bit<48> tmp = src;
    src = dst;
    dst = tmp;
  }
 apply {
    swap_mac(hdr.ethernet.srcAddr,
             hdr.ethernet.dstAddr);
    std_meta.egress_spec =
std_meta.ingress_port;
  }
}
```

SDN/NFV for 5G

james@jslab.kr

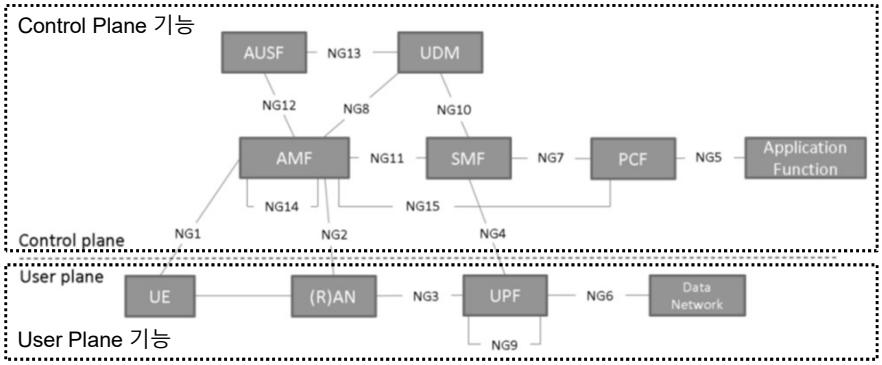
JS Lab

103

IV. SDN 개요

❖ **3GPP proposed architecture and reference points for 5G networks**

❖ **User Plane – Control Plane Split : SDN Architecture**



- User Equipment (UE);
- (Radio) Access Network (RAN);
- User Plane Function (UPF);
- Access and Mobility Function (AMF);
- Session Management Function (SMF);
- Policy Control Function (PCF);
- Authentication Server Function (AUSF);
- User Data Management (UDM);

5G-PPP Software Network Working Group

SDN/NFV for 5G

james@jslab.kr

JS Lab

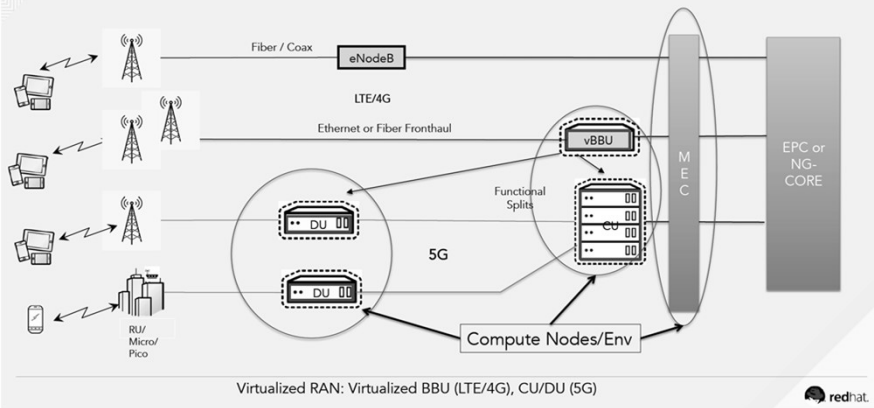
104

SDN/NFV for 5G

james@jslab.kr

IV. SDN 개요

❖ 가상화 RAN (Radio Access Network)
❖ IT 기술 적용 가능 환경



JS Lab

CU (Central Unit), **DU**(Distributed Unit) **RU**(Remote Unit)

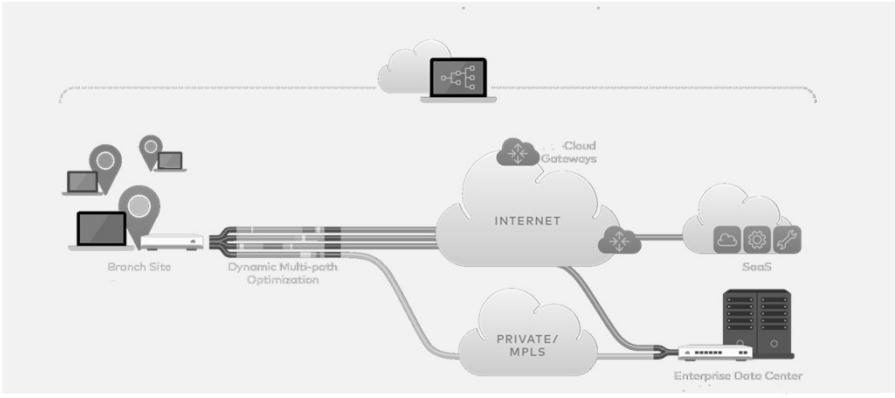
105

SDN/NFV for 5G

james@jslab.kr

IV. SDN 개요

❖ Software Defined WAN
① 가시화 관리 제공 컨트롤러
② 종단간 오버레이 구성
③ GPS / LTE 등을 장착한 지사 기기
④ 퍼블릭/프라이빗 클라우드 접속 지원



JS Lab

106

IV. SDN 개요

❖ SDN 데이터플레인 오픈소스 프로젝트

Project	Description
DPDK	A hardware abstraction library which can be used for fast communication with supported NIC cards. The DPDK library allows robust communication between kernel and NIC cards with less CPU cycles.
FD.io	A set of libraries in the user space that can help you build packet processing applications. FD.io can use hardware acceleration and DPDK to provide a high performance packet processing. FD.io uses the VPP method, which processes an array of packets at the same time.
IO Visor	Similar to FD.io, but it's a set of libraries that provide packet processing features within the kernel.
OpenDataPlane	A hardware abstraction for supported packet processing SoCs. The library provides standard, common APIs which can be used by applications to perform network tasks without the need to consider the hardware. This library supports embedded chips from TI, Marvell, Freescale Semiconductor, etc.
Barefoot Tofino	An Ethernet switch chipset which supports data plane abstraction and programming using the P4 language. Tofino is currently the only switch silicon that supports data plane programming and packet processing for multiple 100G ports.
SmartNICs	NIC cards that allow you to use them for packet processing. You can load a packet switching or modification program on the SmartNIC chipset itself (without even using the system CPU) and let the SmartNIC perform. You can use the SmartNICs for many off-loading functions such as encapsulation (GRE, VXLAN, IPSec, etc.) or other packet classification functions.
Xilinx SDNet	As one of the major FPGA providers, Xilinx is also offering FPGA chipsets that can be used for data plane and fast packet processing. Initially, Xilinx provided SDNet, which is their proprietary SDN for building data plane applications. Later, they added support for the standard P4 language. You can use Xilinx FPGA to build high performance processing.
Open Container Initiative	An open source project for governance and standardizing container formats and runtimes. OCI has defined the specification for filesystem bundle (the open specification for exporting and storing container files) and the runtime bundle (the open specification for running a container).

107

IV. SDN 개요

❖ SDN 콘트롤 플레인

Project/Product	License Type	SDN Method	Main Use Case
OpenDaylight	Open Source	Direct Fabric Programming	Cloud, datacenter, multi-tenant
ONOS	Open Source	Direct Fabric Programming	Telco, service providers
Tungsten Fabric	Open Source	Overlay	Cloud, datacenter, multi-tenant
Project Calico	Open Source	Overlay	Cloud, datacenter, multi-tenant
Cisco ACI	Commercial	Direct Fabric Programming	Cross sites, cloud, datacenter, multi-tenant
Floodlight	Open Source	Direct Fabric Programming	Cloud, datacenter, multi-tenant
Big Cloud Fabric	Commercial	Direct Fabric Programming	Cloud, datacenter, multi-tenant

108

IV. SDN 개요

❖ 제조사 네트워킹 솔루션의 오케스트레이션 연동

- SDN 기반 클라우드 네이티브(Cloud Native)화 데이터센터 네트워킹 분야 집중
- 컨테이너, 서비스 메쉬, 마이크로서비스, 변경 불가능 인프라(Immutable Infrastructure) 및 선언적 API를 사용하는 접근 방식

제조사	솔루션 이름	오케스트레이션 연동
Big Switch Networks	Big Cloud Fabric	쿠버네티스, 오픈스택, VMware, OpenShift
Huawei	CloudFabric	오픈스택, FusionSphere, ManageOne, Red Hat, Mirantis
Lenovo	RackSwitch	오픈스택, VMware vCloud Suite, Microsoft Azure Stack, Tungsten Fabric
Netronome	Agilio SmartNIC	오픈스택
Plexxi	Plexxi HCN	쿠버네티스, 오픈스택, vCloud, Nutanix
ZTE	ZENIC	쿠버네티스, 오픈스택
Dell EMC	Z9100-ON Switch	쿠버네티스, 오픈스택, VMware vCloud Suite
Altoline	99xx/69xx	쿠버네티스, 오픈스택, VMware vCloud Suite
Mellanox	Open Composable Networks	오픈스택, VMware vCloud Suite, NEO
Cisco	Application Policy Infrastructure Controller (APIC)	VMware vCloud Suite
Ericsson	Cloud SDN	쿠버네티스, 오픈스택

제조사	솔루션 이름	오케스트레이션 연동
Juniper Networks	Contrail	오픈스택
Nuage Networks	Virtualized Services Platform(VSP)	쿠버네티스, 오픈스택, VMware vCloud Suite, 클라우드스택
Pluribus	Netvisor OS, Adaptive Cloud Fabric	VMware vCloud Suite, Ansible, Puppet, Chef
FlowEngine	FlowEngine TDE-2000	오픈스택, VMware vCloud Suite
Red Hat	NFV solution	쿠버네티스, 오픈스택
VMware	NSX	쿠버네티스, 오픈스택, VMware vCloud Suite
Wind River	Titanium Cloud	오픈스택
A10	Thunder ADC	오픈스택, VMware vCloud Suite
Cumulus	Cumulus Linux	오픈스택
ipinfusion	OcNOS	오픈스택
Pulse Secure	Pulse Access Suite	쿠버네티스, 오픈스택, VMware vCloud Suite

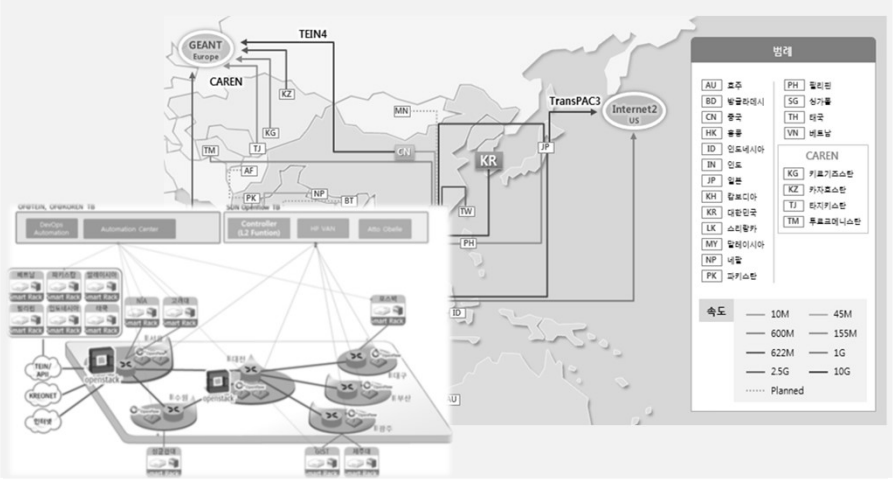
IV. SDN 개요

❖ 적용 사례: 삼정데이터서비스
❖ 데이터센터 운영 : IDC 센터 2개
❖ SDN 제어기 오픈소스: https://github.com/superkkt/cherry
❖ SDN 도입의 이점
- 중앙 집중화된 네트워크 관리 시스템이 필요
- ARP & IP spoofing 등의 보안 위협 원천 차단
- 효율적인 네트워크/시스템 관리

❖ 각 분야의 자동화
- 네트워크 인프라: SDN (OpenFlow)
- OS: 자동 설치/설정 (Cobbler, Puppet)
- 서비스 운영: 컨테이너 (Docker)

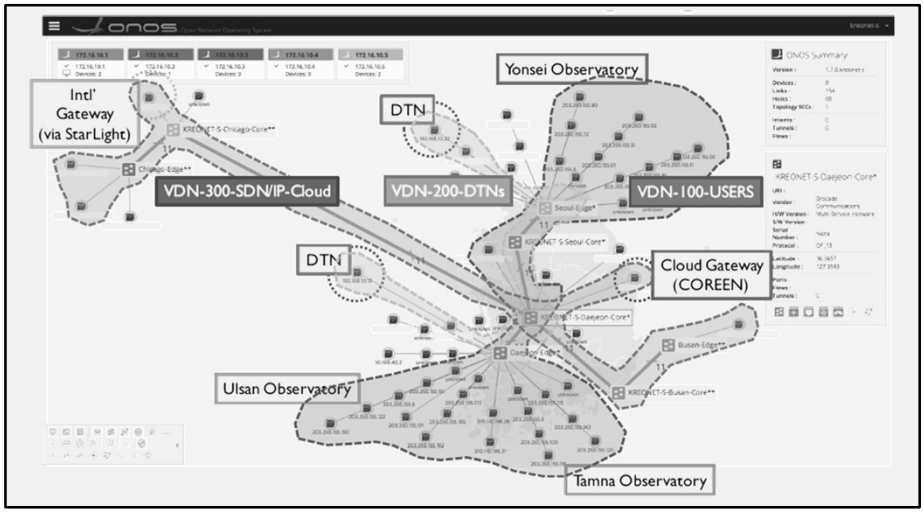
IV. SDN 개요

- ❖ **OF@KOREN (미래네트워크연구시험망),**
- ❖ **OF@TEIN (국제연구망)**



IV. SDN 개요

- ❖ **SDN 컨트롤러 :** CORD는 클라우드 네트워킹을 위한 SDN 컨트롤러 사용 (ONOS)



SDN/NFV for 5G

james@jslab.kr

I. 개요
II. 소프트웨어 정의
III. 가상화와 클라우드 서비스
IV. SDN 개요
V. NFV
VI. 오버레이 / 언더레이
VII. SDN 관련 기술
VIII. 클라우드 네트워크
IX. 텔레콤 환경을 위한 SDN/NFV
X. 관리

❖ 부록: OpenFlow
❖ 실습교재 (별도)

JS Lab

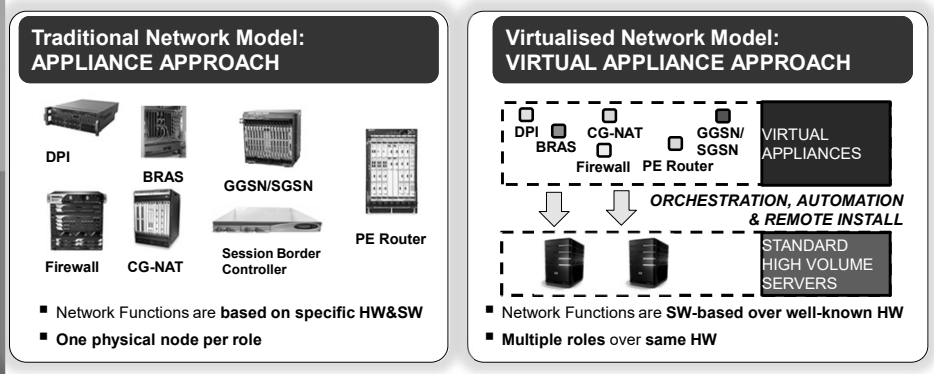
113

SDN/NFV for 5G

james@jslab.kr

V. NFV

❖ 통신 시스템 구성 컴포넌트 (Network Function)을 가상화(Virtualization)
❖ 하드웨어 의존성을 최소화 하는 네트워크 가상화로 독립적이며 유연하고 단순한 네트워크를 만드는 도구 제공
❖ 가상 네트워크의 도입 및 인프라의 클라우드화



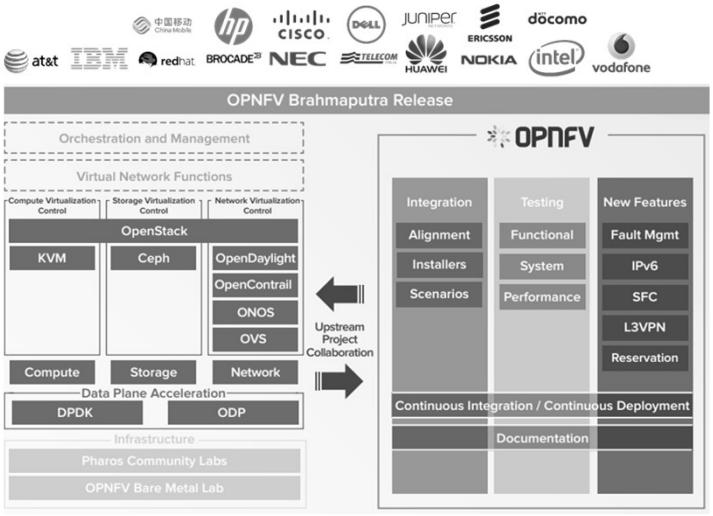
JS Lab

114

SDN/NFV for 5G
james@jslab.kr

V. NFV

❖ 오픈 NFV(OPNFV)



JS Lab

115

SDN/NFV for 5G
james@jslab.kr

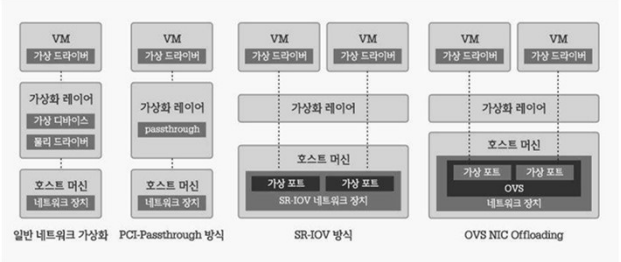
V. NFV

❖ VNF (Virtual Network Function)의 마이크로 서비스화

❖ VNF의 MSA화를 통한 각 모듈의 간소화 (lightweight) 및 각 모듈 간의 SBA (Service Based Architecture)에 기반한 통신

❖ VNF의 컨테이너화 (Containerization) / 마이크로 서비스화 (Micro-Service Architecture – MSA)

- **Kubernetes 도입에 따른 운영 환경 변화**
- **Kubernetes Networking:** 물리 장비와 동일한 네트워크 성능을 보장 해야 하며, 이를 위해서 SR-IOV, OVS-DPDK와 같은 가속 기술 사용이 필요



JS Lab

출처: SDN/NFV 기반 5G 통신망 인프라의 진화, 정보화진흥원, SKT 신상호 매니저

116

V. NFV

❖ **분야별 NFV Use Case:**

- **서비스 사업자**
  - ✓ 코어의 가상라우터
  - ✓ 에지의 vCPE(virtual Customer Premises Equipment)
- **엔터프라이즈**
  - ✓ 방화벽이나 로드밸런서 등의 분산 네트워크 기능
  - ✓ 마이크로서비스/마이크로세그먼트 구성
- **클라우드 데이터센터 제공**
  - ✓ 가상 인프라/테넌트
  - ✓ 방화벽이나 로드밸런서 등의 분산 네트워크 기능

117

V. NFV

❖ **NFV(Network Function Virtualization) 상용 제품:**

Virtual Router / Switch	Virtual Firewall / IPS / IDS	Virtual Load Balancer	SD-WAN	UC-IP Telephony
VyOS, open source router	pfSense, open source firewall	HAproxy, open source	Silver Peak Virtual Unity	Cisco CallManager
Vyatta, commercial router	Juniper vSRX, commercial firewall	F5 vLTM, commercial	Riverbed SteelConnect	Asterisk-based systems
Cisco Nexus 1000v, commercial	Snort, open source IDS	Loadbalancer.org, commercial	Cisco Viptela	Avaya
Cisco CSR	Cisco ASAv	Avi Networks, commercial	VeloCloud Networks	Skype for Business
VMware NSX	VMware NSX	VMware NSX	Versa	Freeswitch

118

V. NFV

❖ **Virtual Firewalls:**

Vendor	Name	Commercial or Open Source
Juniper	vSRX	Commercial
Cisco	ASAv	Commercial
FortiGate	Virtual Appliances	Commercial
Sophos	Virtual Appliance	Commercial
VMware	NSX Firewall	Commercial
Stonegate (ForcePoint)	Virtual Appliance	Commercial
Rubicon Communications	pfSense	Open Source
Palo Alto Networks	Virtual Appliance	Commercial

119

V. NFV

❖ **Virtual Load Balancers:**

Vendor	Name	Commercial or Open Source
F5 Networks	Virtual LTM	Commercial
Citrix	Virtual Load Balancer	Commercial
Avi Networks	Virtual Load Balancer	Commercial
Barracuda	Virtual Load Balancer	Commercial
Kemp	Kemp Virtual	Commercial
Fortigate	Virtual Load Balancer	Commercial
HAproxy Technologies	HA Proxy	Open Source
Facebook	Katran	Open Source

120

V. NFV

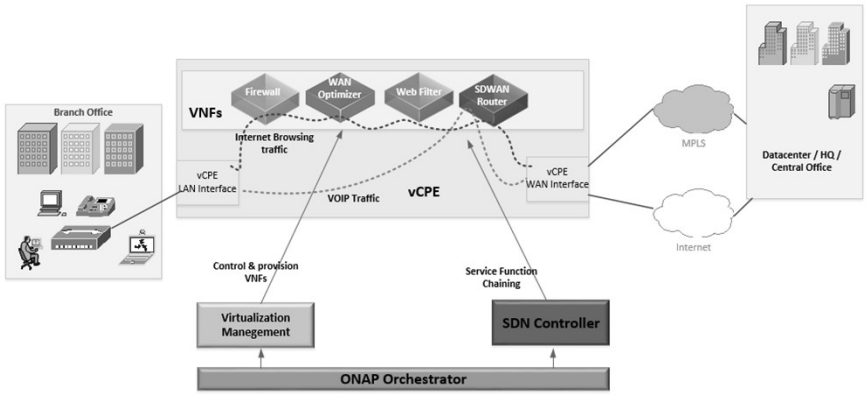
❖ **Virtual Routers:**

Vendor	Name	Commercial or Open Source
Cisco	CSR (Cloud Service Router)	Commercial
Cisco	ISRv (Integrated Services Virtual Router)	Commercial
Juniper	vMX	Commercial
Brocade (acquired)	Vyatta	Commercial
Alcatel Lucent	VSR	Commercial
VMware	NSX	Commercial
Cloud Router	Cloud Router	Open Source
VyOS	VyOS	Open Source
Quagga	Linux Router (Quagga)	Open Source

121

V. NFV

❖ **Service Chaining:**

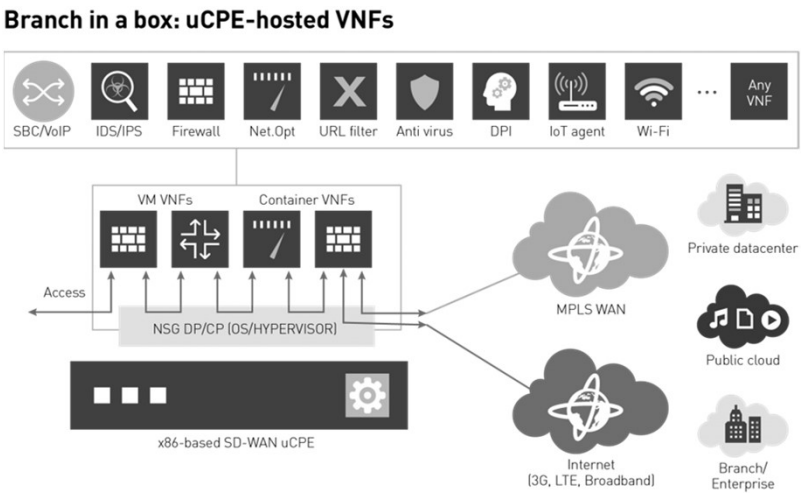


NFV and Service Chaining in a Service Provider Network

122

V. NFV

❖ **uCPE (Universal Customer Premises Equipment):**



출처: https://www.nuagenetworks.net/blog/vsp-release-5-0-part-1/

123

V. NFV

❖ 네트워킹 오픈소스

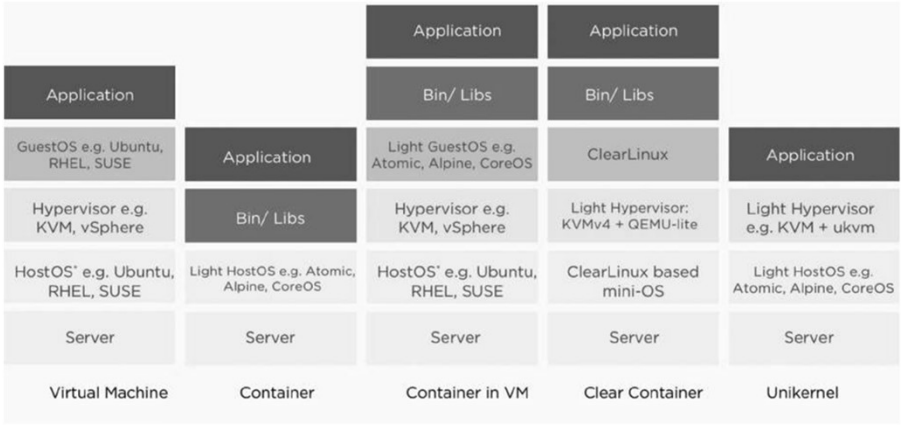
이름	구분	출범일
Edgent	네트워크 분석	2016-12
linkerd	NFVI – 인프라, VNF – L4-7 가속, 캐싱	2016-04
Cilium	NFVI, VNF – L4-7 보안	2017-03
BiRD	NFVI – 스위칭, 라우팅	2013-03
NetBox	NFVI – 스위칭, 라우팅	2016-06
OSM (Open Source MANO)	NFV MANO	2016-05
FBOSS	NFVI – 스위칭, 라우팅, NFVI – NOS	2015-03
Faucet SDN 제어ler	NFVI – 제어	2015-03
GoBGP	NFVI – 스위칭, 라우팅	2017-02
HAProxy	VNF – L4-7 보안, 가속, 캐싱	2001-12
YANFF	NFVI, VNF – L4-7 보안, 가속, 캐싱	2017-03
OpenContrail	NFVI – 스위칭, 라우팅, 제어, NFV MANO	2013-09
OpenDataPlane Project	NFVI – 인프라	2015-02
OpenSwitch	NFVI, 스위칭, 라우팅, NOS	2016
OPNFV	NFVI – 인프라, 하드웨어, 스위칭, 라우팅, NOS, 제어	2017-09
FD.io	NFVI – 인프라, 스위칭, 라우팅, VNF – L4-7 가속, 캐싱	2016-02
OpenDaylight	NFVI – 제어	2013-03

이름	구분	출범일
Open vSwitch	NFVI – 스위칭, 라우팅	2009-07
ONAP	NFVI – 제어, NFV MANO, VNF – L4-7 보안,가속, 캐싱	2017-03
DPDK	NFVI – 인프라, 스위칭, 라우팅	2012-09
FR라우팅 (FRR)	NFVI – 스위칭, 라우팅	2017-10
OpenLSO	NFV MANO	2016-03
NGINX Open Source (OSS)	VNF – L4-7 보안, L4-7 가속, 캐싱	2011-07
Ryu NOS	NFVI – NOS, 제어	2011-12
Open Network Linux	NFVI – NOS	2014-01
ONIE	NFVI – 하드웨어, 설치	2013-06
SONiC	NFVI – 스위칭, 라우팅, NOS	2016-03
OpenConfig Project	NFV MANO	2014-10
CORD	NFVI – 인프라, NOS	미제공
ONOS	NFVI – 제어	2014-12
OpenStack Neutron	NFVI – 인프라	2013-07
OpenStack Tacker	NFV MANO	2015-12
P4	NFVI – 인프라, 스위칭, 라우팅	2015-02
Project Calico	NFVI – 스위칭, 라우팅	2014-07
Open Virtual Network (OVN)	NFVI – 인프라, 스위칭, 라우팅	2015-01

124

V. NFV

❖ NFVI의 가상 계층 선택 종류



SDN/NFV for 5G

james@jslab.kr

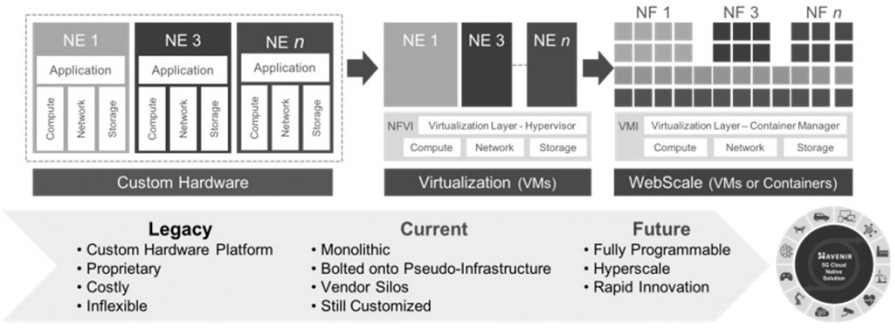
JS Lab

125

V. NFV

❖ 5G는 컴퓨팅 능력이 네트워크 구성을 변화

❖ 네트워크와 서비스 인프라의 경계 불명확



SDN/NFV for 5G

james@jslab.kr

JS Lab

출처: https://mavenir.com/products/packet-computing/5g-core

126

SDN/NFV for 5G

james@jslab.kr

V. NFV

- **ETSI NFV 5G Session Summary**
- **Common features:** NFV & SDN, Slicing, E2E management and orchestration, Security, Multi-access/domain, Low latency

	Common (NFV-SDN, Slicing, etc)	SLA, E2E quality	Edge computing	Autonomous, Self-healing	Softwarization	Flexibility, Plug & Play	Others
NGMN	○	○	○	○		○	Attestability
5G-PPP	○	○	○	○	○	○	Information model
ITU IMT2020	○	○	○	○	○		ICN/CCN, Trust, Traffic models
5GMF	○	○			○	○	
5G Forum	○			○	○	○	Energy efficiency,
TU-Dresden 5G Labs	○		○				Network coding, Compressed sensing
EURECOM	○	○	○			○	Programmability, C/D split, Micro services

JS Lab

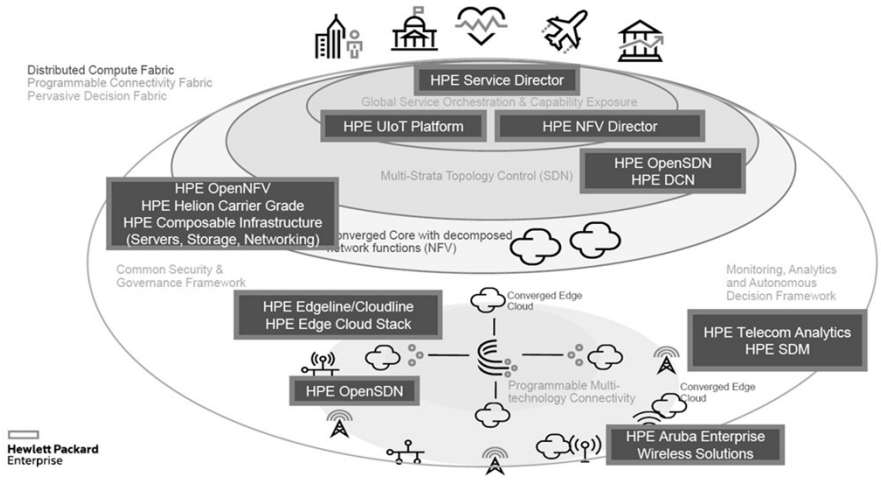
127

SDN/NFV for 5G

james@jslab.kr

V. NFV

- **5G Network Architecture: The HPE Portfolio**

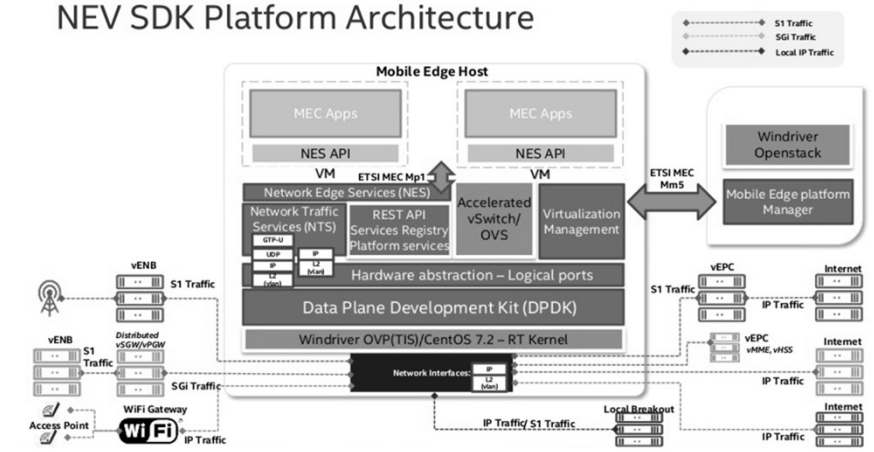


JS Lab

128

V. NFV

- ❖ 인텔의 플랫폼 아키텍처
- ❖ Network Edge Virtualization (NEV)
- ❖ Multi-access Edge Computing (MEC)



JS Lab

V. NFV

- ❖ 제조사 솔루션 기술

항목	내용
설치 자동화 (Ansible등)	• 설치 이미지 저장
	• 템플릿을 통한 자동 설치
	• 설정 관리
	• 라이프사이클 모니터링
오토 스케일링	• 스케일링을 위한 템플릿
	• 설치 및 스케일링 시 오토 리커버리
모니터링	• 모니터링을 위한 표준 인터페이스 연동
	• 보안 로그 수집 및 SIEM 전달
	• 보안 서비스 포탈, GUI 구현
인프라 보안 구성 요소	• vFW 가상방화벽
	• vWAF 가상웹방화벽
	• vIDS 침입탐지
	• vIPS 침입차단
	• 서버 보안(OS)
	• 서버 접근제어
	• DB 암호화
	• DB 접근제어
	• 위 제품들에 대한 오픈소스 솔루션 대체, 성능 검증
SDDC등의 시스템 연동	• 기존 시스템과 연동
	• 보안을 위한 오케스트레이션에 네트워크 통합 (SaaS등)

JS Lab

V. NFV

❖ 가상 보안 기기와 소프트웨어

Firewall	• Cisco (vASA) • Cisco (vNGFW, NGFWv) • PaloAlto (vFW) • Juniper (vSRX) • Juniper (cSRX) • CheckPoint (vFW) • Vyatta (오픈소스) • SOPHOS (vUTM) • pfSense (오픈소스) • Secui (MF2 VE)
IPS	• Cisco (vNGFW + 차단기능) • PaloAlto (VFW) • SOPHOS (vUTM) • Intel Security (McAfee) • Suricata (오픈소스) • TrendMicro (DeepSecurity)
IDS	• IBM (Proventia) • Snort (opensource) • Suricata (opensource) • Cisco (vNGFW + 차단 기능) • OSSEC (Host IDS)
DB Encrypt	• CubeOne • MSSQL/Oracle/PostgreSQL TDE • Vormetric

WAF	• Barracuda (vWAF) • Imperva (vWAF) • Array Networks (vWAF) • WebConx (Host WAF) • F5 (ASM) • mod_security (Host SW) • SOPHOS (vUTM)
SecureOS	• SGA (RedCastle) • CA (PIM) • SELinux/Grsecurity/AppArmor
SAC (Server Access Control)	• HiWare (NetandHI) • Core Security (Core PAM) • Centrify (Privilege Management) • CyberArk for Server (PSM) • CA (Privileged Access Management) • Dell(Quest)
DAC	• PNPSecure (DBSafer) • CyberArk for DB • CA (PAM) • Imperva (DB Security) • Gemalto

SDN/NFV for 5G

james@jslab.kr

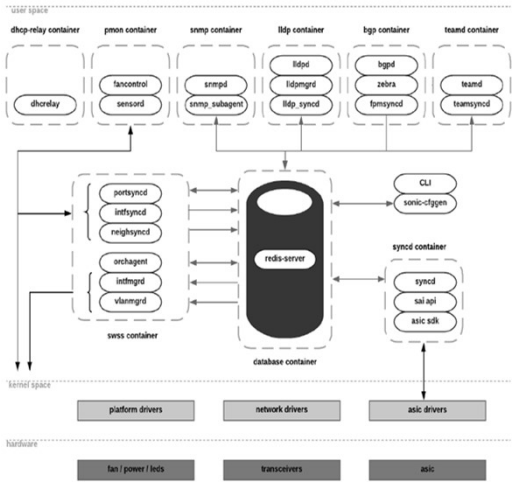
JS Lab

Identity Governance and Administration (IGA) 계정관리 DAC (Discretionary Access Control : 임의적 접근통제)

131

V. NFV

❖ **SONiC** (Software for Open Networking in the Cloud)



SDN/NFV for 5G

james@jslab.kr

JS Lab

132

V. NFV

❖ 오픈스택(OpenStack)의 클라우드 네이티브화(化)

- 사용자/개발팀 증가 시 필요한 멀티테넌트(Multi-tenants) 이슈 해결
- 컨테이너 기술은 소프트웨어 의존성 이슈 해결
- 오픈스택(OpenSteck)의 클라우드 네이티브화(化)를 위한 Kolla 프로젝트는 오픈스택을 마이크로서비스 개념으로 디버그와 업그레이드를 쉽게 함
- 고가용성 클라우드 아키텍처 설계로 오픈스택서비스를 신속하게 적용함
- 서비스를 유지하며 오픈스택 서비스의 유지보수와 업그레이드 가능

SDN/NFV for 5G
james@jslab.kr

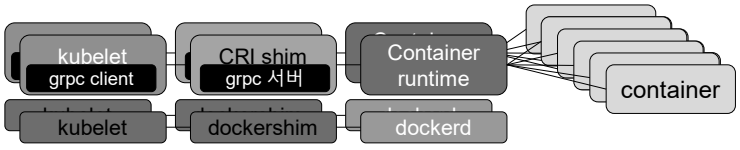
133

V. NFV

❖ CRI (Container Runtime Interface): Kubernetes(K8s)는 특정 런타임과 분리 추상화하며 K8s 1.6(2017년 3월) kublet에서 CRI를 정식 적용

❖ CRI 호환 런타임 프로젝트(배포범위 확대: VM/Hypervisor/베어메탈)

- Docker (CRI shim 라이브러리 사용)
- dockershim
- Rkt (CoreOS에서 파생)
- cri-o (도커와 같이 OCI 호환하는 OCI confirmed 런타임, K8s 프로젝트)
- frakti (하이퍼바이저 용 런타임)
- rktlet (rkt 컨테이너 런타임)
- virtlet VM(QCOW) 런타임
- cri-containerd

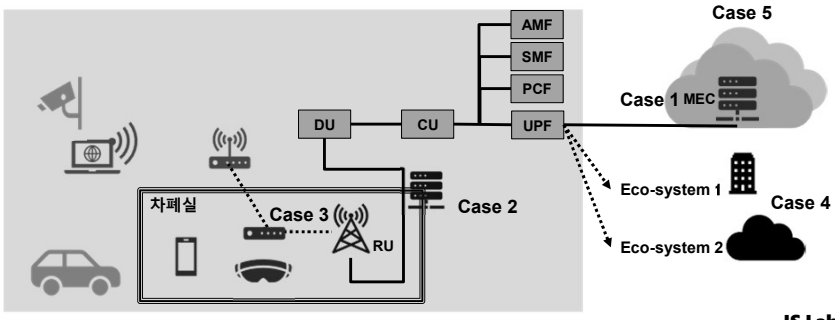


SDN/NFV for 5G
james@jslab.kr

134

V. NFV

- ❖ **Edge Cloud Computing @ 5G Testbed**
- ❖ **5G 비즈니스 고려: SW/HW 개발 vs 출시 서비스**
  - **Case 1: Managed Resources**
  - **Case 2: RU/DU/CU w/Appliances @ Far Edge Cloud** (초에지)
  - **Case 3: w/CPE or 5G Modem**
  - **Case 4: Eco-system**
  - **Case 5: Connecting Multiple Edge Clouds**



SDN/NFV for 5G
james@jslab.kr

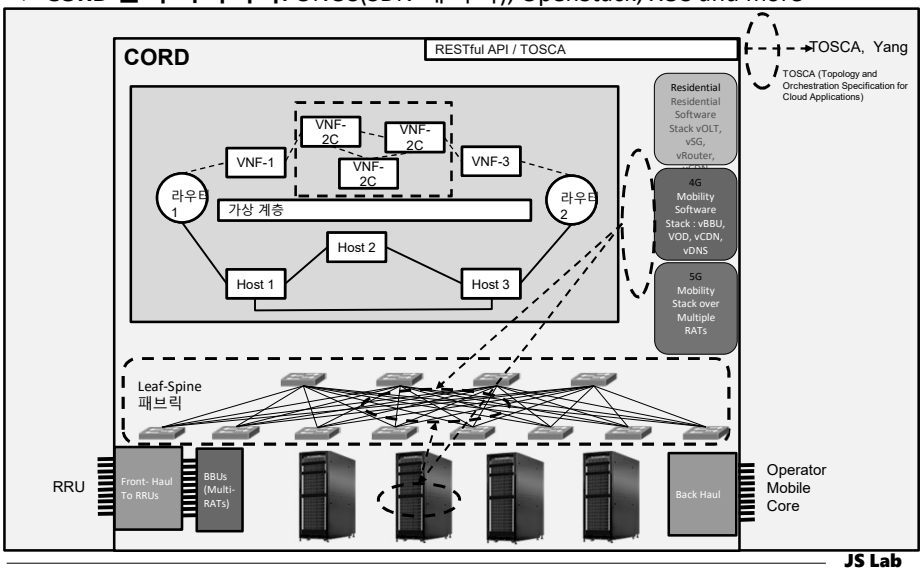
JS Lab

DU(Distributed Unit) **CU**(Centralized Unit) **AMF**(Application Function) **SMF**(Session Management Function) **PCF**(Policy Control Function)

135

V. NFV

- ❖ **CORD 관리 아키텍처:** ONOS(SDN 제어기), OpenStack, XOS and more



SDN/NFV for 5G
james@jslab.kr

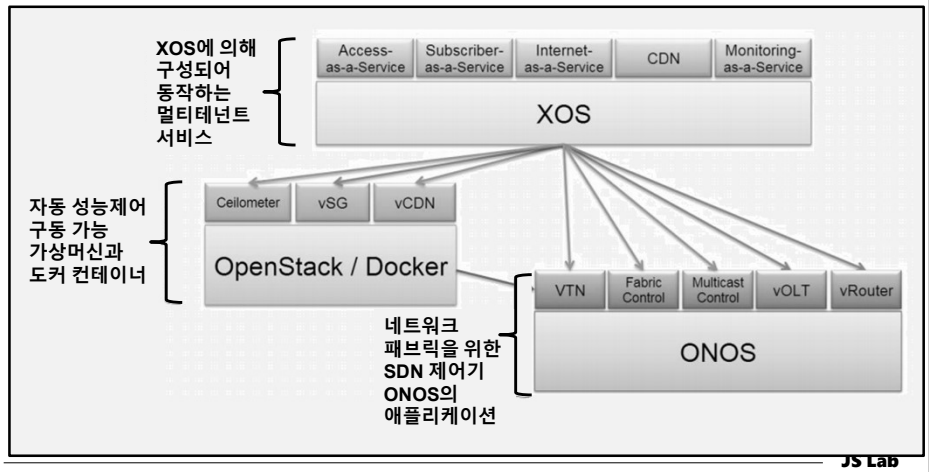
JS Lab

BBU (Baseband Unit) RRU (Remote Radio Unit) vSG (Virtual Subscriber Gateway) RAT (Radio Access Technology)

136

V. NFV

❖ CORD (Central Office Re-architected as a Datacenter) 소프트웨어 구성: R-CORD, M-CORD, E-CORD, A-CORD)



137

V. NFV

❖ TOSCA와 YANG

- CORD와 운영의 인터페이스 표준 TOSCA와 YANG

	TOSCA	YANG
기관	OASIS	IETF
적용 기반	IT, 애플리케이션	네트워킹
목적	애플리케이션 중심, 애플리케이션과 관련 가공 기능 적용, 확장성(Scale), 복구(Healing), 업그레이드, 설계 싯점 초점, 이행을 위한 기술(Description)	운영중인 네트워크 기기와 애플리케이션 컨피규레이션, 운영중을 초점, 블랙박스
슬로건	Bring them up and running	Configure them
예제	CRM 시스템, SQL DB, IPTV 서버, 가상 네트워크 템플릿 적용	새로운 IPTV 서비스 프로비져닝
운영	설치, 시작, 컨피규레이션, 확장성, 복구, 업그레이드, 중지, 삭제	시작한 후 / 정지 전에 운영과 컨피규레이션,
작업 방법	기업은 템플릿을 작성하고, 사용자는 템플릿 린스턴스를 작성함	표준 기구와 기업은 모델을 작성하고, 사용자는 인스턴스를 작성 한다.

TOSCA (Topology and Orchestration Specification for Cloud Applications) YANG (Yet Another Next Generation)

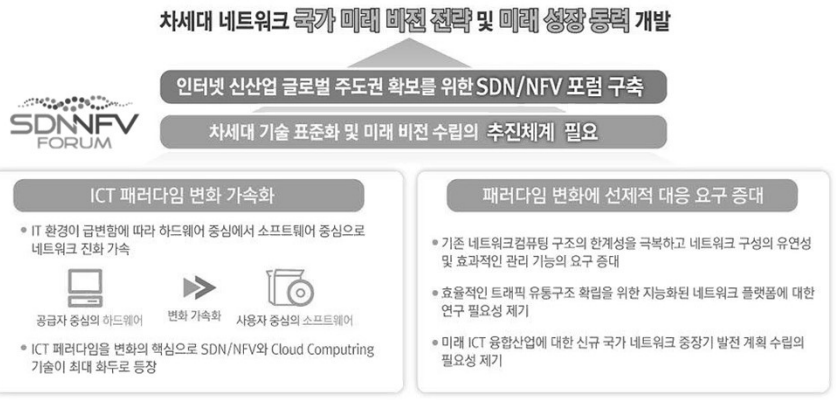
138

V. NFV

❖ **SDN/NFV 포럼(www.sdnnfv.org)**

① **SDN/NFV 포럼 2014년 10월 1일 출범**

② **목표 :** SDN/NFV 등장으로 산업 격변기에 있는 국내 인터넷인프라 산업을 활성



139

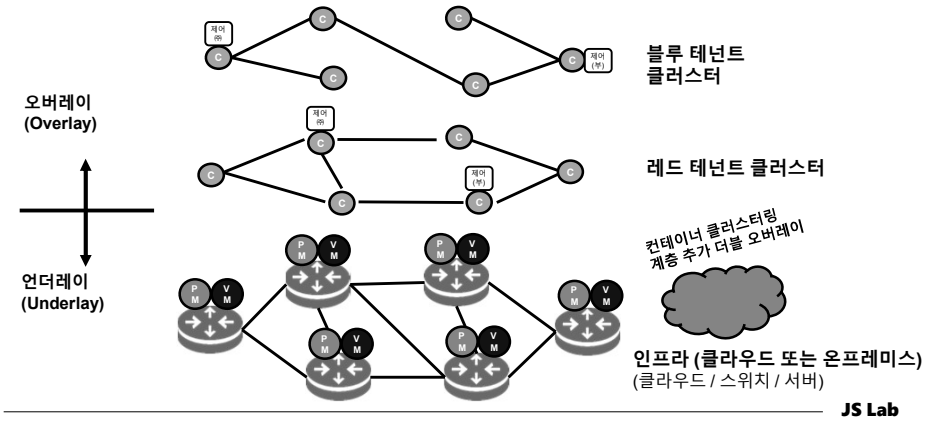
140

VI. 오버레이 / 언더레이

❖ 오버레이(Overlay)

① 오버레이(Overlay)

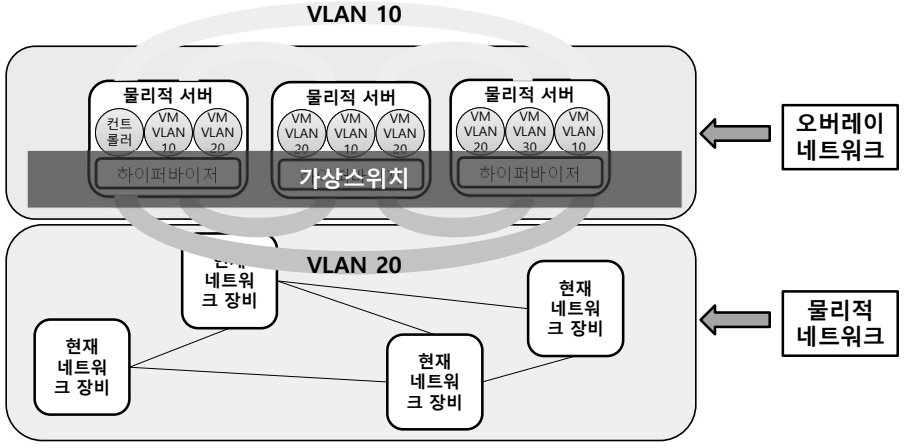
② 언더레이(Underlay)



141

VI. 오버레이 / 언더레이

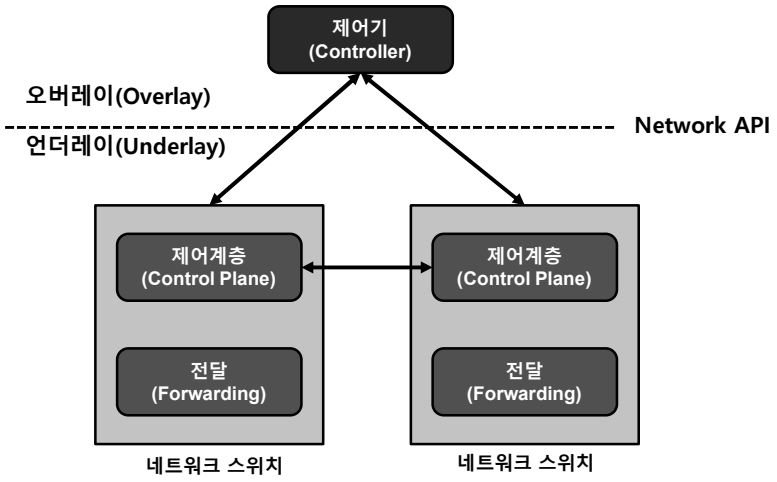
❖ **오버레이 네트워킹** (터널링 프로토콜 이용 가능)



142

VI. 오버레이 / 언더레이

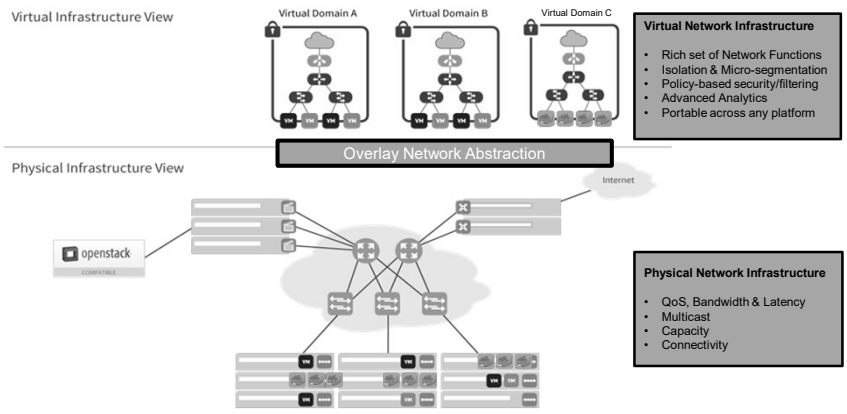
❖ 물리 스위치 기반 SDN 솔루션



143

VI. 오버레이 / 언더레이

❖ Micro-segmentation, VNFs, Security Policies for OpenStack and Containers

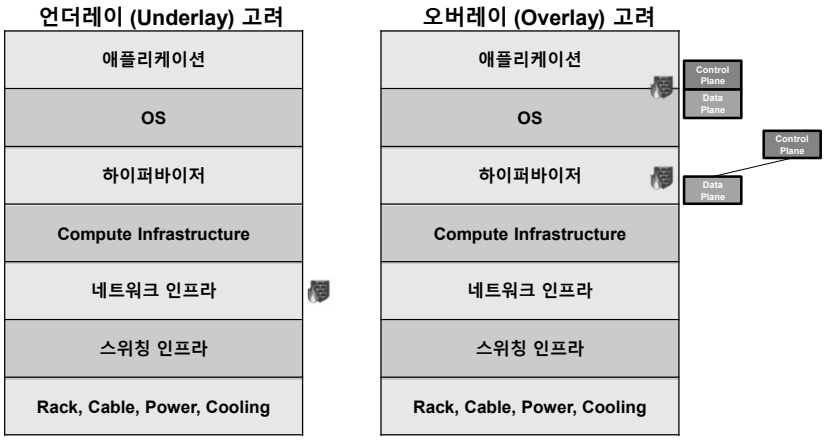


144

VI. 오버레이 / 언더레이

❖ 오버레이 네트워크 인프라 보안

- **가상기기 사용 보안: vFW, vIDS, vIPS** (VM 제공)
- **SDN 기반 분산 보안: dvFW, dvLB** (w / dvRouter, dvSwitch)



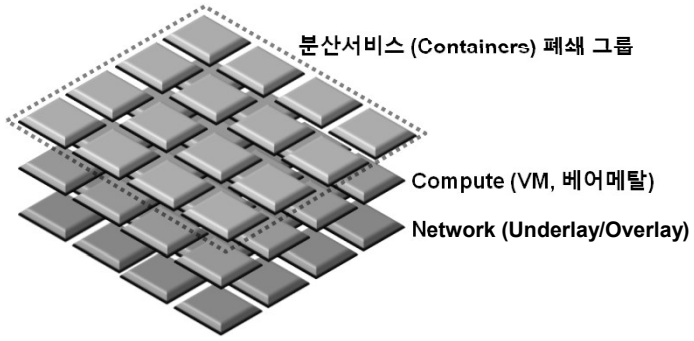
145

VI. 오버레이 / 언더레이

❖ 클라우드 인프라 보안

❖ 클라우드 기반 보안 체계

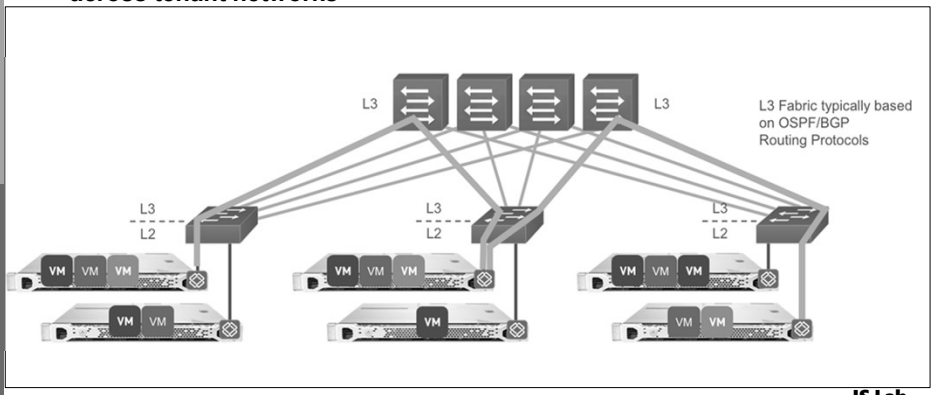
- 클라우드/SDDC/하드웨어 계층간 격리
- 외부 서비스의 노출 정책 지정 (Ingress, LB, DMZ)
- 계층내 신뢰 정책 강화



146

VI. 오버레이 / 언더레이

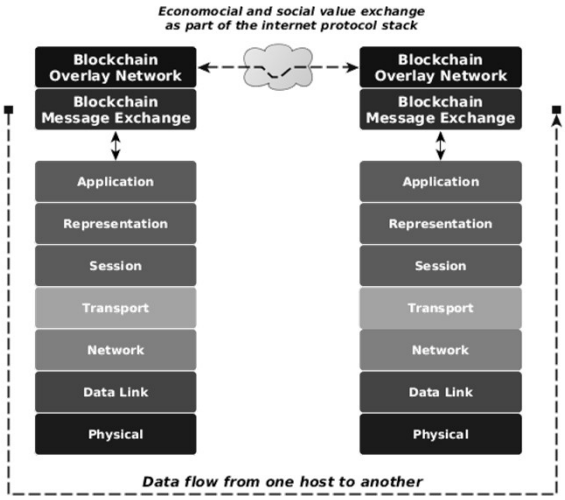
- ❖ **Multi-tenancy achieved by "overlaying" MAC-in-IP 'Tunnels' onto the physical switch fabric (underlay, transport network)**
- ❖ **Encapsulation header (VXLAN, NVGRE, STT) convey tenant network ID to enable full isolation and overlapping IP Address spaces support**
- ❖ **Software layers to implement routing / switching operations within and across tenant networks**



147

VI. 오버레이 / 언더레이

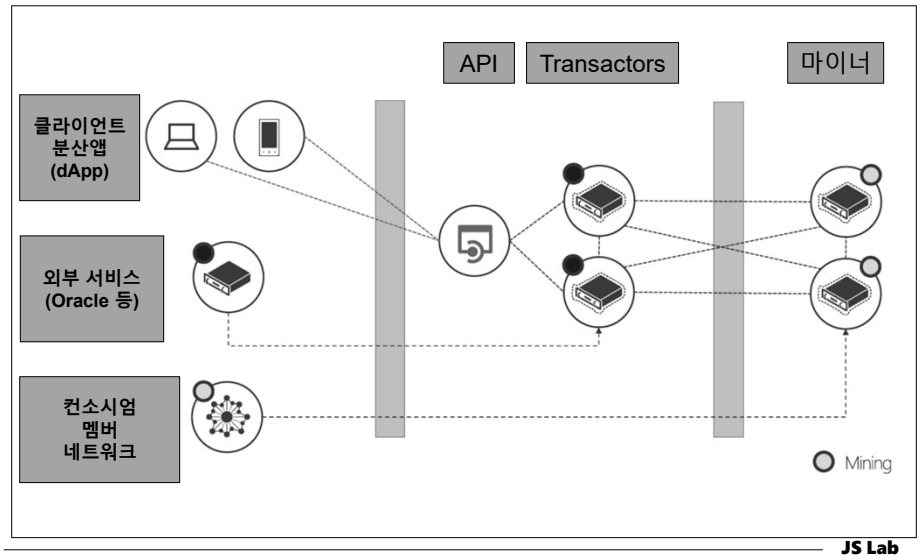
- ❖ **블록체인 오버레이 네트워크 제공:** Amazon AWS, IBM Cloud, MS Azure



148

VI. 오버레이 / 언더레이

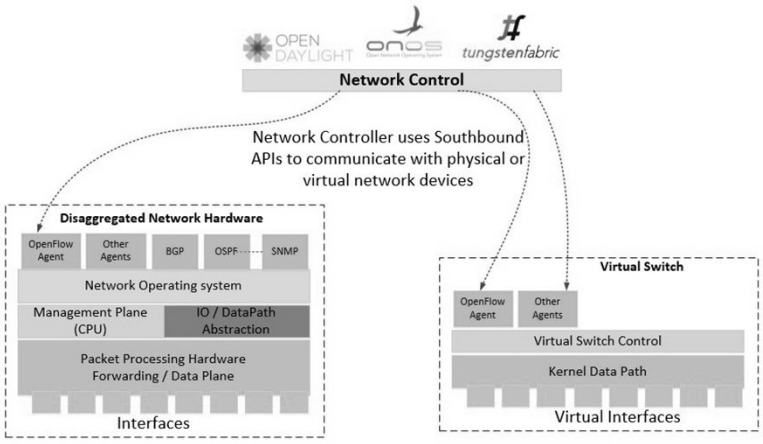
❖ 블록체인 토폴로지



149

VI. 오버레이 / 언더레이

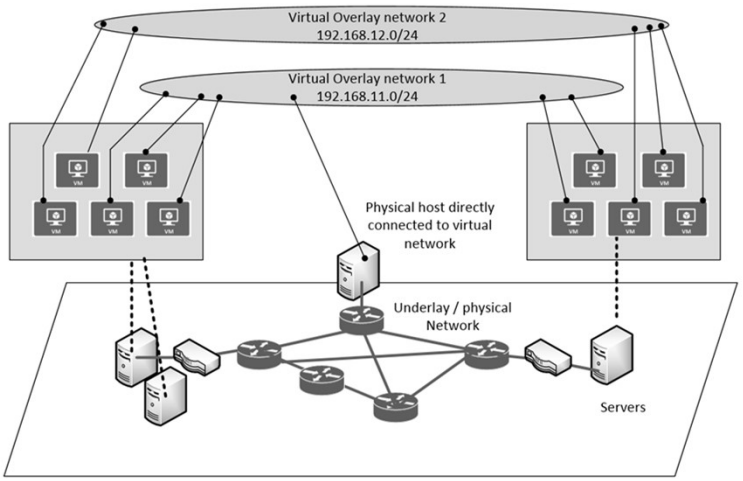
❖ The Network Controller Can Manage Virtual or Hardware Switches/Network Devices



150

VI. 오버레이 / 언더레이

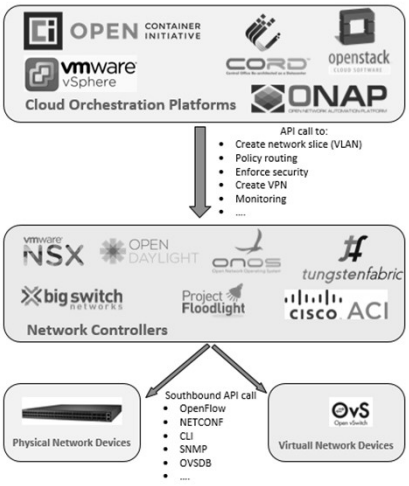
❖ **Overlay Networks Are Virtual Networks on Top of Physical Networks, Built Using Encapsulation and Tunnels**



151

VI. 오버레이 / 언더레이

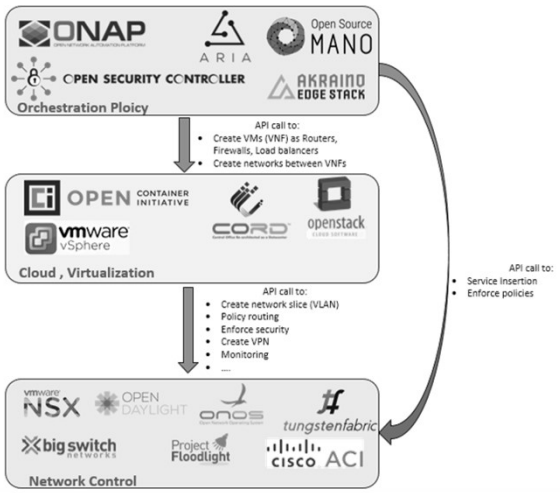
❖ **Cloud and Virtual Management Layer Communicates with Network Controllers**



152

VI. 오버레이 / 언더레이

❖ **Orchestration Platforms Communicate with Network Controller and Cloud Management Layers**



153

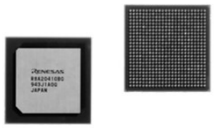
154

VII.SDN 관련 기술

❖ TCAM

1.현재 스위치 장비는 빠른 주소 및 패턴 검색을 위해 TCAM을 사용

2.TCAM : Ternary Content Addressable Memory



	DRAM	TCAM
칩 집적도	✓ 4MB	X MB
MB당 가격	✓ 0.5 ~ $0.75	X 200 ~ $250
접근 속도	X 0ns ~ 80ns	✓ ns ~ 8ns
전기 소모	✓ 5W ~ 2W	X W ~ 30W

155

VII.SDN 관련 기술

❖ 오픈V스위치(OpenVSwitch, OVS)

① 리눅스 기반의 멀티레이어 네트워크 소프트웨어 스위치

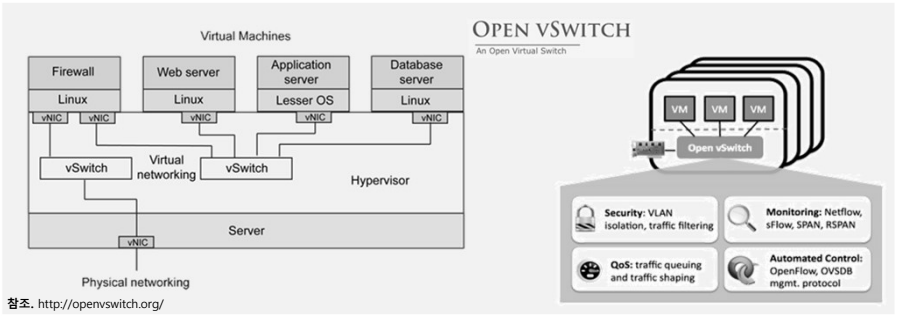
② OpenFlow 뿐만 아니라 현재 스위치 관련 기능도 지원

- VLAN trunk, GRE 터널, NetFlow 등

③ Apache License(BSD type)에 의해 관리

④ 일반적으로 4 x 1Gb/s의 제약사항

⑤ KVM, VirtualBox, Xen, XenServer 등 대부분의 하이퍼바이저 지원



참조. http://openvswitch.org/

156

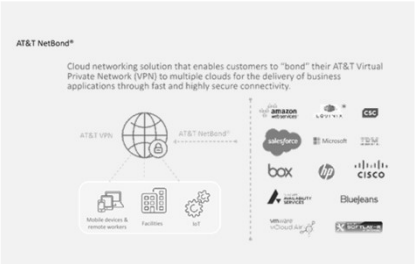
VII.SDN 관련 기술

❖ AT&T 서비스

① Network on Demand (vCPE, vPE, uCPE(vFW) – SDN/NFV
② NetBond – SDN
③ Connected Car – NFV
④ MVNO –NFV
⑤ URL Redirect (차단, 취소) – SDN/NFV
⑥ Mobile Call Recording – NFV

❖ AT&T 내부 서비스

① 제어(DNS, NAT, NTP, DHCP, RADIUS, FW, LB) – NFV/SDN
② Probe - NFV

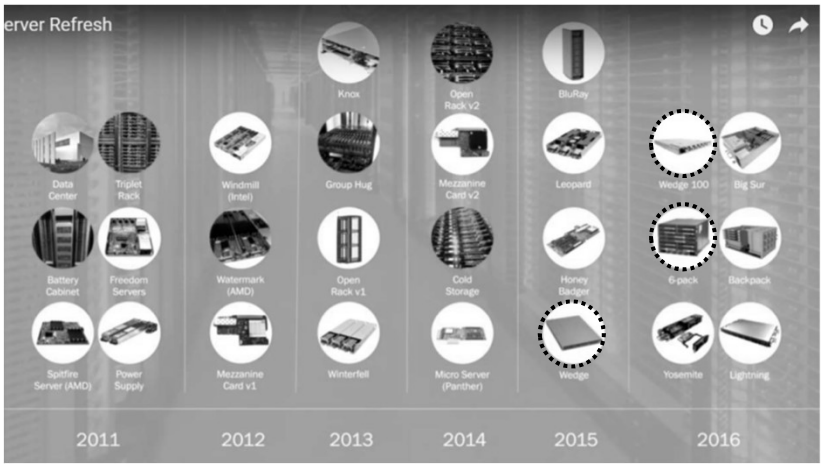


SDN/NFV for 5G
james@jslab.kr
JS Lab

157

VII.SDN 관련 기술

❖ OCP의 발전



SDN/NFV for 5G
james@jslab.kr
JS Lab

158

VII.SDN 관련 기술

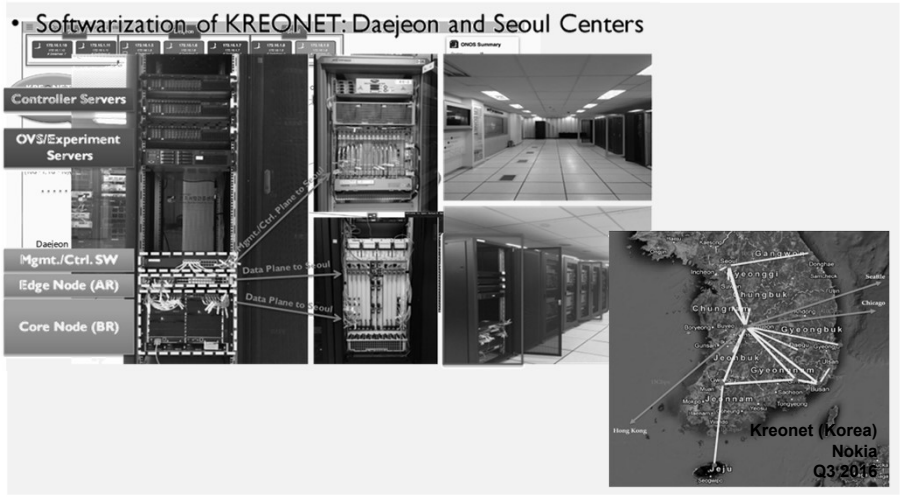
❖ NOS

NOS	Type	Runs on	Features
OpenSwitch	Open Source	Bare metal switch	L2/L3, Linux commands
DANOS	Open Source	Bare metal switch	N/A
Open Network Linux	Open Source	Bare metal switch	Base Linux OS
SONiC	Open Source	Bare metal switch	L2/L3
FBOSS	Open Source	Bare metal switch	Only FBOSS remote co nfiguration agent
Cumulus Linux	Commercial, closed source	Bare metal switch	L2/L3
FRR	Open Source	Linux hosts	L3
Stratum	Open Source	Bare metal switch	L2/L3, SDN

159

VII.SDN 관련 기술

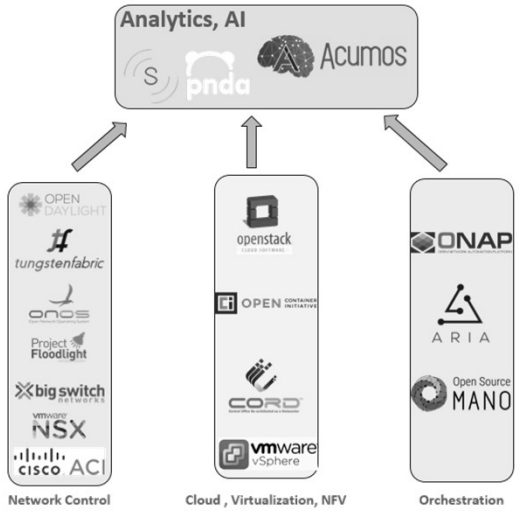
❖ KREONET (오픈소스 기반 관리 소프트웨어 개발)



160

VII.SDN 관련 기술

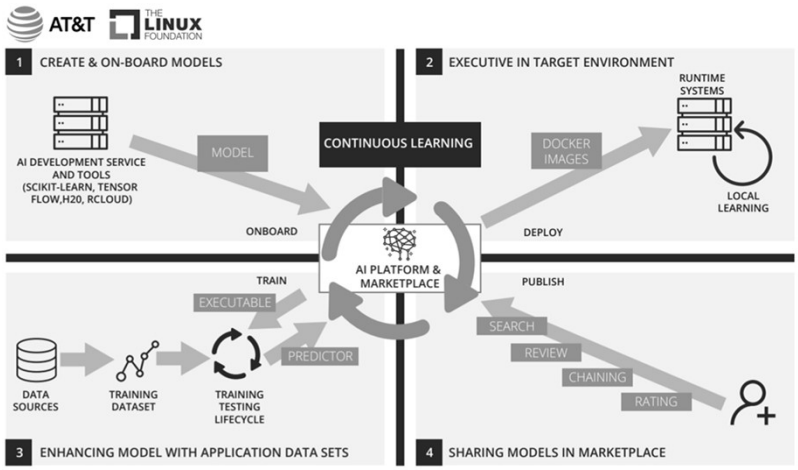
❖ The Linux Foundation Network Analytics Projects



161

VII.SDN 관련 기술

❖ Acumos
❖ An Open Source AI Machine Learning Platform
- By AT&T and The Linux Foundation



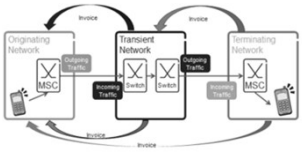
출처: https://www.acumos.org/wp-content/uploads/sites/61/2018/03/acumos_open_source_ai_platform_032518.pdf

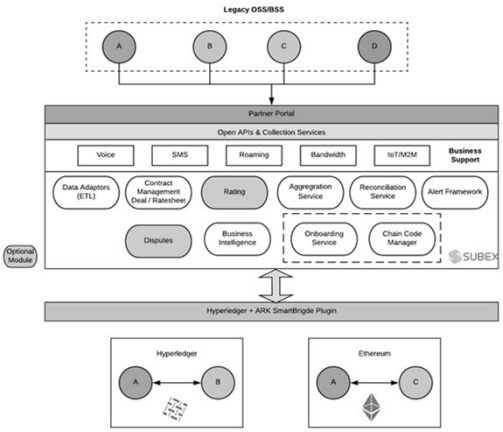
162

VII.SDN 관련 기술

❖ 블록체인 토폴로지 (Inter-carrier Charges @ Telecom)

- Proposed solution is a event agnostic platform that can manage Voice, SMS, Roaming, IoT, Content or any other event settlement scenarios making it is true convergent solution.





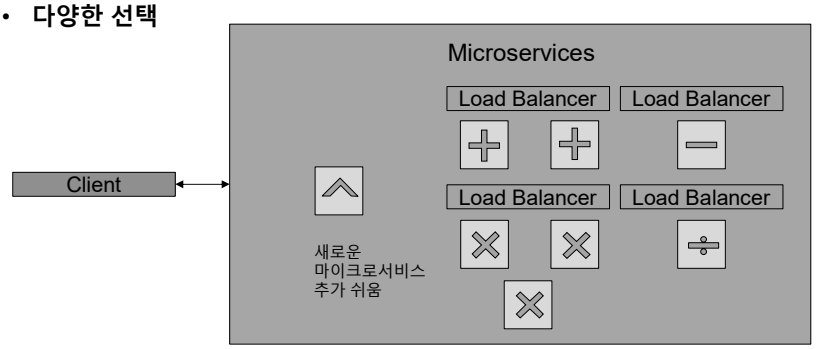
163

164

VIII.클라우드 네트워크

❖마이크로서비스 아키텍처 네트워크

- 단순성(Simplicity)
- 확장성(Scalability)
- 적용(Continuous delivery)
- 낮은 의존성과 더 많은 자유
- 장애 확인
- 데이터 분리와 분산화
- 다양한 선택

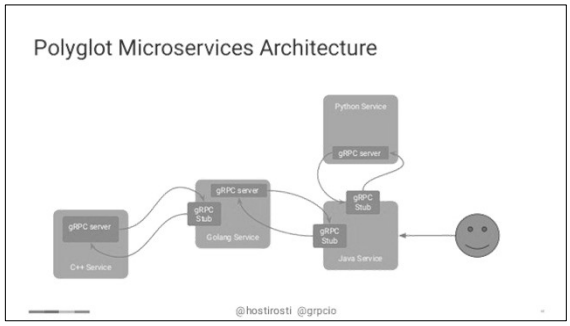


165

VIII.클라우드 네트워크

❖gRPC 사용 마이크로서비스

- **gRPC is faster than REST. gRPC uses HTTP2 by default**
- **gRPC defines relationship between client and server and enforces strict rules of communication between them.** (In REST calls, the request and response are totally de-coupled)
- **gRPC supports useful additions like standard error responses and meta data.**



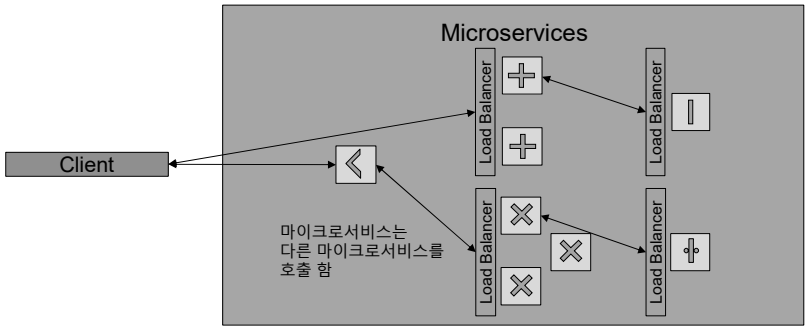
https://medium.com/@akshitjain_74512/inter-service-communication-with-grpc-d815a561e3a1

166

VIII.클라우드 네트워크

❖마이크로서비스 네트워크의 단점

- 트러블슈팅의 복잡성
- Latency 증가
- 운영의 복잡성
- 버전 제어



VIII.클라우드 네트워크

❖Discovery Service

- Client는 기존에는 1개의 monolithic app을 하였으나 MSA에서는 동시에 여러 개의 서비스를 호출해야 함
- Client는 서비스들의 위치를 알아야 함 (Host/IP/Port 등)
- Client가 마이크로서비스 호출시 제공하는 entry point 실시간 위치를 하드코드(Hard code)보다는 유연한 방법으로 현재의 위치를 제공해야 함

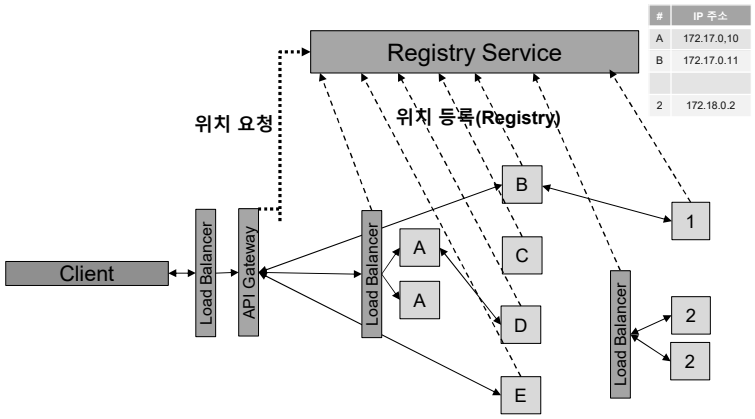
❖API Gateway

- 모든 호출에 대한 1개의 entry point가 필요
- 내부의 복잡성을 숨김
- 마이크로 서비스 변화/결합/구분/추가/제거 유연함
- Client와 Application사이의 왕복 단순화로 효율성 증대

VIII.클라우드 네트워크

❖**Service registry**

- **API Gateway는 모든 서비스에 대한 IP 주소를 알기 위한 DB 필요**
- **Registry 데이터의 안정성을 위한 오픈소스** (Consul이나 SkyDNS)



VIII.클라우드 네트워크

❖**적용 기술 선택**

- **Independency**
- **Source Control**
- **Environment**
- **Failsafe**
- **Reuse**
- **Tagging** (로그 관리 예: Splunk나 ELK or 'Elasticsearch, Logstash, Kibana')

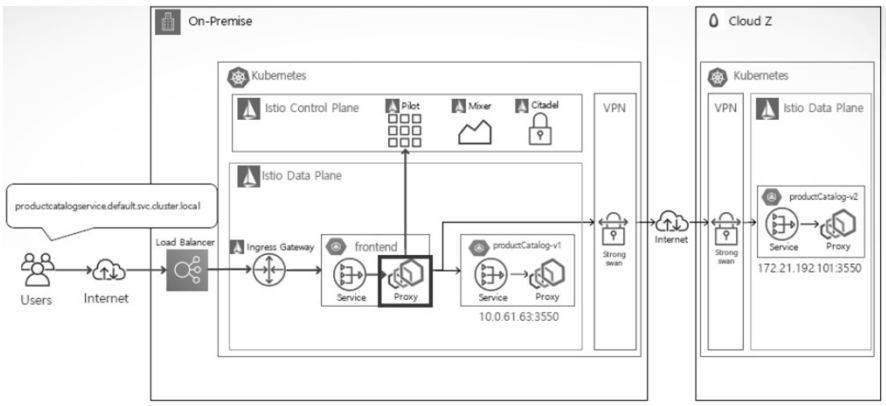
❖**운영 기술 선택**

- **모니터링: 인프라의 모든 요소를 모니터하며 라이브 대쉬보드와 이슈 팝업 사용**
- **온디맨드 확장성: 수동 또는 자동화**
- **API 노출: Netflix의 경우 Zuul 오픈소스로 공개**
- **서킷 브레이커 Circuit Breaker): 모든 시비스에 대해 요청모드를 요구하여 장애를 확인하며, 복수의 장애시 특정 서비스를 차단**(Break the circuit)**함**

VIII.클라우드 네트워크

❖ **Multi-Cloud Service Mesh Architecture (예)**

- **Multi-Cloud(Cross Cluster)간 Service Mesh 연결/확장**



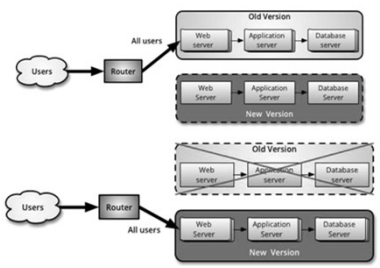
171

VIII.클라우드 네트워크

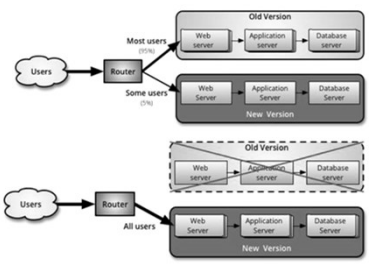
❖**Deployments**

- **Blue/Green Deployments**
- **Canary Deployments**

Blue/Green Deployments



Canary Deployments

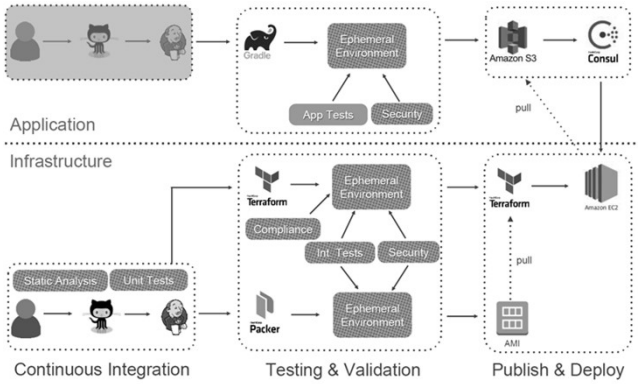


172

VIII.클라우드 네트워크

❖**Containerized Services**

- **Infra Module - Container Management System**
- **Fully Decoupled from Apps**
- **Apps are deployed with Container Management System specific tools**



173

VIII.클라우드 네트워크

❖**클라우드 디자인 패턴**

- **디자인 패턴은 클라우드에서 안정적이고 확장성 있는 안전한 애플리케이션을 빌드하는 데 유용**
- **각 패턴은 패턴이 해결하는 문제, 패턴을 적용하기 위한 고려**
- **클라우드 플랫폼에 호스팅, 분산 시스템과 관련**
  - 가용성
  - 데이터 관리
  - 디자인 및 구현
  - 메시징
  - 관리 및 모니터링
  - 성능 및 확장성
  - 복원력
  - 보안

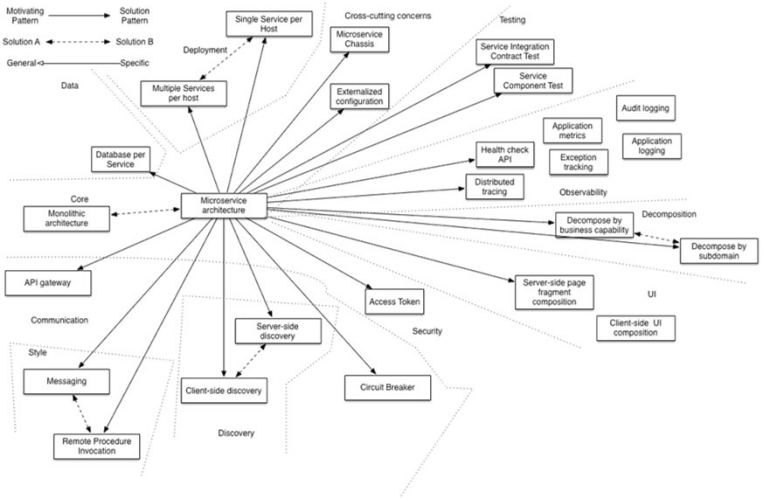
174

SDN/NFV for 5G

james@jslab.kr

VIII.클라우드 네트워크

❖Pattern: Microservice Architecture



https://microservices.io/patterns/microservices.html

JS Lab

SDN/NFV for 5G

james@jslab.kr

VIII.클라우드 네트워크

❖패턴 카탈로그 (MS Azure 예)

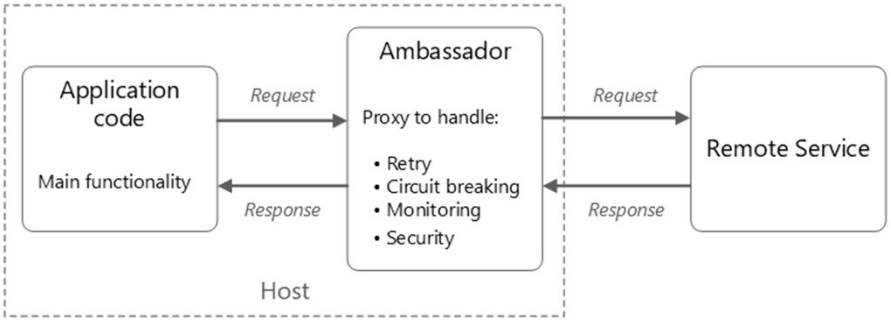
특사	소비자 서비스 또는 애플리케이션을 대신하여 네트워크 요청을 전송하는 도우미 서비스를 만듭니다.
손상 방지 레이어	현대식 애플리케이션과 레거시 시스템 사이에 외관 또는 어댑터 레이어를 구현합니다.
프런트 엔드에 대한 백 엔드	특정 프런트 엔드 애플리케이션 또는 인터페이스에서 사용할 별도의 백 엔드 서비스를 만듭니다.
격벽	하나가 고장 나더라도 나머지는 정상적으로 작동하도록 애플리케이션의 요소를 여러 풀에 격리합니다.
Cache-Aside	필요할 때 데이터를 데이터 저장소에서 캐시로 로드
연속	중앙 오케스트레이터에 의존하는 대신 각 서비스에서 비즈니스 작업이 처리되는 시기와 방법을 결정하도록 합니다.
회로 차단기	원격 서비스 또는 리소스에 연결할 때 해결하는 데 걸리는 시간이 유동적인 오류를 처리합니다.
클레임 검사	큰 메시지를 클레임 검사 및 페이로드로 분할하면 메시지 버스의 과부하를 피할 수 있습니다.
보정 트랜잭션	여러 단계로 나뉘어 있지만 결국에는 일관적인 작업을 정의하는 일련의 단계에서 수행한 작업을 실행 취소합니다.
경쟁 소비자	여러 동시 소비자가 동일한 메시징 채널에 수신된 메시지를 처리할 수 있게 해 줍니다.
컴퓨팅 리소스 통합	여러 작업을 단일 계산 단위로 통합합니다.
CQRS	별도의 인터페이스를 사용하여 데이터를 업데이트하는 작업과 데이터를 읽는 작업을 분리합니다.
이벤트 소싱	추가 전용 저장소를 사용하여 도메인의 데이터에 대해 수행된 작업을 설명하는 일련의 이벤트 전체를 기록합니다.
외부 구성 저장소	구성 정보를 애플리케이션 배포 패키지에서 중앙 위치로 이동합니다.
페더레이션 ID	외부 ID 공급자에게 인증을 위임합니다.
게이트 키퍼	클라이언트와 애플리케이션 또는 서비스 간 브로커 역할을 하며, 요청을 검사 및 정리하고, 요청 및 데이터를 전달하는 전용 호스트 인스턴스를 사용하여 애플리케이션 및 서비스를 보호합니다.
게이트웨이 집계	게이트웨이를 사용하여 여러 개별 요청을 단일 요청으로 집계합니다.
게이트웨이 오프로딩	공유 또는 특수 서비스 기능을 게이트웨이 프록시에 오프로드합니다.
게이트웨이 라우팅	단일 엔드포인트를 사용하여 요청을 여러 서비스에 라우팅합니다.
상태 엔드포인트 모니터링	외부 도구가 노출된 엔드포인트를 통해 주기적으로 액세스할 수 있는 기능 검사를 애플리케이션 내부에 구현합니다.
인덱스 테이블	쿼리에서 자주 참조하는 데이터 저장소의 필드에 대한 인덱스를 만듭니다.
리더 선택	인스턴스 중 하나를 다른 인스턴스를 관리하는 리더로 선택하여 분산된 애플리케이션의 공동 작업 인스턴스 컬렉션이 수행하는 작업을 조정합니다.
구체화된 뷰	데이터가 필요한 쿼리 작업에 대해 이상적으로 포맷되지 않은 경우 하나 이상의 데이터 저장소에 있는 데이터에 대한 미리 채워진 뷰를 생성합니다.
파이프 및 필터	복잡한 처리를 수행하는 작업을 재사용 가능한 일련의 별도 요소로 분류합니다.
우선 순위 큐	우선 순위가 높은 요청을 우선 순위가 낮은 요청보다 먼저 받아서 처리하도록 서비스로 전송된 요청의 우선 순위를 지정합니다.
게시자/구독자	애플리케이션이 발신자와 수신자를 연결하지 않고 여러 관심 있는 소비자에게 이벤트를 비동기적으로 알릴 수 있습니다.
큐 기반 부하 평준화	작업 그리고 그 작업이 일시적인 높은 부하를 부드럽게 처리하기 위해 호출하는 서비스 사이에서 버퍼 역할을 하는 큐를 사용합니다.
다시 시도	이전에 실패한 작업을 투명하게 다시 시도하여 서비스 또는 네트워크 리소스에 연결하려 할 때 애플리케이션을 사용하여 예상된 일시적 오류를 처리합니다.
Scheduler 에이전트 감독자	서비스 및 기타 원격 리소스의 분산된 집합에서 일련의 작업을 조정합니다.
분할	데이터 저장소를 수평 파티션 또는 분할 집합으로 나눕니다.
사이드카	격리 및 캡슐화를 제공하는 별도의 프로세스 또는 컨테이너에 애플리케이션 구성 요소를 배포합니다.
정적 콘텐츠 호스팅	정적 콘텐츠를 클라이언트에 직접 제공할 수 있는 클라우드 기반 스토리지 서비스에 배포합니다.
스트랭글러	특정 기능을 새로운 애플리케이션 및 서비스로 점진적으로 교체하여 레거시 시스템을 단계적으로 마이그레이션합니다.
제한	애플리케이션 인스턴스, 개별 테넌트 또는 서비스 전체의 리소스 사용량을 제어합니다.
발레 키	클라이언트에 특정 리소스 또는 서비스에 대한 제한된 직접 액세스를 제공하는 토큰 또는 키를 사용합니다.

https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

JS Lab

VIII.클라우드 네트워크

❖특사(Ambassador) 패턴



https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

JS Lab

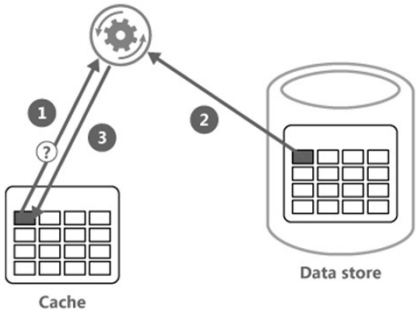
SDN/NFV for 5G

james@jslab.kr

177

VIII.클라우드 네트워크

❖Cache-Aside pattern



1: Determine whether the item is currently held in the cache.
2: If the item is not currently in the cache, read the item from the data store.
3: Store a copy of the item in the cache.

https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

JS Lab

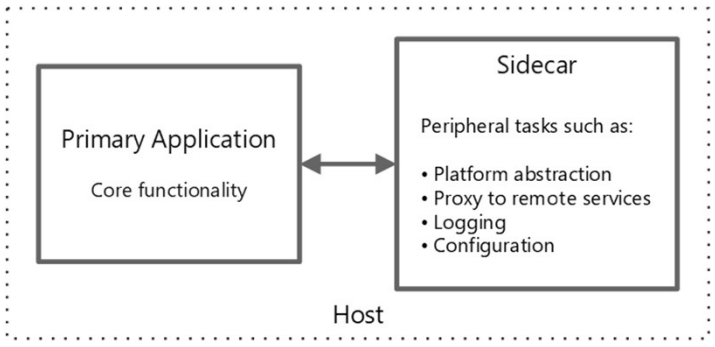
SDN/NFV for 5G

james@jslab.kr

178

VIII.클라우드 네트워크

❖사이드카(Sidecar) 패턴



SDN/NFV for 5G
james@jslab.kr

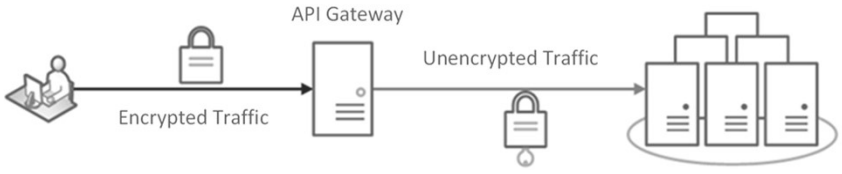
https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

JS Lab

179

VIII.클라우드 네트워크

❖게이트웨이 오프로딩 패턴



SDN/NFV for 5G
james@jslab.kr

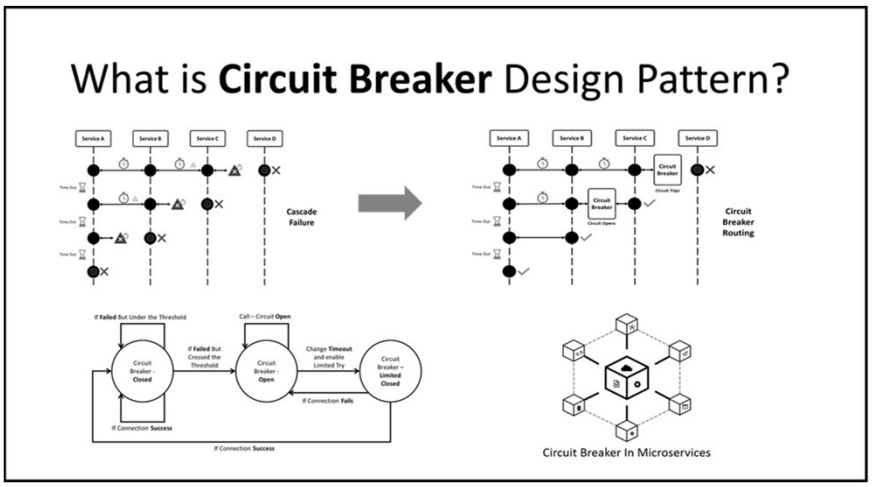
https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

JS Lab

180

VIII.클라우드 네트워크

❖**What is Circuit Breaker Design Pattern?**

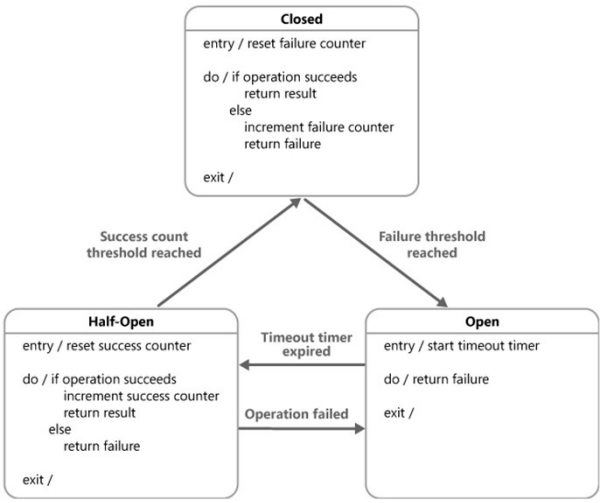


https://digitalvarys.com/what-is-circuit-breaker-design-pattern/?fbclid=IwAR1uukaeNh-vJZXeLhQP2MZvpeq7NVeNVKHmjNEPubzrH8vISlwr_Dedph8

181

VIII.클라우드 네트워크

❖**회로 차단기 패턴**



https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

182

IX. 텔레콤 환경을 위한 SDN/NFV

❖ 5G 코어(Core) 네트워크 아키텍처 변화

- Functional entities
- Single Core
- Dedicated protocols

- Service Based (SBA/SBI/NAPS)
- Virtualization & Slicing
- Softwarization/ Cloudification
- Application Programming Interfaces
- Harmonized protocols (HTTP …)
- Exposure to 3rdParties
- Backward & Forward Compatibility

Source: Georg Mayer
https://cloudifynetwork.com

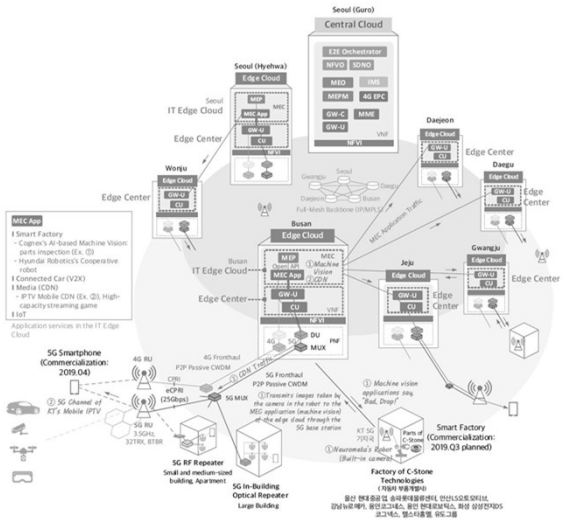
SBA: Service Based Architecture SBI: Service Base Interface NAPS: Northbound APIs

SDN/NFV for 5G

james@jslab.kr

IX. 텔레콤 환경을 위한 SDN/NFV

❖ 현재 통신사업자의 망 구성 (KT) – by Netmanias



JS Lab

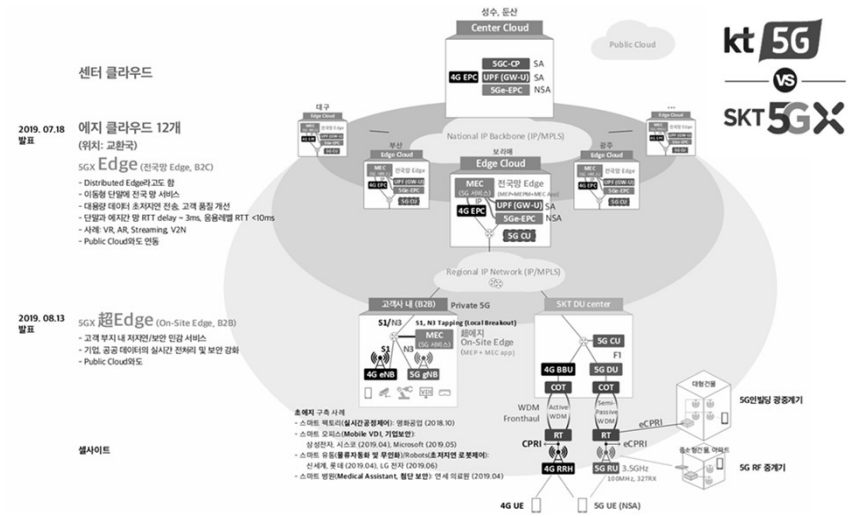
185

SDN/NFV for 5G

james@jslab.kr

IX. 텔레콤 환경을 위한 SDN/NFV

❖ 현재 통신사업자의 망 구성 (SKT) – by Netmanias



JS Lab

출처: https://www.netmanias.com/ko/?m=view&id=oneshot&no=14450

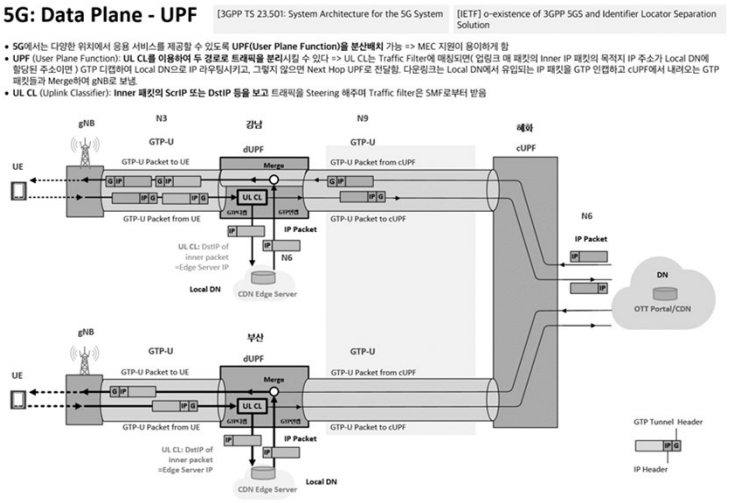
186

SDN/NFV for 5G

james@jslab.kr

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **5G Data Plane - UPF**



JS Lab

https://www.netmanias.com/ko/?m=view&id=oneshot&no=14003

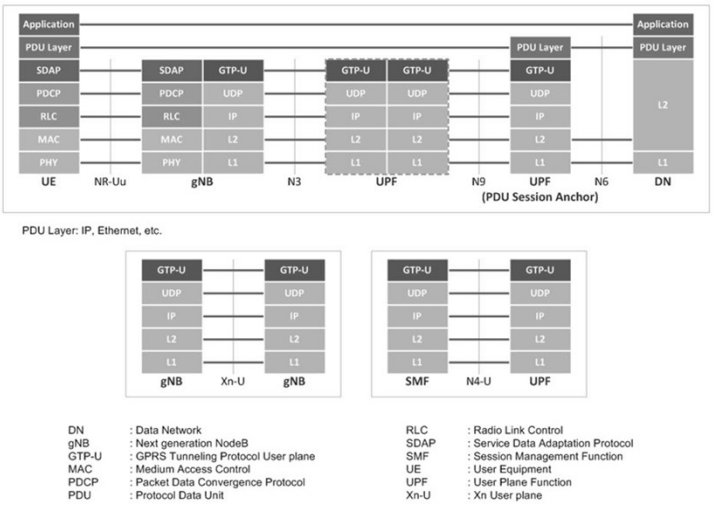
187

SDN/NFV for 5G

james@jslab.kr

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **5G Protocol Stack -** User Plane



DN	: Data Network	RLC	: Radio Link Control
gNB	: Next generation NodeB	SDAP	: Service Data Adaptation Protocol
GTP-U	: GPRS Tunneling Protocol User plane	SMF	: Session Management Function
MAC	: Medium Access Control	UE	: User Equipment
PDCP	: Packet Data Convergence Protocol	UPF	: User Plane Function
PDU	: Protocol Data Unit	Xn-U	: Xn User plane

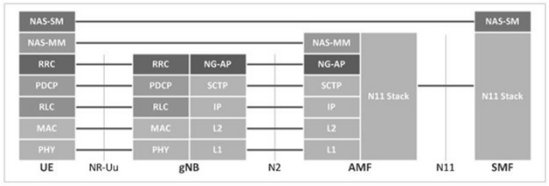
JS Lab

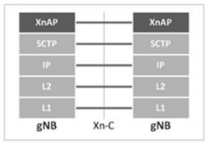
https://www.netmanias.com/ko/?m=view&id=oneshot&no=14036

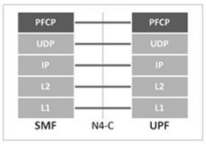
188

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **5G Protocol Stack -** Control Plane







AMF	: Access and Mobility Management Function	PDCP	: Packet Data Convergence Protocol	TLS	: Transport Layer Security
gNB	: Next generation NodeB	PFCP	: Packet Forwarding Control Function	UE	: User Equipment
MAC	: Medium Access Control	RLC	: Radio Link Control	UPF	: User Plane Function
NAS	: Non-Access Stratum	RRC	: Radio Resource Control	XnAP	: Xn Application Protocol
NAS-SM	: NAS – Session Management	SBI*	: Service Based Interface	Xn-C	: Xn Control plane
NAS-MM	: NAS – Mobility Management	SCTP	: Stream Control Transmission Protocol		
NG-AP	: Next Generation Application Protocol	SMF	: Session Management Function		

* SBI: Namf, Nsmf, Nudm, Nnrf, Nnssf, Nausf, Nnef, Npcf, Nudr, Nsmsf, Nudr, etc.

SDN/NFV for 5G

james@jslab.kr

JS Lab

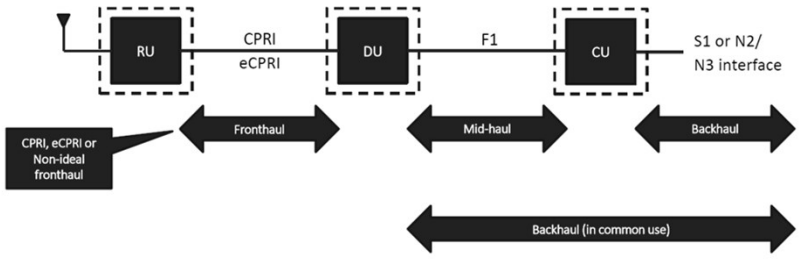
https://www.netmanias.com/ko/?m=view&id=oneshot&no=14036

189

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **프론트홀 / 백홀**

- **Fronthaul**
- **Mid-haul**
- **Backhaul**



SDN/NFV for 5G

james@jslab.kr

JS Lab

RU: Radio Unit DU: Digital Unit CU: Centralized Unit

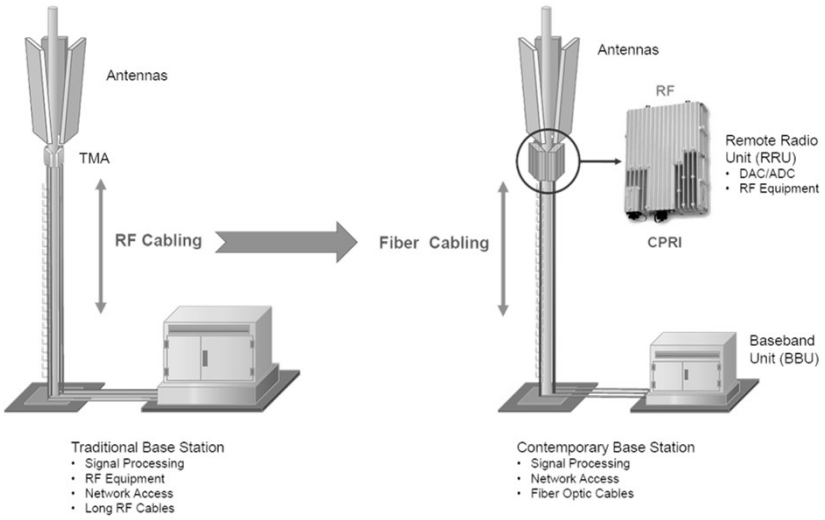
190

SDN/NFV for 5G

james@jslab.kr

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **Comparison of a Traditional and a Contemporary Base Station**



http://www.ni.com/white-paper/53051/en/

JS Lab

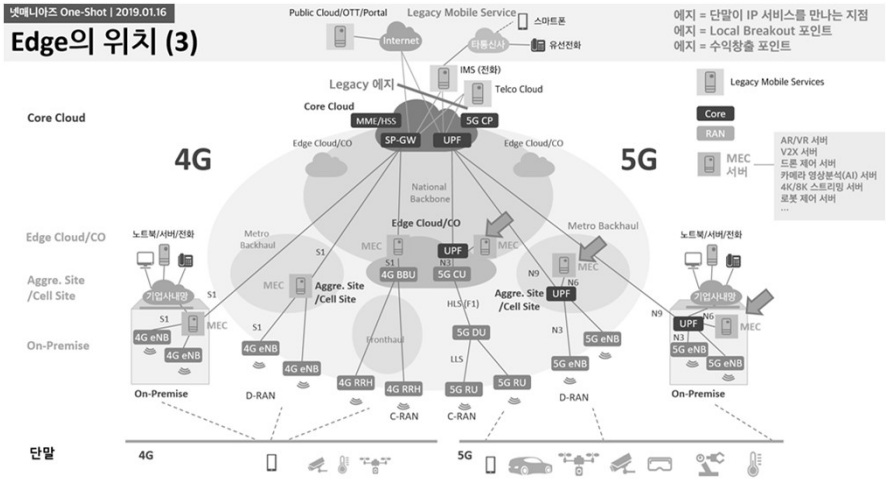
191

SDN/NFV for 5G

james@jslab.kr

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **에지 위치 비교**

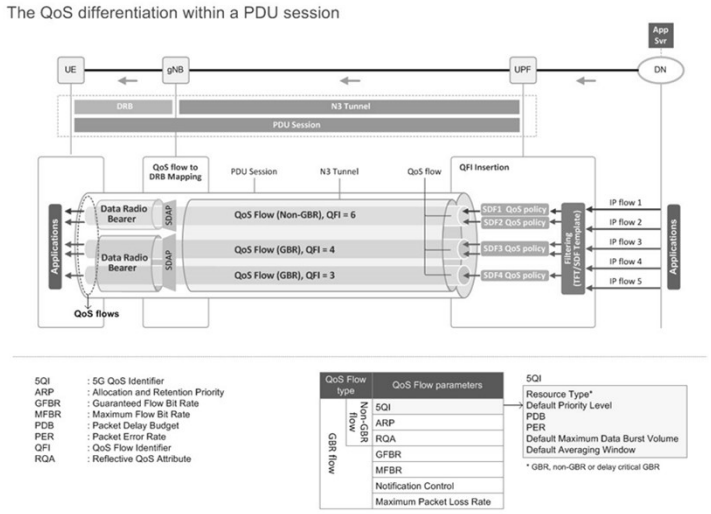


JS Lab

192

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **5G QoS**



5QI : 5G QoS Identifier
ARP : Allocation and Retention Priority
GFBR : Guaranteed Flow Bit Rate
MFBR : Maximum Flow Bit Rate
PDB : Packet Delay Budget
PER : Packet Error Rate
QFI : QoS Flow Identifier
RQA : Reflective QoS Attribute

QoS Flow type	QoS Flow parameters
GBR flow	5QI
Non-GBR flow	ARP
	RQA
	GFBR
	MFBR
	Notification Control
	Maximum Packet Loss Rate

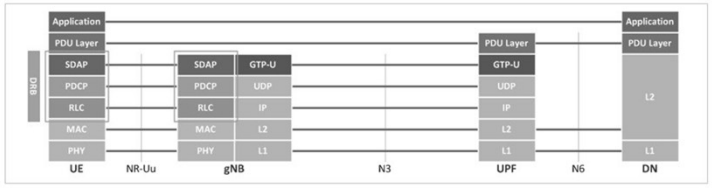
5QI
Resource Type*
Default Priority Level
PDB
PER
Default Maximum Data Burst Volume
Default Averaging Window
* GBR, non-GBR or delay critical GBR

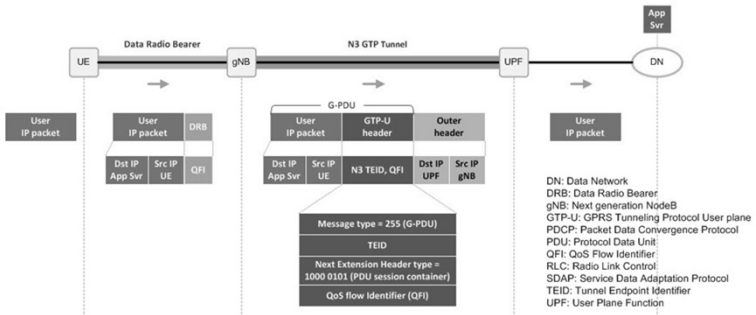
SDN/NFV for 5G
james@jslab.kr
JS Lab

193

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **5G Traffic Flow**





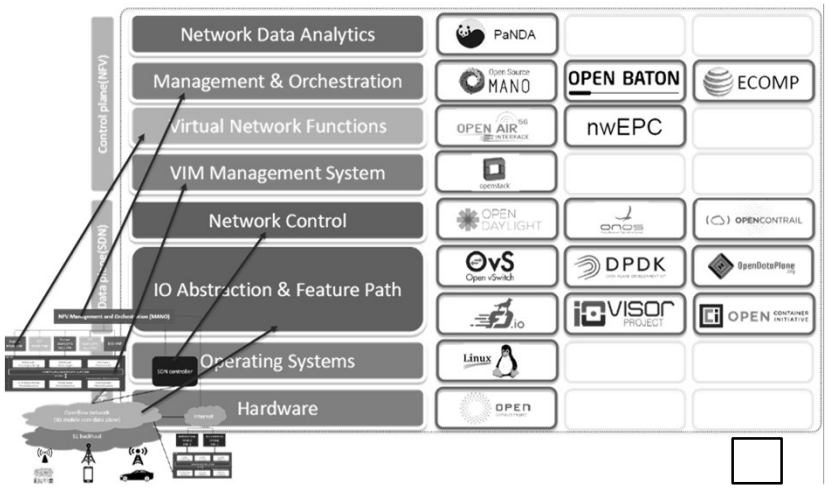
DN: Data Network
DRB: Data Radio Bearer
gNB: Next generation NodeB
GTP-U: GPRS Tunneling Protocol User plane
PDCP: Packet Data Convergence Protocol
PDU: Protocol Data Unit
QFI: QoS Flow Identifier
RLC: Radio Link Control
SDAP: Service Data Adaptation Protocol
TEID: Tunnel Endpoint Identifier
UPF: User Plane Function

SDN/NFV for 5G
james@jslab.kr
JS Lab

194

IX. 텔레콤 환경을 위한 SDN/NFV

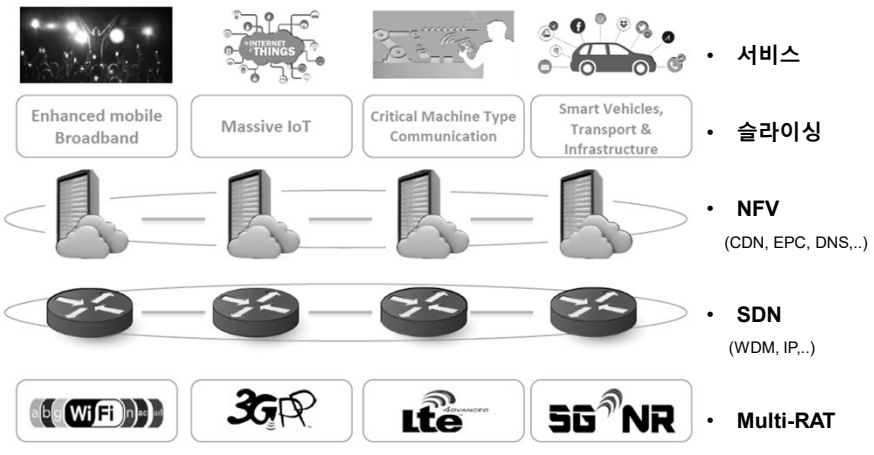
❖ **OpenSource Building Blocks For 5G**



195

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **5G 코어의 SDN/NFV**
❖ **필요 기능 구현을 위한 기술 설계**

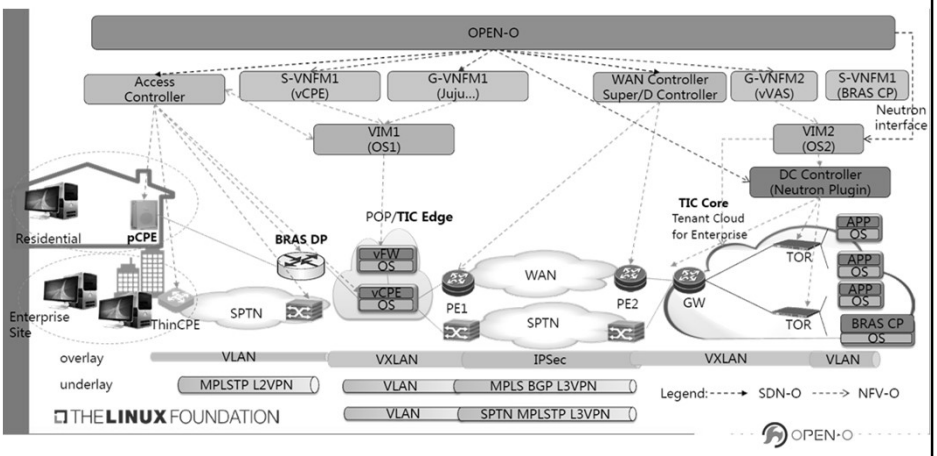


- **서비스**
- **슬라이싱**
- **NFV**
  (CDN, EPC, DNS,..)
- **SDN**
  (WDM, IP,..)
- **Multi-RAT**

196

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **종단간 SDN/NFV 서비스를 위한 오케스트레이터 필요** (OPEN-O)

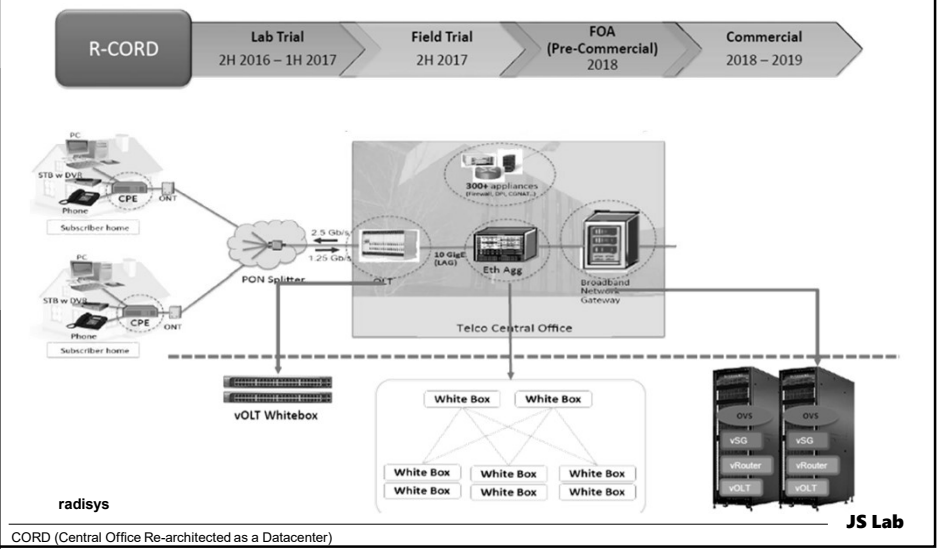


JS Lab

197

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **통신 시설의 변화 (데이터센터化)**



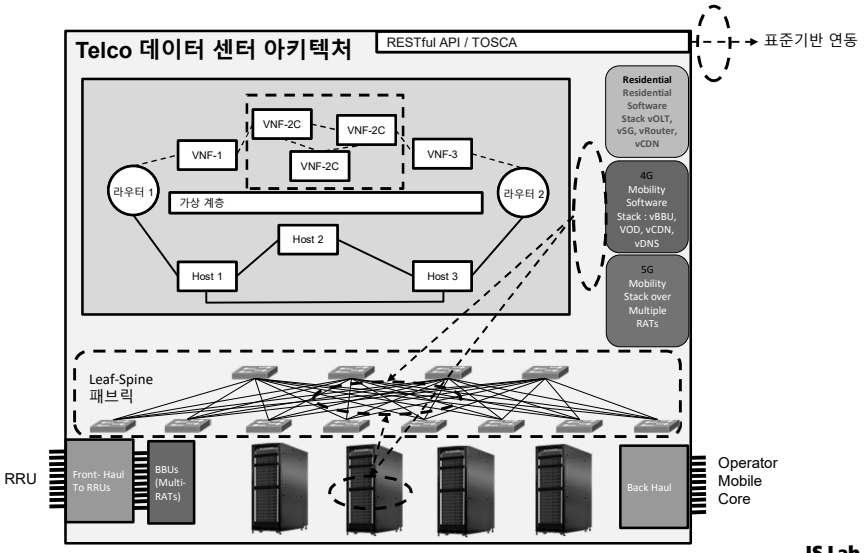
JS Lab

CORD (Central Office Re-architected as a Datacenter)

198

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **데이터센터化 하는 통신국사:** SDN 제어기, OpenStack, XOS and more



SDN/NFV for 5G

james@jslab.kr

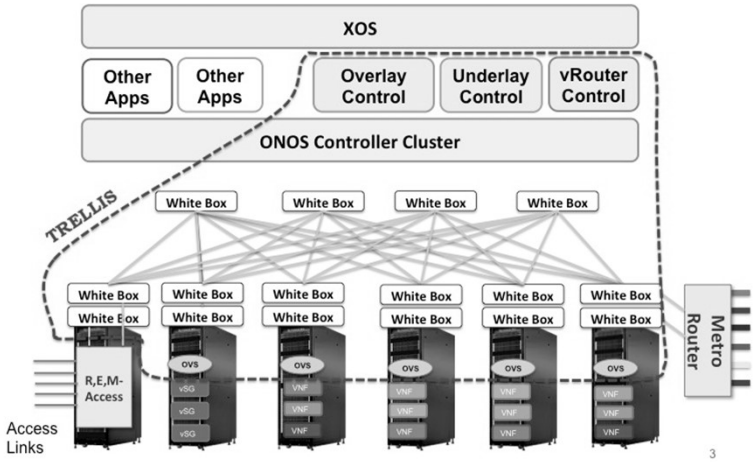
JS Lab

BBU (Baseband Unit) RRU (Remote Radio Unit) vSG (Virtual Subscriber Gateway) RAT (Radio Access Technology)

199

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **Trellis is a networking project designed to deliver a standard L2/L3 leaf-spine switching fabric for datacenters leveraging the ONOS Controller.**



SDN/NFV for 5G

james@jslab.kr

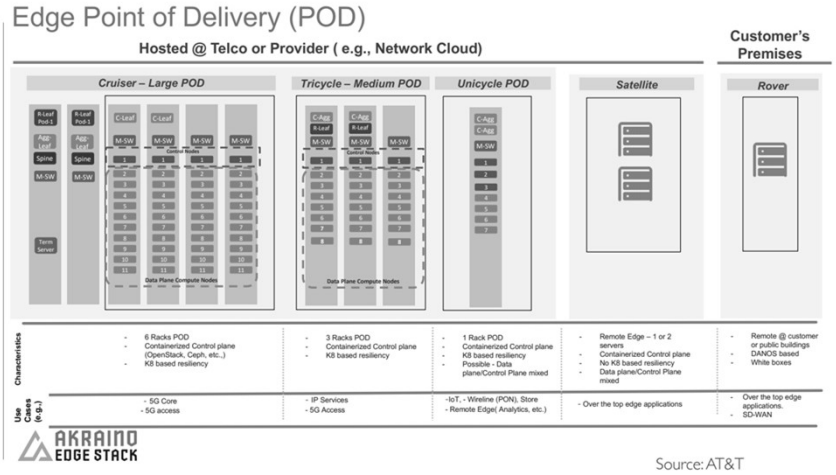
JS Lab

BBU (Baseband Unit) RRU (Remote Radio Unit) vSG (Virtual Subscriber Gateway) RAT (Radio Access Technology)

200

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **Edge Point of Delivery**

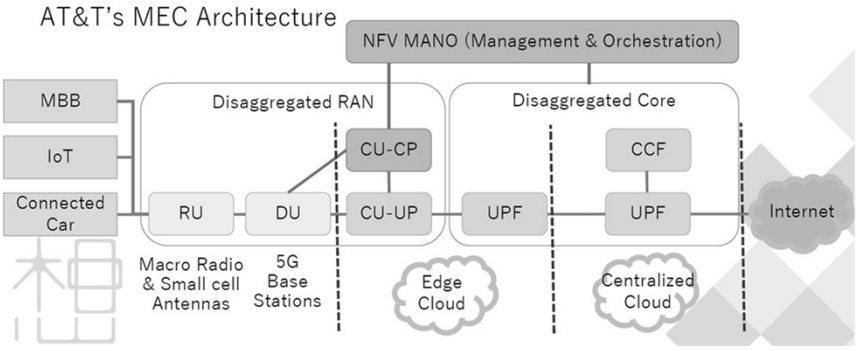


201

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **AT&T's MEC Architecture**

❖ **5G RAN and IoT on OpenStack based edge computing**



RU: Radio Unit
DU: Digital Unit

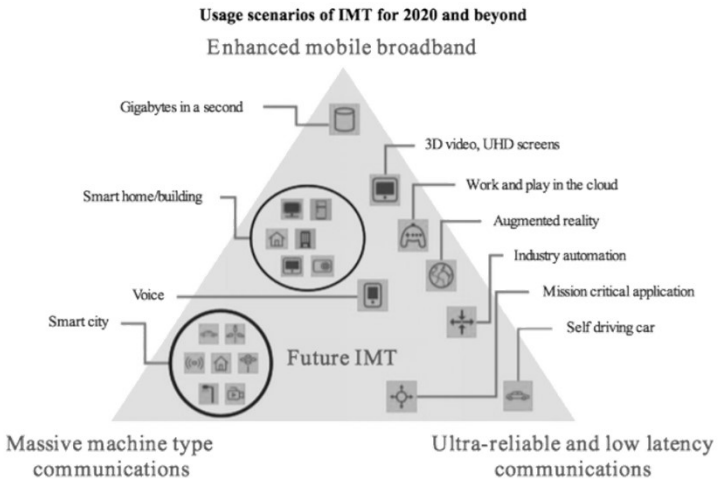
CU: Centralized Unit
CP: Control Plane
UP: User Plane

UPF: User Plane Function
CCF: Core Control Function

202

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **Usage scenarios of IMT-2020 and beyond (Source: ITU-R [4])**



http://556ventures.com

JS Lab

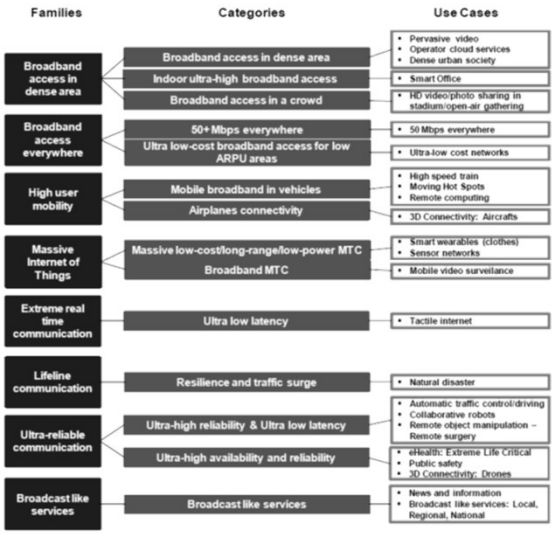
SDN/NFV for 5G

james@jslab.kr

203

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **5G Requirements**



http://www.mgclab.com/

JS Lab

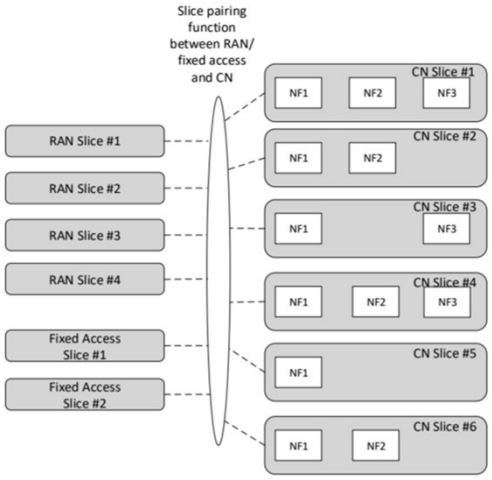
SDN/NFV for 5G

james@jslab.kr

204

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **Network Slicing Architecture**



SDN/NFV for 5G

james@jslab.kr

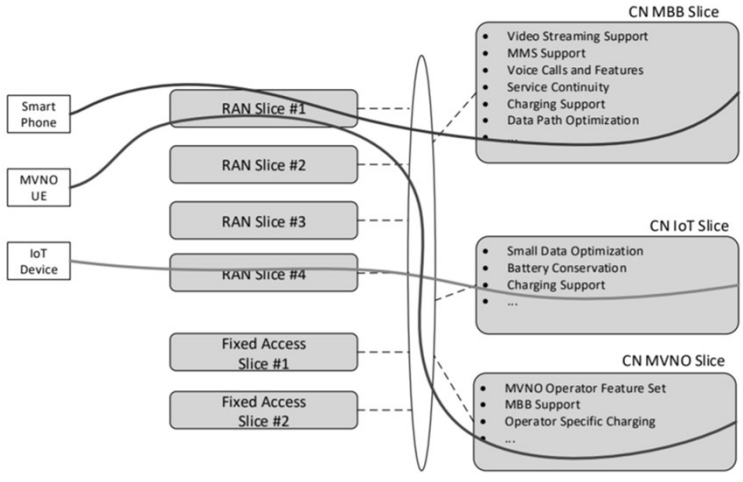
5G Americas White Paper – Network Slicing for 5G and Beyond

JS Lab

205

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **Network Slice Examples**



SDN/NFV for 5G

james@jslab.kr

5G Americas White Paper – Network Slicing for 5G and Beyond

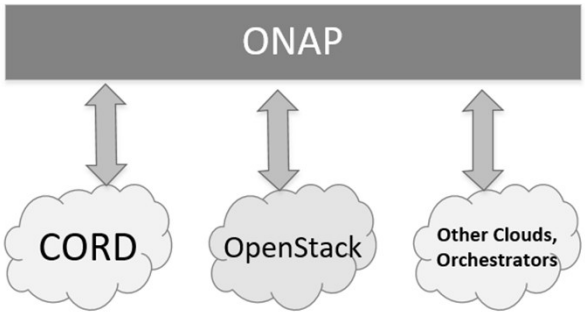
JS Lab

206

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **High-Level Relationship between ONAP and Cloud Platforms**

- **a high-level relationship between ONAP, CORD and other cloud platforms such as OpenStack.**



SDN/NFV for 5G
james@jslab.kr
JS Lab

207

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **ONAP (Open Network Automation Platform)**
❖ **Open Source ECOMP + OPEN-O joining forces**
❖ **Collaboration across providers, vendors and ecosystem with a coherent framework for Automation - Design, Orchestration, Management & Policy**

THE LINUX FOUNDATION

SDN/NFV for 5G
james@jslab.kr
www.onap.org
JS Lab

208

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **A Growing Ecosystem – ONAP members**

SDN/NFV for 5G
james@jslab.kr

JS Lab

209

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **ONAP has adopted a 6 months release cadence**

SDN/NFV for 5G
james@jslab.kr

JS Lab

210

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **ONAP is deployed using the ONAP Operations Manager (OOM)**

❖ **Requirements**

- 14 VM (1 Rancher, 13 K8s nodes) - 8 vCPU - 16 GB RAM
- 160 GB Storage
- 1 Rancher VM that also serves as a shared NFS server
- 3 etcd VMs for the Kubernetes HA etcd plane
- 2 orch VMs for the Kubernetes HA orchestration plane
- 12 k8s VMs for the Kubernetes HA compute hosts

Software	Version
Kubernetes	1.11.2
Helm	2.9.1
kubectl	1.11.2
Docker	17.03.x

SDN/NFV for 5G

james@jslab.kr

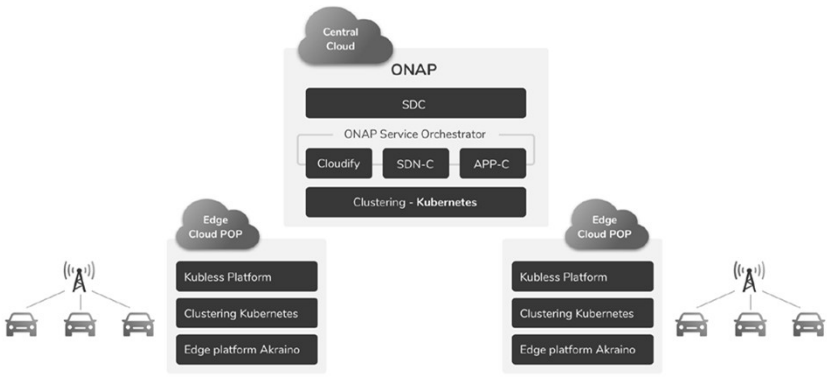
https://docs.onap.org/en/casablanca/guides/onap-developer/settingup/index.html#installing-onap

JS Lab

211

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **ONAP SDC, SO Orchestration and Monitoring (예)**



Kubeless is a Kubernetes-native serverless framework

Kata Containers is an open source project building extremely lightweight virtual machines that seamlessly plug into the containers ecosystem.

SDN/NFV for 5G

james@jslab.kr

www.akraino.org , www.starlingx.io Service Design and Creation (SDC) Service Orchestrator (SO)

JS Lab

212

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **A Growing Ecosystem – ONAP members**

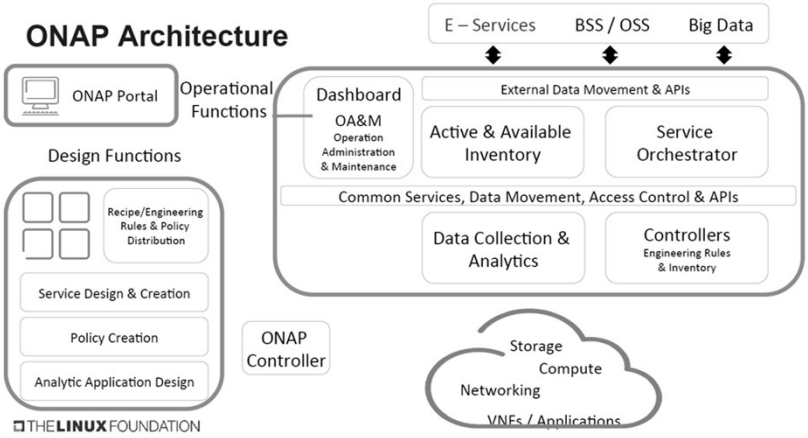
SDN/NFV for 5G
james@jslab.kr

JS Lab

213

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **ONAP Architecture:** Unified Platform, VNF Onboarding, Fully Automated Platform, Agile Operation, Support 5G/IoT Evolution, Improve Customer Experience



SDN/NFV for 5G
james@jslab.kr

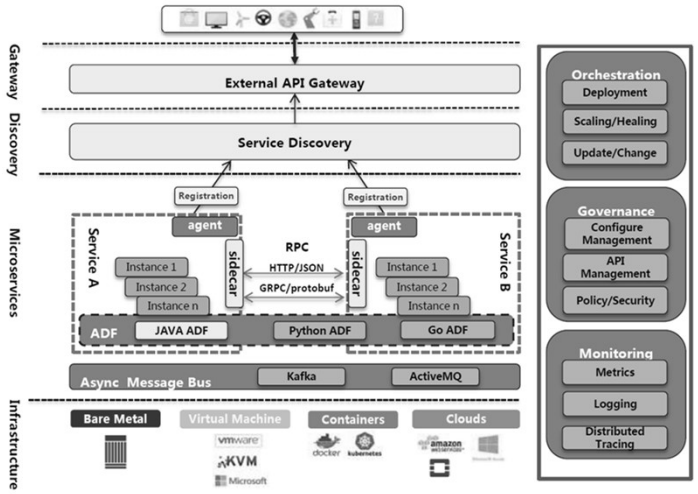
JS Lab

www.onap.org

214

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **OMSA - ONAP Microservice Architecture**



Note: this diagram is a functional view of OMSA, which is not mapped to specific projects

JS Lab

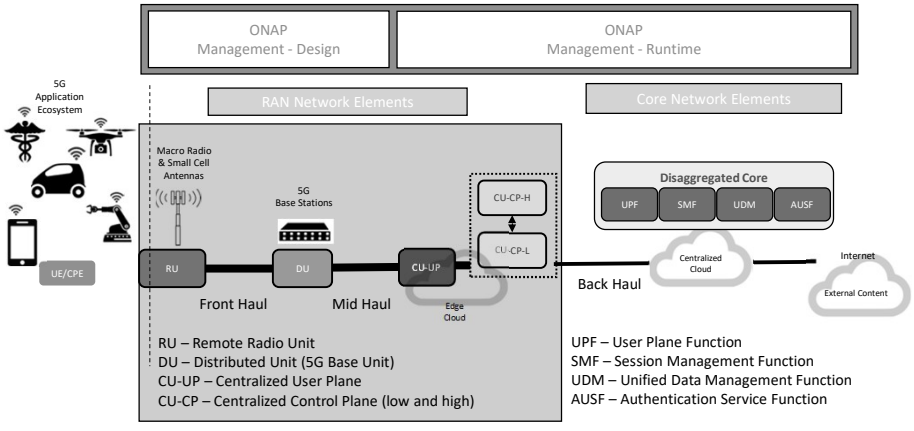
SDN/NFV for 5G

james@jslab.kr

215

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **5G Radio Access by ONAP – Network Architecture**



THE LINUX FOUNDATION

ONAP

JS Lab

SDN/NFV for 5G

james@jslab.kr

216

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **Components for Edge Computing (Edge Computing POC)**
❖ **日本仮想化技術株式会社**

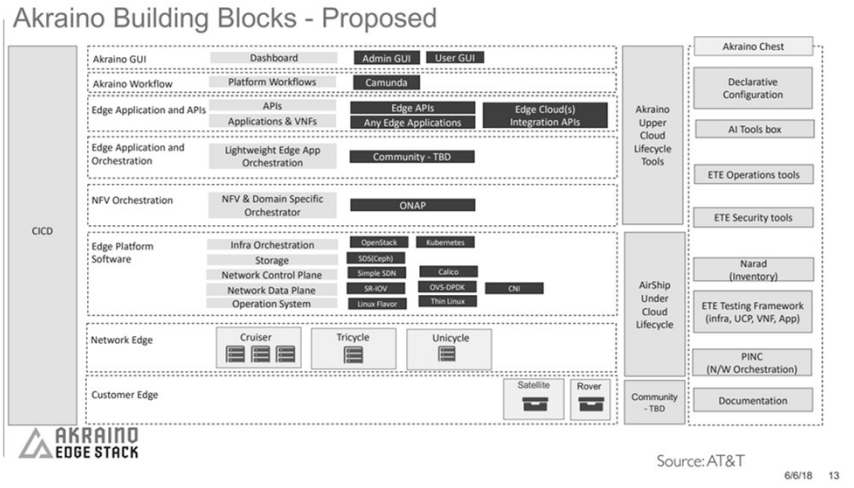
❖ **Components**

- ➢ **Edge Cloud**
  - **Edge Controllers**
    - ✓ Physical Provisioning: Ubuntu MAAS
    - ✓ Application Provisioning: Ubuntu Juju
    - ✓ Orchestrator: Kubernetes
    - ✓ SDN(Software Defined Network): Flannel
    - ✓ Monitoring/Alerting: Prometheus, Grafana
  - **Container nodes**
    - ✓ GPU Server
    - ✓ General Purpose Server: Intel and ARM Server

217

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **The Akraino Edge Stack code is contributed by AT&T and Intel Corporation:**



218

IX. 텔레콤 환경을 위한 SDN/NFV

❖ 빠른 변화

- ETSI의 MEC (Mobile|Multi-access Edge Computing)
- OpenStack Foundation의 'Edge Computing Group'
- Linux Foundation의 LF Edge

➢ ETSI

Mobile Edge Computing → Multi-access Edge Computing (2017)

➢ OpenStack Foundation

Massive distributed Working Group (2016) → Fog Edge Massively Distributed Cloud(FEMDC) SIG (2017) → Edge Computing Group (2018)

➢ Linux Foundation

LF Edge (2019)

- Akraino Edge Stack
- EdgeX Foundry (a common open framework for IoT edge computing)
- Open Glossary of Edge Computing
- Home Edge Project
- EVE (Edge Virtualization Engine, open and agnostic standard edge architecture)

https://www.openstack.org/edge-computing/cloud-edge-computing-beyond-the-data-center?lang=en_US

219

IX. 텔레콤 환경을 위한 SDN/NFV

❖ 리눅스 재단 (Linux Foundation)

❖ 통신사와 통신관련 제조사들은 다양한 오픈소스 프로젝트에서 활동

- LF AI Foundation (인공지능)
- LF Edge Foundation (에지)
- LF Networking Fund (오픈소스 네트워킹)
- Hyperledger (블록체인)

LF AI
LF EDGE
LF NETWORKING
HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

220

IX. 텔레콤 환경을 위한 SDN/NFV

- ❖ 리눅스 재단 (Linux Foundation)
- ❖ LF AI Foundation (인공지능)
  - 실시간 데이터 처리
  - 분산 환경

출처: https://landscape.lfai.foundation/

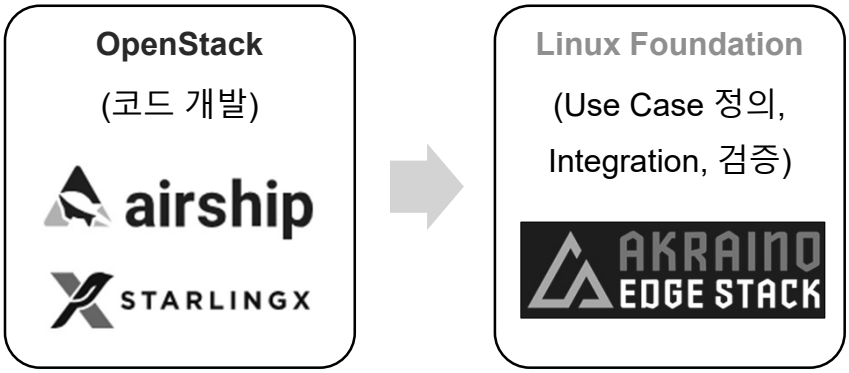
JS Lab

SDN/NFV for 5G
james@jslab.kr

221

IX. 텔레콤 환경을 위한 SDN/NFV

- ❖ Cloud Edge Computing: 단순 데이터센터 보다 큰 의미
- ❖ Akraino, Airship, StalingX (협력)
- ❖ Killer Service Solution 탑재 필요



출처: https://wiki.openstack.org/wiki/Edge_Computing_Group?fbclid=IwAR3GNTB5_2IOJO-SvaGsmhCC2jhLxG9X-ISt021v-mIfG-TxsR7jiPtrM80

JS Lab

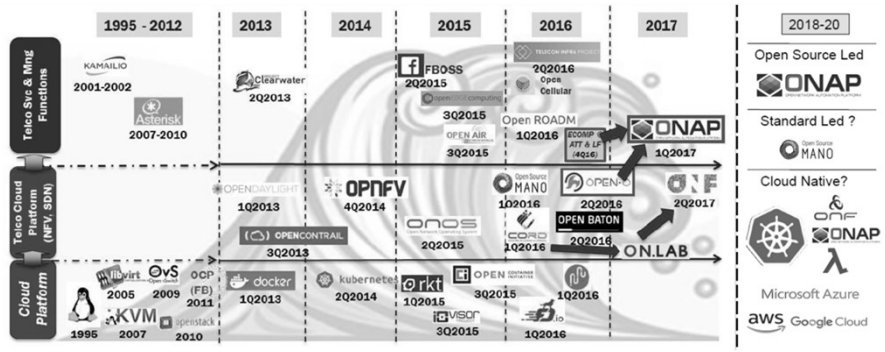
SDN/NFV for 5G
james@jslab.kr

222

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **운영지원 시스템(OSS)의 미래** (클라우드 네이티브화 진행중)

- **오픈소스 (예):** Open sources like Linux, OpenStack, KVM and others coming from the IT cloud platform, are becoming the foundation for a telco cloud platform based on network functions virtualization (NFV) and software-defined networking (SDN).



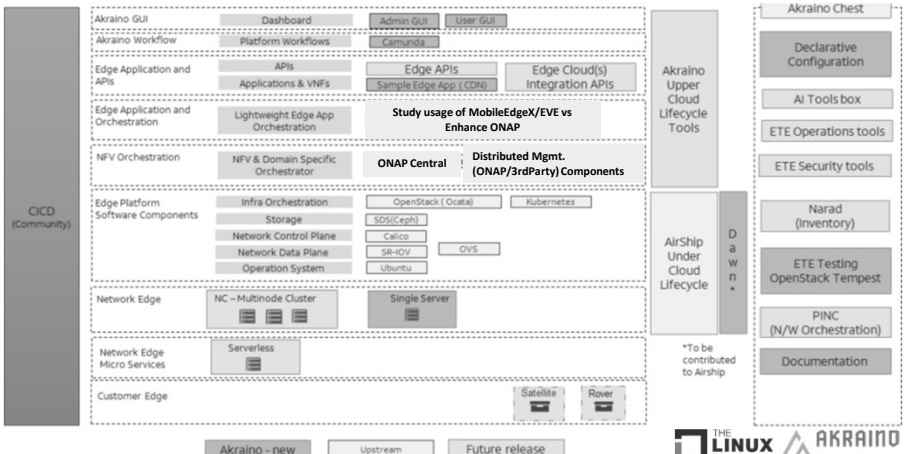
https://inform.tmforum.org/features-and-analysis/2017/03/open-source-future-oss/ **ONAP**(Open Networking Automation Platform)

IX. 텔레콤 환경을 위한 SDN/NFV

❖ **리눅스 재단의 에지 아키텍처 프로젝트 제안**

❖ **통신사의 기지국 및 기업 환경 API 제공 응용 시장** (예)

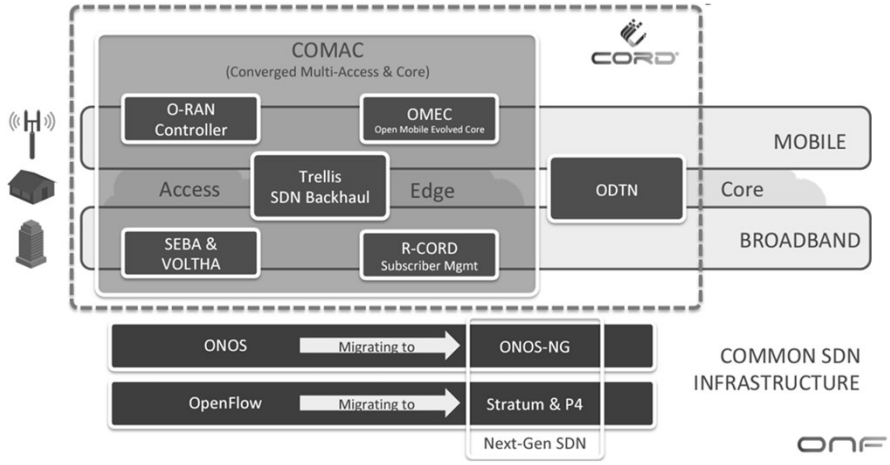
❖ **Akraino Edge Stack:** AT&T 와 Intel 제안



출처: www.akraino.org

IX. 텔레콤 환경을 위한 SDN/NFV

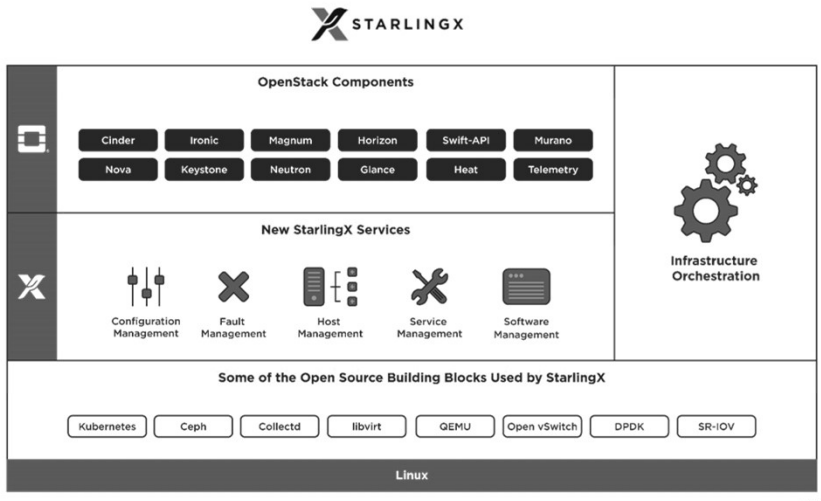
❖ **모바일과 브로즈밴드 통합 엑세스망 환경 고려** (예: COMAC)



COMAC(Converged Multi-Access & Core)

IX. 텔레콤 환경을 위한 SDN/NFV

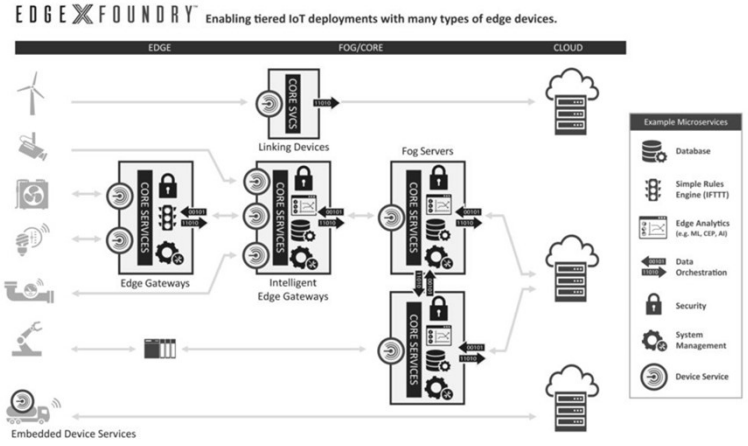
❖ **인프라 오케스트레이션 발전 방향 고려** (예: StarlingX)



출처: https://www.starlingx.io/

IX. 텔레콤 환경을 위한 SDN/NFV

- 적용 방법 고려 Reference 필요
- Optional Reference Services



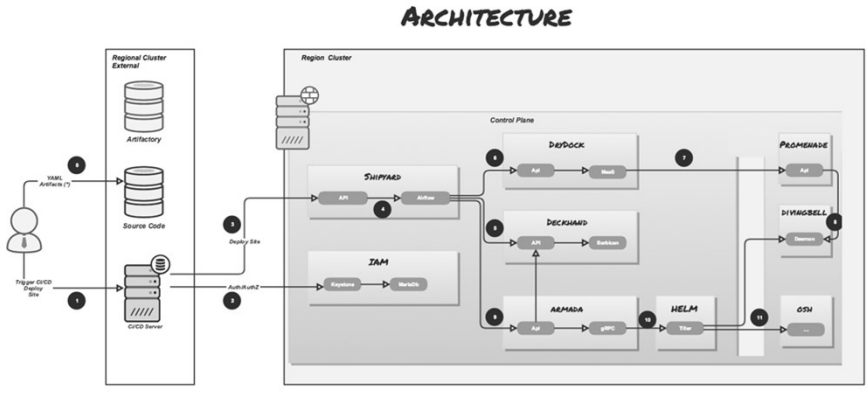
출처: https://www.edgexfoundry.org/about/

JS Lab

227

IX. 텔레콤 환경을 위한 SDN/NFV

- **국내의 리딩 오픈소스 프로젝트 레퍼런스(예) 'Airship':** a collection of components that coordinate to form means of configuring and deploying and maintaining a Kubernetes environment using a declarative set of yaml documents. More specifically, the current focus of this project is the implementation of OpenStack on Kubernetes (OOK).



출처: https://www.airshipit.org/

JS Lab

228

IX. 텔레콤 환경을 위한 SDN/NFV

- ❖ **LF Edge: 리눅스 재단이 2019년 1월 시작한 프로젝트**
- ❖ **60 members, including Arm, AT&T, Dell, Ericsson, IBM, Intel, Huawei, Red Hat, Samsung**
- ❖ **관리 프로젝트:**
  - **Akraino Edge Stack**
  - **EdgeX Foundry** (a common open framework for IoT edge computing)
  - **Open Glossary of Edge Computing**
  - **Home Edge Project**
  - **EVE** (Edge Virtualization Engine, open and agnostic standard edge architecture)

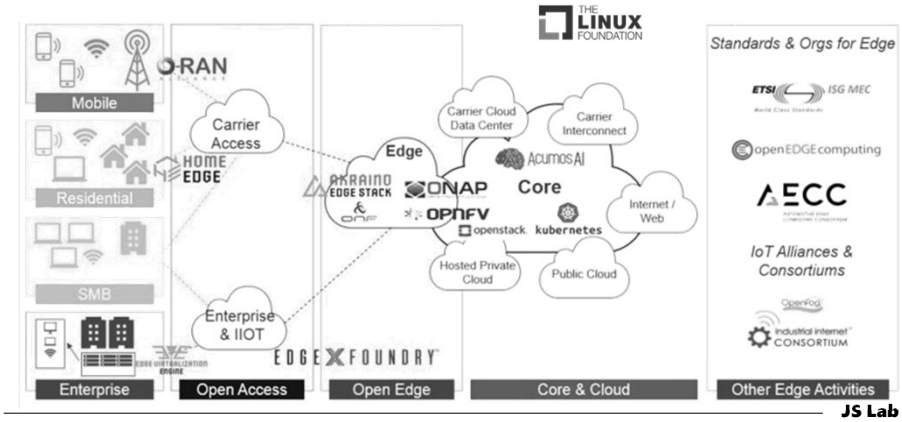
LF EDGE
AKRAINO EDGE STACK
HOME EDGE
OPEN GLOSSARY OF EDGE COMPUTING
EDGE VIRTUALIZATION ENGINE
EDGE X FOUNDRY

출처: https://www.lfedge.org/

229

IX. 텔레콤 환경을 위한 SDN/NFV

- ❖ **리눅스 재단의 에지 관련 프로젝트**
  - **Standards** (표준)
  - **Ref Arch** (레퍼런스 아키텍처)
  - **Ref Implementation** (적용 레퍼런스)



230

IX. 텔레콤 환경을 위한 SDN/NFV

❖ The Status of Open Source for 5G (1 of 2) - 5G Americas

5G Network Area	Focus	Brief Description	Open Source Effort References
Infrastructure	Hardware	High performance at lower cost by programmability and specialization of tasks	Open Compute Project: https://www.opencompute.org P4: https://p4.org
Infrastructure	Networking	Fast rate packet processing by acceleration techniques	DPDK: http://dpdk.org VPP: https://fd.io
Infrastructure	Operating System	Enabling white box use in carrier grade networks	Linux: https://www.linuxfoundation.org/projects/linux/ Berkle Software Distribution: http://www.bsd.org Disaggregated Network Operating System: https://www.danosproject.org
Access Network	Radio	Implementing 4G LTE and 5G Radio Access Network for NodeB and/or User Equipment	openair5G: https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/home O-RAN: https://www.o-ran.org/
Core Network	Wireless Core Network	Implementing 4G LTE EPC and 5G NGC	openairCN: https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/home M-CORD NGIC: https://software.intel.com/en-us/articles/an-interactive-demo-of-the-next-generation-infrastructure-core-reference-implementation
Management & Control	Networking	Carrier grade packet processing and flow control	OpenDaylight: https://www.opendaylight.org ONOS: https://onosproject.org Open vSwitch: https://www.openvswitch.org M-CORD NGIC: https://software.intel.com/en-us/articles/an-interactive-demo-of-the-next-generation-infrastructure-core-reference-implementation FD.io: https://fd.io
Management & Control	Virtualization	Abstraction of general compute resources to be shared across multiple applications and logical networks	OpenStack: https://www.openstack.org Kubernetes: https://kubernetes.io Docker: https://www.docker.com
Management & Control	Orchestration	Frameworks for describing dynamic function and network deployment policies with specific performance characteristics	Open Source MANO (OSM): https://osm.etsi.org MEF Lifecycle Service Orchestration (LSO): XOS: https://www.opennetworking.org/xos/

IX. 텔레콤 환경을 위한 SDN/NFV

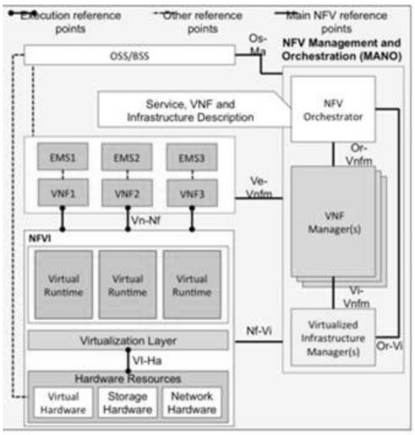
❖ The Status of Open Source for 5G (2 of 2) - 5G Americas

5G Network Area	Focus	Brief Description	Open Source Effort References
Management & Control	Automation	Frameworks and middleware for enabling Orchestration and Management tools to configure general compute and networking components via virtualization layers	xRAN: http://www.xran.org ONAP: https://www.onap.org Ansible: https://www.ansible.com Terraform: https://www.terraform.io/
Management & Control	Modeling	Modeling tools and languages for defining function and network services for deployment used by Orchestration Frameworks	TOSCA: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca JuJu: http://jujucharms.com YAML: http://yaml.org YANG: https://tools.ietf.org/html/rfc6020
Management & Control	DevOps	Software development methods to automate process of building, validating and deploying workloads into NFV environments for service agility	Elasticsearch, Logstash, Kibana (ELK): https://www.elastic.co/elk-stack Consul: https://www.consul.io Etcd: https://coreos.com/etcd/ Jenkins: https://jenkins.io/ Puppet: https://puppet.com Chef: https://www.chef.io/chef/
Management & Control	Testing Tools		
Management & Control	Analytics	Data streaming protocols for continuous analysis of the service monitoring	Apache Kafka: https://kafka.apache.org/ Apache Spark: https://spark.apache.org/
Management & Control	AI	Framework for use of AI in Network	Automation https://www.acumos.org/
Management & Control	Edge Compute	Open source software for Edge	Computing https://www.akraino.org/
Management & Control	Cybersecurity	Security framework for Virtual network infrastructures	SHIELD: https://torsec.github.io/shield-h2020/about/summary.html

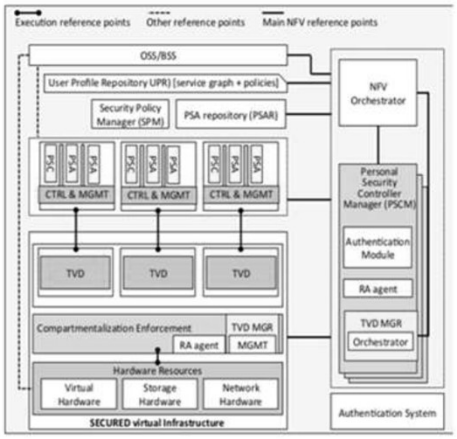
IX. 텔레콤 환경을 위한 SDN/NFV

- ❖ SECURED and ETSI NFV
- ❖ SECURity at the network Edge

ETSI NFV



SECURED



https://www.secured-fp7.eu/

JS Lab

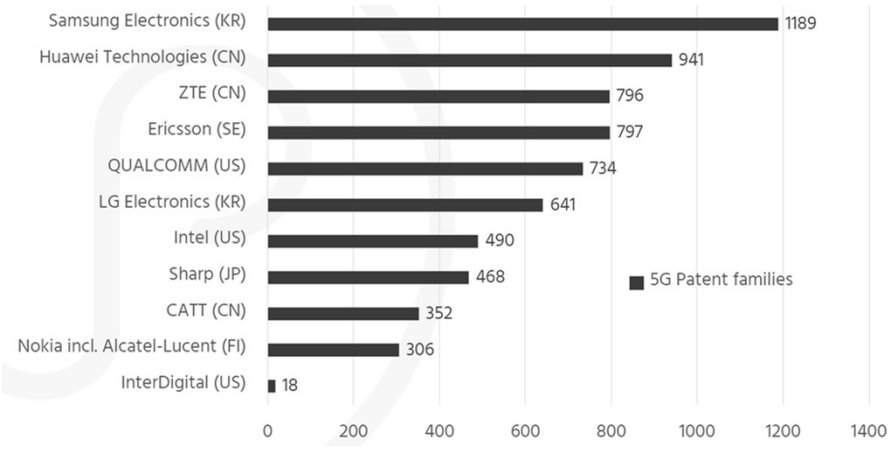
SDN/NFV for 5G

james@jslab.kr

233

IX. 텔레콤 환경을 위한 SDN/NFV

- ❖ 기업들의 5G 특허 소유



www.iplytics.com (2019)

JS Lab

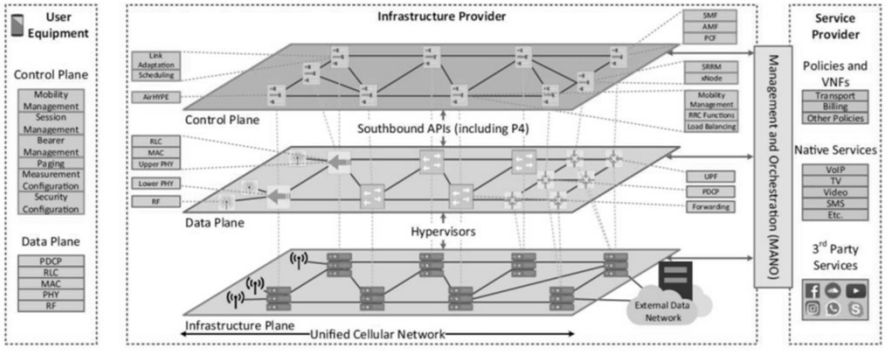
SDN/NFV for 5G

james@jslab.kr

234

IX. 텔레콤 환경을 위한 SDN/NFV

❖ 5G Cellular Network Architecture (mgclab)



The ARBAT architecture.

SDN/NFV for 5G

james@jslab.kr

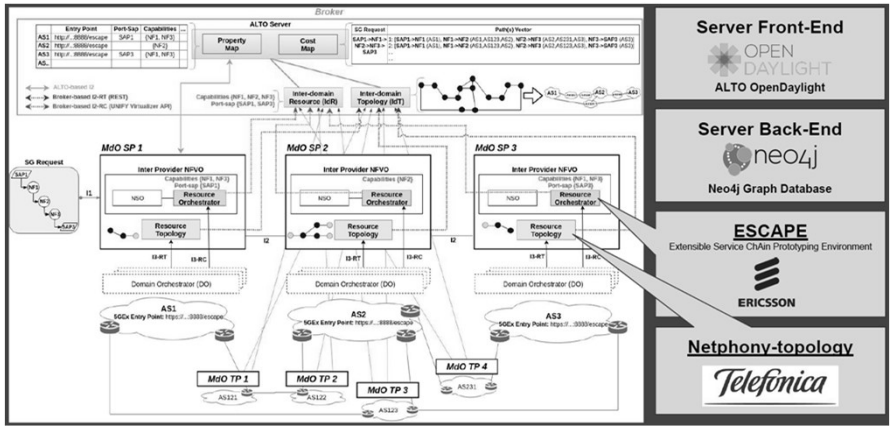
JS Lab

http://www.mgclab.com/

235

IX. 텔레콤 환경을 위한 SDN/NFV

❖ 5G-Ex: Use Case for ALTO protocol [RFC7285]
❖ Multi-domain Orchestration
❖ Leveraging the Application-Layer Traffic Optimization Protocol



SDN/NFV for 5G

james@jslab.kr

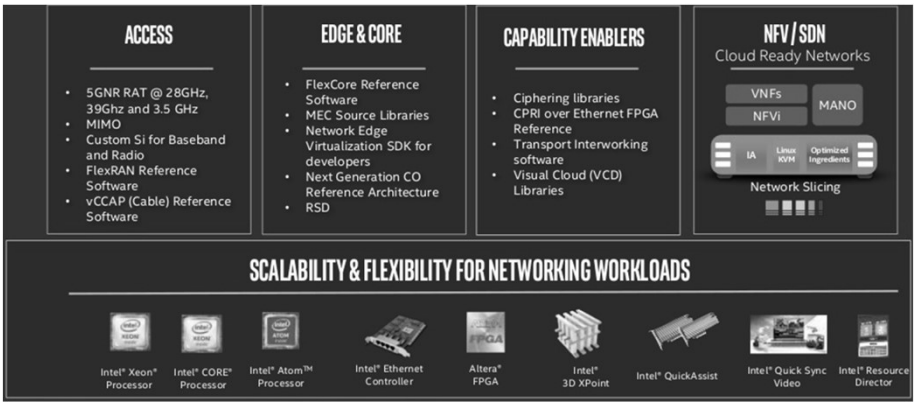
JS Lab

Workshop pre-IETF - CSBC 2018

236

IX. 텔레콤 환경을 위한 SDN/NFV

❖ 인텔의 5G 인프라 구성 블록



SDN/NFV for 5G

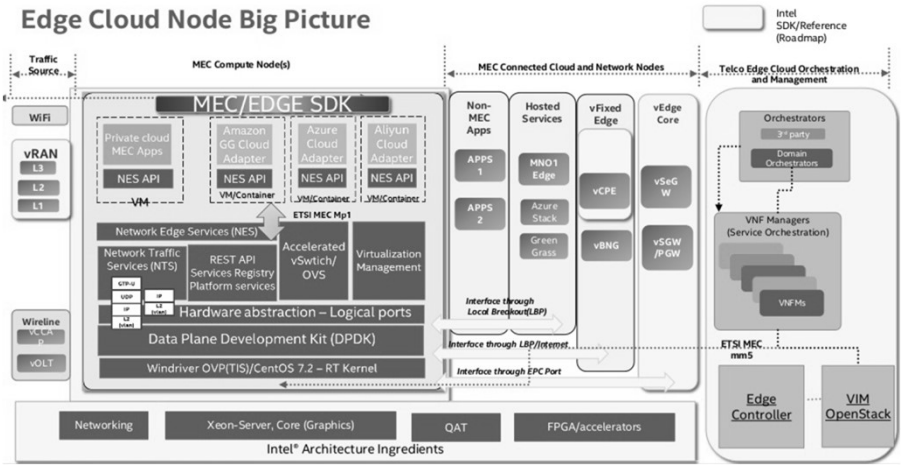
james@jslab.kr

JS Lab

237

IX. 텔레콤 환경을 위한 SDN/NFV

❖ 인텔의 에지 클라우드 구성



SDN/NFV for 5G

james@jslab.kr

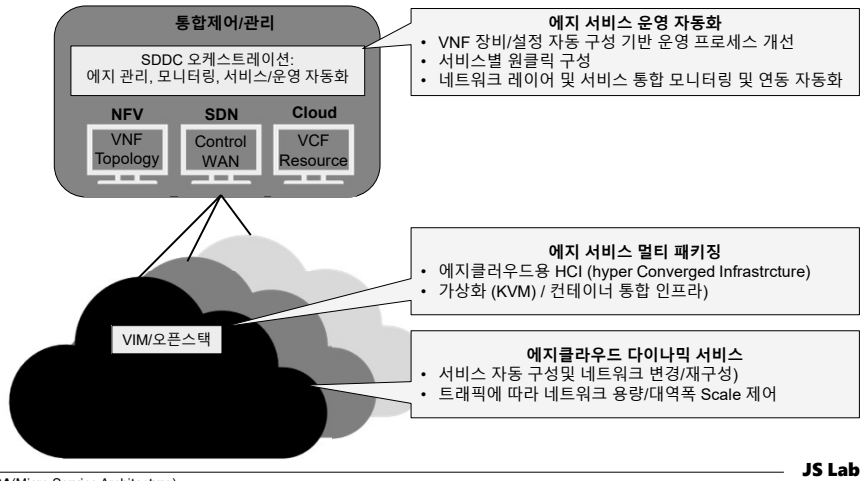
JS Lab

238

X. 관리

❖ 5G의 종단간 통합제어/관리 수용

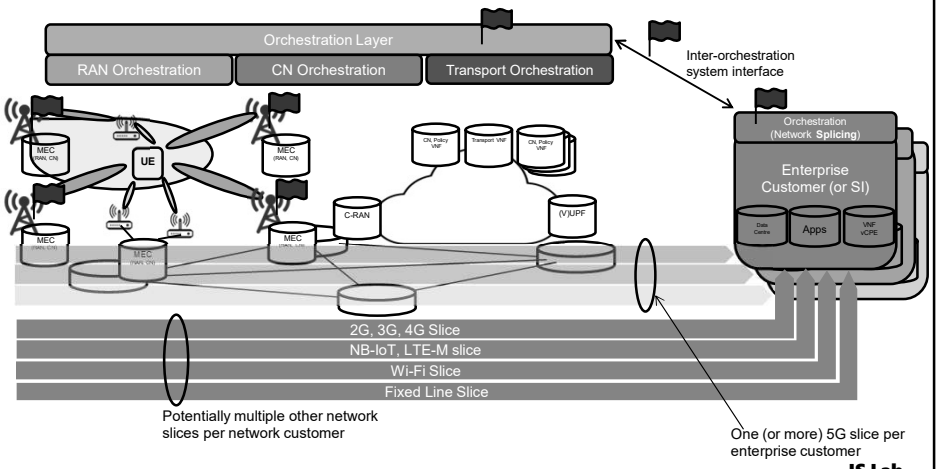
❖ 목표 아키텍처 MSA와 출시 가상화 서비스



MSA(Micro Service Architecture)

X. 관리

- **Technical Gap**
- **Pain points for commercial slicing** (by Samsung)
- **코어의 All IP 기반 제어 → SDN/NFV 기반 제어**



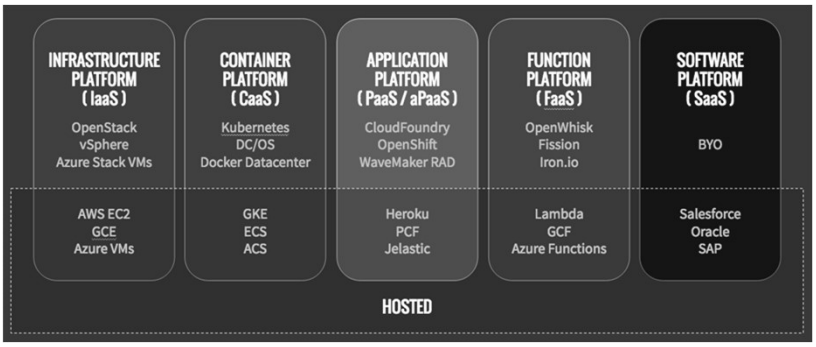
RAN (Radio Access Network), TN (Transport Network), and CN (Core Network)

241

X. 관리

- **멀티 클라우드 관리 기술 분류**

1. **클라우드 관리 플랫폼:** RedHat 'Cloudforms', IBM, Right Scale 'Cloud Management', VMware, Mopheus, SCALR
2. **애플리케이션 플랫폼:** RedHat 'OpenShift', Rancher, Cloud Foundry, Kubernetes
3. **자동화/오케스트레이션 플랫폼:** Teraform, Cloudify



https://dzone.com/articles/iaas-vs-caas-vs-paas-vs-faas-choosing-the-right-platform

242

X. 관리

❖ 멀티 클라우드 관리 기술 분류 별 장단점

	클라우드 관리 플랫폼 (Infrastructure First)	애플리케이션 플랫폼 (Application First Approach)	자동화/오케스트레이션 플랫폼 (Automation First)
장점	• Single Pane of Glass • 비용 분석과 제어	• 쿠버네티스(Kubernetes or K8s)와 컨테이너는 주요 클라우드에서 지원 • 애플리케이션과 마이크로서비스 중심	• 퍼블릭/프라이빗 클라우드등 다양하고 폭넓은 지원 • 모든 클라우드나 애플리케이션 지원 구조 • 커스터마이징 가능
단점	• 제한적인 애플리케이션 인식 • DevOps프로세스에 부적합 • 제한적인 클라우드 서비스	• 대부분 그린필드에 적합 • 다른 영역에 호환성 부족	• 애플리케이션과 클라우드 단위로 커스터마이징 필요
	RED HAT CLOUDFORMS, RIGHTSCALE CLOUD MANAGEMENT, IBM, vmware, MORPHEUS, SCALR	OPENSHIFT, RANCHER, CLOUD FOUNDRY, kubernetes	TERRAFORM, CLOUDIFY, ARIA, OASIS TOSCA

243

X. 관리

❖ 표준과 오픈소스

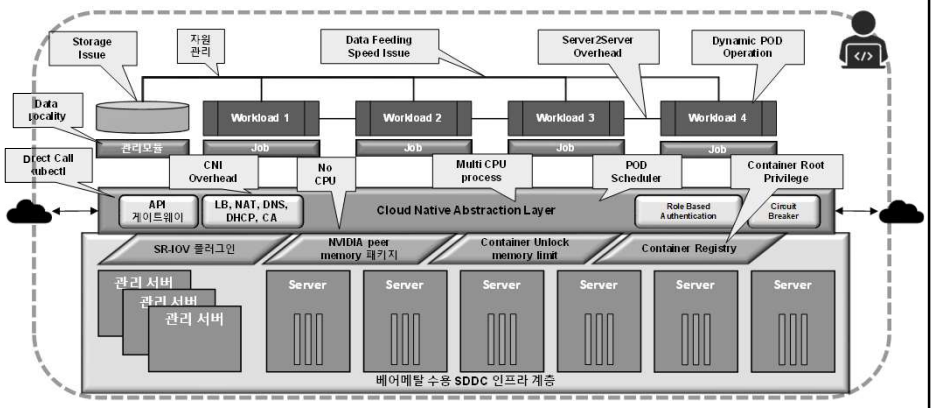
표준	오픈소스
OASIS Advancing open standards for the information society	OPEN-O, Open Source MANO
OASIS TOSCA Topology and Orchestration Specification for Cloud Applications (TOSCA)	CLOUDIFY, ARIA
tmforum, MEF METRO ETHERNET FORUM	ONAP OPEN NETWORK AUTOMATION PLATFORM
ETSI, IETF	

244

X. 관리

❖ 클라우드 네이티브 아키텍처 'CNA' 체계 운영/관리

- 클라우드 기반 서비스를 위한 K8s Based Platform
- 가상화 환경 이슈를 모니터링 관리하는 운영 체계



245

X. 관리

❖ Declarative(선언)는 무엇(What)와 어떻게(How)를 분리

❖ Declarative(선언)는 모델(What)와 오케스트레이터(How)를 정의

❖ Model은 Orchestrator에 충분한 정보를 일반적인 형식으로 제공

❖ 장점:

1. No Coding의 매력
2. 오케스트레이터 적용 독립적
3. 단순 버전 부여 가능한 가공물 관리
4. 설계는 가치와 솔루션에 집중

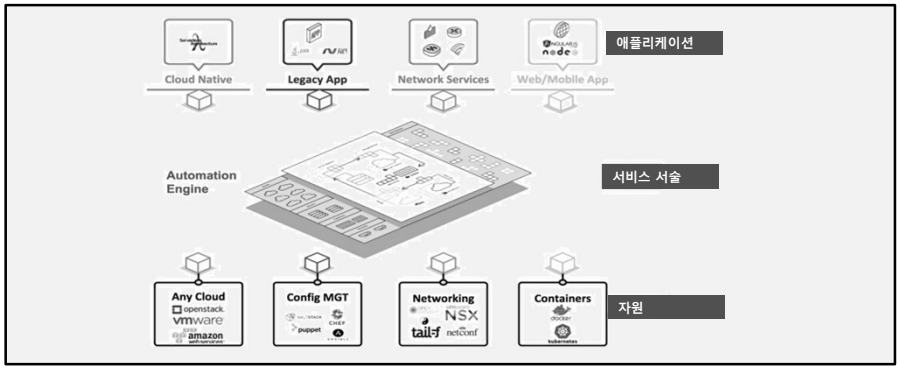
❖ 예: TOSCA, Docker Compose, Kubernetes, HOT, CloudFormation 등등

246

X. 관리

❖ **필요한 멀티클라우드 관리 기능 채택** (예)

1. **컨테이너 사용 가능**
2. **대부분의 주요 클라우드에서 지원하는 쿠버네티스(K8s) 사용**
3. **멀티 클라우드와 플랫폼 호환성과 통합이 가능한 자동화 프레임워크**
4. **애플리케이션의 요청에 따라 추상화한 인프라 자원을 제공**



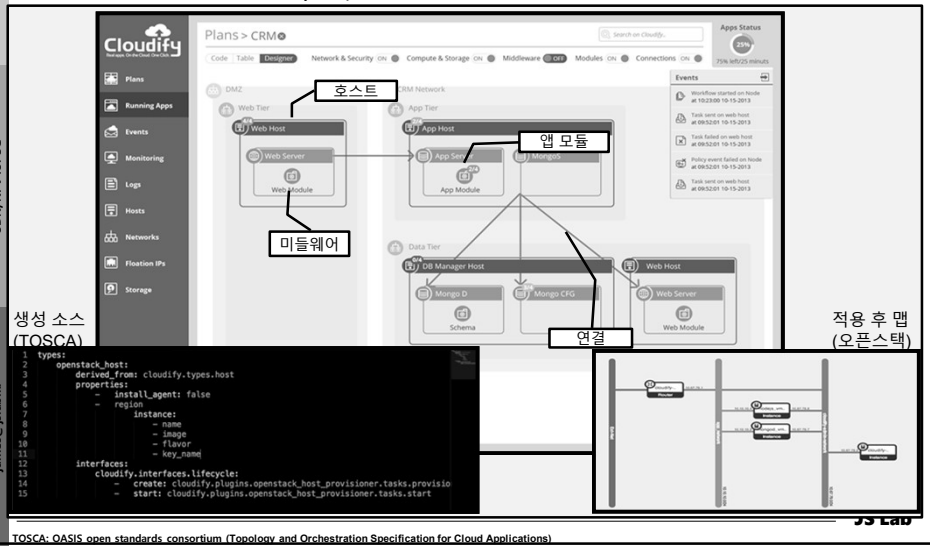
SDN/NFV for 5G
james@jslab.kr

JS Lab

247

X. 관리

❖ **멀티 클라우드 오케스트레이션 :** 표준 TOSCA 기반 GUI 서비스 (TOSCA 표준 적용 오픈소스 Cloudify 예)



TOSCA: OASIS open standards consortium (Topology and Orchestration Specification for Cloud Applications)

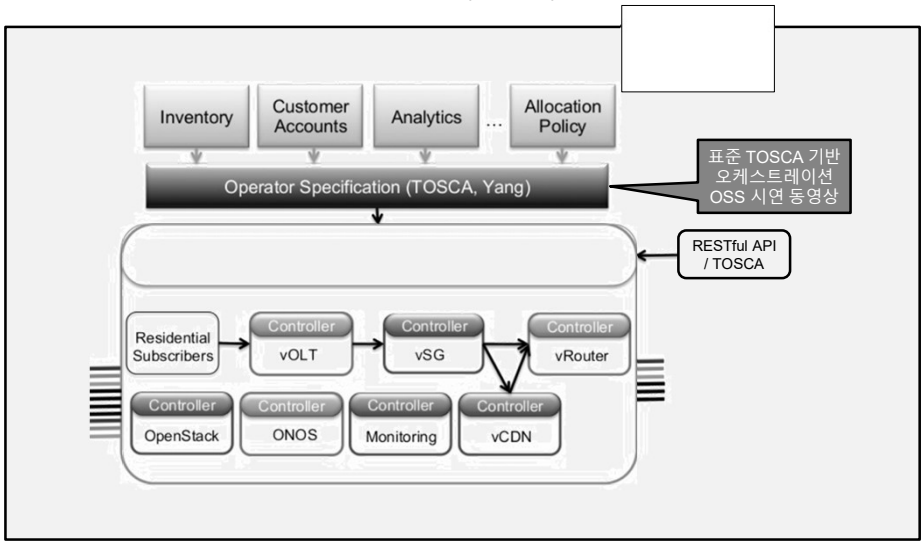
SDN/NFV for 5G
james@jslab.kr

JS Lab

248

X. 관리

❖ **표준 TOSCA 기반 오케스트레이션** (동영상)



SDN/NFV for 5G
james@jslab.kr

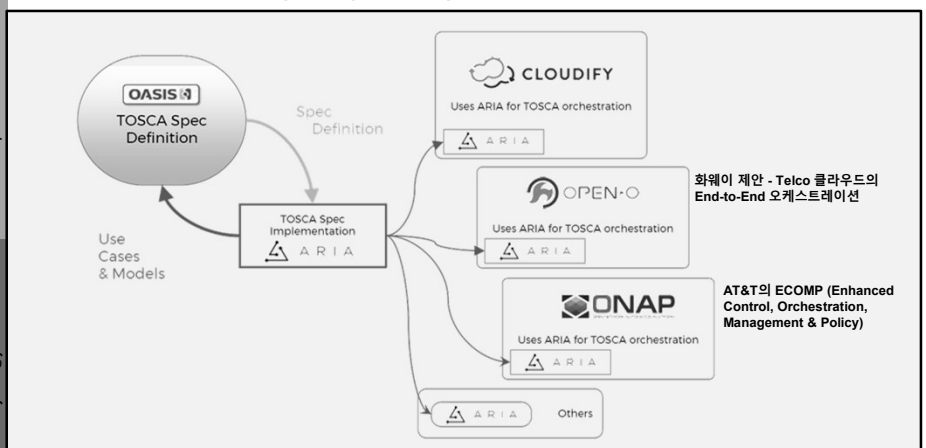
JS Lab

249

X. 관리

❖ **표준 TOSCA 스펙 적용 오픈소스 'ARIA'**

1. **오케스트레이션이 TOSCA 프로파일 지원을 위한 Python 라이브러리**
2. **TOSCA 애플리케이션 생성을 위한 SDK**
3. **CLI Tools: 오케스트레이션을 위한 TOSCA 템플릿**



SDN/NFV for 5G
james@jslab.kr

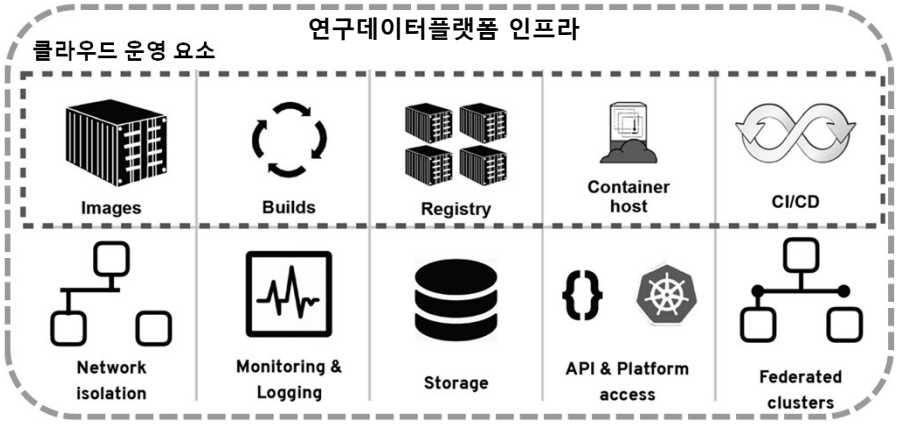
JS Lab

250

X. 관리

❖ 클라우드 인프라 구성 요소 고려 보안

- SDDC 추상화 운영 요소: 컴퓨팅, 스토리지, 네트워크, 관리, API 접속
- 클라우드 운영 요소: Images, Builds, Registry, Container Host, CI/CD



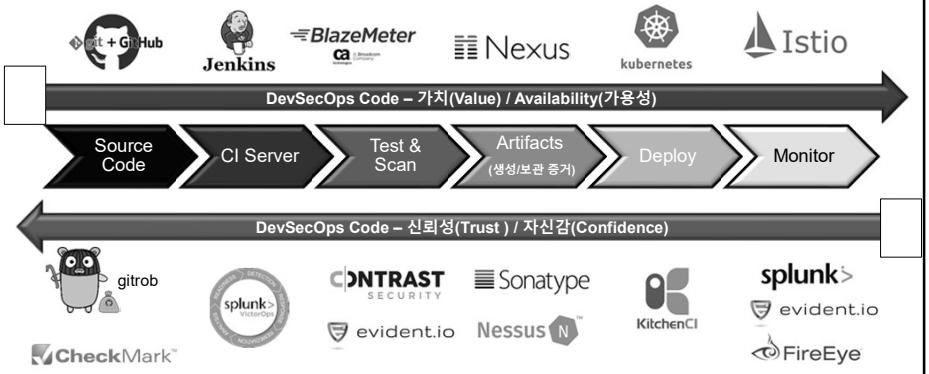
251

X. 관리

❖ 클라우드 인프라 요소 보안

❖ CNA 기반 단계별 보안 자동화 체계 (DevSecOps)

- Images, Builds, Registry, Container Host, CI/CD 등의 오픈소스 도구를 위한 전문 보안 솔루션 적용

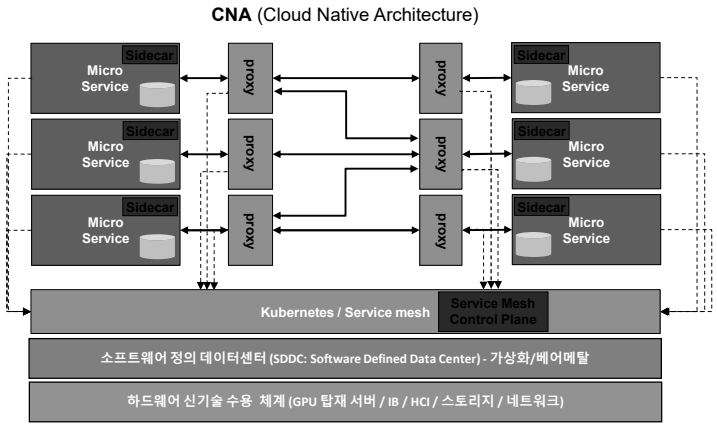


252

X. 관리

❖ 서비스 메시 관리 수용 체계

- Sidecar Design Pattern: 라우터 내장 CB, LB, SD 내장
- 서비스 메시용 오픈소스인 CNCF의 'Istio'는 정책 강화 Telemetry 제공



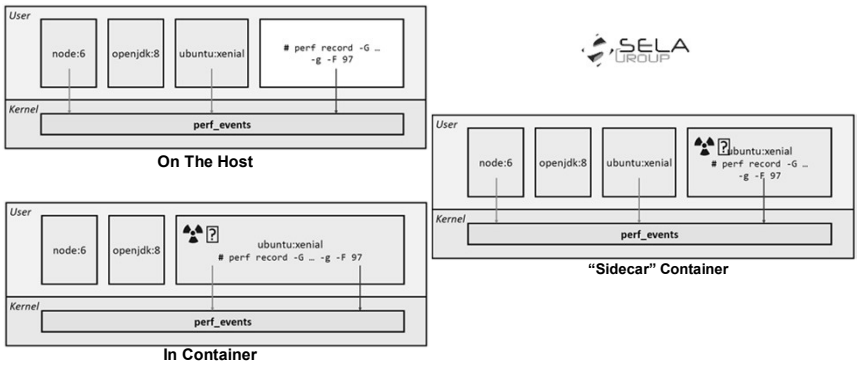
Circuit Breaker(CB), Load Balancer(LB), Service Discovery(SD)

253

X. 관리

❖ Tool Deployment

- 호스트 내 설치 (On The Host)
- 컨테이너 내 설치 (In Container)
- 도구 전용 컨테이너 ("Sidecar" Container)

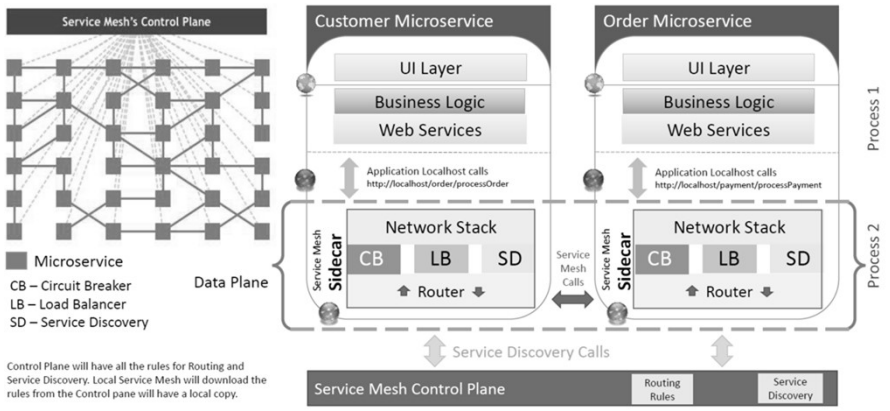


254

X. 관리

❖ Istio

❖ Service Mesh – Sidecar Design Pattern



https://github.com/meta-magic/kubernetes_workshop

JS Lab

SDN/NFV for 5G

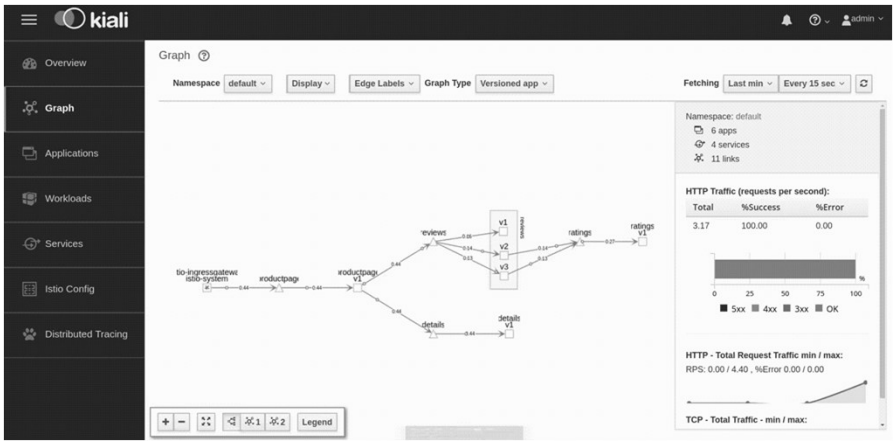
james@jslab.kr

255

X. 관리

❖ 클라우드 네이티브 서비스 메시 관리도구 'Istio'

- Service mesh is a network communication infrastructure layer for a microservices-based application.



JS Lab

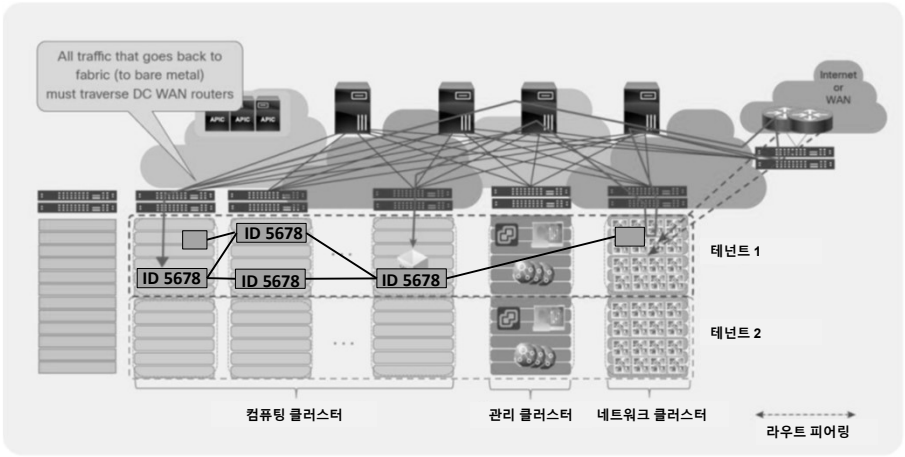
SDN/NFV for 5G

james@jslab.kr

256

X. 관리

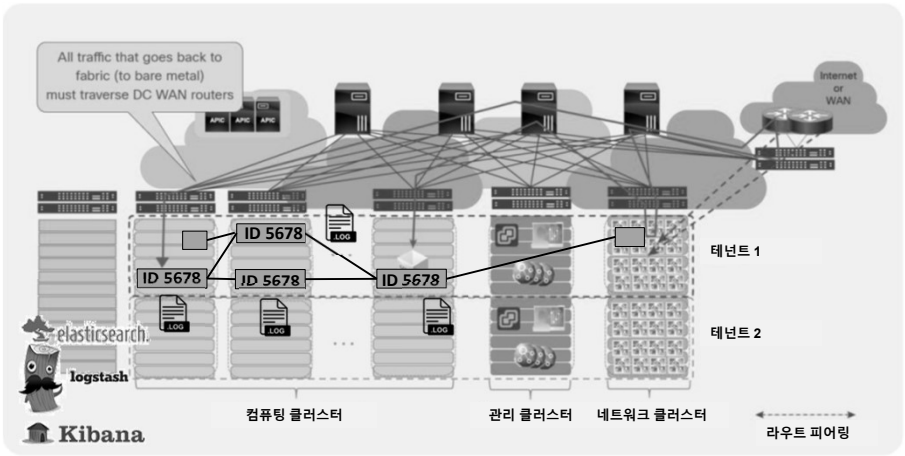
❖ Correlation IDs over Physical Infrastructure



257

X. 관리

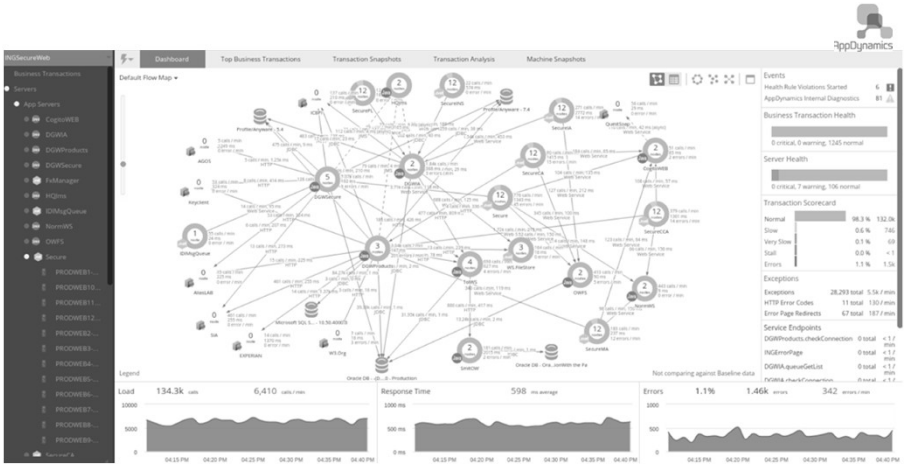
❖ Centralized logging



258

X. 관리

❖ **monitoring**



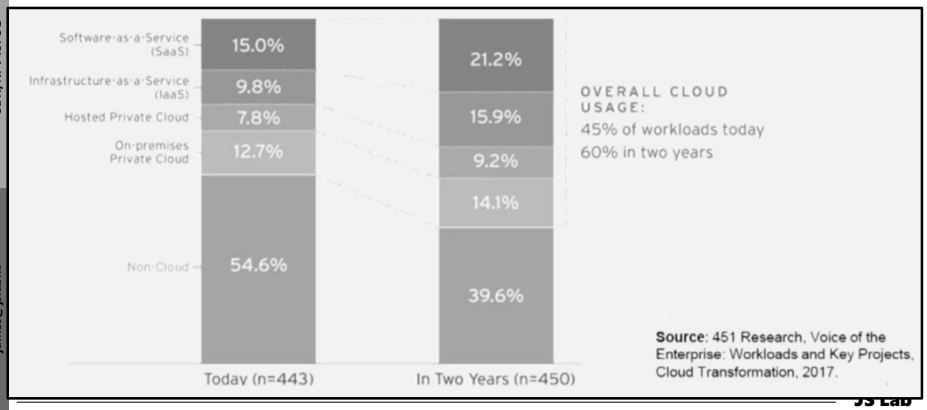
https://www.zdnet.com/article/appdynamics-launches-analytics-collaboration-tools/

X. 관리

❖ **통합 관리 필요**

❖ **멀티 클라우드 (Multi-Cloud):** 여러 개의 애플리케이션을 위해 여러 클라우드 서비스를 사용

❖ **하이브리드 클라우드 (Hybrid Cloud):** 하나의 애플리케이션(서비스)이 여러 플랫폼(Public, Private, On-premises)에 연결



X. 관리

❖클라우드의 실행 자원 단위는 컨테이너

❖컨테이너 보안 - Application Container Security Guide (NIST)

- NIST(National Institute of Standards and Technology) 특별 공개 '800-190'
- 컨테이너에 의해 실행 하는 애플리케이션 지원은 새로운 방법의 개발, 실행, 지원을 위해 조직의 운영 문화와 기술 처리를 위해 보완해야 함
- 범용 OS 보다 '컨테이너 전용 호스트 OS'가 공격 표면을 축소 할 수 있어 사용을 권장
- 동일 목적/민감성/위협 대응의 컨테이너 들을 하나의 로스트 커널에 적용하여 수준을 높이는 추가 방어를 허용
- 컨테이너 전용 취약관리 도구를 사용하여 이미지 처리의 보안 강화
- 하드웨어 기반 신뢰제공 컴퓨팅을 기반으로 상응 보완하는 것을 고려
- 컨테이너를 인식하는 방어 도구를 사용

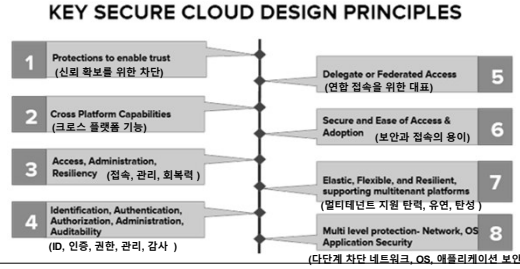
SDN/NFV for 5G
james@jslab.kr

261

X. 관리

❖ 보안 표준 적용 클라우드 구축 (해외 예)

1. ISO/IEC 27001:2013
2. ISO/IEC 27002:2013
3. ISO/IEC 27017:2015
4. SOC 1/SOC 2/SOC 3
5. NIST SP 800-53
6. PCI DSS



Securing the Telco Cloud
June 2017 @ OPNFV Summit

Ng Hwee Ming
Principal Technologist
APAC Office of Technlogy, Red Hat

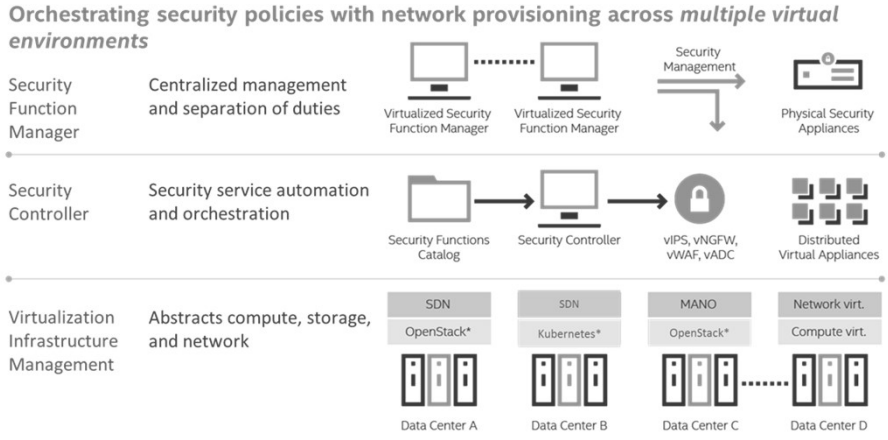
Abhilash S V
Cloud Solution Architect, Red Hat

SDN/NFV for 5G
james@jslab.kr

262

X. 관리

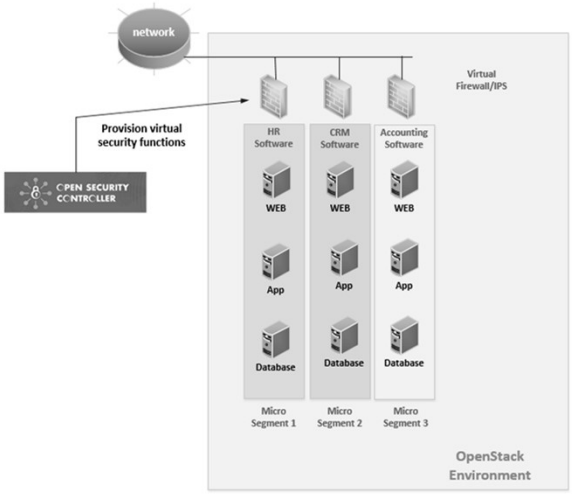
❖ **Open Security Controller Conceptual Architecture**



263

X. 관리

❖ **Open Security Controller - Microsegmentation**



264

X. 관리

❖ **TOSCA Concepts**

1. **모델러와 오케스트레이터의 역할을 정의**
2. **오브젝트 모델링 기반 (Java like) :** 시스템에 요구되는 객체(Object)들을 보여줌으로써 주로 적용시스템의 정적인 구조를 포착하는 데 사용
3. **모델을 적용에 연결(Bind)하는 API**
4. **오케스트레이터로 명령 가능한 워크플로우를 모델링한 인스턴스 Runtime을 해석**

SDN/NFV for 5G

james@jslab.kr

JS Lab

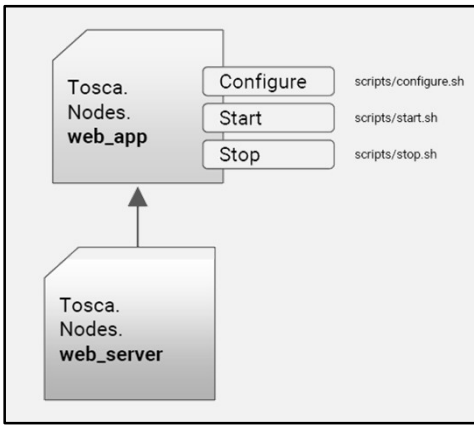
Runtime: 컴퓨터 프로그램이 실행되고 있는 동안의 동작

265

X. 관리

❖ **표준 TOSCA 적용 'ARIA' Hello World 예 (TOSCA/ARIA)**

1. **소스 공개**
2. **helloworld.yaml**




```
1.  tosca_definitions_version: tosca_simple_yaml_1_0
2.  node_types:
3.    HelloWorld:
4.      derived_from: tosca:WebApplication
5.      requirements:
6.        - host:
7.            # Override to allow for 0 occurrences
8.            capability: tosca:Container
9.            occurrences: [ 0, UNBOUNDED ]
10. topology_template:
11.   inputs:
12.   node_templates:
13.     hello_world:
14.       type: HelloWorld
15.       capabilities:
16.         app_endpoint:
17.           properties:
18.             protocol: http
19.             port: 9090
20.       interfaces:
21.         Standard:
22.           configure: scripts/configure.sh
23.           start: scripts/start.sh
24.           stop: scripts/stop.sh
25.   outputs:
26.     port:
27.       type: integer
28. value: { get_property: [ hello_world, app_endpoint, port ] }
```

SDN/NFV for 5G

james@jslab.kr

JS Lab

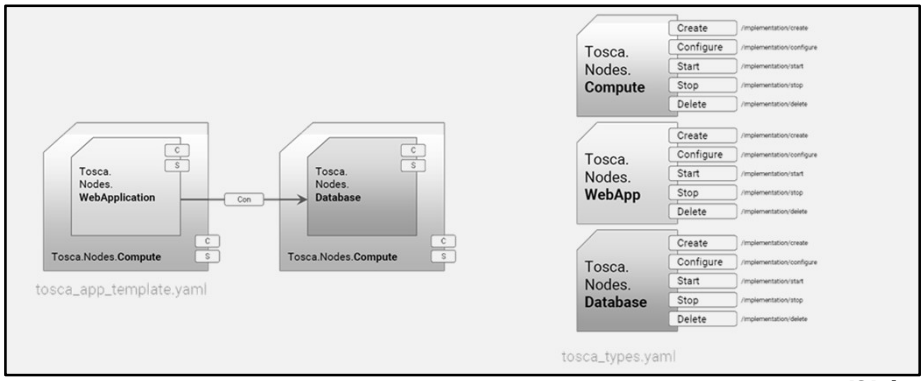
https://github.com/apache/incubator-ariatosca/blob/master/examples/hello-world/hello-world.yaml

266

X. 관리

❖ **TOSCA Types**

1. 기준 노드 Type (Compute, Network, etc)
2. 관련성 Type
3. 라이프사이클 운영과 적용
4. 커스텀 Type (ARIA 지원)



267

X. 관리

❖ **TOSCA .yaml**

Node Type

Inputs

Node Templates

Requirements and Capabilities

Lifecycle Interfaces

Outputs

```
1.  tosca_definitions_version: tosca_simple_yaml_1_0

2.  node_types:

3.    HelloWorld:
4.      derived_from: tosca:WebApplication
5.      requirements:
6.        - host:
7.            # Override to allow for 0 occurrences
8.            capability: tosca:Container
9.            occurrences: [ 0, UNBOUNDED ]

10. topology_template:
11.   inputs:

12.   node_templates:
13.     hello_world:
14.       type: HelloWorld
15.       capabilities:
16.         app_endpoint:
17.           properties:
18.             protocol: http
19.             port: 9090
20.       interfaces:
21.         Standard:
22.           configure: scripts/configure.sh
23.           start: scripts/start.sh
24.           stop: scripts/stop.sh

25.   outputs:
26.     port:
27.       type: integer
28. value: { get_property: [ hello_world, app_endpoint, port ] }
```

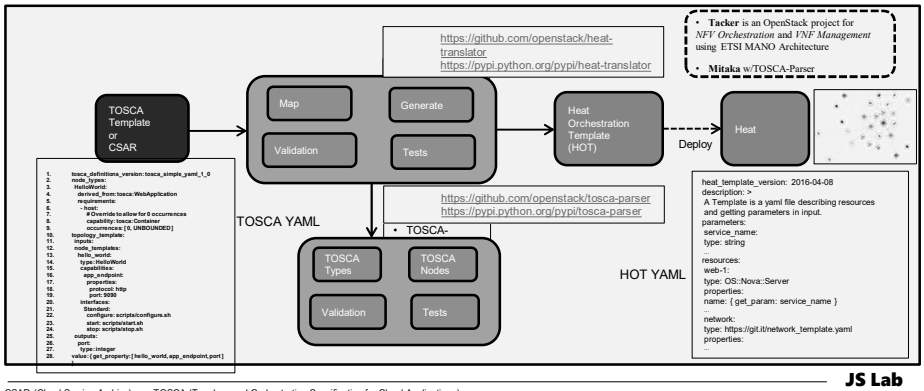

https://github.com/apache/incubator-ariatosca/blob/master/examples/hello-world/hello-world.yaml

268

X. 관리

❖ **TOSCA 처리 과정 (오픈스택 예)**

- **TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0 Committee Specification Draft 03 (17 March 2016)**
- **토스카 파서(TOSCA-Parser): Parser for TOSCA Simple Profile in YAML**
- **히트번역기(Heat-Translator): An OpenStack project to map and translate non-Heat (e.g. TOSCA) templates to Heat Orchestration Template (HOT)**



CSAR (Cloud Service Archive) TOSCA (Topology and Orchestration Specification for Cloud Applications)

X. 관리

❖ **도커 컨테이너 관리 (예)**

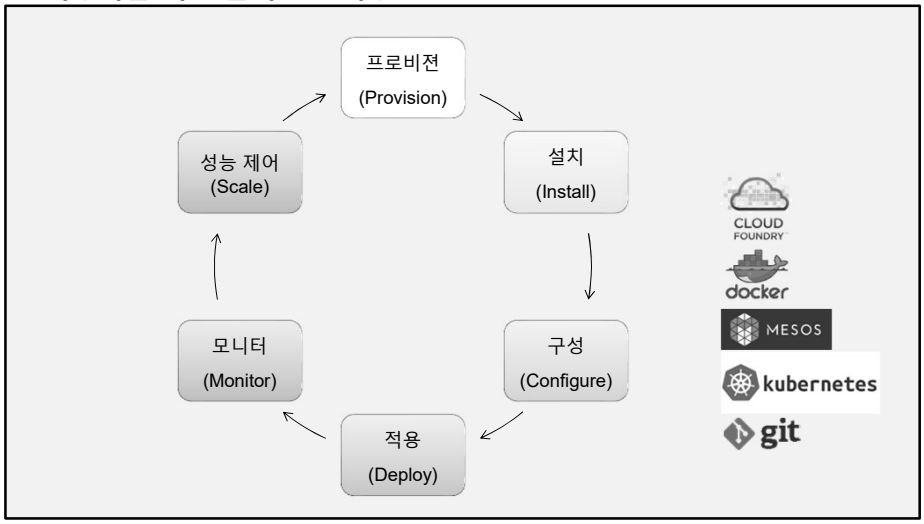
1. **CLI**
2. **Swarm** (UCP, Docker Datacenter, Docker Cloud)
3. **Kubernetes** (Cloud Eco-system)
4. **기타**

Docker .yaml

```
1.  version: '3'
2.  services:
3.    micro:
4.      image: gianarb/micro:1.2.0
5.      deploy:
6.        mode: replicated
7.        replicas: 2
8.        resources:
9.          limits:
10.           cpus: '0.25'
11.           memory: 512M
12.       reservations:
13.         cpus: '0.25'
14.         memory: 256M
15.       restart_policy:
16.         condition: on-failure
17.         delay: 5s
18.         max_attempts: 3
19.         window: 120s
```

X. 관리

❖ **DevOps 기반 관리의 자동화**
❖ **자동화를 위한 플러그인 허용**



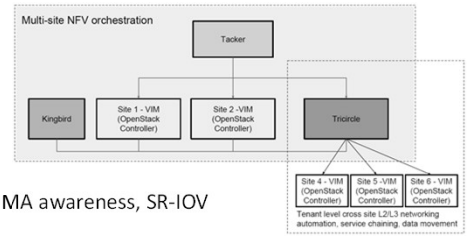
271

X. 관리

❖ **OpenStack**

1. **Tacker:** 멀티 사이트를 위한 오픈 NFV 오케스트레이터 (OpenStack)

- VNF 라이프 사이클 관리
  - ✓ 프레임워크 모니터
  - ✓ 프레임워크 컨피규레이션
- VNF 카탈로그 관리
  - ✓ TOSCA 탬플릿 지원
- EPA 지원
  - ✓ CPU-Pinning, Huge Page, NUMA awareness, SR-IOV
- 자원 생성 자동화
  - ✓ Flavor, Network and Image creation



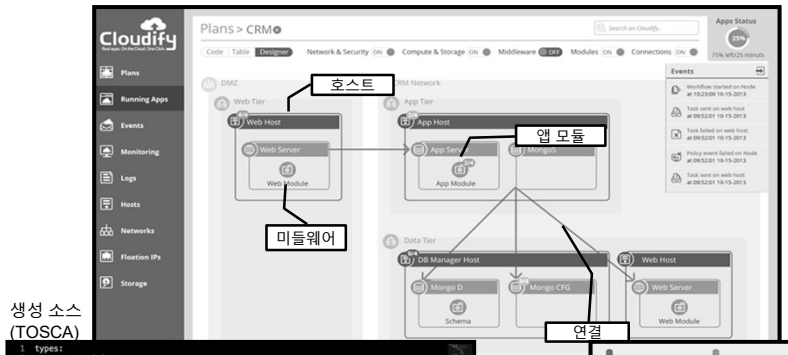
2. **Kingbird:** 멀티 리전을 관리하며 자원을 동기화
3. **Tricircle:** OpenStack L2/L3 네트워크를 연결하여 volume/VM 이동, 이미지 배포, 글로벌 자원 가시화 분산 쿼타 관리

EPA (Enhanced Platform Awareness) NUMA (Non-uniform memory access)

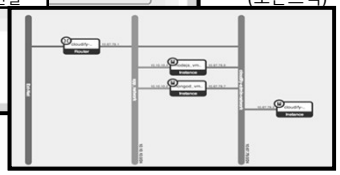
272

X. 관리

❖ **멀티 클라우드 오케스트레이션 :** 표준 TOSCA 기반 GUI 서비스 (TOSCA 표준 적용 오픈소스 Cloudify 예)




```
types:
  openstack_host:
    derived_from: cloudify.types.host
    properties:
      - install_agent: false
      - region
          instance:
              - name
              - image
              - flavor
              - key_name
    interfaces:
      cloudify.interfaces.lifecycle:
        - create: cloudify.plugins.openstack_host_provisioner.tasks.provisio
        - start: cloudify.plugins.openstack_host_provisioner.tasks.start
```

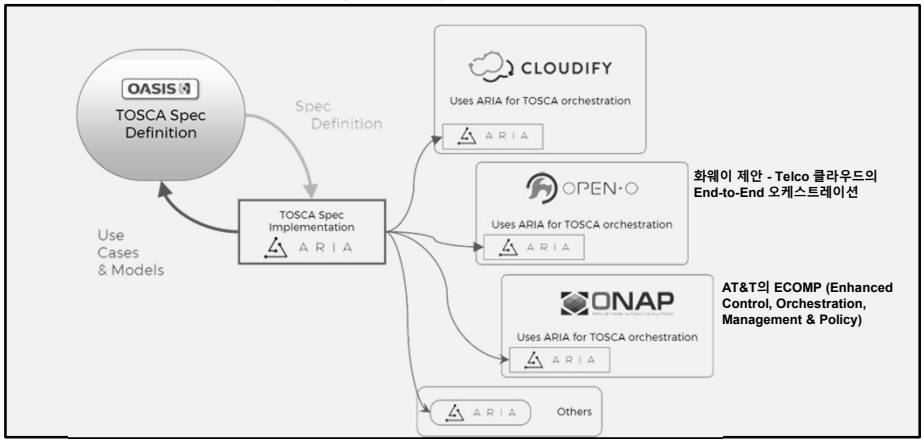


TOSCA: OASIS open standards consortium (Topology and Orchestration Specification for Cloud Applications)

X. 관리

❖ **표준 TOSCA 스펙 적용 오픈소스 'ARIA'**

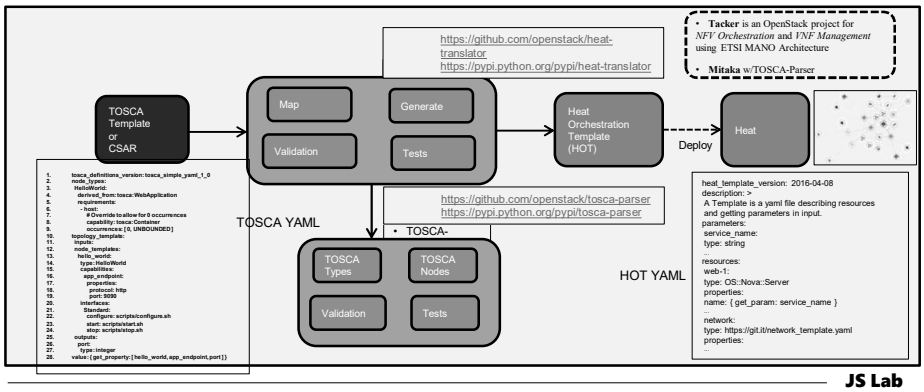
1. **오케스트레이션이 TOSCA 프로파일 지원을 위한 Python 라이브러리**
2. **TOSCA 애플리케이션 생성을 위한 SDK**
3. **CLI Tools: 오케스트레이션을 위한 TOSCA 템플릿**



X. 관리

❖ **TOSCA 처리 과정 (오픈스택 예)**

- TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0 Committee Specification Draft 03 (17 March 2016)
- 토스카 파서(TOSCA-Parser): Parser for TOSCA Simple Profile in YAML
- 히트번역기(Heat-Translator): An OpenStack project to map and translate non-Heat (e.g. TOSCA) templates to Heat Orchestration Template (HOT)

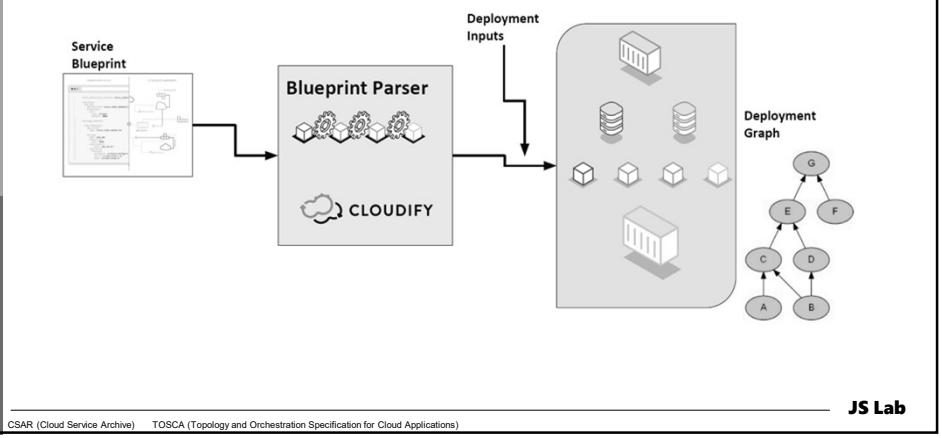


275

X. 관리

❖ **Service Instantiation**

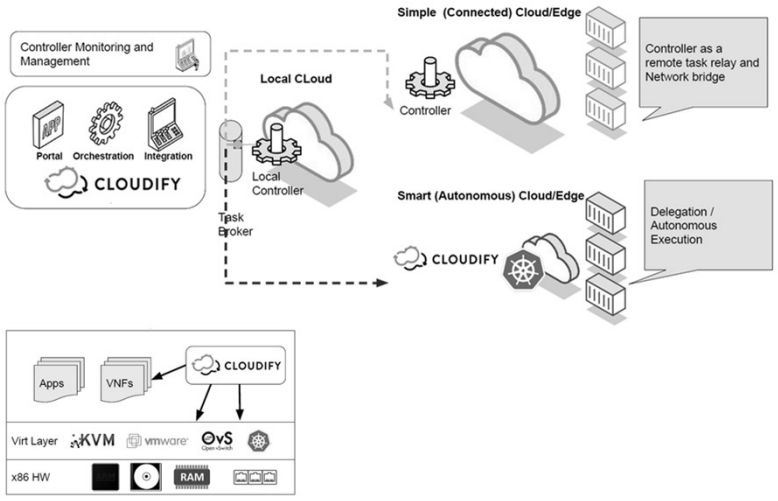
❖ TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0



276

X. 관리

❖ **Edge Computing Orchestration**

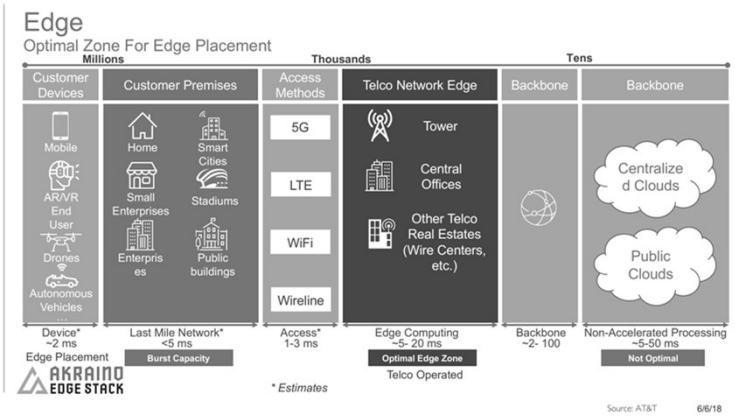


CSAR (Cloud Service Archive) TOSCA (Topology and Orchestration Specification for Cloud Applications)

277

X. 관리

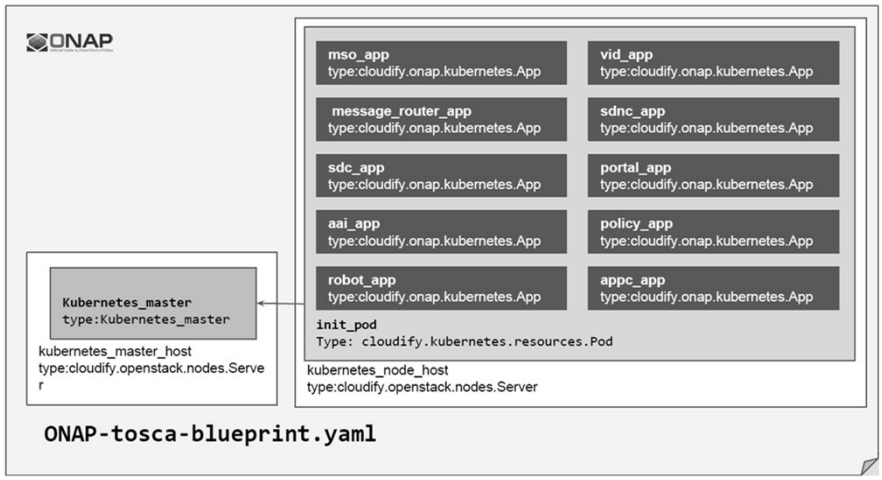
❖ **Network Edge in service providers is where the service provider logically connects to their customers. It is controlled by the service provider. Akraino also includes the CPE device, which is generally located on customer premises.**



278

X. 관리

❖ **ONAP TOSCA Template**

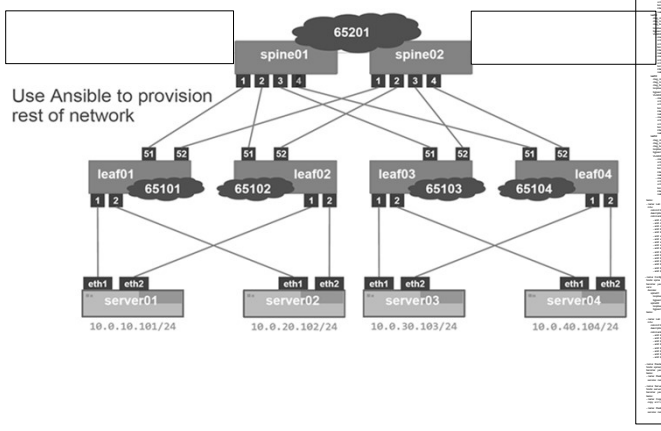


https://gerrit.onap.org/r/gitweb?p=oom.git;a=blob;f=onap-blueprint.yaml

X. 관리

❖ **Ansible Template** (Playbook)

❖ **ansible-playbook.yml**



https://gerrit.onap.org/r/gitweb?p=oom.git;a=blob;f=onap-blueprint.yaml

X. 관리

❖ **Operation for Ansible "ansible-playbook.yml"**

```
cumulus@oob-mgmt-server:~/NetworkAutomation$ ansible-playbook.yml
...
PLAY RECAP *********************************************************************************
leaf01                     : ok=10   changed=7    unreachable=0    failed=0
leaf02                     : ok=10   changed=6    unreachable=0    failed=0
leaf03                     : ok=10   changed=6    unreachable=0    failed=0
leaf04                     : ok=10   changed=6    unreachable=0    failed=0
server01                   : ok=3    changed=1    unreachable=0    failed=0
server02                   : ok=3    changed=1    unreachable=0    failed=0
server03                   : ok=3    changed=1    unreachable=0    failed=0
server04                   : ok=3    changed=1    unreachable=0    failed=0
spine01                    : ok=10   changed=6    unreachable=0    failed=0
spine02                    : ok=10   changed=6    unreachable=0    failed=0
Wednesday 11 October 2017  19:42:59 +0000 (0:00:05.074)       0:01:16.673 *****
===============================================================================
reset : Clear config ---------------------------------------------------------- 16.03s
Net 6 -- Deploy Configuration To All Leafs ------------------------------------ 13.61s
Net 6 -- Deploy Configuration to All Spines ----------------------------------- 9.70s
reset : Restore NTP ----------------------------------------------------------- 7.97s
Restart Networking ------------------------------------------------------------ 5.07s
Restart PTM Daemon to Apply new Topology.dot file ----------------------------- 3.87s
Gathering Facts --------------------------------------------------------------- 3.21s
Gathering Facts --------------------------------------------------------------- 3.01s
Restart the netq-agent -------------------------------------------------------- 2.93s
reset : Copy the Default Interface Configuration in Place --------------------- 2.83s
Download the topology.dot file from the OOB-MGMT-SERVER ----------------------- 2.11s
Gathering Facts --------------------------------------------------------------- 1.80s
reset : Apply Default Interface Configuration --------------------------------- 1.76s
Copy Interfaces Configuration File -------------------------------------------- 1.52s
Gathering Facts --------------------------------------------------------------- 1.20s
cumulus@oob-mgmt-server:~/NetworkAutomation$
```

SDN/NFV for 5G
james@jslab.kr
JS Lab

281

SDN/NFV for 5G
james@jslab.kr
JS Lab

282

SDN/NFV for 5G

james@jslab.kr

부록. 오픈플로우

I. 오픈플로우 버전
II. 구성과 동작
III. 오픈플로우 컨피그
IV. SDN 컨트롤러
V. 오픈플로우 기술

JS Lab

283

SDN/NFV for 5G

james@jslab.kr

I. 오픈플로우 버전

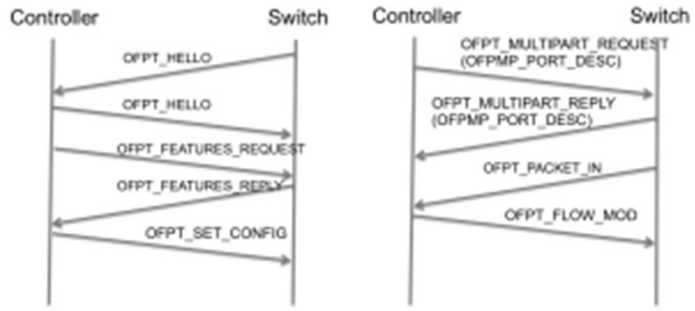
JS Lab

284

I. 오픈플로우 버전 1) 개요

1.오픈플로우 버전

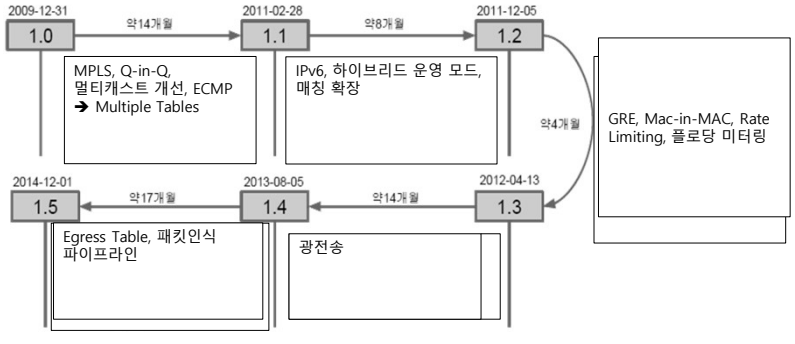
① SDN 사우스바운드를 위한 표준으로 ONF에서 관리
② 버전별로 플로우 테이블등이 다름
③ 하드웨어 제조 등을 고려해 v1.3을 장기 지원(long term support)로 확정



285

I. 오픈플로우 버전 1) 개요

2.버전별 주요 변화



286

I. 오픈플로우 버전 1) 개요

3. 병렬 개발

① **목적**
- 증가하는 1.x에 대한 요청에 대한 확장
- V1.3.x를 장기간 지원하도록 결정

② **요청**
- 소프트웨어 : 빠른 변화 요청
- 하드웨어 : 지속적이고 고정된 형태 요청

③ **대상**
- 모든 종류의 네트워크 장비
- 현실에 기반을 둔 시제품 필요

④ **계획**
- 오픈플로우 1.3.x : 장기간 지원
- 오픈플로우 1.3.x를 위한 ONF 확장판(Extention) : 신기능 추가
- 오픈플로우 1.x : 연장성, 증가하는 요청사항에 대한 확장
- 오픈플로우 1.0.x : 개발 계획 없음

I. 오픈플로우 버전 2) 버전의 발전

언제	누가	무엇을	어떻게
2008년	스탠포트대학 오픈플로우팀	v0.8.9 v0.9.0	• 표준 연구 시작
2009년 12월 31일	OpenFlow.org	v1.0	• 최초 공식 릴리즈 • 싱글 Flow Table 지원
2011년 2월 28일	OpenFlow.org	v1.1	• MPLS shim header 지원 • 다중 Flow Table, Group 지원(멀티캐스트, 브로드캐스트 지원 가능) • Multipath 지원 • 다중 VLAN Tagging 지원 • 가상포트 지원
2011년 3월	Open Networking Foundation		• ONF 설립
2011년 12월 5일	Open Networking Foundation	v1.2	• OpenFlow에서 ONF로 표준화 기관 이관 • Extensible match 지원 • IPv6 지원 • 다중 컨트롤러 지원

I. 오픈플로우 버전 2) 버전의 발전

언제	누가	무엇을	어떻게
2014년 4월 13일	Open Networking Foundation	v1.3	• Flexible Table Miss 지원 • IPv6 확장 헤더 지원 • Provider Backbone Bridge(PBB) 지원 • MPLS Bottom of Stack bit(BoS) 지원 • 컨트롤러, 스위치 간 보조 연결에 대한 사양 정의
2012년 9월 6일	Open Networking Foundation	v1.3.1	• 컨트롤러, 스위치 간 오픈플로우 버전 협상 방식 개선
2013년 4월 25일	Open Networking Foundation	v1.3.2	• 오픈플로우 메시지 개선
2014년	Open Networking Foundation	v1.4	• 확장 헤더 지원 향상 • 광 포트 지원 • 컨트롤러에 의한 향상된 스위치 관리 • 용량 관리와 관련된 룰 지원
2015년	Open Networking Foundation	v1.5	• 통신사등 요구사항 개선

289

I. 오픈플로우 버전 2) 버전 1.0

3. 버전 1.0

① 2009년 12월 공개

② 최초의 공개용 정식 설치 버전

③ 대부분 제조사에서 지원

④ 테이블

- Match : L1~ L4 정책을 위한 18개 2개의 선택 match
- 상태 테이블 : 4개

⑤ 단점

- 단일 플로우 테이블 : 테이블 공간의 제약
- 제한된 조건 옵션 : 12개
- 제한된 패킷 전달 옵션 : 브로드캐스트, 멀티캐스트, 누락(Drop)

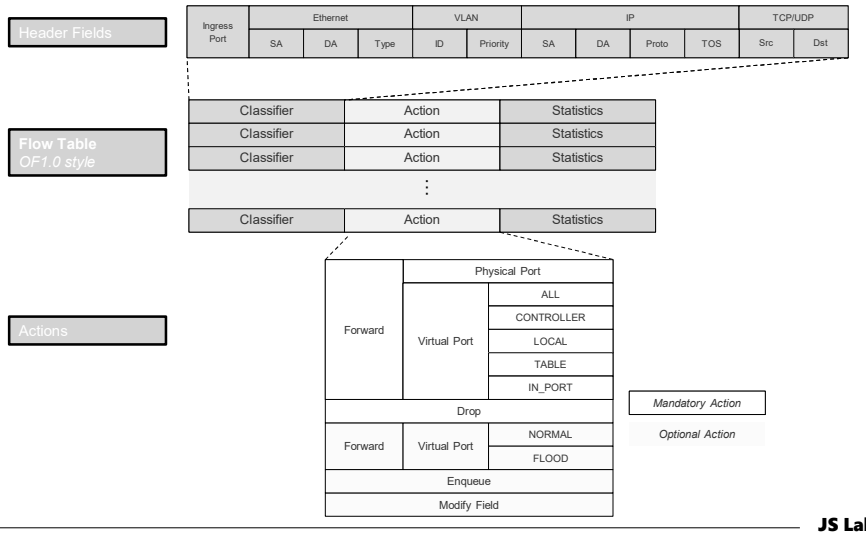
⑥ 업그레이드

- 버전 1.0.1 : 2012년 6월 공개, 오류 수정
- 버전 1.0.2 : 2013년 11월 공개, TCP 포트 6653으로 수정

290

I. 오픈플로우 버전 2) 버전 1.0

2. 프로토콜 구조



291

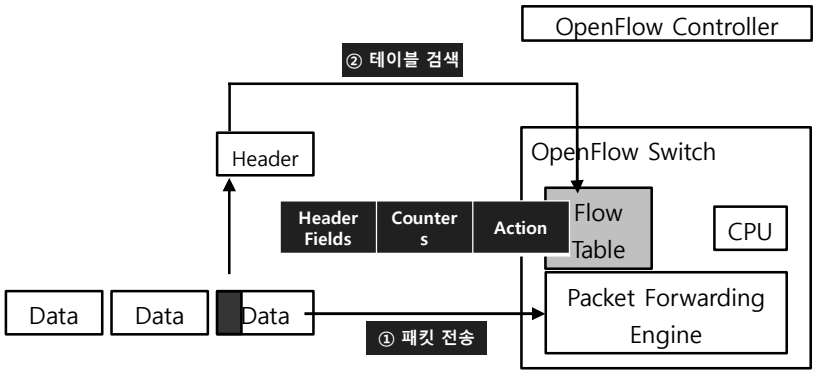
I. 오픈플로우 버전 2) 버전 1.0

3. 동작

① 패킷 전송 : 스위치 포트로 데이터 패킷이 진입

② 테이블 검색

- 들어온 패킷의 헤더 정보를 활용
- 플로우 테이블은 우선순위를 가진 플로우 엔트리로 구성



292

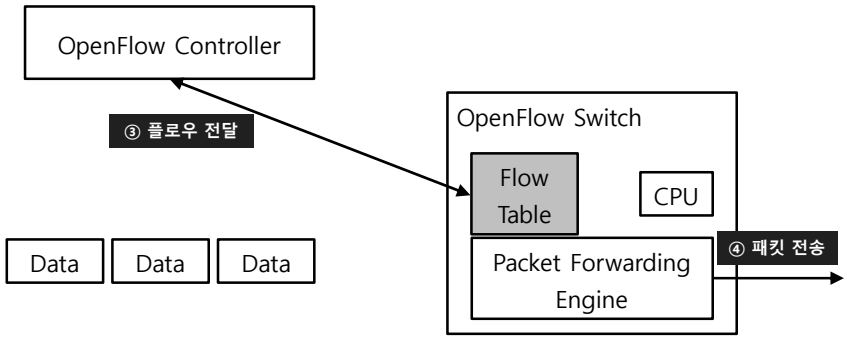
I. 오픈플로우 버전 2) 버전 1.0

3. 동작

① 플로우 전달

- 플로우테이블에 없을 경우 컨트롤러에 문의
- 컨트롤러가 해당 플로우에 대한 경로 전달

② 스위치는 패킷을 지정된 포트로 전송



I. 오픈플로우 버전 2) 버전 1.0

3. 플로우 테이블

① 플로우를 보내기 위한 조건 및 동작의 집합

- 플로우 엔트리의 집합
- 플로우 엔트리는 하나의 행으로 존재

② 플로우 엔트리는 3가지 요소를 포함

- Headers : 12개의 튜플에 대한 조건
- Counters : 통계 정보
- Actions : 동작, 여러 개의 동작을 위해 프로그램 가능

Header Fields	Counters	Action
	플로우 엔트리1	
	플로우 엔트리 2	
	플로우 엔트리 3	
	플로우 엔트리 n	

I. 오픈플로우 버전 2) 버전 1.0

4. 플로우 테이블

③ 헤더 필드

- **기본적인 12개의 튜플(tuple)로 구성**
  - ✓ Source Port
  - ✓ EtherSource : source MAC
  - ✓ Ether DST
  - ✓ ther Type
  - ✓ VLAN ID
  - ✓ VLAN Priority
  - ✓ IP SRC
  - ✓ IP DEST
  - ✓ IP Protocol
  - ✓ IP ToS
  - ✓ TCP/UCP SRC : ICMP type
  - ✓ TCP/UDP Dest : ICMP code

Ingress Port	Ether Source	Ether DST	Ether Type	VLAN ID	VLNA Priority	IP SRC	IP DST	IP Proto	IP ToS bits	TCP/UDP SRC	TCP/UDP Dest

295

I. 오픈플로우 버전 2) 버전 1.0

4. 플로우 테이블

④ 카운트 필드

- Per Table
  - ✓ Active Entries : 32 bits
  - ✓ Packet Lookups : 64 bits
  - ✓ Packet Matched : 64 bits
- Per Flow
  - ✓ Received Packets : 64 bits
  - ✓ Received Bytes : 64 bits
  - ✓ Duration(seconds) : 32 bits
  - ✓ Duration(nanoseconds) : 32 bits
- Per Queue
  - ✓ Transmit Packets : 64 bits
  - ✓ Transmit Bytes : 64 bits
  - ✓ Transmit Overrun Errors : 64 bits

296

I. 오픈플로우 버전 2) 버전 1.0

4. 플로우 테이블

④ 카운트 필드

- Per Port
  - ✓ Received Packets : 64 bits
  - ✓ Transmitted Packets : 64 bits
  - ✓ Received Bytes : 64 bits
  - ✓ Transmitted Bytes : 64 bits
  - ✓ Receive Drops : 64 bits
  - ✓ Transmit Drops : 64 bits
  - ✓ Receive Errors : 64 bits
  - ✓ Transmit Errors : 64 bits
  - ✓ Transmit Errors : 64 bits
  - ✓ Receive Frame Alignment Errors : 64 bits
  - ✓ Receive Overrun Errors : 64 bits
  - ✓ Receive CRC Errors : 64 bits
  - ✓ Collisions : 64 bits

I. 오픈플로우 버전 2) 버전 1.0

4. 플로우 테이블

⑤ 액션 필드

- Forward : 패킷 내보내기
  - ✓ ALL : 모든 포트로 보내기
  - ✓ Controller : 컨트롤러로 보내기
  - ✓ Local : 스위치의 내부 스택으로 보내기
  - ✓ Table : 테이블로 보내기, Packet Out 메시지에만 해당
  - ✓ IN_Port : 들어온 포트로 보내기
  - ✓ NORMAL : 스위치의 기본 동작방식으로 보내기(옵션)
  - ✓ FLOOD : 들어온 포트를 제외하고 내보내기(옵션)
- Enqueue : QoS 적용등을 위해 포트의 큐로 보내기(옵션)
- Drop : 특별한 동작 없이 버리기

I. 오픈플로우 버전 2) 버전 1.0

4. 플로우 테이블

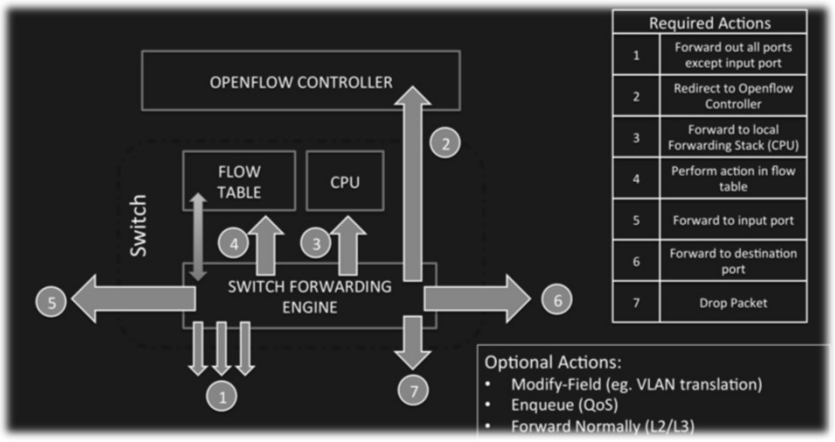
⑤ 액션 필드

- Modify-Field : 필드 값 수정하기(옵션)
  - ✓ Set VLAN ID
  - ✓ Set VLAN Priority
  - ✓ Strip VLAN header
  - ✓ Modify Ethernet source MAC address
  - ✓ Modify Ethernet destination MAC address
  - ✓ Modify IPv4 source address
  - ✓ Modify IPv4 destination address
  - ✓ Modify IPv4 ToS bits
  - ✓ Modify transport source port
  - ✓ Modify transport destination port

299

I. 오픈플로우 버전 2) 버전 1.0

5. 오픈플로우 스위치 기능



300

I. 오픈플로우 버전 3) 버전 1.1

❖ **버전 1.1**

① 2011년 2월 공개

② V1.0과 호환되지 않으며, 대중적이지 않음

③ 신규 기능 추가

- 다중 테이블
- 그룹 테이블
- Q-in-Q
- MPLS 태그 지원 : Telco 연동

④ 특징

- 다중 플로우 테이블
- 다중 파이프 라인

301

I. 오픈플로우 버전 4) 버전 1.2

❖ **버전 1.2**

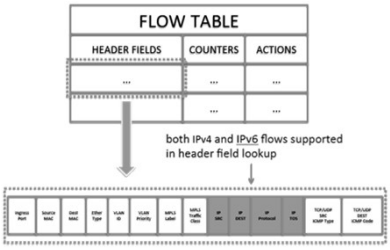
① 2011년 12월 공개

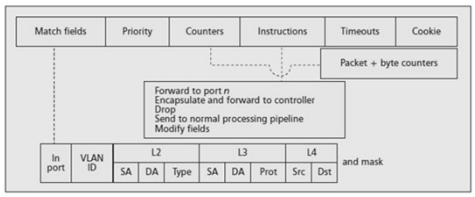
② ONF를 통한 최초 공개, 대중적이지 않음

③ 과거버전과 호환 불가

④ 특징

- IPv6 지원 : Source, Destination 필드
- 다중 컨트롤러 간 이중화 : Master, Slave 컨트롤러 지원
- TLV(Type, Length, Value) 통한 헤더 확장성





302

I. 오픈플로우 버전 5) 버전 1.3

1. 버전 1.3

① **2012년 4월 공개**

② **하드웨어 개발 및 안정화 등을 위해 장기간 지원 예정 : v1.3.1, v1.3.2, v1.3.3**

③ **기능 추가 : 모니터링 및 운영, 관리 기능 추가**

- MAC-in-MAC(Provider Backbone Brigdge)
- Per-flow metering : 사용량에 따른 제어 가능, 통신사용
- 미터테이블 추가 : 통신사 QoS등 적용 가능
- 다중 컨트롤러 지원 확장 : 임의의 보조 컨트롤러 설정
- IPv6 확장 헤더 지원 : ESP IPv6 헤더, Authentication 헤더, Hop-by-Hop 헤더

④ **업그레이드**

- 버전 1.3.1 : 2012년 9월, 오류 수정
- 버전 1.3.4 : 2014년 4월
- 버전 1.3.5 : 2015년 3월 26일

Meter Identifier	Meter Bands	Counters

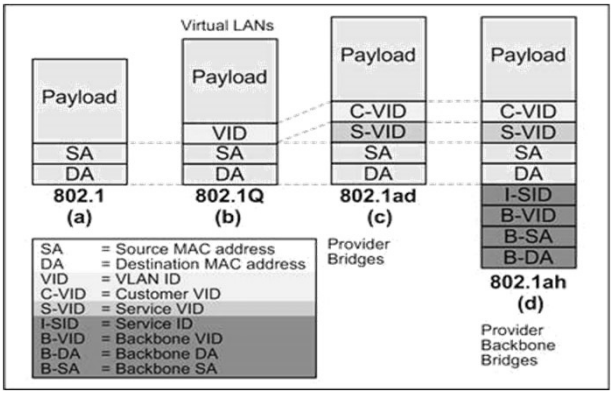
SDN/NFV for 5G
james@jslab.kr

303

I. 오픈플로우 버전 5) 버전 1.3

2. 특징

① **PBB & SPB : MAC in MAC**



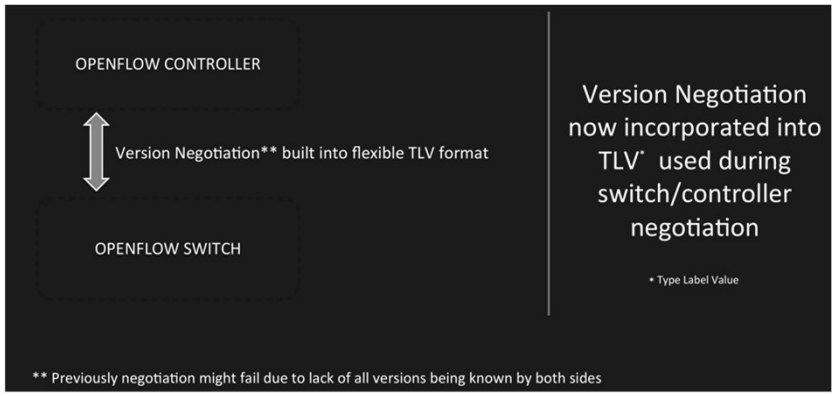
SDN/NFV for 5G
james@jslab.kr

304

I. 오픈플로우 버전 5) 버전 1.3

2. 특징

② 상호 버전 확인을 위한 TLV 지원

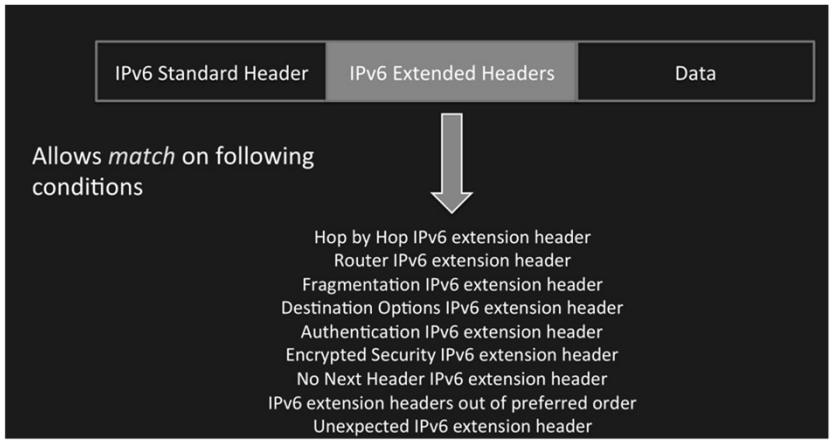


I. 오픈플로우 버전 5) 버전 1.3

2. 특징

③ **IPv6 지원 개선**

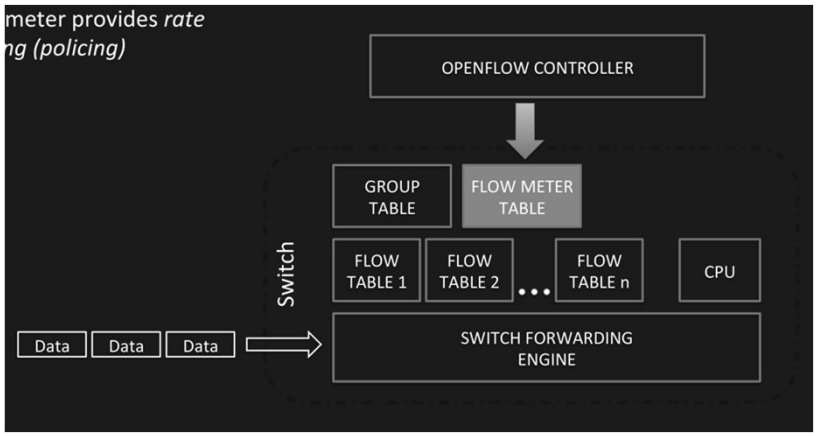
- IPv6 확장 헤더 지원



I. 오픈플로우 버전 5) 버전 1.3

2. 특징

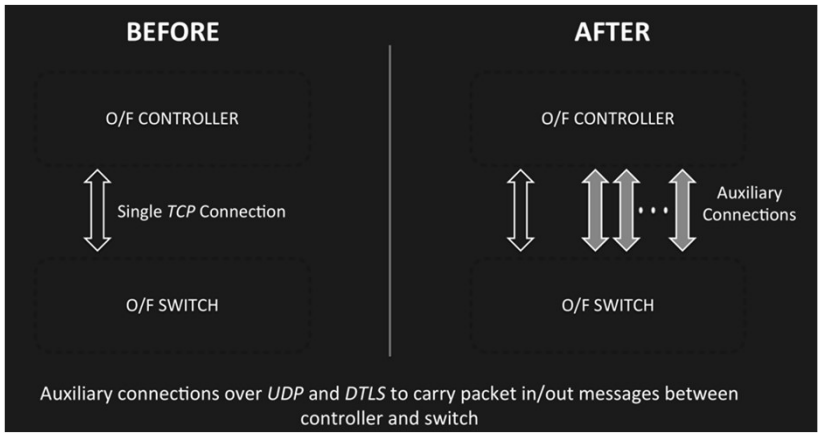
④ 미터링 기능 제공



I. 오픈플로우 버전 5) 버전 1.3

2. 특징

⑤ 보조 포트 접속 기능 제공



I. 오픈플로우 버전 6) 버전 1.4

SDN/NFV for 5G

james@jslab.kr

1. 버전 1.4

① **2013년 10월 공개**

② **기본 포트 변경 : 6633 -> 6653**

③ **기능 추가**

- 확장성 향상 : 포트와 테이블, 큐를 위한 TLV 구조 추가
- 광 포트 설명, 설정, 모니터 기능 추가
- 플로우 모니터링
- 테이블 동기화

2. 특징

① 확장성 향상 : 프로토콜의 더 많은 부분이 TLV를 이용한 옵션으로 확장 가능

② 광 포트 설정 : 주파수나 출력 같은 광 포트와 관련된 세부 사항 지원

③ 향상된 스위치 관리 : 스위치가 다른 컨트롤러에 의해 생성된 플로우에 대해 확인 가능

JS Lab

309

I. 오픈플로우 버전 7) 버전 1.5

SDN/NFV for 5G

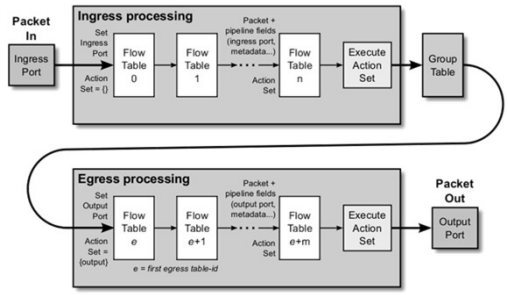
james@jslab.kr

1. 버전 1.5

① **2014년 12월 19일 공개**

② **기능 추가**

- Egress Tables : 출력 포트에서도 별도의 Flow 프로세싱 가능
- Packet Type aware pipieline : 패킷 타입별 처리, 기존에는 이더넷만 처리
- TCP flags matching : TCP 헤더의 flags bits 식별
- Scheduled Bundles : 컨트롤러가 스위치의 정보 획득을 위한 번들 기능 확장

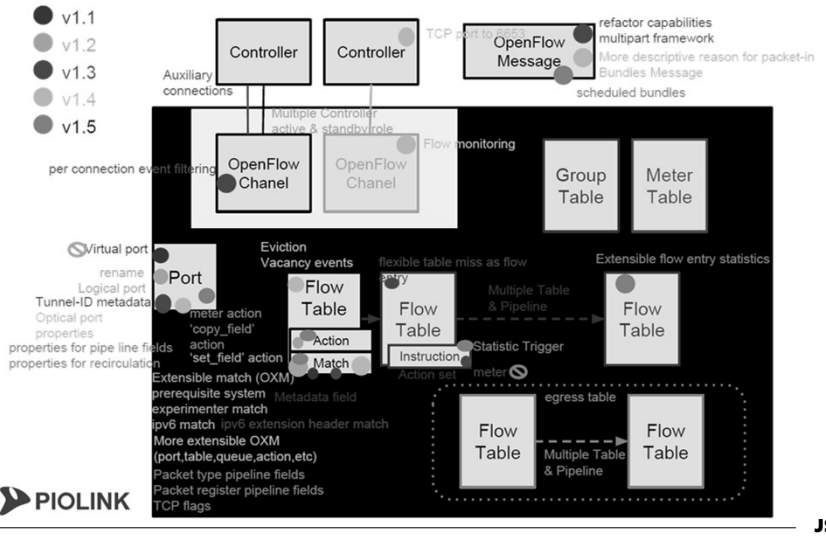


JS Lab

310

I. 오픈플로우 버전 8) 버전별 기능 추가

❖ 프로토콜 세부 내용



311

II. 구성과 동작

SDN/NFV for 5G
james@jslab.kr
JS Lab

312

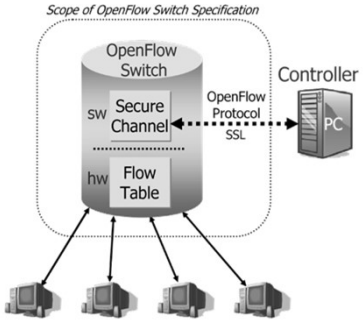
II. 구성과 동작 1) 논리적 구성요소

❖ 오픈플로우 필수 구성 요소

① 플로우 테이블(Flow Table) : 스위치에 설정

② 보안 연결(Secure Channel) : 컨트롤러와 스위치간 연결

③ 오픈플로우 프로토콜(OpenFlow Protocol) : 컨트롤러가 스위치에게 메시지 전달



II. 구성과 동작 2) 플로우 테이블

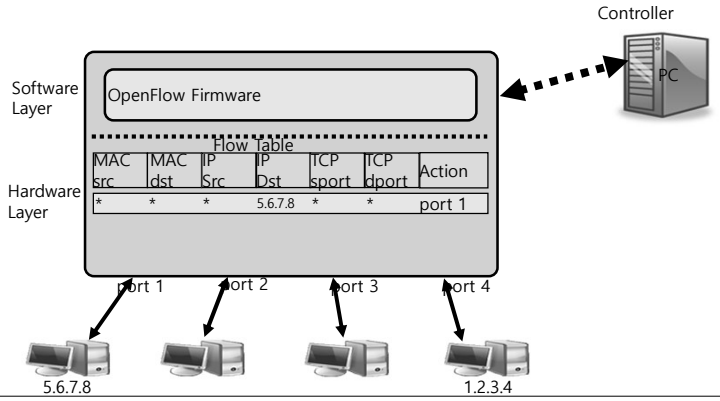
1. 플로우테이블

① 플로우 테이블의 추상화(Abstraction)

② 우선순위가 있는 플로우 엔트리의 집합

③ 네트워크 장비의 칩셋 안에 존재

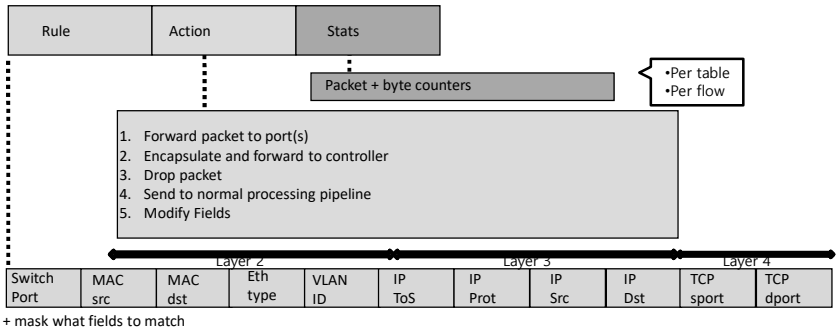
④ 각 플로우는 타임아웃을 보유



II. 구성과 동작 2) 플로우 테이블

2. 플로우엔트리

① **Match : 패킷에 대한 조건**
② **Action : 조건에 맞는 패킷을 어떻게 처리할 것인가**
③ **Stats : 조건에 맞는 패킷에 대한 통계 업데이트**



II. 구성과 동작 2) 플로우 테이블

3. Action Set

① **Forward**
- Physical ports (Required)
- Virtual ports : All, Controller, Local, Table, IN_PORT (Required)
- Virtual ports : Normal, Flood (Required)

② **Enqueue (Optional)**
③ **Drop (Required)**
④ **Modify Field (Optional)**
- Set/Add VLAN ID
- Set VLAN priority
- Strip VLAN Header
- Modify Ethernet source/destination address
- Modify IPv4 source/destionation address
- Modify IPv4 type of service bits
- Modify IPv4 TCP/UDP source/destination port

II. 구성과 동작 3) 보안연결

1. 보안 연결(Secure Channel)

① 오픈플로우 스위치와 컨트롤러간의 연결을 위한 인터페이스
② 컨트롤러가 설정하고, SC 인터페이스를 통해 스위치를 관리
- 스위치로부터 이벤트 수집
- 스위치로 패킷 전달

③ 오픈플로우 스위치와 절차를 사용하는 컨트롤러 간의 연결을 맺고 끊음
- 접속 설정
- 접속 중단

④ 보안 연결은 TLS 연결, 스위치와 컨트롤러는 상호간에 특별히 인증된 보안키를 나누어가지고 정보 교환

SDN/NFV for 5G
james@jslab.kr

II. 구성과 동작 4) 오픈플로우

1. Match 필드

① 기본 12개 요소(오픈플로우 1.0)
② MAC 출발지/도착지, IP 출발지, 도착지, VLAN, TCP/UDP 포트, 물리 스위치 포트
③ 와일드카드 사용 가능

2. 플로우 엔트리(Flow Entries)

① 플로우 테이블은 들어오는 패킷에 대하여 비교해 볼 수 있는 엔트리 집합으로 구성
② 개별 플로우 엔트리는 match, counters, actions로 구성
③ 비교는 플로우 테이블의 첫번째 엔트리부터 시작
④ 플로우 엔트리는 우선순위에 의해 동작
⑤ Match에 성공하면 actions를 수행
⑥ Match에 실패하면
- 오픈플로우 채널을 통해 컨트롤러에게 전달
- Drop
- 다음 플로우 테이블을 검색

SDN/NFV for 5G
james@jslab.kr

II. 구성과 동작 4) 오픈플로우

3. 플로우 테이블(Flow Tables)

① **우선 순위가 있는 플로우 엔트리의 리스트**
② **순서대로 처리되며, 첫번째 부합하는 조건의 Action을 따른다.**
③ **각 플로우는 타임아웃을 가지고 있다(idle, hard)**
④ **플로우 엔트리는 Proactive이거나 Reactive일수 잇다.**
- Proactive : 플로우를 지속적으로 가지고 있거나, 기본적으로 가지고 있어 일반적인 age out이 발생하지 않음
- Reactive : 플로우가 요청에 의해 생성되며, 동작이 발생한 이후에는 age out이 진행

SDN/NFV for 5G
james@jslab.kr

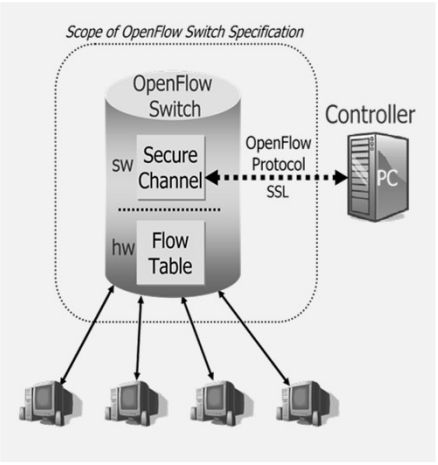
II. 구성과 동작 5) 물리적 구성요소

1. 오픈플로우 컨트롤러

① **오픈플로우 프로토콜 메시지**
② **채널 관리**
③ **처리**
- 파이프라인 프로세싱
- 패킷 match
- Instructions & Action Set

2. 오픈플로우 스위치

① **보안 연결(SC : Secure Connection)**
② **플로우 테이블**
- 플로우 엔트리



SDN/NFV for 5G
james@jslab.kr

II. 구성과 동작 6) 컨트롤러

❖ 오픈플로우 컨트롤러

① 하드웨어에 탑재 또는 단일 소프트웨어 : 프로그램으로 스위치에게 플로우 관련 정책 전달
② 플로우 테이블의 삭제, 수정, 삽입을 통해 패킷 전달에 대한 스위치의 동작을 관리
③ 오픈플로우 버전에 따라 스위치에서 가능한 행동이 정의'
- 버전간 하위호환성 불가

④ 컨트롤러와 스위치는 보안 연결(TLS/SSL)을 통해 연결

SDN/NFV for 5G
james@jslab.kr

II. 구성과 동작 7) 스위치

1. 오픈플로우 스위치

① 기본적인 데이터 포워딩 장비로 플로우 테이블에 의해 패킷 전달
② 플로우 테이블은 각 플로우 엔트리의 집합
③ 각 플로우는 match, counter, Instruction으로 구성(v1.2 이상)
④ 필드, 카운터, 신스트럭션즈로 구성
⑤ 플로우 테이블은 플로우 룰 또는 플로우 엔트리로도 불림
⑥ 플로우 테이블에 처음에 있는 헤더 필드는 어떤 패킷에 적용해야 하는지 나타낸다.
⑦ 와일드카드를 사용한 match를 통해
⑧ 오픈플로우에서도 빠른 패킷 전달을 위해 TCAM을 사용기를 권장한다.
⑨ 헤더 필드는 오픈플로우 규정에 의해 다른 프로토콜을 구별할 수 있다. 이더넷 버전포식스 엠피엘에스등
⑩ 카운터는 플로우 통계를 위해 지정되어 있으며, 받은 패킷 수, 바이트를 저장
⑪ 액션은 패킷을 어떻게 처리할까? 기본적으로 전달, 드랍, 필드 수정 등이 있다.

SDN/NFV for 5G
james@jslab.kr

II. 구성과 동작 7) 스위치

2. 스위치 부팅 단계

① **스위치는 최초 부팅시 최소한의 설정이 필요**
- IP 주소
- 컨트롤러 IP 주소
- 게이트웨어 IP 주소

② **스위치가 컨트롤러에 접속**

③ **스위치는 포트에 대한 정보를 컨트롤러에게 제공**

④ **컨트롤러는 LLDP 응답을 보낼 수 있는 규칙을 설치**

⑤ **LLDP 요청을 모든 이웃 장비들에게 전달**

⑥ **컨트롤러는 LLDP로 얻은 정보를 기반으로 네트워크 구성을 파악**

II. 구성과 동작 7) 스위치

3. 오픈플로우 스위치 구성요소

① **플로우 테이블& 그룹테이블**
- 패킷을 비교하고 전달하는 것을 수행

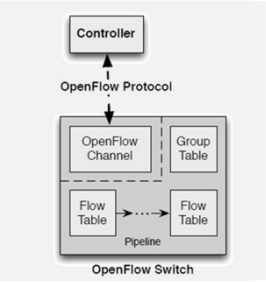
② **오픈플로우 채널**
- 스위치와 컨트롤러가 연결되는 인터페이스

③ **파이프라인 프로세스**
- 플로우 테이블 사이에서 플로우 엔트리 비교에 따라 패킷의 이동을 관리

4. 오픈플로우 스위치의 종류

① **순수 오픈플로우 스위치 : OpenFlow only**

② **하이브리드 스위치 : OpenFlow Capable**

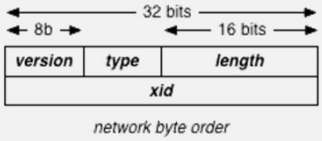


II. 구성과 동작 8) 오픈플로우 메시지

1. 오픈플로우 메시지

① 컨트롤러와 스위치간에 전달되는 메시지
② 3가지 종류의 메시지
- Controller-to-Switch
- 비동기 메시지(Asynchronous)
- 동기 메시지(Symmetric)

③ 메시지 패킷의 구조
- 버전 : 8비트
- 타입 : 8비트
- 길이 : 16비트
- Xid : 32비트



II. 구성과 동작 8) 오픈플로우 메시지

2. Controller-to-switch

① 컨트롤러가 스위치로 전달
② 직접적인 관리나 스위치의 상태 점검, 카운터 정보 검색
③ 새로운 플로우 생성 후 스위치에게 처리에 대한 답변 요청
④ 스위치로부터 응답이 있을 수도, 없을수도 있음
⑤ 메시지 종류 : 기능, 설정, 준비 상태, 수정상태 등

II. 구성과 동작 8) 오픈플로우 메시지

3. 비동기 메시지(Asynchronous)

① 스위치가 보내서 컨트롤러에게 네트워크 이벤트나 스위치 상태변화를 스위치로부터 컨트롤러의 요청 없이 생성되어 전달
② 존재하는 플로우에 맞지 않을 경우 컨트롤러에게 패킷 전달
③ 컨트롤러에게 패킷 도착, 플로우 제거, 포트 상태 변화, 에러 컨트롤 전달
④ Time to Live 또는 Inactivity 타이머가 지나 플로우가 삭제될 경우 컨트롤러에게 알림
⑤ 메시지 타입 : 패킷인, 플로우 제거, 포트 상태, 에러 등

327

II. 구성과 동작 8) 오픈플로우 메시지

4. 동기 메시지(Symmetric)

① 스위치와 컨트롤러 서로간에 서로의 요청 없이 전송
② 향후 기술 확장을 위해 경로를 제어하기 위한 시험 메시지 전송
③ 주로 Hello를 위해 사용
- 처음 스위치와 컨트롤러 연동시
- 컨트롤러와 스위치 연결이 유효한지 확인
- 지연 시간 결정 전에 에코 메시지 사용

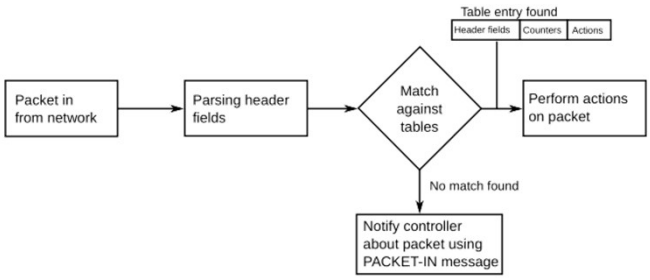
④ 메시지 타입 : hello, echo request/reply

328

II. 구성과 동작 9) 기본 동작

❖ 기본 동작

① 플로우 테이블에 맞는 조건이 있을 경우 수행
② 플로우 테이블에 맞는 조건이 없을 경우 컨트롤러에 문의
- 모든 플로우 관련 제어는 컨트롤러로부터 시작



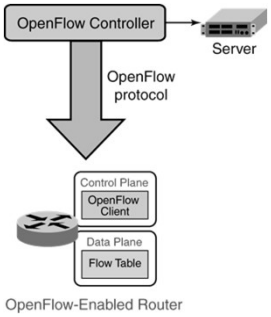
II. 구성과 동작 10) 플로우 전달

1. 플로우 엔트리(Flow Entries)

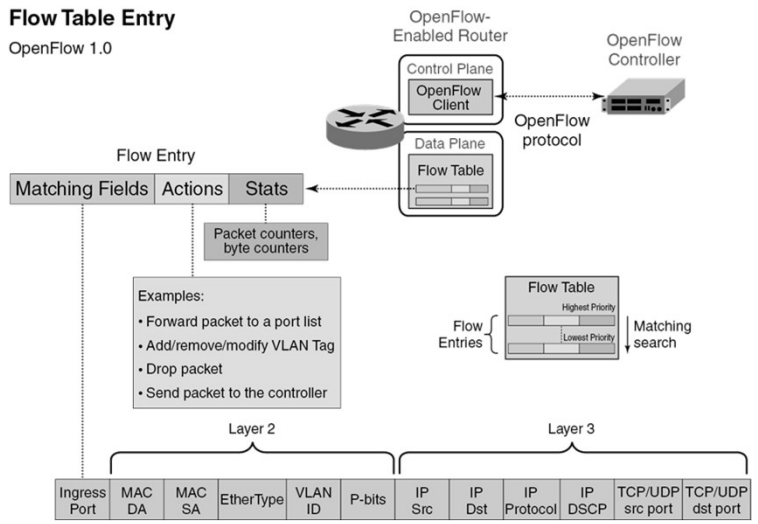
① Match Field : 들어오는 패킷에 대한 조건
② Stats : 패킷 비교 결과에 대한 기록
③ Actions : 패킷이 조건에 맞다면 어떻게 해야하는지

2. 플로우 테이블(Flow Tables)

① Match : 연관된 Actions이나 Instruction을 수행
② No Match : Drop 또는 컨트롤러에게 전달
③ Actions : Forward, Drop, Normal, Flood,...



II. 구성과 동작 11) 플로우 전달



II. 구성과 동작 12) 플로우 전송

1. Proactive

① 컨트롤러가 패킷이 오기 전에 미리 플로우 테이블을 스위치에 전송
② 패킷이 스위치에 도착하면 플로우 테이블이 있음으로 플로우 설정 시간이 불필요, 지연(latency)에 있어 매우 좋음
③ 주로 wildcard 마스크를 이용해서 처리
④ 관리자 뿐만 아니라, 특정 애플리케이션에 의한 플로우 생성 가능, 예를 들어 특정 트래픽 과부하시 경로 우회를 애플리케이션 통해 해결
⑤ 플로우는 기본적으로 계속 지속되며, 시간이 지나더라도 사라지지 않는다.
⑥ 컨트롤러가 플로우 엔트리를 추가
⑦ 플로우 테이블의 효율적인 사용
⑧ 모든 플로우는 추가적인 플로우 설정 시간이 필요
⑨ 만일 컨트롤러와 연결이 끊긴다면, 스위치는 최소한의 동작만 가능

II. 구성과 동작 13) 플로우 전송

SDN/NFV for 5G

james@jslab.kr

2. Reactive

① 첫번재 패킷이 스위치에 도착하면 스위치는 패킷에 대한 경로를 알지 못함으로 컨트롤러에 전송, 컨트롤러에 의해서 파악된 최적의 경로로 패킷이 전달되도록 해당 스위치에 플로우 테이블 전송, 플로우 설정이 이루어지는 동안 지연 발생

② 동적으로 설정되고, 장비의 상태가 변화되면 변경될 수 있다.

③ 사용되지 않으면 일정시간 이후 삭제될 수 있다.

④ 컨트롤러가 스위치에 사전에 플로우를 전달

⑤ 플로우 설정에 추가 시간이 필요치 않음

⑥ 컨트롤러와의 단절이 트래픽에 영향을 미치지 않음

⑦ 일반적으로 통합된 룰이 필요(wildcard를 이용)

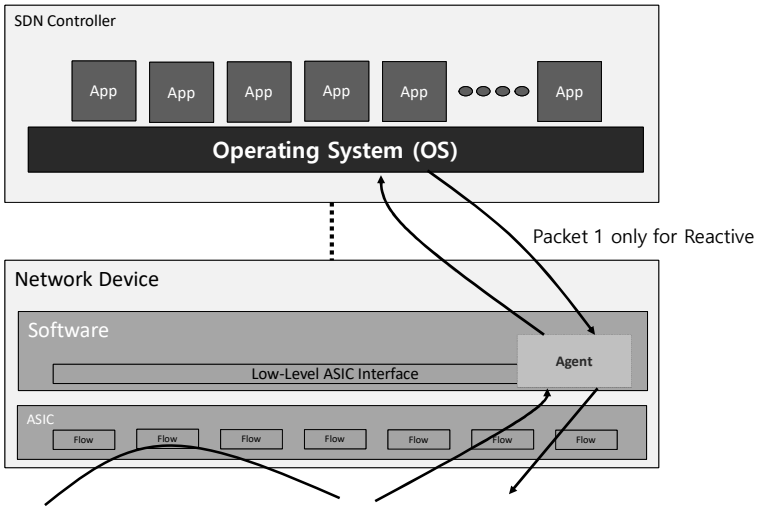
JS Lab

333

II. 구성과 동작 13) 플로우 전송

SDN/NFV for 5G

james@jslab.kr



JS Lab

334

II. 구성과 동작 14) 오픈플로우 포트전송

1. 물리적 포트

① 스위치에 실제 있는 포트의 집합

2. 논리적 포트

① 높은 수준의 추상화

② 포트 어그리게이션, 터널링 등

3. 예약 포트

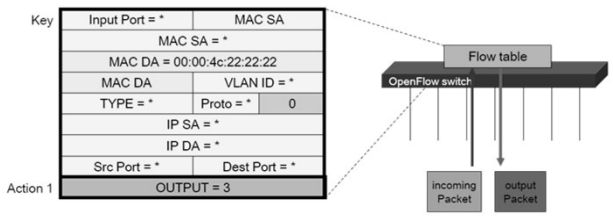
① 오픈플로우 내부 사용을 위해 지정

② ALL, CONTROLLER, LOCAL, NORMAL, FLOOD 등

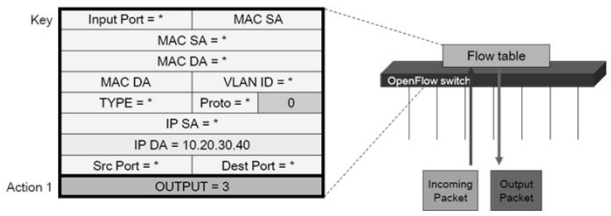
335

II. 구성과 동작 15) 동작 예제

1. L2 스위칭



2. L3 라우팅



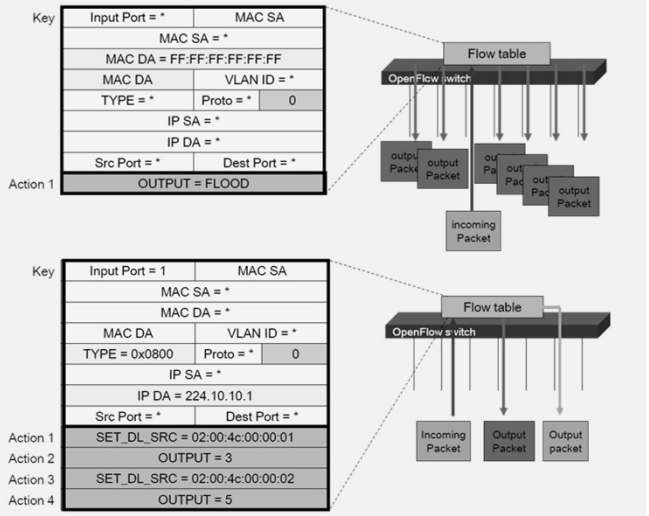
336

II. 구성과 동작 15) 동작 예제

SDN/NFV for 5G

james@jslab.kr

3. 브로드캐스트



4. 멀티캐스트

337

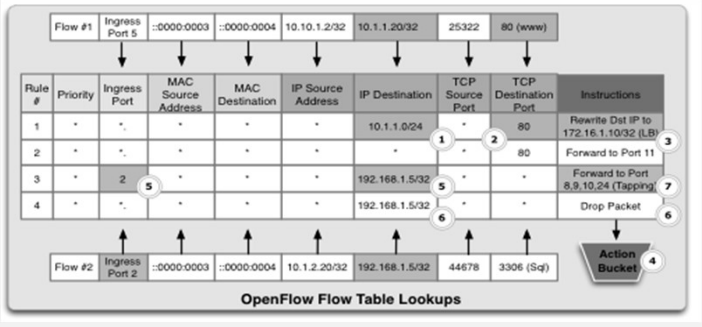
II. 구성과 동작 15) 동작 예제

SDN/NFV for 5G

james@jslab.kr

5. 방화벽

protocol	destination IP	destination port	action	priority*
TCP	192.168.1.0/24	80	ALLOW	1
TCP	192.168.1.0/24	wildcard	DENY	2



338

II. 구성과 동작 16) 플로우 구성

1. 플로우 구성

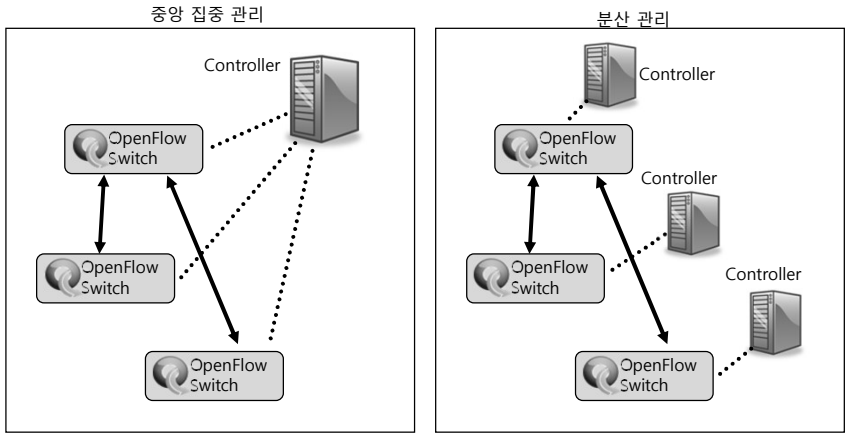
① **개별 플로우 기반**
- 컨트롤러에 의해 모든 플로우가 생성
- 플로우 엔트리에 의한 정확한 비교
- 플로우 테이블은 플로우별로 하나의 엔트리를 보유
- 캠퍼스 네트워크 같은 사용자 기반의 네트워크에 적합

② **통합 플로우 기반**
- 여러 그룹을 포함하나는 하나의 플로우 엔트리
- 와일드카드를 사용한 엔트리
- 플로우 테이블은 플로우 카테고리 별로 하나의 플로우를 보유
- 데이터센터 네트워크 값은 많은 수의 플로우를 가진 네트워크에 적합

II. 구성과 동작 16) 플로우 구성

2. 컨트롤러 구성 방법

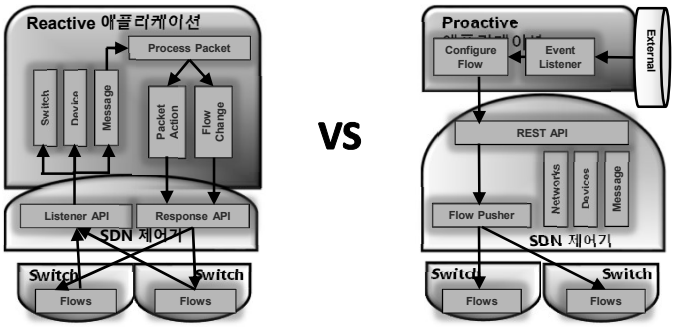
① **컨트롤러 중앙 집중 관리 : 하나의 컨트롤러로 모든 스위치 관리**
② **컨트롤러 분산 관리 : 여러개의 컨트롤러로 개별 스위치 관리**



II. 구성과 동작 17) 응용

- **Reactive 애플리케이션 (예):** 보안/관리 이슈등 비동기 이벤트에 반응
- **Proactive 애플리케이션 (예):** 사전 정책이 부팅시 적용

Reactive	Proactive
대부분 low-level API를 이용하여 컨트롤러 사용 언어 프로그램 (ex. Java, Python)	대부분 high-level API를 사용하는 추상화 계층 이용 (ex. REST API)
예: 사용자별 방화벽, security access	예: 스패닝 트리, 다중 경로
다양한 유형의 서비스를 위해 많은 플로우 자원이 필요 (ex. NAC)	사전 정의 가능한 서비스로 비교적 적은 플로우 필요 (ex. TCP port-specific)



341

III. 오픈플로우 컨피그

342

III. 오픈플로우 컨피그 1) 개요

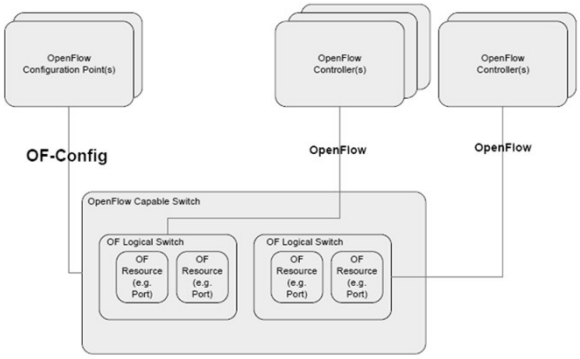
1. OF-Config

① **목적**

- 오픈플로우 스위치 설정을 위한 프로토콜 및 스키마 제공

② **누가**

- ONF의 Config-mgmt Working Group에서 표준화 진행



343

III. 오픈플로우 컨피그 2) 구성

1. 구성요소

① **OpenFlow Control Point**

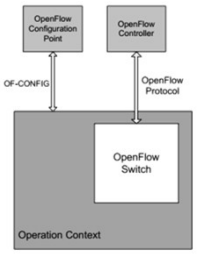
- 오픈플로우 스위치를 설정하기 위한 값들

② **OF-Config: 오픈플로우 스위치를 설정하고 관리하기 위한 프로토콜**

- 컨트롤러의 IP 주소
- 컨트롤러의 포트 정보
- 전송 프로토콜 : TLS 또는 TCP
- 큐(최대/최소) 및 포트 설정
- 포트의 전송/수신 속도, 미디어에 대한 활성화/비활성화

③ **하나 이상의 물리적 스위치**

- 오픈플로우 컨트롤러에 의해 여러개의 컨트롤러에 관리를 받는 하나 이상의 논리적인 스위치로 동작 가능
- OF-Config는 논리적 스위치의 설정을 도와 줌

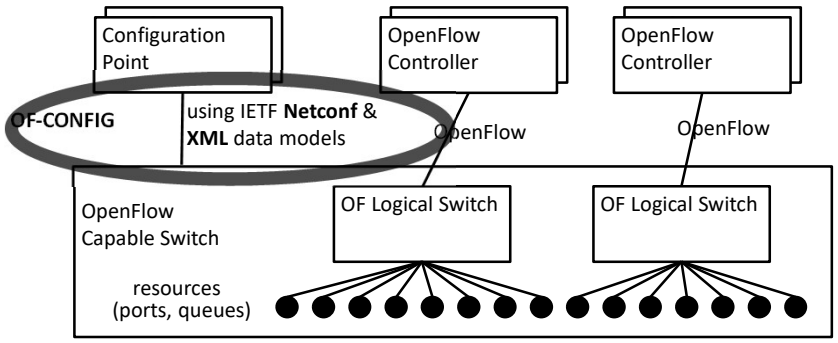


344

III. 오픈플로우 컨피그 2) 구성

2. 구성 기본 모델

① 설정 포인트(Configuration Point)
② 오픈플로우 가능한 스위치(한개 이상의 논리적 스위치 포함)
③ 오픈플로우 컨트롤러



345

III. 오픈플로우 컨피그 3) 버전

1. 버전 1.0

① 2011년 12월 23일 공개
② 오픈플로우 v1.2 기반으로 작성
③ 논리적 스위치에게 컨트롤러 할당
④ 논리적 스위치에게 설정값 전달
⑤ 포트 및 큐의 일부 세부사항 설정

346

III. 오픈플로우 컨피그 3) 버전

2. 버전 1.1

① **2012년 1월 25일 공개**

② **오픈플로우 v1.3 기반으로 작성**

- 인증에 대한 설정
- 논리적 스위치의 능력치에 대한 설정
- 논리적 터널링 지원(VXLAN, NVGRE등)

③ **업데이트**

- 버전 1.1.1 : 2013년 3월 23일 공개, 버전 1.1 단종 후 오픈플로우 v1.3.1 기반으로 오류 수정

SDN/NFV for 5G
james@jslab.kr

III. 오픈플로우 컨피그 3) 버전

3. 버전 1.2

① **2014년 공개**

② **오픈플로우 v1.3 기반으로 작성**

③ **간편한 네트워크 구성 감지**

④ **논리적 스위치에 대한 자원 사용 할당**

SDN/NFV for 5G
james@jslab.kr

III. 오픈플로우 컨피그 3) 버전

4. 버전별 비교

언제	무엇을	어떻게
2011년 12월	OF-Config v1.0	▪ 2012년 1월에 발표 ▪ 오픈플로우 스위치를 위한 구성/관리 (Configuration/Management) 지정
2012년 1월	OF-Config v1.1	▪ Discovery 기능 추가 ▪ 터널링 구성 추가 ▪ Error 처리
2013년 3월	OF-Config v1.1.1	▪ v1.1 수정 강화
2014년	OF-Config v1.2	▪ 논리적 가상 스위치를 위한 자원 지정 ▪ 토폴로지 감지 ▪ 이벤트 발생

III. 오픈플로우 컨피그 4) Netconf & Yang

1. Netconf는 관리용 프로토콜로 선정

① 이상적인 솔루션으로 받아드릴 필요는 없음
② 여전히 대안에 대해 논의 중

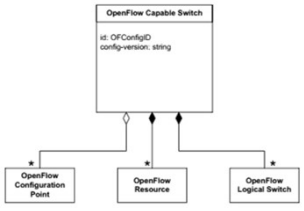
2. XML구조는 모델링 언어로서 선정

① Yang(데이터모델링 언어)도 사용될 수 있으나, XML이 일반적이다
② 일반적인 XML 구조는 Yang을 이용해서 생성
③ OF-CONFIG 데이터모델링은 YANG과 호환

3. 현재까지 설정에 초점을 맞추고 진행

4. 오픈플로우 네트워크의 초기 설정은제일 먼저 해결해야 할 과제


```
<capable-switch>
  <id>CapableSwitch0</id>
  <configuration-points>
    ...
  </configuration-points>
  <resources>
    ...
  </resources>
  <logical-switches>
    ...
  </logical-switches>
</capable-switch>
```



SDN/NFV for 5G

james@jslab.kr

IV. SDN 컨트롤러

JS Lab

351

SDN/NFV for 5G

james@jslab.kr

IV. SDN 컨트롤러 1) 개요

❖ **컨트롤러**

① 네트워크 전체에 대한 완전한 시각을 가진 중앙 집중형 네트워크 제어 시스템

② 어플리션과 네트워크 장비 사이를 연결시켜주는 인터페이스

③ 주변 장치

- 스위치 : 들어오는 패킷을 처리하기 위한 플로우 테이블을 소유
- 애플리케이션 : 직접적인 정책과 네트워크에서의 행동을 정의

④ 기본적인 기능

- 사용자 인터페이스 : 대부분 그래픽 기반(GUI)
- 플로우 제어 : 스위치 및 라우팅
- 애플리케이션을 위한 API 제공

⑤ 추가적인 기능 예

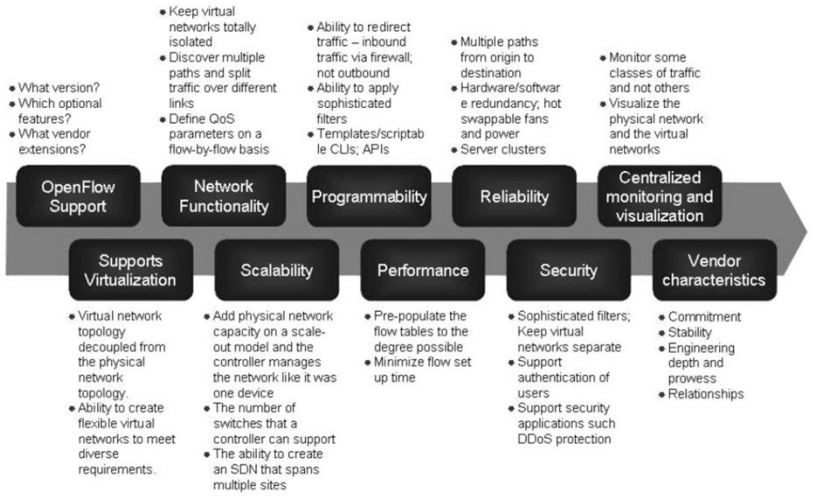
- L4 스위치(로드밸런서)
- 방화벽
- 라우터

JS Lab

352

IV. SDN 컨트롤러 2) 특징

1. 필수요소



SDN/NFV for 5G

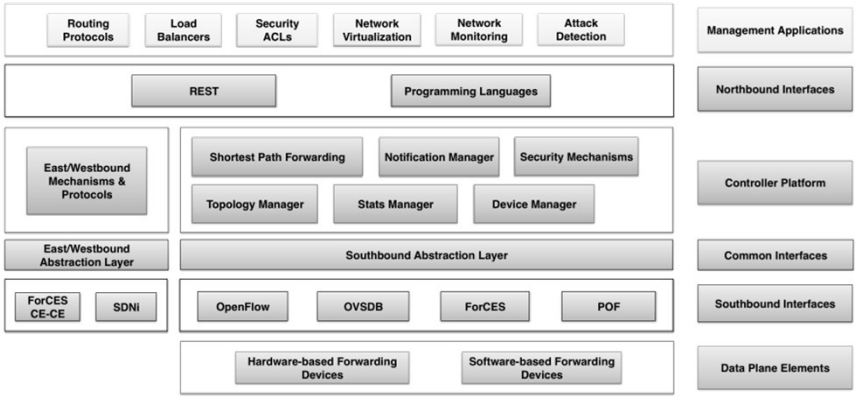
james@jslab.kr

JS Lab

353

IV. SDN 컨트롤러 2) 특징

2. 아키텍처



SDN/NFV for 5G

james@jslab.kr

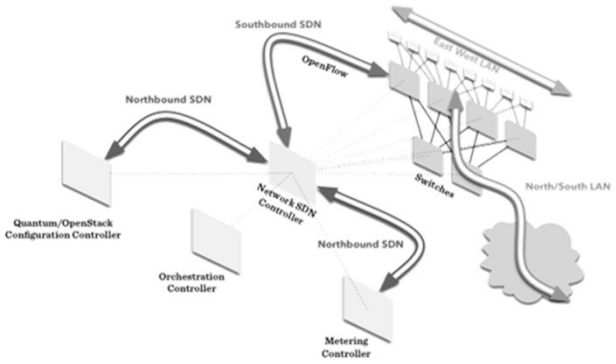
JS Lab

354

IV. SDN 컨트롤러 3) API

1. 사우스바운드 API

① 위치 : 컨트롤러와 스위치 사이 사우스바운드 인터페이스 사이
② 종류 : 오픈플로우, SMTP, BGP, XMPP 등
③ 적용 : 오픈플로우, 특정제조사의 기술, 둘다



IV. SDN 컨트롤러 3) API

2. 노스바운드 API

① 아직 표준화 된 API 없음
② 위치 : 컨트롤러와 애플리케이션 사이
③ 종류 : C++, JAVA, 파이썬, RESTful등
④ 주요 이벤트
- RESTful로는 불가능
- 스위치나 사용자 장비에 대한 이벤트
- 패킷 이벤트

⑤ 동작
- 플로우 추가, 제거, 수정
- 이벤트를 받는 즉시 동작 수행 : 패킷 제거(drop), 수정, 전달, 플로우 추가, 제거, 수정

IV. SDN 컨트롤러 3) API

3. RESTful API

① **REST : REpresentational Status Transfer, 표현 상태 정의**

② **RESTful API : REST 방식의 API**

③ **W3C에 의해 개발된 소프트웨어 아키텍쳐 스타일**

④ **HTTP 위에서 부가적인 전송 계층 없이 프로그램과 서로 정보 전달하기 위한 간단한 구조**

- HTTP URL + HTTP Method
  - ✓ HTTP URL : 특정 서비스의 자원을 명시
  - ✓ HTTP Method : 명시된 자원으로 무엇을 할지 정의

⑤ **특징 :**

- 개발자가 매우 직관적인 서비스를 제작 가능
- 직관적이기 때문에 유지보수도 매우 편리
- URI의 설계에 따라 복잡성이 결정되기 때문에 개발 단계에서 설계가 중요
  - ✓ http://*domain*/members
  - ✓ GET(확인), POST(생성), DELETE(삭제), PUT(수정)

IV. SDN 컨트롤러 4) 종류

1. 개방형 컨트롤러

① **해외**

- Floodlight : 자바&REST, 버전 1.0, BigSwitch Networks에서 공개
- Beacon : JAVA, floodligh와 OpenDayLight로 발전
- RYU : 파이썬, 버전 1.0, 1.3, 빠르고 간편한 제품 개발, NTT 지원
- OpenContrail : 주니퍼에서 공개하는 컨트롤러로 XMPP만 지원
- OpenDayLight : 오픈플로우 지원안되는 스위치도 지원, 시스코 기반
- ONOS : 대형네트워크에서 사용가능한 분산형, ON.Lab 주도

② **국내**

- Open IRIS : ETRI에서 Floodlight를 멀티코어에서 동작하도록 성능 및 UI 개선
- OpenMUL : 쿨클라우드에서 C를 기반으로 제작

IV. SDN 컨트롤러 4) 종류

2. 상용 컨트롤러

① **해외**

- XNC : Cisco, Floodlight의 성능 및 UI 개선 후 오픈데이라이트에 기증
- VAN : HP, SDN 전용 앱스토어 제공
- ProgrammableFlow : NEC, 물리적 환경 및 논리적 UI를 하나의 화면으로 제공
- Contrail : 주니퍼, Contrail을 인수하여 제품화, XMPP만 지원
- NSX : VMware, Nicira를 인수하여 오버레이 기반 SDN 구현
- BNC : BigSwitch, Floodlight의 상용버전으로 현재 Big Cloud Fabric의 일부분

② **국내**

- BEEM : 쿨클라우드, C 기반 상업용 SDN 컨트롤러
- Obelle : 아토리서치, 고성능, 안정성을 지향

IV. SDN 컨트롤러 4) 종류

3. 도입시 고려대상

① **개방형 vs 상용 컨트롤러**
② **하드웨어 일체형 vs 소프트웨어**
③ **API의 선택 : 사우스바운드 선택, 노스바운드 API 표준이 없음**
④ **애플리케이션 사이에서의 조화 : 기존 및 신규 애플리케이션, 개발 능력**
⑤ **성능 : 확장성, 고가용성, 플로우 지원 성능**
⑥ **도입 및 유지보수 비용**

SDN/NFV for 5G

james@jslab.kr

V. 오픈플로우 기술

JS Lab

361

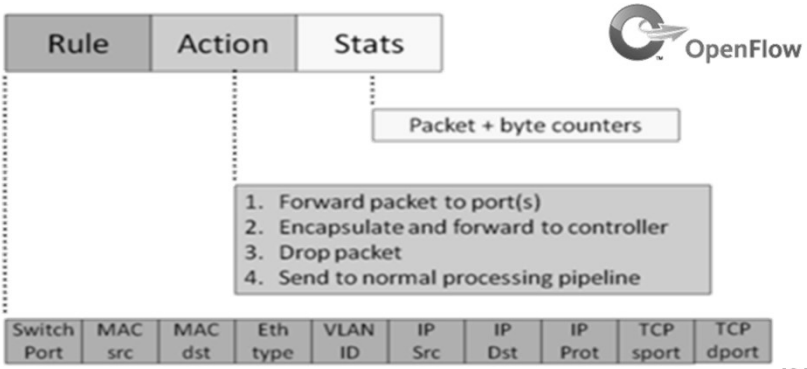
SDN/NFV for 5G

james@jslab.kr

V. 오픈플로우 기술 1) 오픈플로우

1. 오픈플로우(OpenFlow)란?

① 네트워크 장비에서 패킷 전달 기능과 제어 기능을 분리하여 이들 두 기능간의 통신을 위한 프로토콜
② ONF에 의해 관리되는 SDN의 표준 사우스바운드 프로토콜
③ 버전별로 관리되고 있으며, 버전별로 지속적인 업데이트



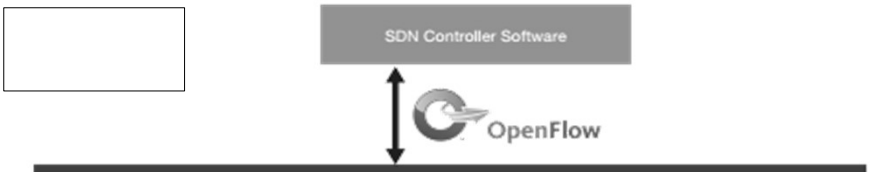
JS Lab

362

V. 오픈플로우 기술 1) 오픈플로우

2. 오픈플로우 테이블

① 오픈플로우 헤더는 총 12개의 세부 항목(Tuple)로 구성
② 각 항목의 값은 특정한 의미를 가지는 것이 아니라 값으로만 존재



OpenFlow-enabled Network Device

Flow Table comparable to an instruction set

MAC src	MAC dst	IP Src	IP Dst	TCP dport	...	Action	Count
*	10:20:..	*	*	*	*	port 1	250
*	*	*	5.6.7.8	*	*	port 2	300
*	*	*	*	25	*	drop	892
*	*	*	192.*	*	*	local	120
*	*	*	*	*	*	controller	11

363

V. 오픈플로우 기술 1) 오픈플로우

3. 오픈플로우 예제

① Switching : 목적지 MAC 주소가 00:1f:af:ab:cd:ef면 6번 포트로 보내라

Ingress Port	Ethernet			VLAN			IP				TCP/UDP		Action
	Source	Destination	Type	ID	Priority	Source	Destination	Protocol	TOS	Source	Destination	
*	*	00:1f:af:ab:cd:ef	*	*	*	*	*	*	*	*	*	Port 6

② Flow Switching : 1.2.3.4 IPv4와 TCP 포트 17264번을 가지고 3번 포트의 VLAN 1으로 들어오는 00:2e:df:ee:ee:ee의 이더넷 패킷이 00:1f:af:ab:cd:ef를 사용하는 5.6.7.8 IPv4의 TCP 포트 80으로 가고자 하면 포트 7번으로 보내라

Ingress Port	Ethernet			VLAN			IP				TCP/UDP		Action
	Source	Destination	Type	ID	Priority	Source	Destination	Protocol	TOS	Source	Destination	
Port 3	00:2e:df:ee:ee:ee	00:1f:af:ab:cd:ef	0x0800	Vlan 1	*	1.2.3.4	5.6.7.8	4	*	17264	80	Port 7

③ Firewall : 목적지 포트가 22번이면 모두 차단하라

Ingress Port	Ethernet			VLAN			IP				TCP/UDP		Action
	Source	Destination	Type	ID	Priority	Source	Destination	Protocol	TOS	Source	Destination	
*	*	*	*	*	*	*	*	*	*	*	22	Drop

364

V. 오픈플로우 기술 1) 오픈플로우

3. 오픈플로우 예제

④ Routing : 목적지 주소가 5.6.7.8이면 6번 포트로 보내라

Ingress Port	Ethernet			VLAN			IP			TCP/UDP		Action
	Source	Destination	Type	ID	Priority	Source	Destination	Protocol	TOS	Source	Destination	
*	*	*	*	*	*	*	5.6.7.8	*	*	*	*	Port 6

⑤ VLAN Switching : VLAN 1번을 달고 들어오면, 2,4번 포트로 보내라

Ingress Port	Ethernet			VLAN			IP			TCP/UDP		Action
	Source	Destination	Type	ID	Priority	Source	Destination	Protocol	TOS	Source	Destination	
*	*	*	*	1	*	*	*	*	*	*	*	Port 2,4

⑥ Mirroring : 80포트에 대한 통신은 웹방화벽(1번 포트)으로 보내고, 모니터링을 위해 5번 포트로 복사하고, 나머지 포트는 다 차단하라

Ingress Port	Ethernet			VLAN			IP			TCP/UDP		Action
	Source	Destination	Type	ID	Priority	Source	Destination	Protocol	TOS	Source	Destination	
*	*	*	*	*	*	*	*	*	*	*	80	Port 1,5
*	*	*	*	*	*	*	*	*	*	*	*	Drop

V. 오픈플로우 기술 1) 오픈플로우

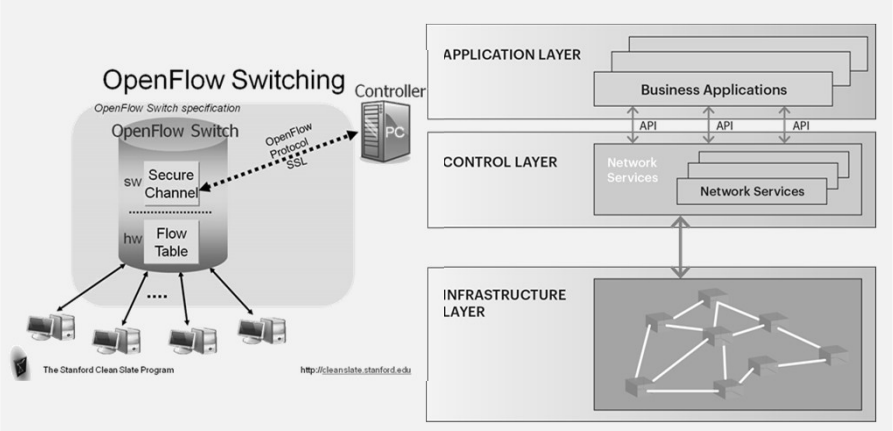
4. Flow Table Entries

Match Fields	Counters	Instructions
1. Ingress Port 2. Metadata 3. Ethernet source address 4. Ethernet destination address 5. Ether type 6. VLAN id 7. VLAN priority 8. MPLS label 9. MPLS traffic class 10. IPv4 source address 11. IPv4 destination address 12. IPv4 protocol / ARP opcode 13. IPv4 ToS bits 14. TCP / UDP / SCTP src port 15. ICMP Type 16. TCP / UDP / SCTP dst port 17. ICMP Code	A. Per Table 1. Reference count (active entries) 2. Packet Lookups 3. Packet Matches B. Per Flow 1. Received Packets 2. Received Bytes 3. Duration (seconds) 4. Duration (nanoseconds) C. Per Port 1. Received Packets 2. Transmitted Packets 3. Received Bytes 4. Transmitted Bytes 5. Received Drops 6. Transmitted Drops 7. Received Errors 8. Transmitted Errors 9. Receive Frame Alignment Errors 10. Receive Overrun Errors 11. Receive CRC Errors 12. Collisions D. Per Queue 1. Transmitted Packets 2. Transmitted Bytes 3. Transmit Overrun Errors E. Per Group 1. Reference Count (flow entries) 2. Packet Count 3. Byte Count F. Per Bucket 1. Packet Count 2. Byte Count	1. Forward packet to zero or more ports 2. Encapsulate and forward to controller 3. Send to normal processing pipeline 4. Modify Fields 5. Any extensions you add!

V. 오픈플로우 기술 2) 오픈플로우 동작

1. 오픈플로우 구성

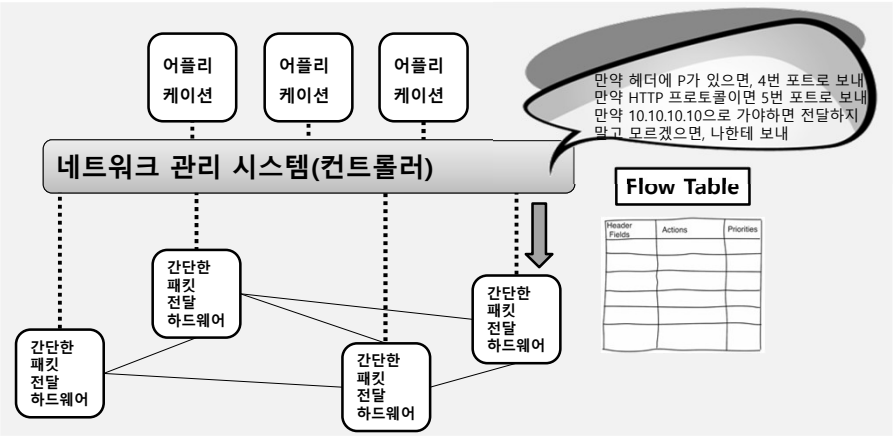
① 사용 : 컨트롤러와 스위치 사이에서 사우스바운드 API로서 동작
② 방법 : SSL을 이용하여 서로간의 메시지 전달



V. 오픈플로우 기술 2) 오픈플로우 동작

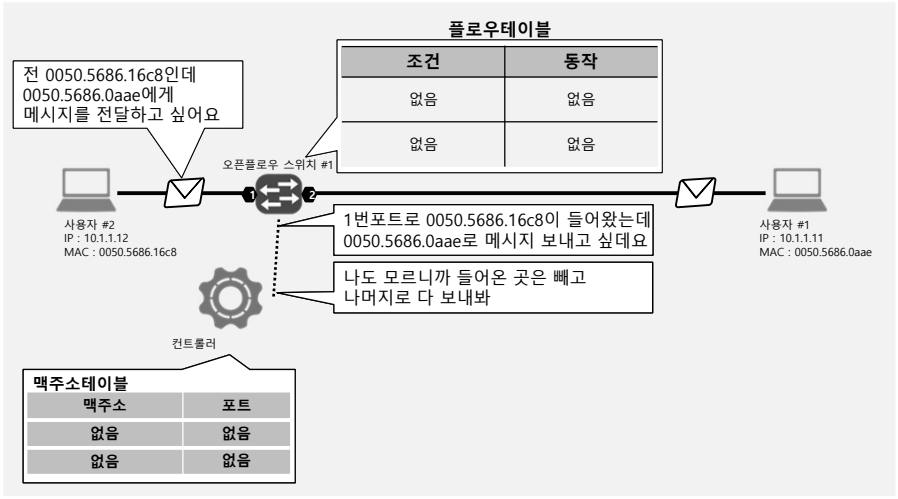
2. 오픈플로우 동작

① 스위치는 경로를 모르는 패킷은 컨트롤러로 문의
② 컨트롤러는 스위치에게 정책에 따른 경로 전달
③ 스위치는 컨트롤러로부터 받은 경로로 패킷 전달



V. 오픈플로우 기술 2) 오픈플로우 동작

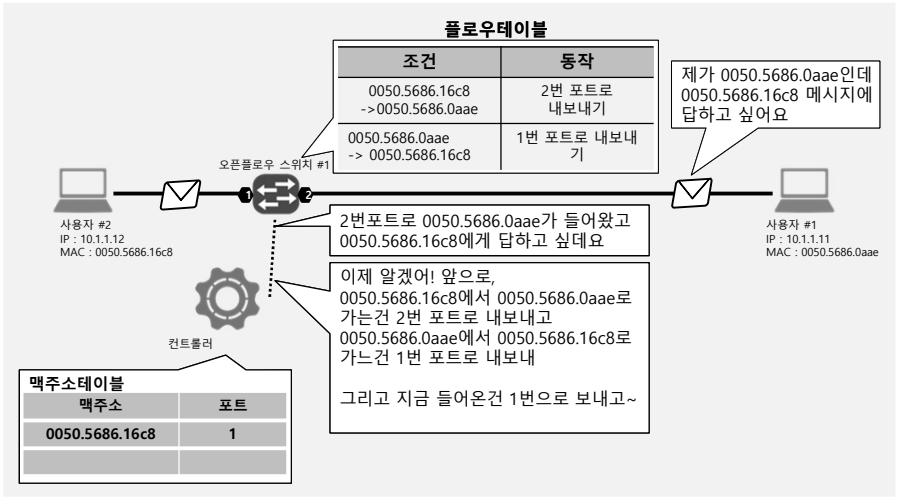
3. 오픈플로우 동작 절차



369

V. 오픈플로우 기술 2) 오픈플로우 동작

3. 오픈플로우 동작 절차



370

V. 오픈플로우 기술 3) SDN 토폴로지 디스커버리

1. SDN 토폴로지 디스커버리

① 컨트롤러에서 Topology Discovery 수행

② OFDP = OpenFlow Discovery Protocol
대부분 SDN 컨트롤러에서 적용 → de facto standard
OFDP는 LLDP 패킷 포맷을 사용
OFDP 동작은 전혀 다름

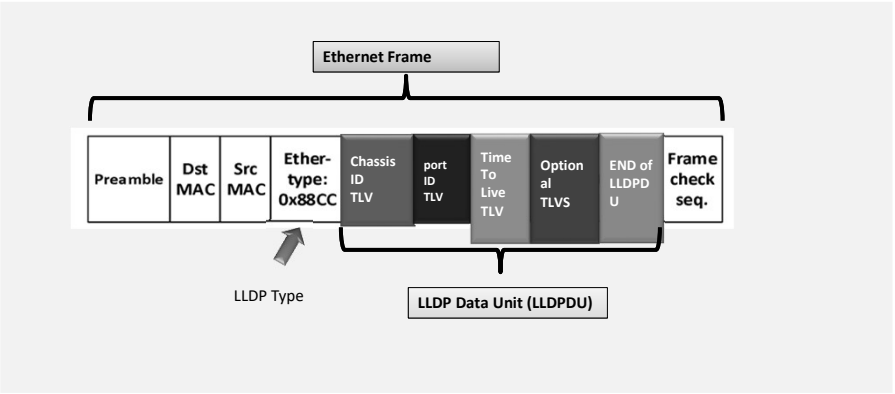
③ LLDP = Link Layer Discovery Protocol
IEEE 802.1AB
Used in traditional Ethernet network devices

371

V. 오픈플로우 기술 3) SDN 토폴로지 디스커버리

2. Topology Discovery Frame structure

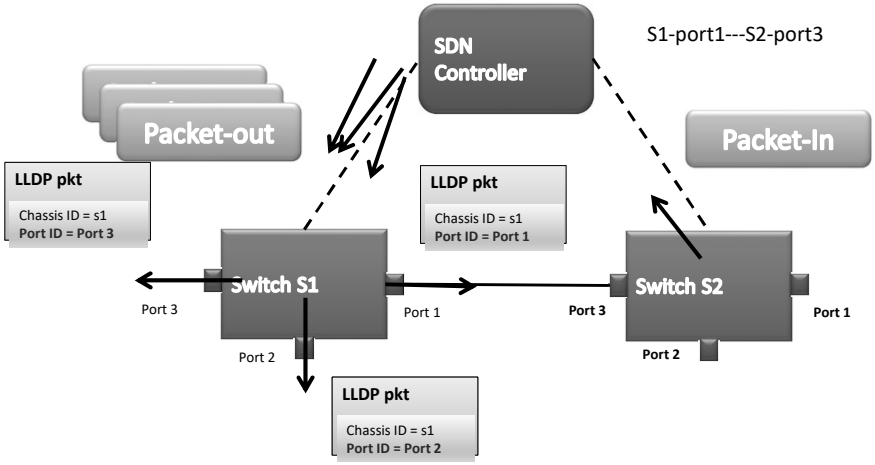
① LLDP 정보는 이더넷 프레임 형태로 기기들에 의해 전송
② 각 프레임은 하나의 LLDP Data Unit (LLDPDU) 포함
③ 각 LLDPDU 는 type-length-value (TLV) 구성 순서에 따름



372

V. 오픈플로우 기술 3) SDN 토폴로지 디스커버리

3. OFDP (OpenFlow Discovery Protocol)

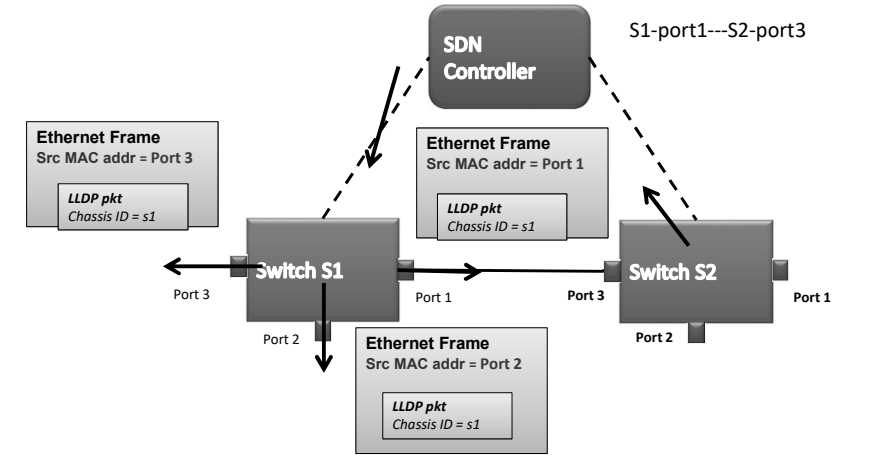


SDN/NFV for 5G
james@jslab.kr
JS Lab

373

V. 오픈플로우 기술 3) SDN 토폴로지 디스커버리

4. OFDP v2 (OpenFlow Discovery Protocol)



SDN/NFV for 5G
james@jslab.kr
JS Lab

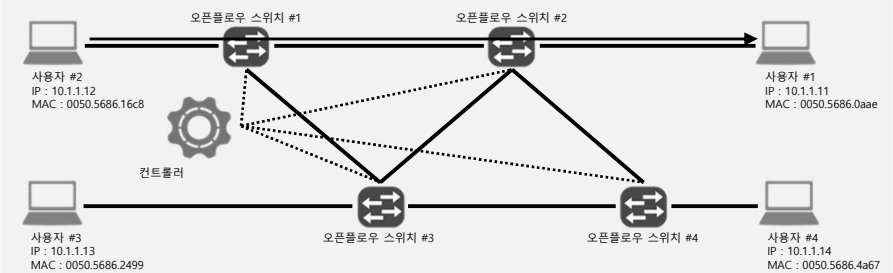
374

SDN/NFV for 5G

james@jslab.kr

V. 오픈플로우 기술 4) 오픈플로우 통신

1. 사용자 #2가 사용자 #1에게 Ping을 시도



ICMP 패킷(L3)		TYPE	CODE	CHECKSUM	DATA	
		.....	출발지 IP	목적지 IP		
IP 패킷(L3)		.....	10.1.1.12	10.1.1.11	ICMP	
	프리앰블/SFD	목적지 MAC	출발지 MAC			
이더넷 프레임 (L2)	10101011	?	0050.5686.16c8	Type/Length	DATA	FCS

JS Lab

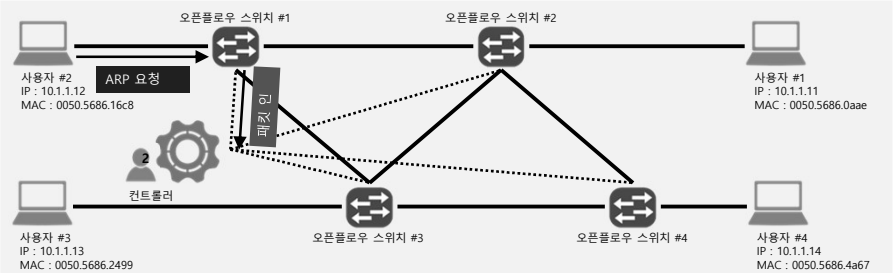
375

SDN/NFV for 5G

james@jslab.kr

V. 오픈플로우 기술 4) 오픈플로우 통신

2. 사용자 #2는 ARP 캐쉬 테이블 생성을 위한 'ARP 요청' 패킷을 전송



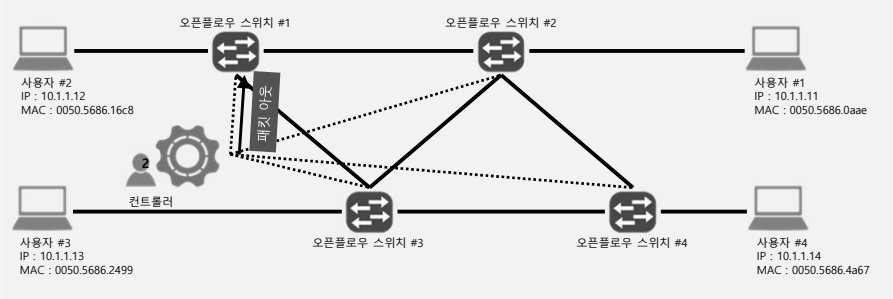
	목적지 MAC	출발지 MAC				
ARP 요청	FFFF:FFFF:FFFF	0050.5686.16c8	Type/Length	ARP	FCS	
패킷 인	이더넷	IP	TCP	오픈플로우	DATA	FCS

JS Lab

376

V. 오픈플로우 기술 4) 오픈플로우 통신

3. 사용자 #1의 주소를 모르는 컨트롤러는 '패킷 아웃(Packet Out)' 답변



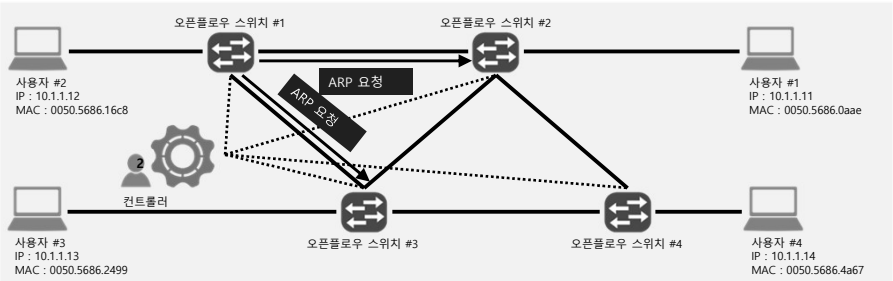
	목적지 MAC	출발지 MAC				
ARP 요청	FFFF:FFFF:FFFF	0050.5686.16c8	Type/Length	ARP	FCS	
패킷 아웃	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

377

V. 오픈플로우 기술 4) 오픈플로우 통신

4. '패킷 아웃' 메시지를 받은 스위치 #1은 'ARP 응답' 을 전달



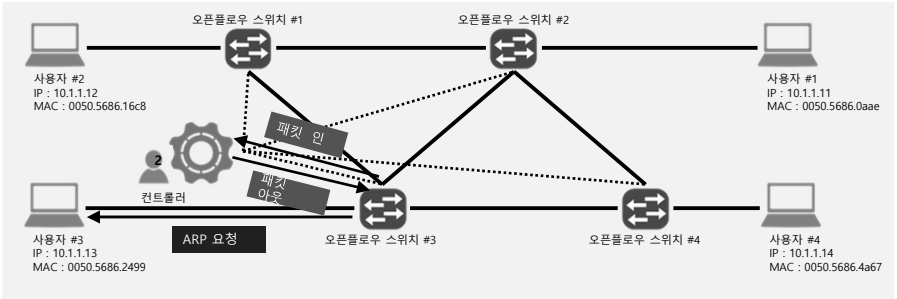
		발신자 MAC	발신자 IP	원하는 상대 MAC	원하는 상대 IP	
ARP 패킷(L3)		0050.5686.16c8	10.1.1.12	0000.0000.0000	10.1.1.11	
	프리엠블/SFD	목적지 MAC	출발지 MAC			
이더넷 프레임 (L2)	10101011	ffff.ffff.ffff	0050.5686.16c8	Type/Length	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

378

V. 오픈플로우 기술 4) 오픈플로우 통신

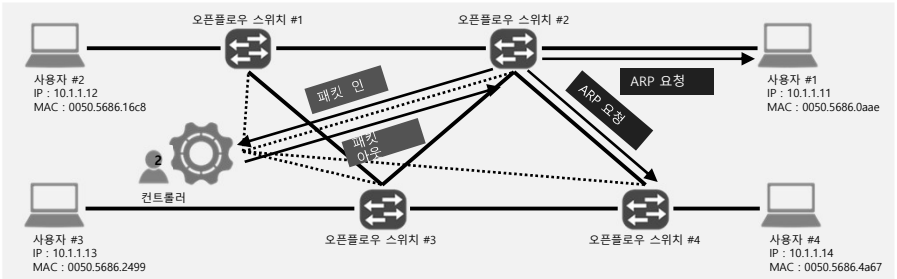
5. 'ARP 요청'을 받은 스위치 #3은 스위치#1과 동일 방식으로 동작



	목적지 MAC	출발지 MAC				
ARP 요청	FFFF:FFFF:FFFF	0050.5686.16c8	Type/Length	ARP	FCS	
패킷 인/아웃	이더넷	IP	TCP	오픈플로우	DATA	FCS

V. 오픈플로우 기술 4) 오픈플로우 통신

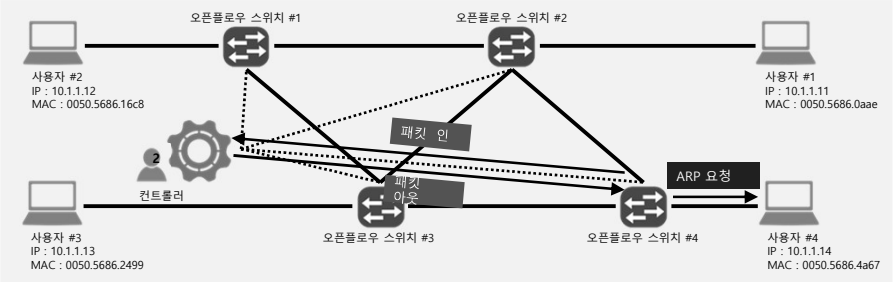
6. 'ARP 요청'을 받은 스위치 #2은 스위치#1과 동일 방식으로 동작



	목적지 MAC	출발지 MAC				
ARP 요청	FFFF:FFFF:FFFF	0050.5686.16c8	Type/Length	ARP	FCS	
패킷 인/아웃	이더넷	IP	TCP	오픈플로우	DATA	FCS

V. 오픈플로우 기술 4) 오픈플로우 통신

7. 'ARP 요청'을 받은 스위치 #4는 스위치#1과 동일 방식으로 동작

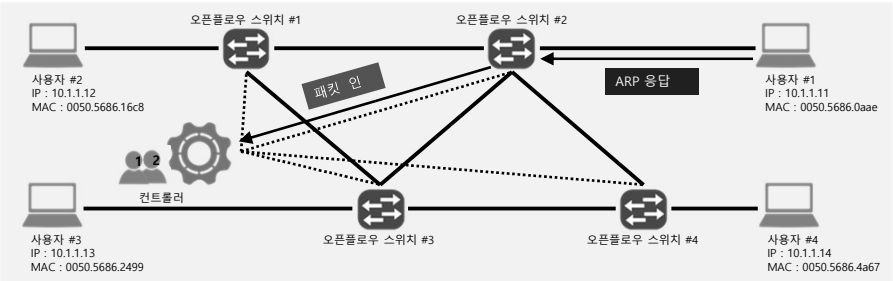


	목적지 MAC	출발지 MAC				
ARP 요청	FFFF:FFFF:FFFF	0050.5686.16c8	Type/Length	ARP	FCS	
패킷 인/아웃	이더넷	IP	TCP	오픈플로우	DATA	FCS

381

V. 오픈플로우 기술 4) 오픈플로우 통신

8. 사용자 #1은 'ARP 요청'에 응답하기 위해 'ARP 응답'을 스위치 #2에게 전송하고, 스위치는 '패킷 인' 메시지로 컨트롤러에게 전송

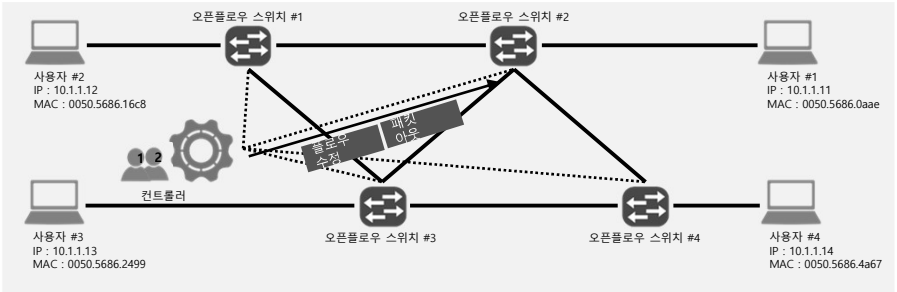


	목적지 MAC	출발지 MAC				
ARP 응답	0050.5686.16c8	0050.5686.0aae	Type/Length	ARP	FCS	
패킷 인	이더넷	IP	TCP	오픈플로우	DATA	FCS

382

V. 오픈플로우 기술 4) 오픈플로우 통신

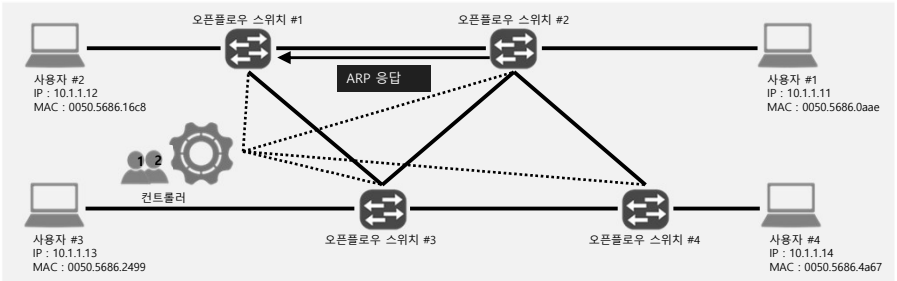
9. 컨트롤러는 '플로우 수정(Modification)' 메시지를 스위치 #2에게 전송



	목적지 MAC	출발지 MAC				
ARP 응답	0050.5686.16c8	0050.5686.0aae	Type/Length	ARP	FCS	
플로우수정/패킷아웃	이더넷	IP	TCP	오픈플로우	DATA	FCS

V. 오픈플로우 기술 4) 오픈플로우 통신

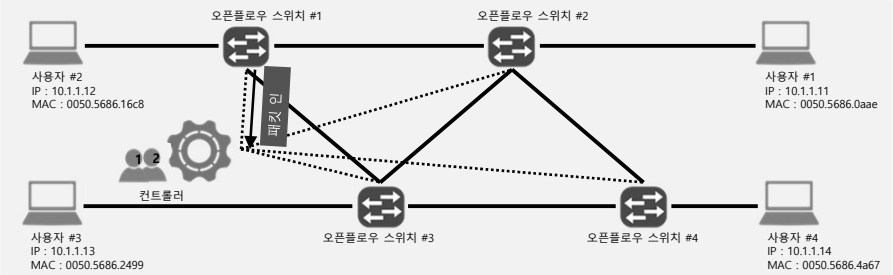
10. 스위치 #2는 수신한 '플로우 수정(Modification)' 메시지를 이용하여 플로우 테이블을 생성, 지정된 동작에 따라 'ARP 응답' 패킷을 스위치 #1에게 전달



		발신자 MAC	발신자 IP	원하는 상대 MAC	원하는 상대 IP	
ARP 패킷(L3)		0050.5686.0aae	10.1.1.11	0050.5686.16c8	10.1.1.12	
	프리앰블/SFD	목적지 MAC	출발지 MAC			
이더넷 프레임 (L2)	10101011	0050.5686.16c8	0050.5686.0aae	Type/Length	DATA	FCS

V. 오픈플로우 기술 4) 오픈플로우 통신

11. 'ARP 응답' 메시지를 수신한 스위치 #1은 스위치 #2와 같은 방식으로 동작



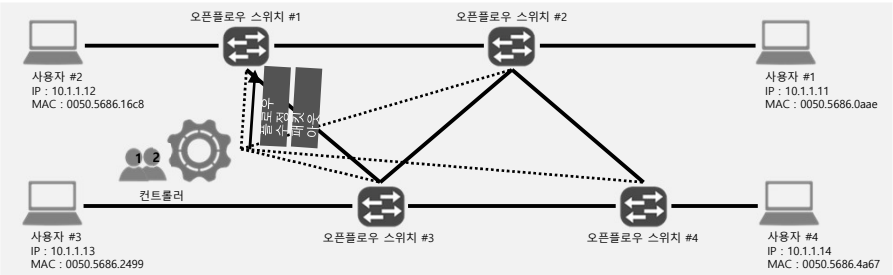
	목적지 MAC	출발지 MAC				
ARP 응답	0050.5686.16c8	0050.5686.0aae	Type/Length	ARP	FCS	
패킷 인	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

385

V. 오픈플로우 기술 4) 오픈플로우 통신

12. 컨트롤러는 '플로우 수정(Modification)' 메시지를 스위치 #1에게 전송



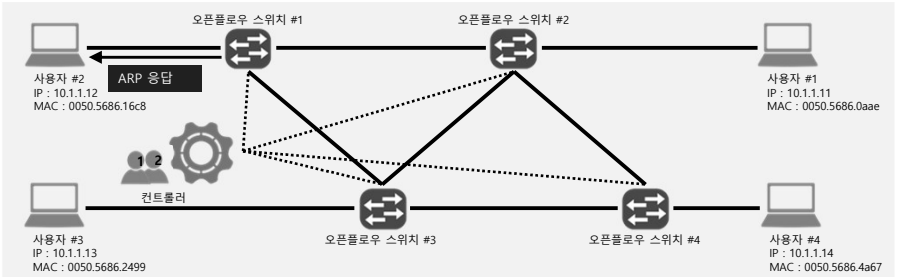
	목적지 MAC	출발지 MAC				
ARP 응답	0050.5686.16c8	0050.5686.0aae	Type/Length	ARP	FCS	
플로우수정/패킷 아웃	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

386

V. 오픈플로우 기술 4) 오픈플로우 통신

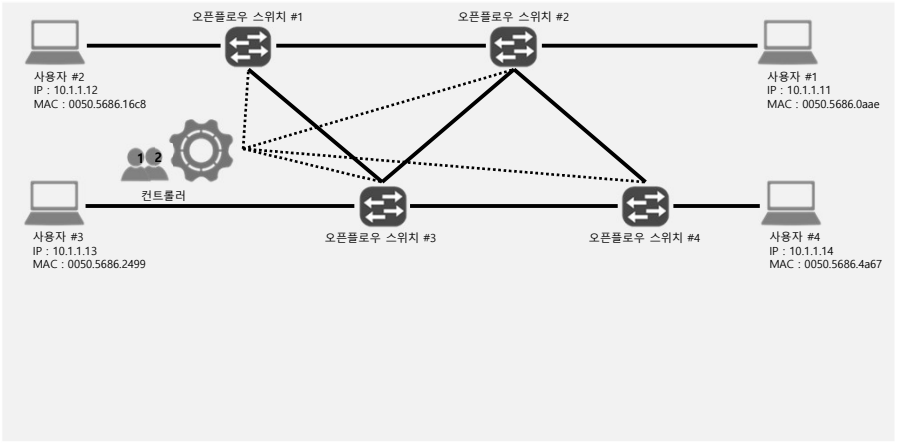
13. 스위치 #1는 수신한 '플로우 수정(Modification)' 메시지를 이용하여 플로우 테이블을 생성하고 지정된 동작에 따라 'ARP 응답'을 사용자 #2에 전달



		발신자 MAC	발신자 IP	원하는 상대 MAC	원하는 상대 IP	
ARP 패킷(L3)		0050.5686.0aae	10.1.1.11	0050.5686.16c8	10.1.1.12	
	프리엠블/SFD	목적지 MAC	출발지 MAC			
이더넷 프레임 (L2)	10101011	0050.5686.16c8	0050.5686.0aae	Type/Length	DATA	FCS

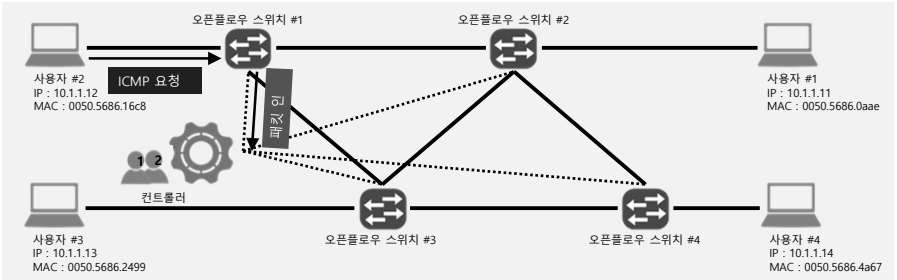
V. 오픈플로우 기술 4) 오픈플로우 통신

14. 사용자 #2는 'ARP 응답'을 수신하여 ARP 캐쉬 테이블을 생성



V. 오픈플로우 기술 4) 오픈플로우 통신

15. 사용자 #2는 'ICMP 요청' 패킷을 전송



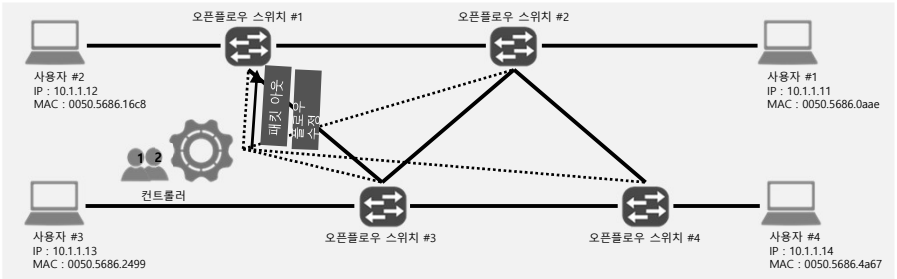
	목적지 MAC	출발지 MAC				
ICMP 요청	0050.5686.0aae	0050.5686.16c8	Type/Length	ARP	FCS	
패킷 인	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

389

V. 오픈플로우 기술 4) 오픈플로우 통신

16. 컨트롤러는 '패킷 아웃' 메시지와 함께 '플로우 수정' 메시지를 생성하여 스위치 #1에게 전송



	목적지 MAC	출발지 MAC				
ICMP 요청	0050.5686.0aae	0050.5686.16c8	Type/Length	ARP	FCS	
플로우수정/패킷 인	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

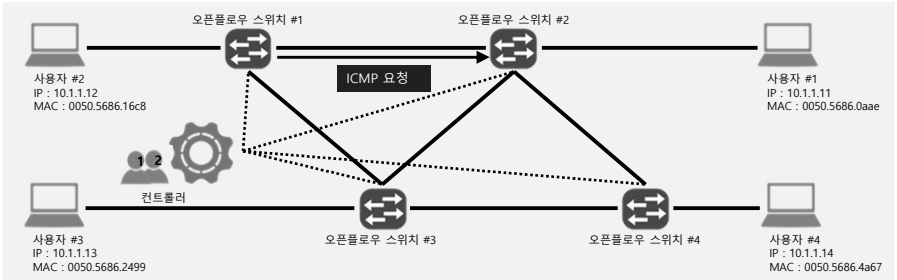
390

SDN/NFV for 5G

james@jslab.kr

V. 오픈플로우 기술 4) 오픈플로우 통신

17. 스위치 #1은 수신 '플로우 수정(Modification)' 를 이용하여 플로우 테이블을 생성하고 지정된 동작에 따라 'ICMP 요청' 메시지를 스위치 #2에게 전달



ICMP 패킷(L3)		TYPE	CODE	CHECKSUM	DATA	
		.....	출발지 IP	목적지 IP		
IP 패킷(L3)		.....	10.1.1.12	10.1.1.11	ICMP	
	프리엠블/SFD	목적지 MAC	출발지 MAC			
이더넷 프레임 (L2)	10101011	0050.5686.0aae	0050.5686.16c8	Type/Length	DATA	FCS

JS Lab

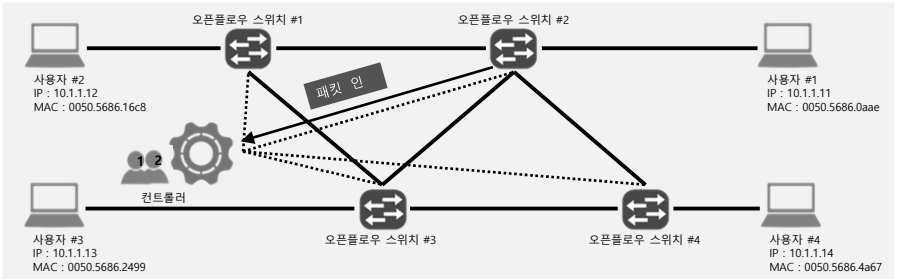
391

SDN/NFV for 5G

james@jslab.kr

V. 오픈플로우 기술 4) 오픈플로우 통신

18. 'ICMP 요청' 메시지를 수신한 스위치 #2은 스위치 #1과 같은 방식으로 동작



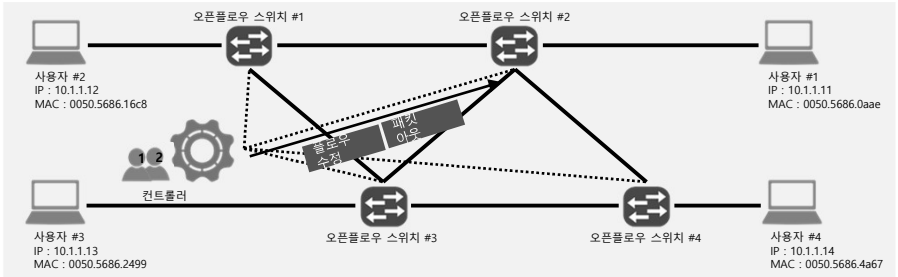
	목적지 MAC	출발지 MAC				
ICMP 요청	0050.5686.0aae	0050.5686.16c8	Type/Length	ARP	FCS	
패킷 인	이더넷	IP	TCP	오픈플로우	DATA	FCS

JS Lab

392

V. 오픈플로우 기술 4) 오픈플로우 통신

19. 컨트롤러는 '패킷 아웃' 메시지와 함께 '플로우 수정' 메시지를 생성하여 스위치 #2에게 전송



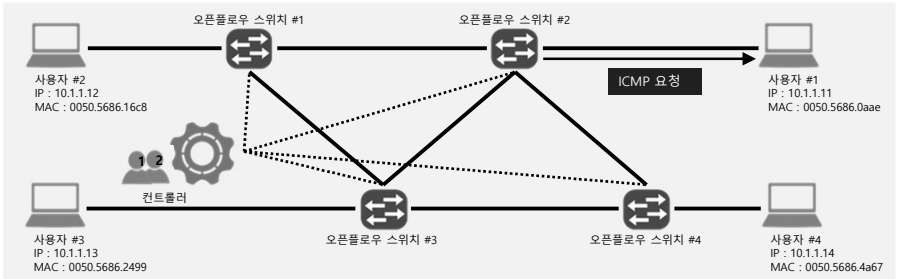
	목적지 MAC	출발지 MAC				
ICMP 요청	0050.5686.16c8	0050.5686.0aae	Type/Length	ARP	FCS	
플로우수정/패킷 아웃	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

393

V. 오픈플로우 기술 4) 오픈플로우 통신

20. 스위치 #2은 수신한 '플로우 수정(Modification)' 를 이용하여 플로우 테이블을 생성 지정된 동작에 따라 'ICMP 요청'을 사용자 #1에게 전달



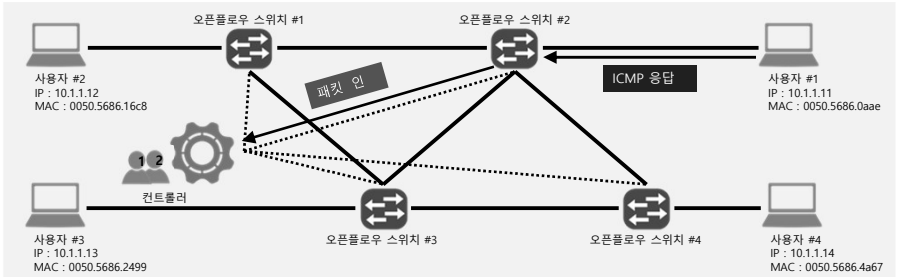
ICMP 패킷(L3)		TYPE	CODE	CHECKSUM	DATA	
		.....	출발지 IP	목적지 IP		
IP 패킷(L3)		.....	10.1.1.12	10.1.1.11	ICMP	
	프리엠블/SFD	목적지 MAC	출발지 MAC			
이더넷 프레임 (L2)	10101011	0050.5686.0aae	0050.5686.16c8	Type/Length	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

394

V. 오픈플로우 기술 4) 오픈플로우 통신

21. 사용자 #1은 'ICMP 요청'에 응답하기 위한 'ICMP 응답' 패킷을 스위치 #2에게 전송하고 이것은 '패킷 인' 메시지로 생성되어 컨트롤러에게 전송



	목적지 MAC	출발지 MAC				
ICMP 응답	0050.5686.16c8	0050.5686.0aae	Type/Length	ARP	FCS	
패킷 인	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G

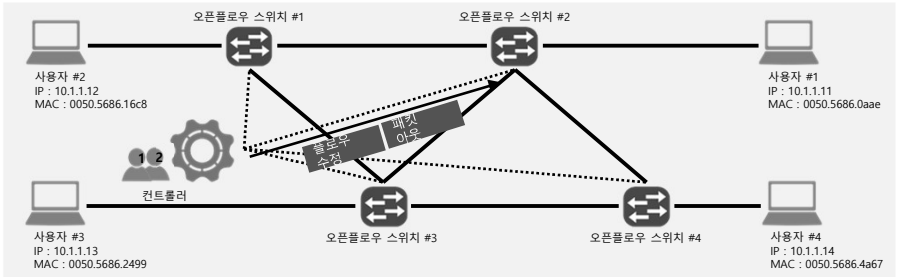
james@jslab.kr

JS Lab

395

V. 오픈플로우 기술 4) 오픈플로우 통신

22. 컨트롤러는 '패킷 아웃' 메시지와 함께 '플로우 수정' 메시지를 생성하여 스위치 #2에게 전송



	목적지 MAC	출발지 MAC				
ICMP 응답	0050.5686.16c8	0050.5686.0aae	Type/Length	ARP	FCS	
플로우수정/패킷 아웃	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G

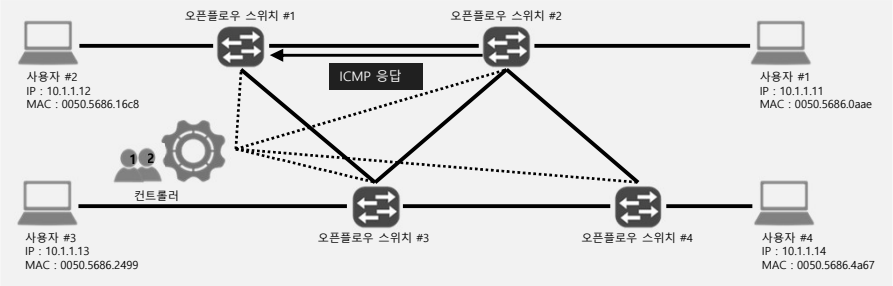
james@jslab.kr

JS Lab

396

V. 오픈플로우 기술 4) 오픈플로우 통신

23. 스위치 #2는 수신한 '플로우 수정(Modification)' 메시지를 이용하여 플로우 테이블을 생성하고 지정된 동작에 따라 'ICMP 응답' 을 스위치 #1에게 전달



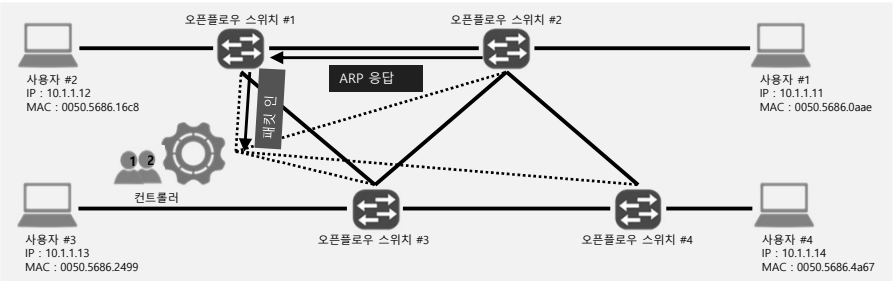
ICMP 패킷(L3)		TYPE	CODE	CHECKSUM	DATA	
		.....	출발지 IP	목적지 IP		
IP 패킷(L3)		.....	10.1.1.11	10.1.1.12	ICMP	
	프리엠블/SFD	목적지 MAC	출발지 MAC			
이더넷 프레임 (L2)	10101011	0050.5686.16c8	0050.5686.0aae	Type/Length	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

397

V. 오픈플로우 기술 4) 오픈플로우 통신

24. 'ICMP 응답' 메시지를 수신한 스위치 #1은 스위치 #2와 같은 방식으로 동작



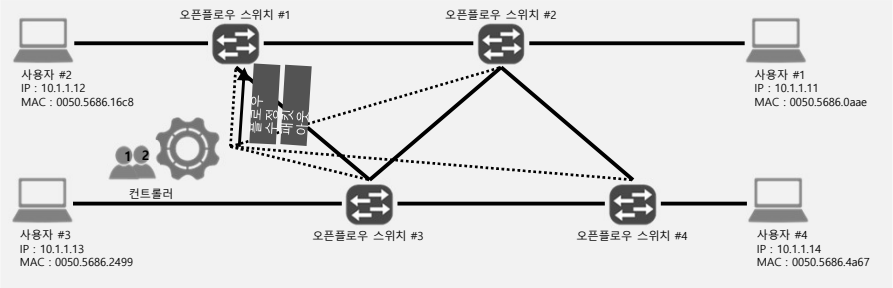
	목적지 MAC	출발지 MAC				
ICMP 응답	0050.5686.16c8	0050.5686.0aae	Type/Length	ARP	FCS	
패킷 인	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

398

V. 오픈플로우 기술 4) 오픈플로우 통신

25. 컨트롤러는 '플로우 수정(Modification)' 메시지를 스위치 #1에게 전송



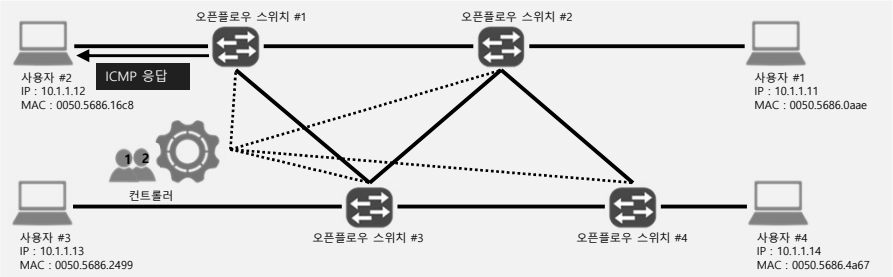
	목적지 MAC	출발지 MAC				
ICMP 응답	0050.5686.16c8	0050.5686.0aae	Type/Length	ARP	FCS	
플로우수정/패킷 아웃	이더넷	IP	TCP	오픈플로우	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

399

V. 오픈플로우 기술 4) 오픈플로우 통신

26. 스위치 #1는 수신한 '플로우 수정(Modification)' 메시지를 이용하여 플로우 테이블을 생성하고 지정된 동작에 따라 'ICMP 응답'을 사용자 #2에게 전달



ICMP 패킷(L3)		TYPE	CODE	CHECKSUM	DATA	
		.....	출발지 IP	목적지 IP		
IP 패킷(L3)		.....	10.1.1.11	10.1.1.12	ICMP	
	프리엠블/SFD	목적지 MAC	출발지 MAC			
이더넷 프레임 (L2)	10101011	0050.5686.16c8	0050.5686.0aae	Type/Length	DATA	FCS

SDN/NFV for 5G
james@jslab.kr
JS Lab

400

SDN/NFV for 5G

james@jslab.kr

I. 개요
II. 소프트웨어 정의
III. 가상화와 클라우드 서비스
IV. SDN 개요
V. NFV
VI. 오버레이 / 언더레이
VII. SDN관련 기술
VIII. 가상네트워크
IX. 텔레콤 환경을 위한 SDN/NFV
X. 관리

❖ 부록: OpenFlow
❖ 실습교재 (별도)

JS Lab

401

Community for KOREN AI Network Lab

James Ahn @ james@jslab.kr

JS Lab

402