Opensource Networking

2019. 10.

(2020년 4월까지 사용 권장)

안 종 석
james@jslab.kr
JS Lab

1

목차

2

목차

별첨

1. 리눅스 네트워킹
2. 오픈스택
3. 컨테이너 네트워킹
4. 쿠버네티스 네트워킹
5. OPNFV
6. 5G 코어네트워킹
7. AI 네트워킹 인프라

Opensource Networking
james@jslab.kr

Opensource Networking
james@jslab.kr

Opensource Networking

james@jslab.kr

I. 오픈소스 네트워킹 개요

- 개요
- Hierarchy of a Network Device
- Legacy Networks
- Opensource at the Market
- Disaggregation
- Modern Networking and SDN

https://www.linuxfoundation.org/projects/networking/

5

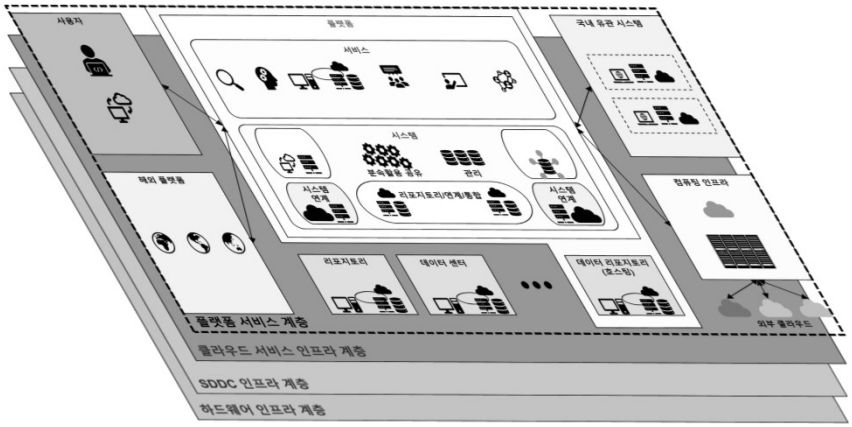
Opensource Networking

james@jslab.kr

I. 오픈소스 네트워킹 개요

❖ 개요

- 계층별 네트워크 연계
- 네트워크 간 연결을 위한 오픈소스 프로젝트 활동 활발



6

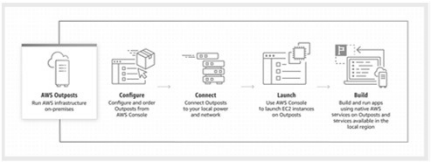
I. 오픈소스 네트워킹 개요

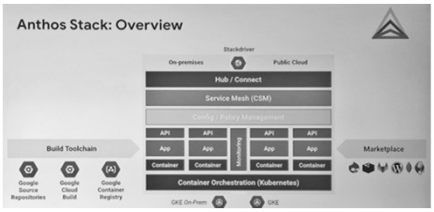
- ❖ **Disaggregation**
- ❖ **Modern Networking and SDN**
  - **Rip-and-Replace**, **Direct Fabric Programming** (Cloud Networking)
  - **Overlay**
  - **Hybrid**

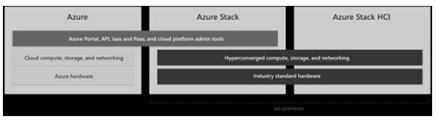
Rip-and-Replace

I. 오픈소스 네트워킹 개요

- ❖ **클라우드 서비스 회사들의 하이브리드 솔루션**
  - **AWS Outpost**
  - **MS Azure Stack**
  - **Google Anthos**
  - **IBM Cloud Pak**



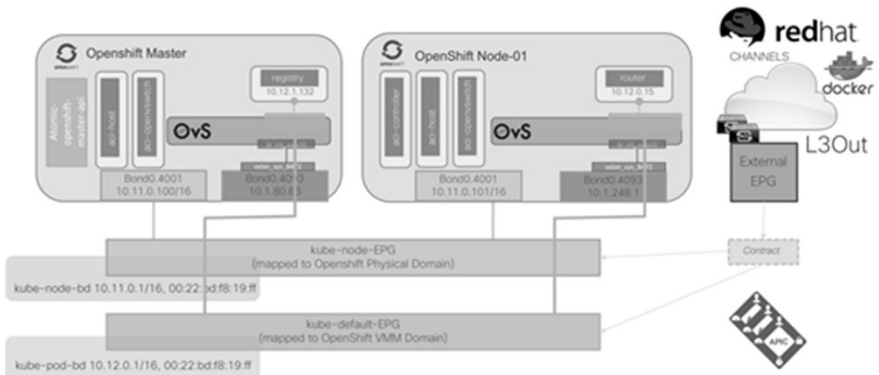




I. 오픈소스 네트워킹 개요

❖ 제조사 솔루션 연동

- 제조사들의 멀티클라우드 기반 아키텍처에 오픈소스 네트워킹 기술 채택
- Cisco, VMware 등

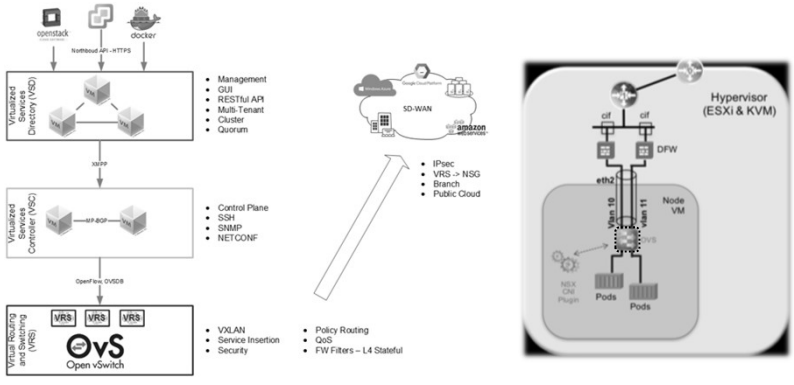


https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/white_papers/Cisco-ACI-CNI-Plugin-for-OpenShift-Architecture-and-Design-Guide.html

I. 오픈소스 네트워킹 개요

❖ 제조사 솔루션 연동

- 제조사들의 멀티클라우드 기반 아키텍처에 오픈소스 네트워킹 기술 채택
- Cisco, VMware 등

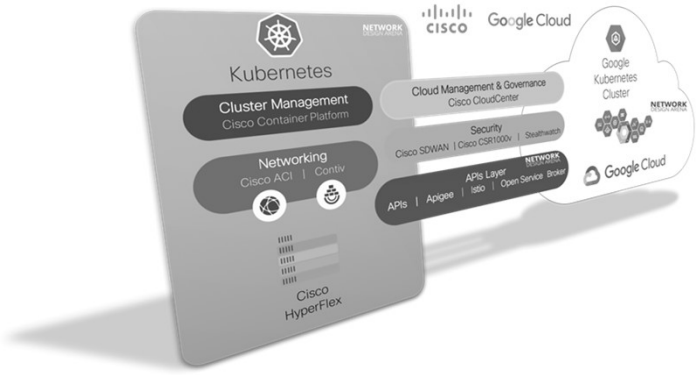


https://thedataplumber.net/nsx-t-vs-nsx-v-and-a-little-bit-of-nuage-vsp/ https://www.routetocloud.com/category/nsx-t/

I. 오픈소스 네트워킹 개요

❖ 클라우드 서비스 / 제조사 협력 솔루션

- 서비스/제조사간 협력 또는 인수를 통한 솔루션 확대
- 네트워킹 오픈소스 활용



http://www.netdesignarena.com/index.php/2018/05/30/why-cisco-multi-cloud-a-cloud-architect-perspective/

I. 오픈소스 네트워킹 개요

❖ Telco News @ Market

Comcast Deploys Open Source Trellis in 'Multiple Markets'
Jessica Lyons Hardcastle | Managing EditorSeptember 14, 2019 1:30

Comcast today said it deployed Trellis, the Open Networking Foundation's (ONF) open source SDN fabric, in "multiple markets."

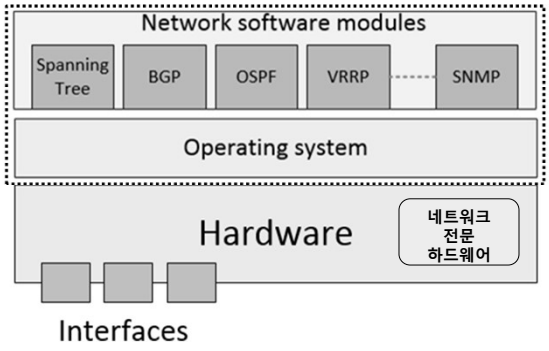
Trellis is an SDN-based, multi-purpose spine-leaf switching fabric designed for access-and-edge networks, NFV, and edge cloud applications. It use the Open Network Operating System (ONOS) open source SDN controller, the OpenFlow protocol, and white box switches.

https://www.opennetworking.org/

I. 오픈소스 네트워킹 개요

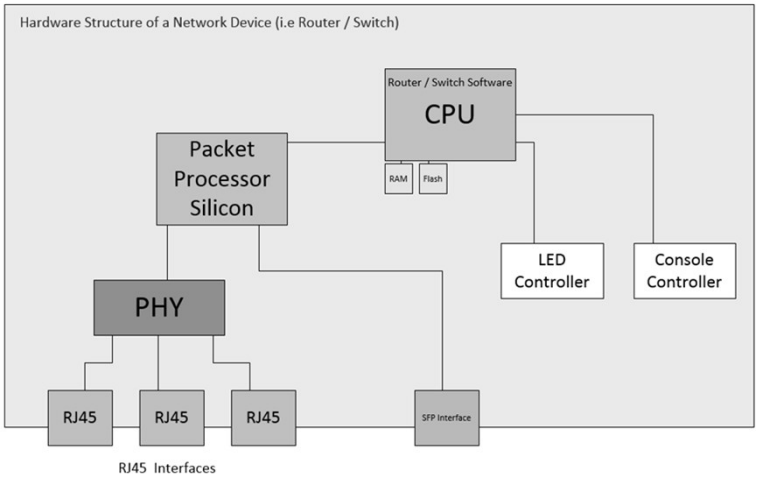
❖ **Hierarchy of a Network Device**

- **Packet processor and forwarding**
- **CPU, along with RAM and flash**
- **Other controllers, such as a fan controller, console port, LED controller, interface controllers, etc.**
- **Software.**



I. 오픈소스 네트워킹 개요

❖ **Legacy Networks**



I. 오픈소스 네트워킹 개요

❖ 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

Opensource Networking

james@jslab.kr

II. 오픈소스와 SDN Landscape

- Disaggregated Hardware Layer
- IO 추상화와 Datapath Layer
- Network Operating Systems
- Network Control Layer
- Network Virtualization
- Cloud and Virtual Management Layer
- Orchestration, Management, Policy Layer
- Network Data Analytics

JS Lab

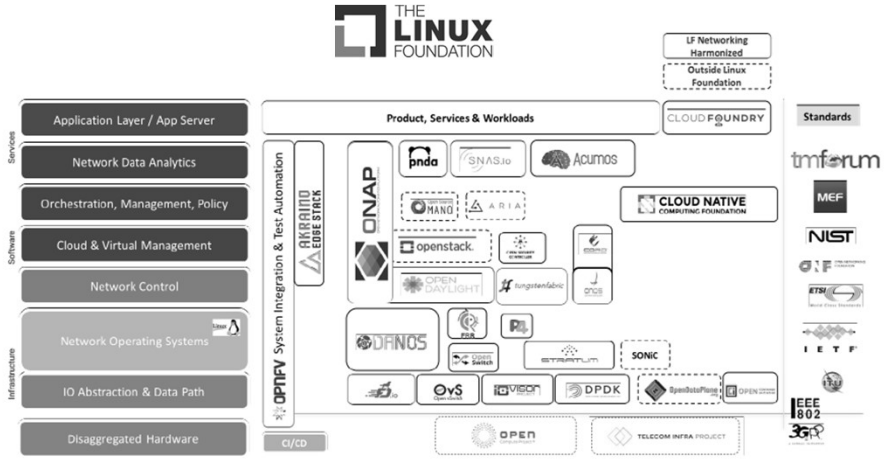
17

Opensource Networking

james@jslab.kr

II. 오픈소스와 SDN Landscape

❖ Open Source and Software Defined Networking



JS Lab

https://www.linuxfoundation.org/ https://www.nist.gov/

18

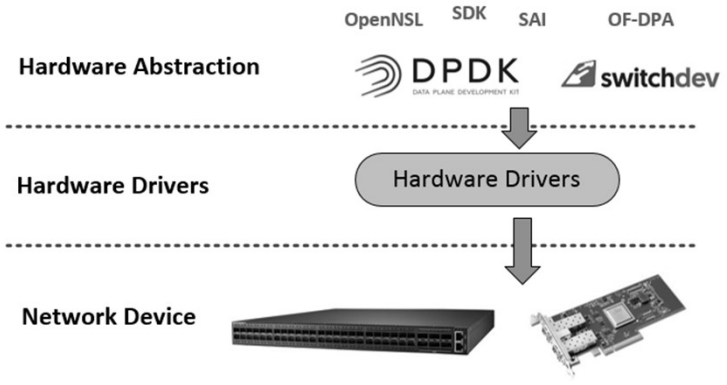
II. 오픈소스와 SDN Landscape

- ❖ **Open Source and Software Defined Networking**
- ❖ **Open Networking - 계층구조** (리눅스재단의 오픈소스네트워킹)
  - **Disaggregated Hardware**
  - **IO Abstraction and Datapath**
  - **Network Operating Systems**
  - **Network Control**
  - **Network Virtualization**
  - **Cloud and Virtual Management**
  - **Orchestration, Management, Policy**
  - **Network Data Analytics**
  - **Application Layer.**

Opensource Networking
james@jslab.kr

II. 오픈소스와 SDN Landscape

- ❖ **IO Abstraction and Datapath Layer**
- ❖ **Hardware Abstraction in an Open Network Device**



Opensource Networking
james@jslab.kr

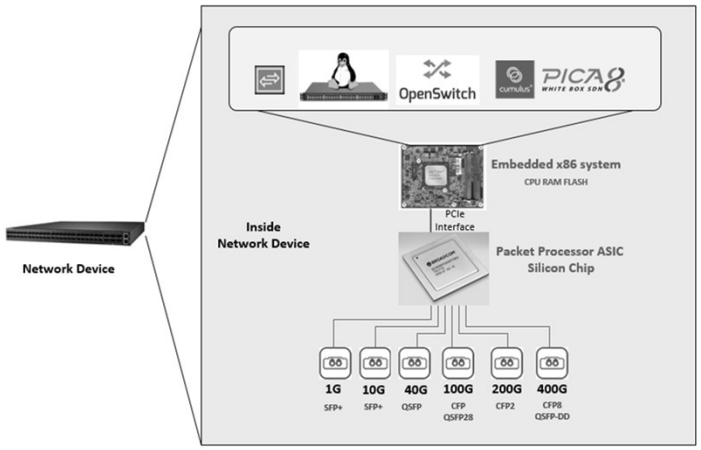
II. 오픈소스와 SDN Landscape

❖ **Disaggregated Hardware Layer**

기능 (Function)	리눅스 탑재 x86 서버 (An x86 server running Linux)	이더넷 스위치/라우터 (Dedicated Ethernet switch or router)
패킷처리 (Packet Processing)	Packets are sent to CPU and OS to make forwarding, routing or firewalling decisions	Packets are processed in packet processor ASIC (Application-Specific Integrated Circuit) silicon, not the CPU
처리량 (Throughput)	Limited to server hardware, such as CPU, I/O bus, kernel. Normally, limited to Gbps	Depends on the ASIC model. Varies from Gbps to tens of Tbps
부팅 (Boot process)	System boots as a normal PC, loads the Linux kernel, OS and networking software (for example, iptables, etc.)	A tiny OS runs on CPU and starts driving the ASIC
포트 수 (Port density)	Limited to the number of ports on the server, or additional ports via a PCIe (Peripheral Component Interconnect Express) card	Ethernet switches can support 48 or 52 ports in 1U form factor

II. 오픈소스와 SDN Landscape

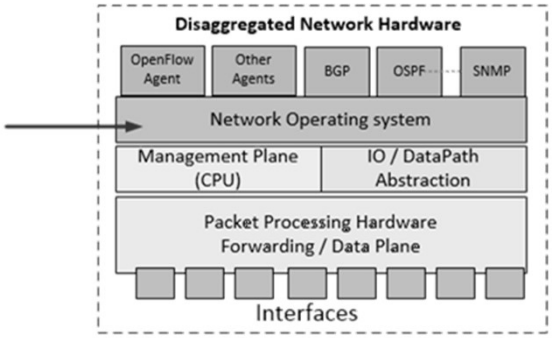
❖ **Disaggregated Network Device inside an Open Hardware**



II. 오픈소스와 SDN Landscape

❖ **Network Operating Systems**

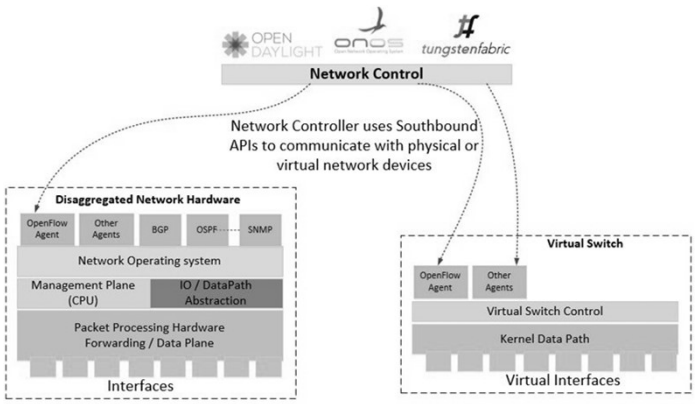
- **A network operating system is an operating system that runs on the management plane of a network device. This operating system is designed to drive the packet processor hardware chipset, such as a switch silicon, and perform the tasks required for forwarding, routing, and switching.**



II. 오픈소스와 SDN Landscape

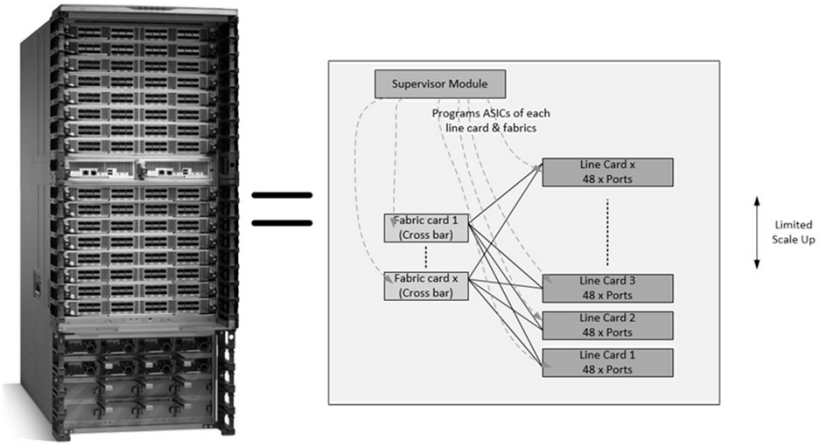
❖ **Network Control Layer**

- **The Network Control layer is about SDN controllers that can manage multiple network operating systems via agents and protocols (aka Southbound APIs).**



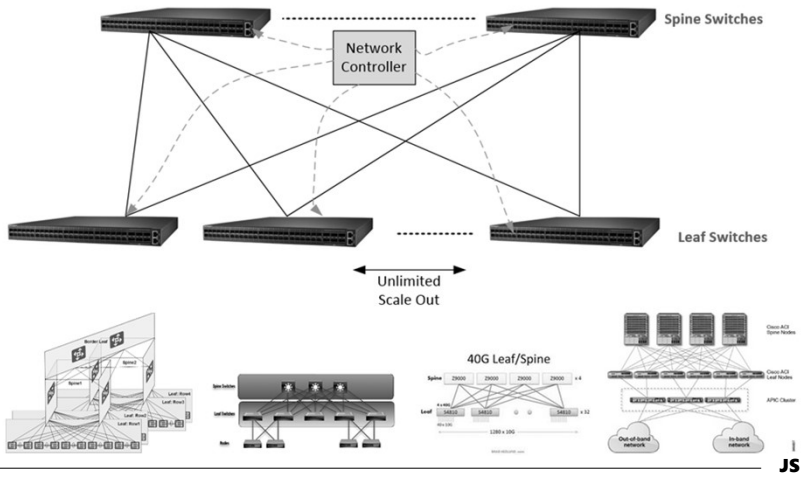
II. 오픈소스와 SDN Landscape

- Network Control Layer
- A Chassis-Based Switch



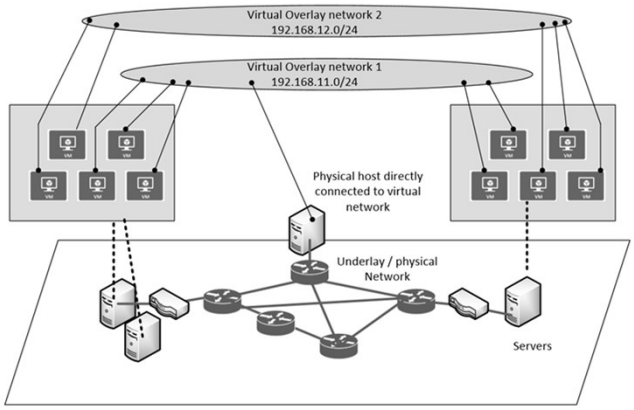
II. 오픈소스와 SDN Landscape

- Network Control Layer
- An SDN Network Can Scale Out



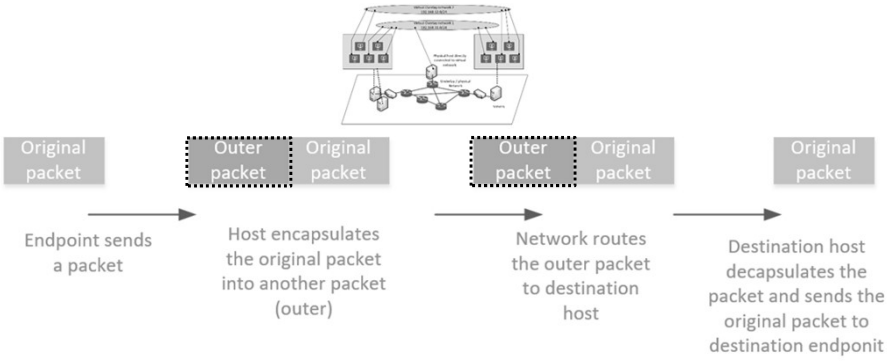
II. 오픈소스와 SDN Landscape

- **Network Virtualization**
- **Overlay Networks Are Virtual Networks on Top of Physical Networks, Built Using Encapsulation and Tunnels**



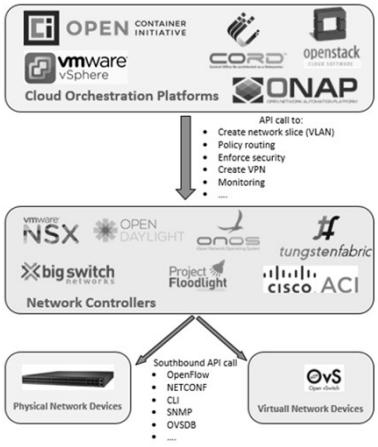
II. 오픈소스와 SDN Landscape

- **Packet Transfer Steps in an Overlay Network**



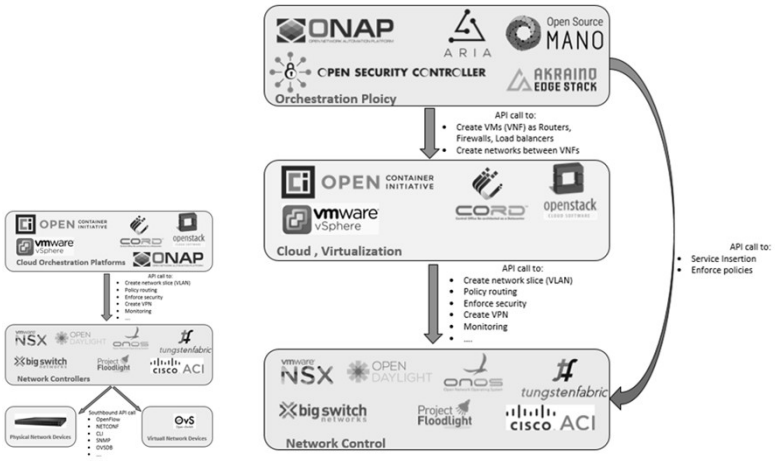
II. 오픈소스와 SDN Landscape

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



II. 오픈소스와 SDN Landscape

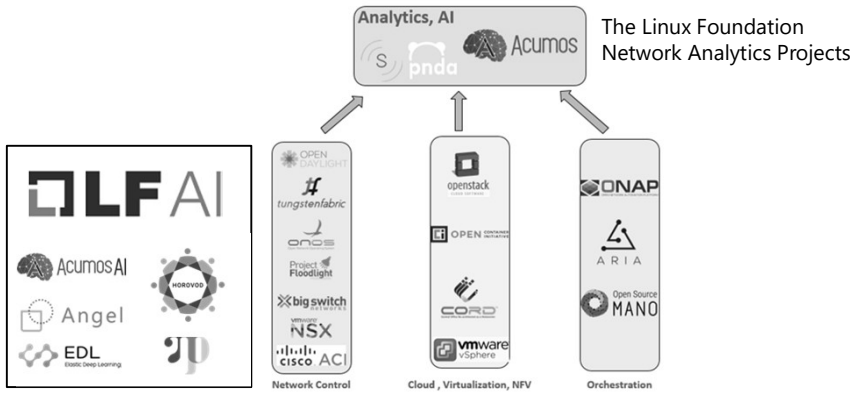
- ❖ Orchestration, Management, Policy Layer
- ❖ Orchestration Platforms Communicate with Network Controller and Cloud Management Layers



II. 오픈소스와 SDN Landscape

❖ **Network Data Analytics**

- **SNAS (Streaming Network Analytics System, formerly OpenBMP)**
- **PNDA (Platform for Network Data Analytics)**
- **Acumos AI**



https://lfai.foundation/

JS Lab

JS Lab

III. 소프드웨어/하드웨어 분리

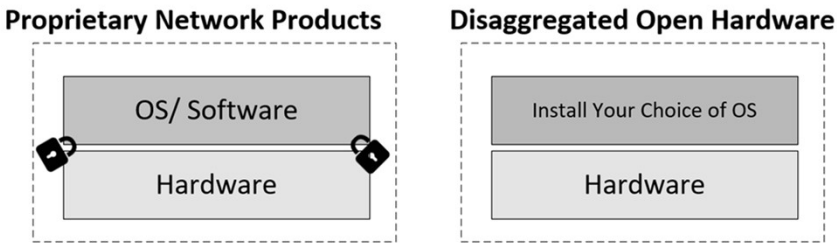
- Proprietary Network Products vs. Disaggregated Open Hardware
- Open Compute Project
- OCP Projects
- Telecom Infra Project
- How Ethernet Switches Are Built
- Types of Switches
- Bare Metal Ethernet Switches
- White Box Ethernet Switch Hardware
- Bare Metal Wireless Access Points

Opensource Networking
james@jslab.kr

III. 소프드웨어/하드웨어 분리

❖ Proprietary Network Products vs. Disaggregated Open Hardware

❖ Commercial Networking Products vs Open Source Hardware



Opensource Networking
james@jslab.kr

III. 소프드웨어/하드웨어 분리

❖ **Differentiate between proprietary devices**

❖ **Disaggregated devices**

Proprietary Network Device	Details
Cisco Catalyst 3750 Switch	A combination of hardware and Cisco IOS software
Juniper M10iRouter	A combination of hardware and Juniper Junos OS
Arista 7170 Switch	A combination of hardware and Arista EOS software

Disaggregated Network Device	Details
Edge-Core AS5712 (48 x 10G switch)	Comes with no software/OS. You can check compatibility and install OpenSwitch/Open Network Linux (ONL)/Cumulus Linux/Pica8/Big Switch, etc.
Mellanox SN2700	Comes with no software/OS. You can check compatibility and install Cumulus Linux.
Alpha Networks SNX-60x0-486F (48-port 10G SFP)	Comes with no software/OS. You can check compatibility and install ONL/OpenSwitch/Cumulus Linux, etc.
Inventec DCS7032Q28 32 x 100GB	Comes with no software/OS. You can check compatibility and install ONL/OpenSwitch/Cumulus Linux, etc.

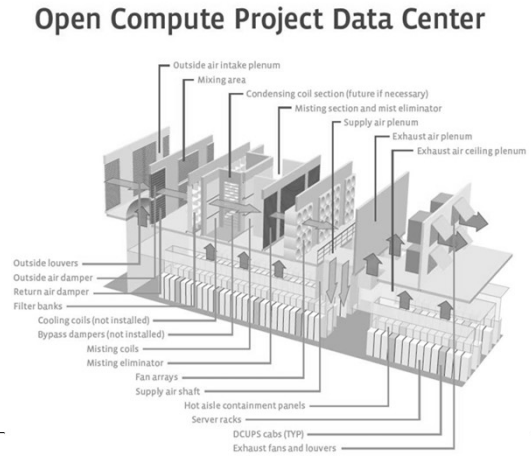
III. 소프드웨어/하드웨어 분리

❖ **Open Compute Project**

- **The Open Compute Project (OCP) was announced by Facebook, along with Intel, RackSpace, Goldman Sachs, and Andy Bechtolsheim, in April 2011. The effort was the result of a redesign of Facebook's data center in Prineville, Oregon. The aim of OCP is to create open source standards for high density and highly-efficient IT equipment for data centers, including server, storage, network, and security.**
- **OCP publishes the open hardware specifications for the data center and enterprise IT systems. There are multiple project workgroups within OCP, each including a project charter and a team working towards producing and enhancing the open source technologies within that project.**

III. 소프트웨어/하드웨어 분리

❖ **All Facebook data centers are 100% OCP**



Facebook OCP Prineville DataCenter
PUE = 1.06

Typical DataCenter
PUE > 1.4

37

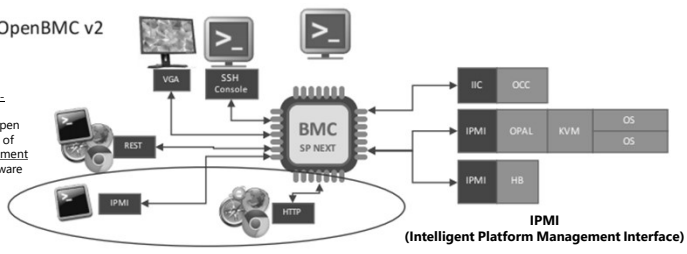
III. 소프트웨어/하드웨어 분리

❖ **OCP Projects**

- **Data Center Facility Power**
- **Data Center Facility Cooling**
- **IT Space Layout and Design**
- **Data Center Facility Monitoring and Control**
- **Data Center Facilities Operation.**

The OpenBMC:

- Linux Foundation open-source project
- goal is to produce an open source implementation of the Baseboard Management Controllers (BMC) Firmware Stack.



IPMI
(Intelligent Platform Management Interface)

38

III. 소프드웨어/하드웨어 분리

❖ **Telecom Infra Project @ OCP**

- **Backhaul Projects**
- **Access Projects**
- **Core & Management Projects**

39

III. 소프드웨어/하드웨어 분리

❖ **OCP Rack and Servers and Storage System**

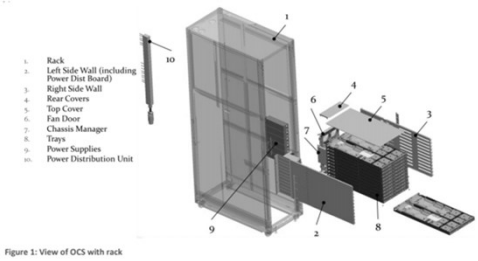
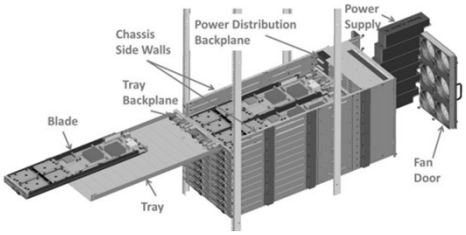
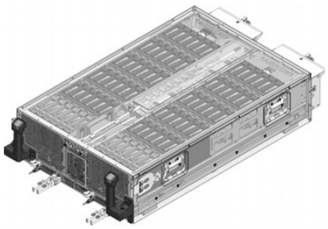


Figure 1: View of OCS with rack





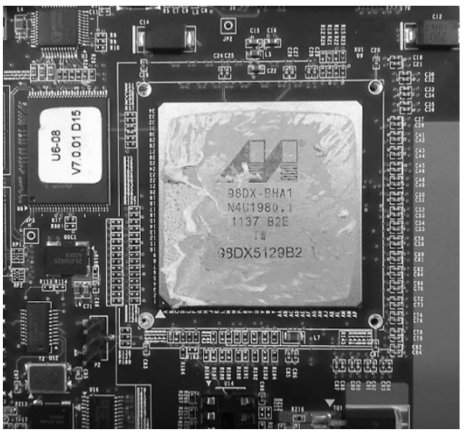
http://files.opencompute.org/oc/public.php?service=files&t=62279808dbea0cf380632c3042246979&path=/V2

40

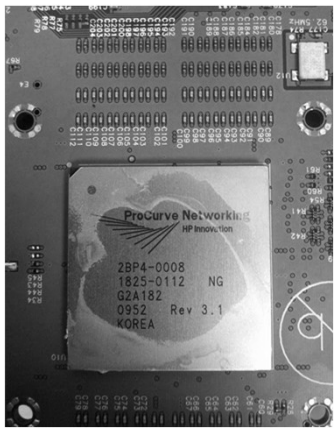
III. 소프트웨어/하드웨어 분리

- **A Merchant Silicon Chipset**
- **Non-merchant silicones**

A Merchant Silicon Chipset from Marvel



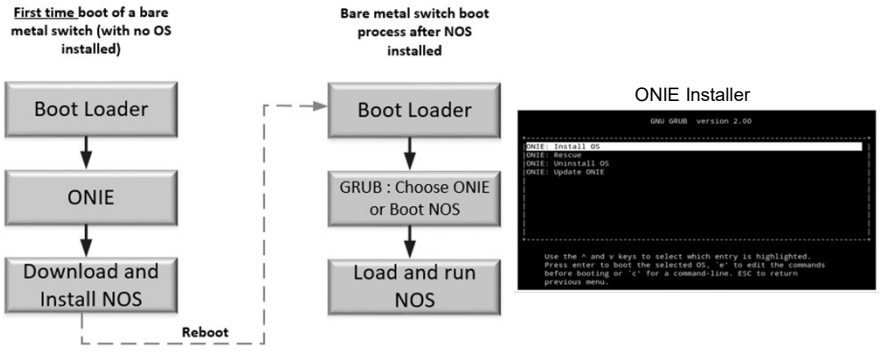
A Non-Merchant Silicon from HP



41

III. 소프트웨어/하드웨어 분리

- **The Open Network Install Environment (ONIE)**
- **Bare Metal Switch to Boot Up Process and Execution of ONIE**



42

III. 소프드웨어/하드웨어 분리

❖ How Ethernet Switches Are Built

Chip Manufacturer	Chipset
Broadcom Inc.	• Strata SGX Family: ✓ Helix (1G) ✓ Trident 2, 2+, 3 (10G/40G) ✓ Tomahawk 2, 3 (100G/200G/400G) • Strata DNX Family (Large buffer): ✓ Qumran ✓ Jericho
Mellanox Technologies	Spectrum, Spectrum 2 (10G/40G/100G/200G/400G)
Cavium	XPliant (1G/10G/40G/100G)
Barefoot Networks	Tofino (10G/40G/100G)
Marvell Technology Group	Prestera switching family
Microsemi (Vitesse)	Gigabit switch chipsets
Intel Corporation	FM6000 series (10G/40G)

JS Lab

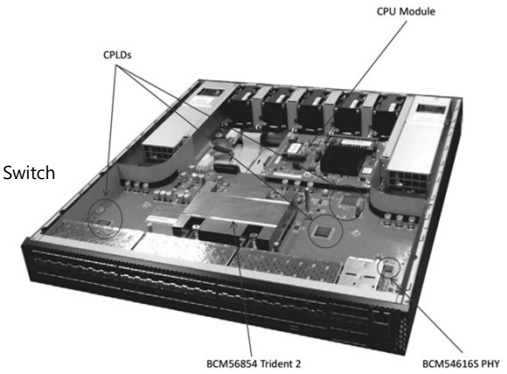
https://www.intel.com/content/www/us/en/ethernet-products/switch-silicon/ethernet-switch-fm5000-fm6000-datasheet.html

III. 소프드웨어/하드웨어 분리

❖ How Ethernet Switches Are Built

- **Chassis**
- **Power supplies**
- **Fans**
- **Control System**
- **CPU PCBA**
- **Switch main board PCBA**

An Edge-Core AS5712 48 x 10G, 6 x 40G Switch

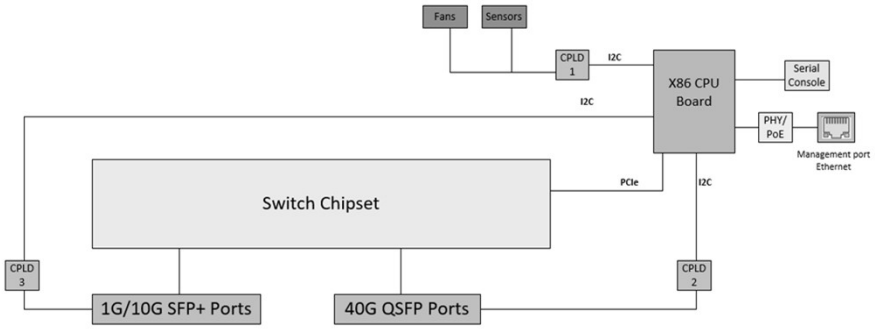


JS Lab

http://files.opencompute.org/oc/public.php?service=files&t=b96cbe20907b1a99edbecbeac3e92c4d

III. 소프드웨어/하드웨어 분리

❖ **Ethernet Switches**



http://files.opencompute.org/oc/public.php?service=files&t=b96cbe20907b1a99edbecbeac3e92c4d

III. 소프드웨어/하드웨어 분리

❖ **Ethernet Switches**

Type	You can install any NOS	What is included with purchase	Hardware Support	NOS	NOS Support
Bare metal switch	Yes	Switch hardware only	Hardware manufacturer	Purchased separately	NOS vendor
White-box switch	Yes	Switch hardware only	Hardware manufacturer	Purchased separately	NOS vendor
Brite-box switch	Not supported by its vendor	Switch hardware and a NOS	Company selling the brite-box switch	Already included	Company selling the brite-box switch

https://www.dell.com/ae/business/p/open-networking-switches/pd

III. 소프드웨어/하드웨어 분리

❖ **Edge-Core Switches**

Switch Model	Main Ports	Switch Chipset
AS4610	48 x 1G RJ45	Broadcom Helix 4
AS5712	48 x 10G	Broadcom Trident 2
AS5812	48 x 10G	Broadcom Trident 2+
AS5912	48 x 10G	Broadcom Qumran-MX
AS6712	32 x 40G	Broadcom Trident 2
AS6812	32 x 40G	Broadcom Trident 2+
AS7816	64 x 100G	Broadcom Tomahawk 2
AS7712	32 x 100G	Broadcom Tomahawk
AS7512	32 x 100G	Cavium Xpliant
AS7900	32 x 400G	Broadcom Tomahawk 3

http://files.opencompute.org/oc/public.php?service=files&t=b96cbe20907b1a99edbecbeac3e92c4d

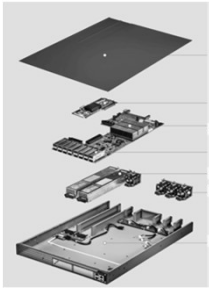
47

III. 소프드웨어/하드웨어 분리

❖ **Facebook Switches:**

Switch Model	Main Ports	Switch Chipset
Wedge	16 x 40G	Broadcom Trident 2
Wedge 100	32 x 100G	Broadcom Tomahawk
Backpack (Chassis-based)	128 x 100G	Broadcom Tomahawk
Wedge 100C	32 x 100G	Cavium Xpliant
Wedge 100B	32 x 100G / 65 x 100G	Barefoot Tofino T10

Wedge System Building Blocks



http://files.opencompute.org/oc/public.php?service=files&t=80dde82c251e1c3f5ad8ff7fd2675061

Facebook Wedge 100B based on Barefoot Networks Tofino Chip

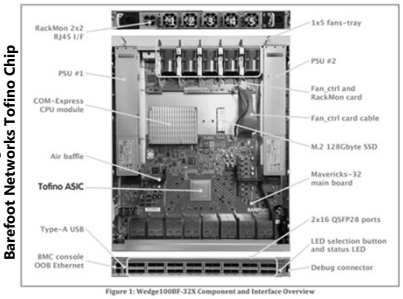


Figure 1: Wedge100BF-32X Component and Interface Overview

http://files.opencompute.org/oc/public.php?service=files&t=226d6609f6a58925148ed01289a8c517

48

III. 소프드웨어/하드웨어 분리

- Edgecore Networks Wedge100-32X 100GbE
- Facebook - Wedge-100 Switch (19-in vs 21-in)

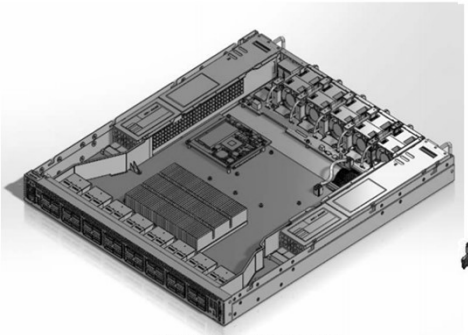


Figure 2: ISO view of Standard 19-in SKU

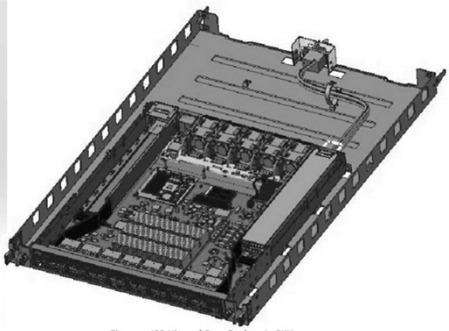
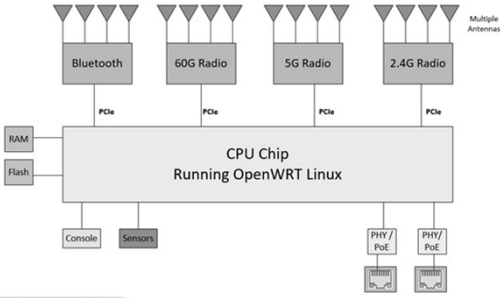


Figure 4: ISO View of OpenRack 21-in SKU

III. 소프드웨어/하드웨어 분리

- Bare Metal Wireless Access Points



Top View of ECW7212-L

Front View of ECW7212-L

http://files.opencompute.org/oc/public.php?service=files&t=cc7242c818159bfa2c1d69825eb6bb1b

III. 소프드웨어/하드웨어 분리

❖ **Mellanox Bare Metal Switches:**

Switch Model	Main Ports	Switch Chipset
MSX1410	48 x 10G / 12 x 40G	Mellanox Switch-X2
MSX1710	48 x 10G / 36 x 40G	Mellanox Switch-X2
SN 270	48 x 10G / 32 x 100G	Mellanox Spectrum

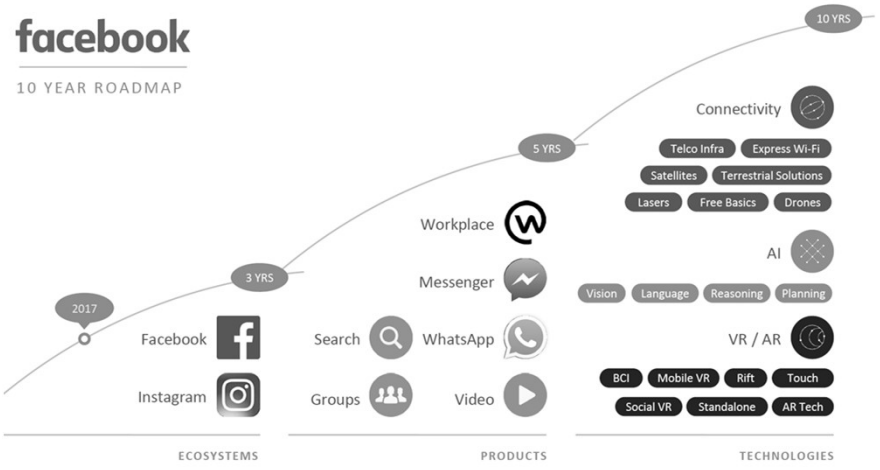
❖ **Barefoot Tofino-Based Switches:**

Switch Model	Main Ports	Switch Chipset
Wedge 100B	32 x 100G / 65 x 100G	Barefoot Tofino T10

http://files.opencompute.org/oc/public.php?service=files&t=80dde82c251e1c3f5ad8ff7fd2675061

III. 소프드웨어/하드웨어 분리

❖ **페이스북 로드맵:**



I. 오픈소스 네트워킹 개요
II. 오픈소스와 SDN Landscape
III. 소프드웨어/하드웨어 분리
IV. IO 추상화와 Data Path
V. **NOS** (Network Operating systems)
VI. 네트워크 제어 (Network Control)
VII. 클라우드와 가상화 관리
VIII. 네트워크 가상화
IX. **NFV** (Network Function Virtualization)
X. 네트워크 자동화
XI. 네트워크 데이터 분석
XII. Use Case
❖ 실습교재 (별도)

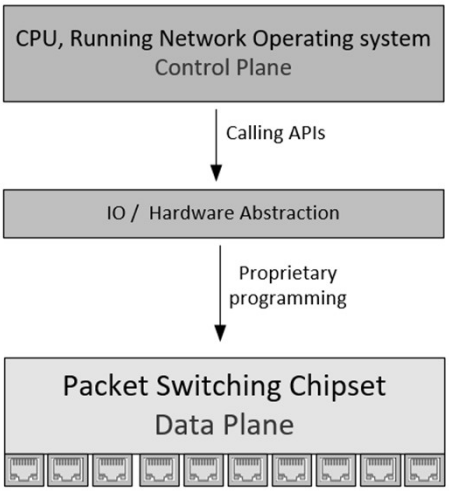
IV. IO 추상화와 Datapath

- **Types of Planes in a Network Device**
- **DPDK** (Data Plane Development Kit)
- **FD.io** (The Fast Data Project)
- **IO Visor Project**
- **Open vSwitch** (OVS)
- **OpenDataPlane** (ODP)
- **Open Container Initiative** (OCI)
- **Open Container Initiative and Open Virtualization Format**
- **SmartNICs**
- **Working with FPGAs**
- **Barefoot Networks Tofino Programmable Switch Silicon**

https://www.linuxfoundation.org/projects/networking/

IV. IO 추상화와 Datapath

❖ **Types of Planes in a Network Device:**



IV. IO 추상화와 Datapath

❖ **Types of Planes in a Network Device (Continued):**

Task	API call from Control Plane to hardware abstraction	Call from hardware abstracti on to chipset
Create new VLAN ID 500	`vlan_id=500;` `Create_vlan(vlan_id);`	0x8721; 0x8734; 0x876772829283;

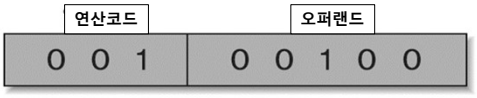
Add port eth1, eth2 to vlan 500	`vlan_id=500;`
	`port=1;`
	`vlan_add_port(vlan_id,port)`
	`port=2;`
	`vlan_add_port(vlan_id,port)`
		0x8722;
		0x8973;
		0x2389202;
		0x8722;
		0x8973;
		0x2389201;

IV. IO 추상화와 Datapath

❖ 기계명령어 형식 - 연산 코드(Opcode) / 오퍼랜드(Operand)

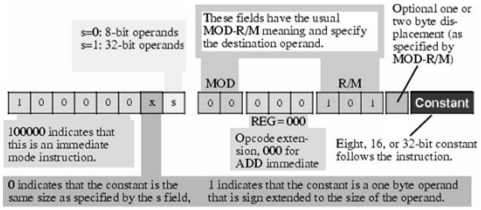
- Opcode: CPU가 수행할 연산을 지정해 주는 비트들 (Operation code)
- Operand: 데이터가 저장된 기억장치 주소 혹은 연산에 사용될 데이터 비트

8bit 기계어(예)



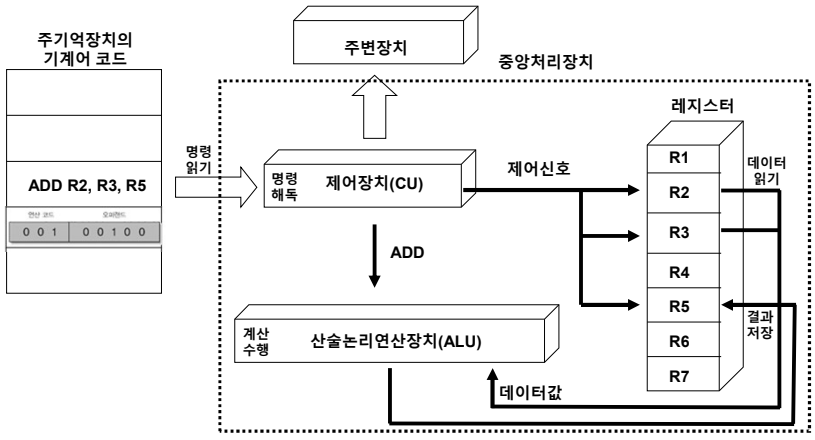
기계 명령어 형식

x86 Instructions



IV. IO 추상화와 Datapath

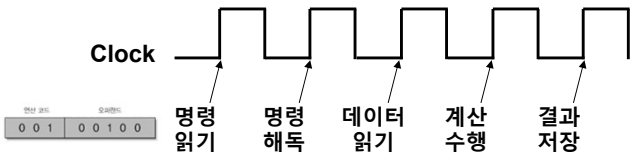
❖ 프로그램에 의한 중앙 처리 장치 동작 과정



IV. IO 추상화와 Datapath

❖ 중앙 처리 장치

- 클럭 (Clock)
- 컴퓨터 동작을 위한 진동
- 중앙 처리 장치가 작업을 수행하는 단위
- 같은 종류의 CPU라면 초당 클럭 수가 많을 수록 속도가 빠름
- Opcode별 필요 클럭 수는 다를 수 있음

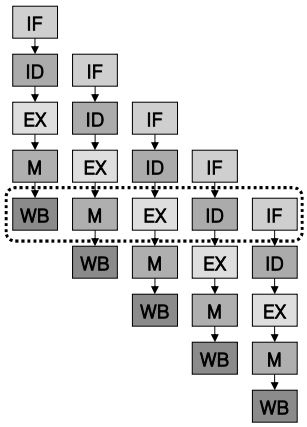


IV. IO 추상화와 Datapath

❖ **Pipeline 병렬 처리** (5 단계 예)

- **1 단계 : 명령읽기 IF(Instruction Fetch)**
  - 명령어를 메모리에서 가져옴
- **2 단계 : 명령해독 ID(Instruction Decode)**
  - 명령어를 해석
- **3 단계 : 계산수행 EX(Execution)**
  - 명령어 실행
- **4 단계 : 데이터 읽기 M(Memory access)**
  - 읽거나 쓸 메모리 특정 위치에 접근
- **5 단계 : 결과저장 WB(Write Back)**
  - 레지스터에 다시 씀

병렬로 명령어 동시 수행



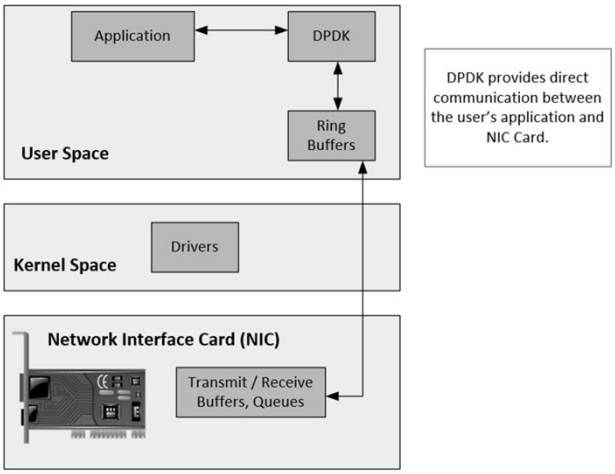
IV. IO 추상화와 Datapath

❖ **DPDK (Data Plane Development Kit):**

DPDK - Quick Summary	
Name	Data Plane Development Kit
By	The Linux Foundation
Where it runs	Linux, any x86, VM, or embedded system
What it does	Packet processing, routing, switching, encapsulation on the NIC card
Features	Environment-independent. Mostly used on appliances or x86 servers
What it can do out-of-the-box	You can build networking applications using DPDK libraries that can process packets at a high speed. DPDK APIs are very comprehensive, start from NIC functions such as bonding and network protocol parsing (Ethernet, ARP, ICMP, IPv4, IPv6, TCP, UDP, etc.), classification, QoS, ACL, etc.
Pros	DPDK is a low-level library and one of the most robust frameworks for packet processing. DPDK can achieve very high speeds in packet processing.
Caveats	Coding and using DPDK requires low-level C programming skills, requires the programmer to consider numerous parameters when using it.

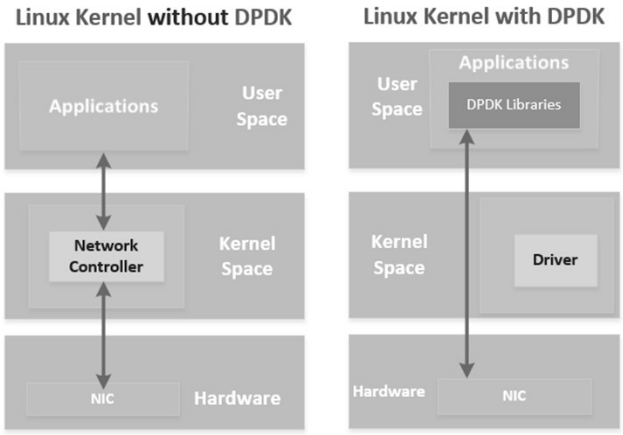
IV. IO 추상화와 Datapath

❖ **DPDK Setup and Architecture:**



IV. IO 추상화와 Datapath

❖ **DPDK Usage**



Differences between a System with and without DPDK

63

IV. IO 추상화와 Datapath

❖ **FD.io (The Fast Data Project)**

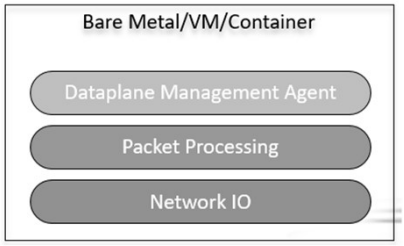
FD.io - Quick Summary	
Name	The Fast Data Project (FD.io)
By	The Linux Foundation
Where it runs	Linux, any x86, VM, or embedded system (Runs in the User Space)
What it does	Packet processing, routing, switching, NAT
Features	It uses vector packet processing (VPP) mechanisms to achieve high performance
What it can do out-of-the-box	FD.io's VPP provides a command line tool called vppctl, which can be used to interface with VPP to create interconnects, manage routing tables, create tunnel interfaces, manage hardware acceleration, etc. You can use FD.io APIs to build virtual switches, virtual routers, virtual firewalls, or other packet processing applications.

64

IV. IO 추상화와 Datapath

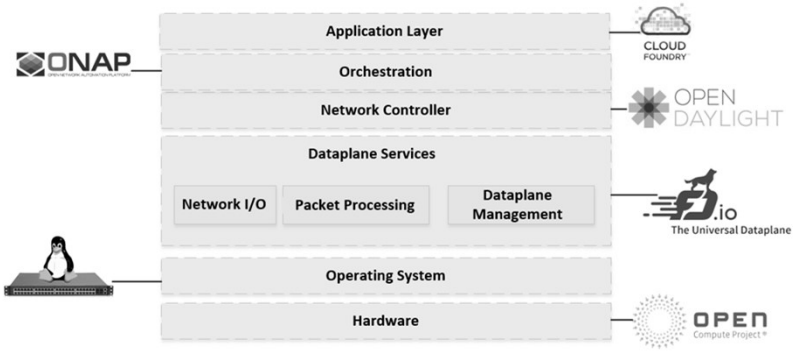
❖ **FD.io Components:**

- **Data Plane Management Agent:** An agent software that allows a Control Plane software or an SDN controller (such as OpenDaylight) to control and communicate with FD.io.
- **Packet Processing:** The packet processing engine of FD.io to classify, transform, prioritize, forward, terminate packets.
- **Network IO:** The hardware acceleration driver, connecting FD.io with the network hardware (for example, DPDK).



IV. IO 추상화와 Datapath

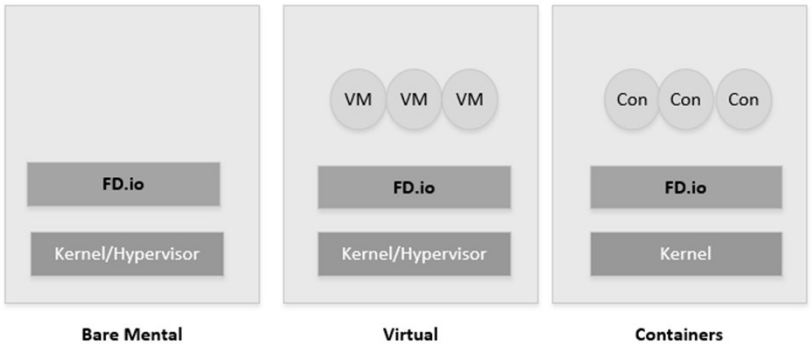
❖ **FD.io Communication with Other Networking Subsystems:**



IV. IO 추상화와 Datapath

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

- **Bare metal servers directly**
- **Virtual machines**
- **Containers**



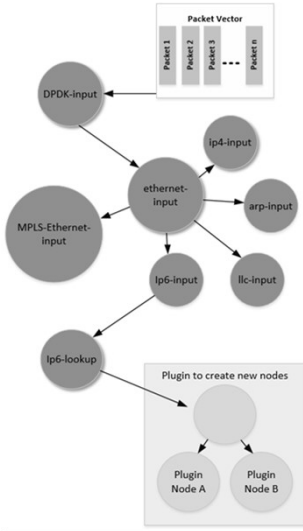
IV. IO 추상화와 Datapath

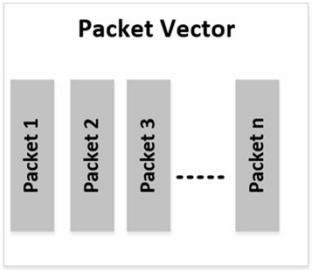
❖ **FD.io - Vector Packet Processing:** Vector Packet Processor (VPP) is the core component of FD.io. VPP can use DPDK for network IO and hardware-accelerated packet processing. VPP is a packet processing platform that can perform the following (below is a summarized list):

- **Interfaces:** VPP supports the following interfaces to be used as input an out: DPDK, TunTap, vhost.
- **Tunnels/Encapsulation:** VPP supports the following tunneling technologies: GRE, VXLAN, IPSec, MPLS over Ethernet or GRE, deep label stacks. User space applications can directly perform such encapsulation and decapsulation without the need to use the Linux kernel network controller.
- **Routing and switching (no routing protocol):** VPP provides direct access to many routing and switching features. Networking programs such as routing protocols (BGPD, OSPFD) will be able to use VPP to modify routing tables and other tasks: IPv4/IPv6, hierarchical FIB, VRFs, multi-paths, source RPF, segment routing, VLAN support, MAC learning, inbound ACL, proxy ARP.
- **Network services and security:** VPP supports NAT and other networking features and filters which can be used by security applications (source NAT, per-interface filters, DHCP, LLDP, BFD, policer, mirror/SPAN ports, IP flow export).

IV. IO 추상화와 Datapath

❖ **FD.io - Vector Packet Processing:**





IV. IO 추상화와 Datapath

❖ **How Does FD.io Relate to DPDK?**

- **FD.io provides an abstract environment for building virtual routers, switches and packet processors. FD.io can use DPDK to communicate directly with NIC cards. DPDK provides hardware acceleration to FD.io. However, hardware acceleration and usage of DPDK is optional - an FD.io-based application can still run without DPDK.**

IV. IO 추상화와 Datapath

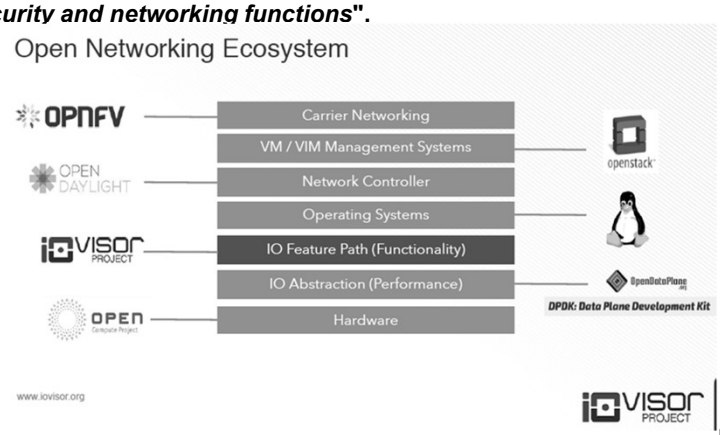
❖ **How Does FD.io Relate to IO Visor?**

- **The FD.io project and IO Visor (more about IO Visor in a little bit) are seen as complementary projects. The IO Visor Project focuses on dynamic runtime extensibility of data plane capabilities in the kernel. IO Visor aims to create a repository of IO modules that are portable across multiple possible data planes (like eBPF in the Linux kernel) and frameworks (like FD.io).**

IV. IO 추상화와 Datapath

❖ **IO Visor Project**

- "***The IO Visor Project is an open source project and a community of developers to accelerate the innovation, development, and sharing of virtualized in-kernel IO services for tracing, analytics, monitoring, security and networking functions***".



https://www.iovisor.org/

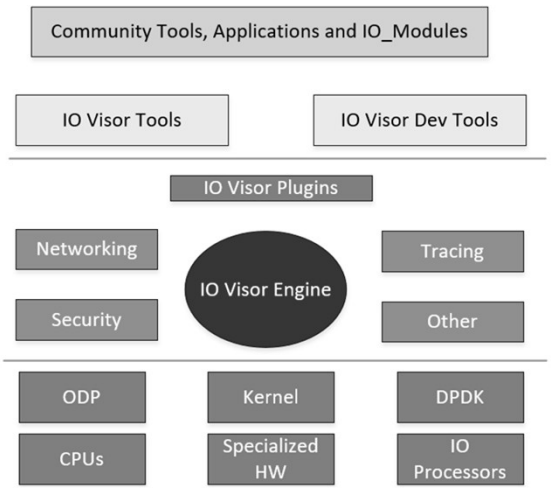
IV. IO 추상화와 Datapath

❖ **IO Visor Project**

IO Visor - Quick Summary	
Name	IO Visor Project
By	The Linux Foundation
Where it runs	Linux, any x86, VM, or embedded system (in-kernel)
What it does	Packet processing, routing, switching, NAT
Features	Supports three data path methods: XDP (eXpress Data Path), BCC (BPF Compiler Collection), and eBPF
What it can do out-of-the-box	Since IO Visor is mainly user in an in-kernel mode, it can help build robust networking applications for networking within a host, between a host and virtual machines or containers. It accelerates the in-host networking functions.

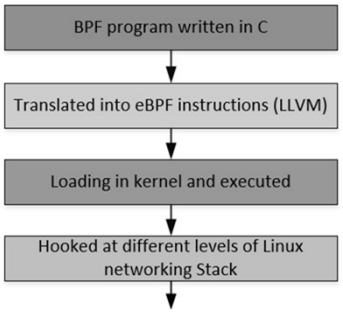
IV. IO 추상화와 Datapath

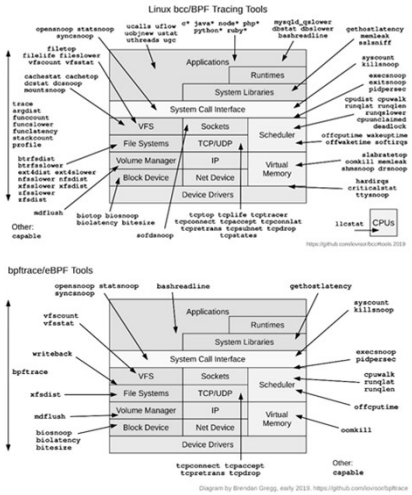
❖ **IO Visor Components**



IV. IO 추상화와 Datapath

❖ **IO Visor - eBPF**





http://www.brendangregg.com/ebpf.html

IV. IO 추상화와 Datapath

❖ **eBPF:** extended Berkeley Packet Filter. eBPF 는 x86-64 와 arm64 의 공통점을 따온 것처럼 보이는 별도의 어셈블리 언어입니다. 실제로 이런 코드를 사람이 직접 작성하지는 않고, 성능 측정을 위한 C 언어 코드를 작성하면 이를 eBPF 프로그램으로 트랜스파일

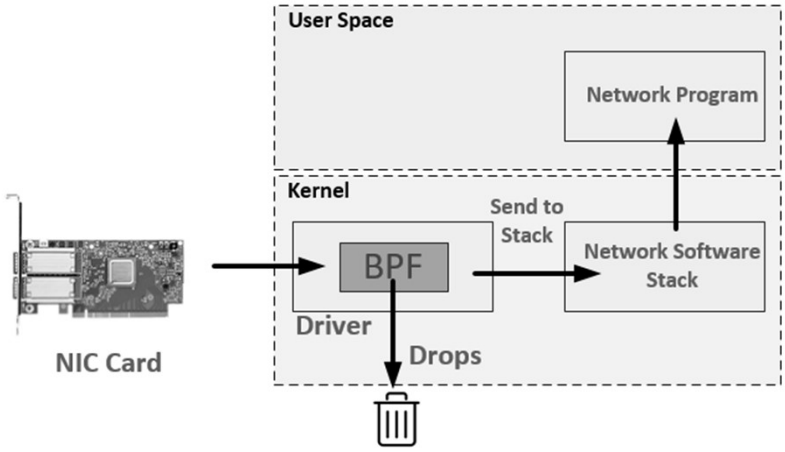
❖ **IO Visor – eBPF** (개발자의 생각 @ blog)**:** 운영 체제 수준에서 제공하는 기능인 eBPF와 이를 사용한 BCC 툴킷을 써서 실행 중인 서비스를 멈추거나 수정하지 않고도 성능을 측정하고 문제의 원인을 찾을 수 있음

- **가설 #1: 어디선가 CPU를 많이 쓴다.**
- **가설 #2: 데이터베이스가 충분히 빠른가?**
- **가설 #3: 다른 대기 코드가 있다**
- **가설 #4: DNS 조회가 오래 걸리고 있다**

https://blog.ifunfactory.com/2018/03/29/linux-%EA%B2%8C%EC%9E%84-%EC%84%9C%EB%B2%84-%EC%84%B1%EB%8A%A5-%EB%B6%84%EC%84%9D%EC%97%90-ebpf-bcc-%ED%99%9C%EC%9A%A9%ED%95%98%EA%B8%B0/?fbclid=IwAR1EaKA5gT9CbLWPBxOqSDzi9DDcEjbiFU8kO9dDfq1yycHtSa9ZSH3vWFA

IV. IO 추상화와 Datapath

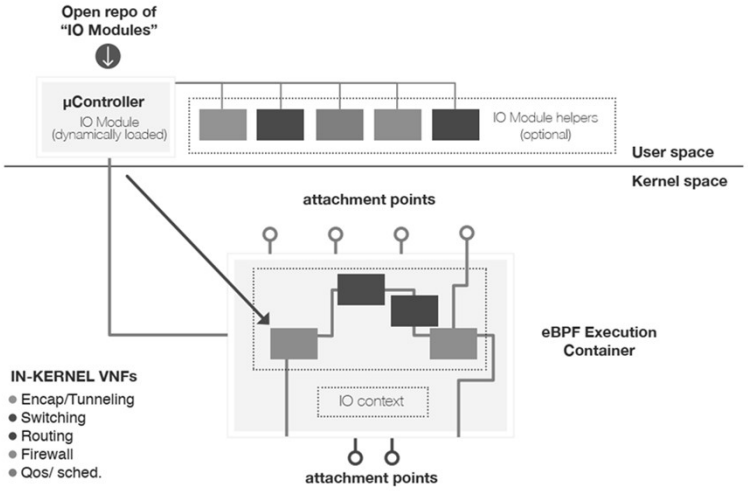
❖ **IO Visor - XDP**



77

IV. IO 추상화와 Datapath

❖ **Networking Based on IO Visor**

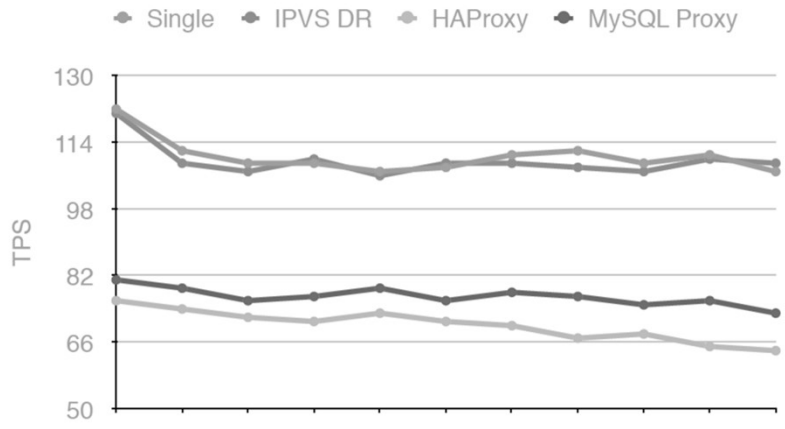


http://www.iovisor.org/wp-content/uploads/sites/8/2016/09/use_case_condensed.pdf

78

IV. IO 추상화와 Datapath

❖**IP Virtual Server** (IPVS)
❖리눅스 기반에 설치되는 서버로 L4 기능을 대체



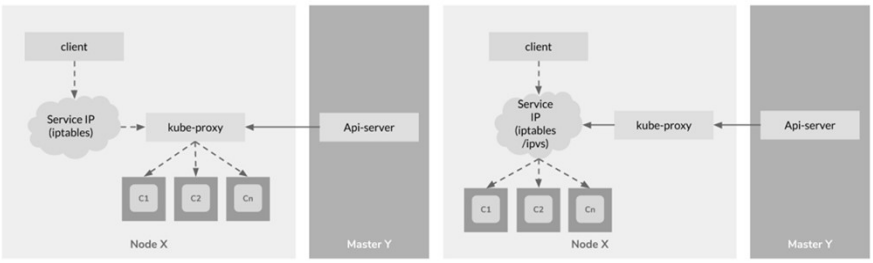
IPVS: (IP Virtual Server): 리눅스 커널에 내장한 트랜스포트 계층의 로드밸런싱

79

IV. IO 추상화와 Datapath

❖ **Kubernetes Service Proxy Modes (Userspace, iptables, & ipvs)**

- **kubectl expose deployment** <application-name> **--type=**LoadBalancer **--name=**<service-name>

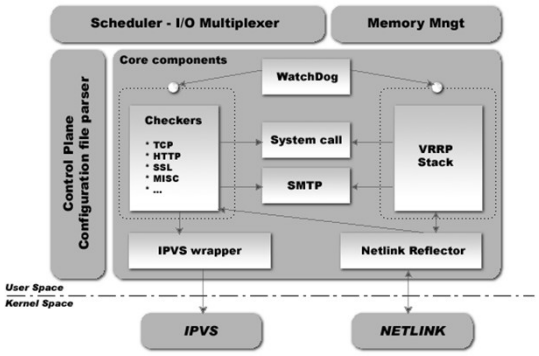


https://medium.com/containermind/a-beginners-guide-to-kubernetes-7e8ca56420b6

80

IV. IO 추상화와 Datapath

❖**IP Virtual Server** (IPVS)
❖**지원 기능**
- **HTTP_GET :** HTTP로 요청을 보내서 응답 확인
- **SSL_GET :** HTTPS로요청을 보내서 응답 확인
- **TCP_CHECK :** TCP로 접속할 수 있는지 여부를 확인
- **SMTP_CHECK :** SMTP로 HELO 명령을 보내서 응답 확인
- **MISC_CHECK :** 외부 명령을 실행해서 종료코드를 확인



http://www.softfactory.org/architecture/loadbalancing/ipvs-ip-virtual-server

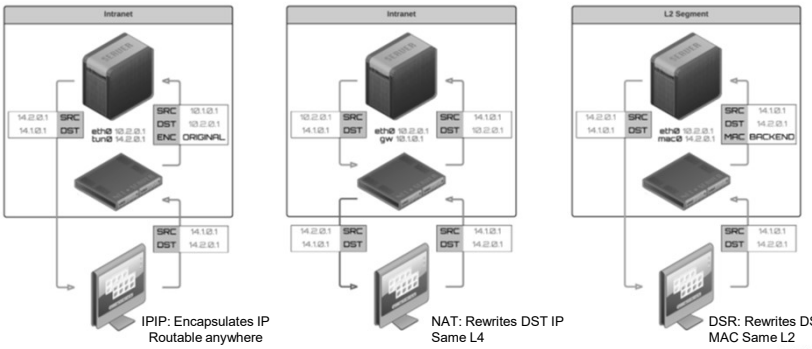
JS Lab

Opensource Networking
james@jslab.kr

81

IV. IO 추상화와 Datapath

❖**IP Virtual Server @ Docker**
- **Works inside the Linux Kernel, based onNetfilter.**
- **Supports TCP, SCTP & UDP, v4 and v6.**
- **8+ methods: WRR, WLC, LBLCR, SH and much more – plugins.**
- **NAT, Tunneling, Direct Routing.**
- **Address bundling via FWMark services.**



IPIP: Encapsulates IP
Routable anywhere

NAT: Rewrites DST IP
Same L4

DSR: Rewrites DST
MAC Same L2

Round Robin**(rr)** Weighted Round Robin**(wrr)** Least Connection**(lc)** Weighted Least Connection**(wlc)** Locality-Based Least Connection**(lblc)** Source Hashing **(sh)**

JS Lab

Opensource Networking
james@jslab.kr

82

IV. IO 추상화와 Datapath

❖ **Open vSwitch (OVS)**

OVS - Quick Summary	
Name	Open Virtual Switch (OVS)
By	The Linux Foundation
Where it runs	Linux
What it does	Virtual switch/router with physical and virtual interfaces
Features	L2 switching, VXLAN encapsulation, distributed switches controlled by a co ntroller
What it can do out-of-the-box	OVS is one of the core components of virtualization hypervisors

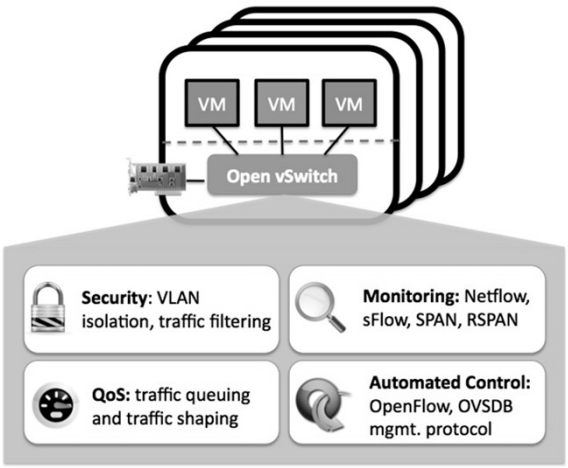
Opensource Networking james@jslab.kr

https://www.openvswitch.org/

83

IV. IO 추상화와 Datapath

❖ **Open vSwitch (OVS)**



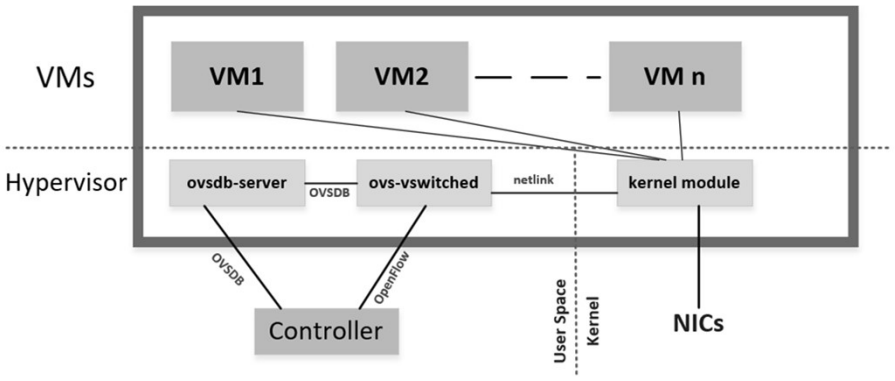
Opensource Networking james@jslab.kr

https://www.openvswitch.org/

84

IV. IO 추상화와 Datapath

❖ **Open vSwitch (OVS) Architecture**



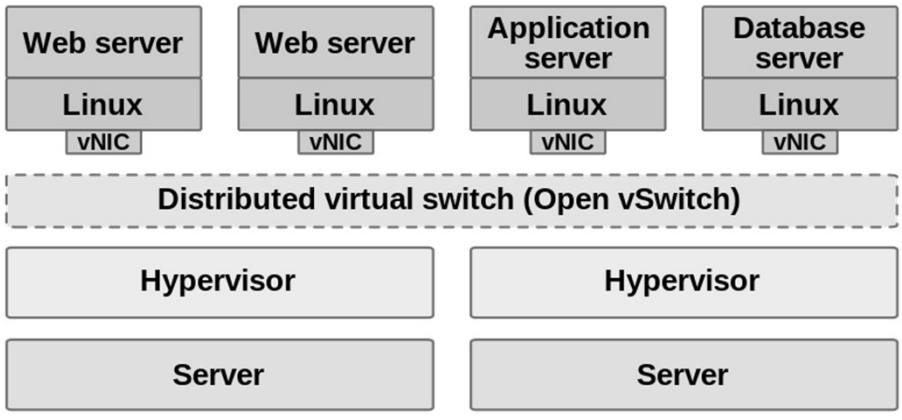
https://www.openvswitch.org/

JS Lab

IV. IO 추상화와 Datapath

❖ **Open vSwitch (OVS)**

❖ **Distributed Open vSwitch Instance**



https://www.openvswitch.org/

JS Lab

IV. IO 추상화와 Datapath

❖ **Open vSwitch (OVS)**

❖ **OVS - Supported Features**

- Visibility into inter-VM communication via NetFlow, sFlow(R), IPFIX, SPAN, RSPAN, and GRE-tunnelled mirrors
- LACP (IEEE 802.1AX-2008)
- Standard 802.1Q VLAN model with trunking
- Multicast snooping
- IETF Auto-Attach SPBM and rudimentary required LLDP support
- BFD and 802.1ag link monitoring
- STP (IEEE 802.1D-1998) and RSTP (IEEE 802.1D-2004)
- Fine-grained QoS control
- Support for HFSC qdisc
- Per VM interface traffic policing
- NIC bonding with source-MAC load balancing, active backup, and L4 hashing
- OpenFlow protocol support (including many extensions for virtualization)
- IPv6 support
- Multiple tunnelling protocols (GRE, VXLAN, STT, and Geneve, with IPsec support)
- Remote configuration protocol with C and Python bindings
- Kernel and user-space forwarding engine options
- Multi-table forwarding pipeline with flow-caching engine
- Forwarding layer abstraction to ease porting to new software and hardware platforms.

https://www.openvswitch.org/

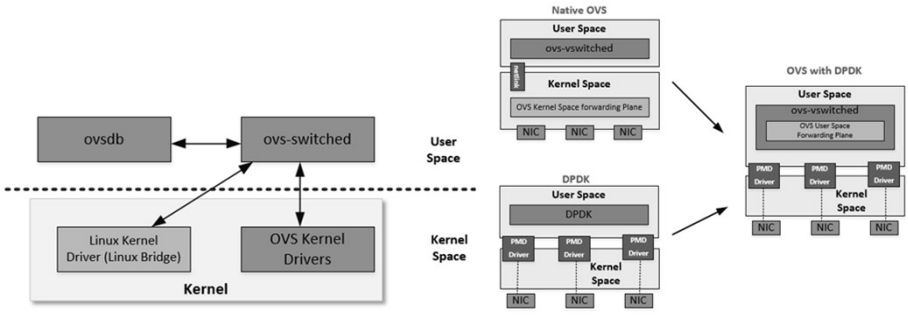
JS Lab

87

IV. IO 추상화와 Datapath

❖ **OVS vs Linux Bridge**

❖ **OVS supports advanced features and distributed environment comparing to standard Linux bridge.**



Integration of DPDK Data Plane with Open vSwitch

https://www.openvswitch.org/

JS Lab

88

IV. IO 추상화와 Datapath

❖ **OpenDataPlane (ODP)**

ODP - Quick Summary	
Name	OpenDataPlane
By	The independent open source community is mainly driven by the Linaro Network Group
Where it runs	An abstraction layer, runs on x86, ARM and other embedded SoCs
What it does	Robust network programming for embedded systems
Features	Provides networking APIs
What it can do out-of-the-box	ODP is similar to DPDK in x86, but it supports embedded systems and SoCs. ODP provides a standard network abstraction for developers to write their applications. Applications can be ported between different hardware by re-compiling them. You can implement applications such as NAT, switching, routing, classifications, IPsec encapsulation using ODP over the supported platforms.

https://www.opendataplane.org/about/

IV. IO 추상화와 Datapath

❖ **OpenDataPlane (ODP)**

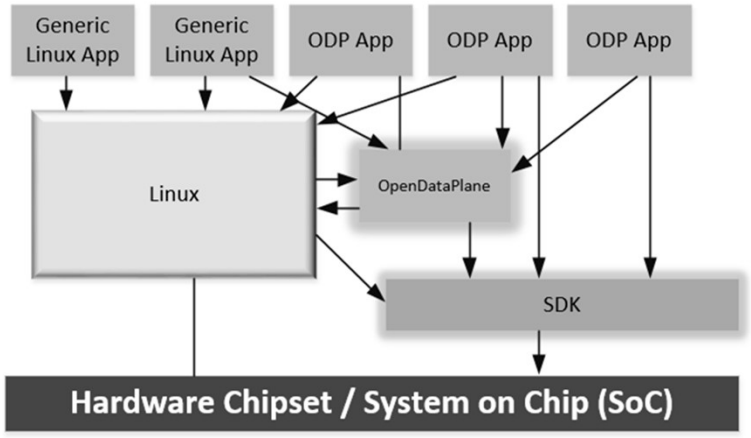
❖ **ODP is supported on the following chipsets:**

- **Cavium Octeon™ SoCs**
- **Cavium ThunderX™ SoCs**
- **Kalray MPPA (Massively Parallel Processor Array)**
- **Freescale QorIQ – ARM-based DPAA2 architecture LS2080, LS2085**
  QorIQ – ARM and PowerPC-based DPAA architecture LS1043
- **Texas Instruments (TI): Keystone II SoCs**
- **Marvell ARMADA SoC implementation**
- **Linaro ODP-DPDK - software-optimized implementation using DPDK**
- **NXP QorIQ SoCs**
- **HiSilicon platforms.**

https://www.opendataplane.org/about/

IV. IO 추상화와 Datapath

- ❖ OpenDataPlane (ODP)
- ❖ ODP and other components:



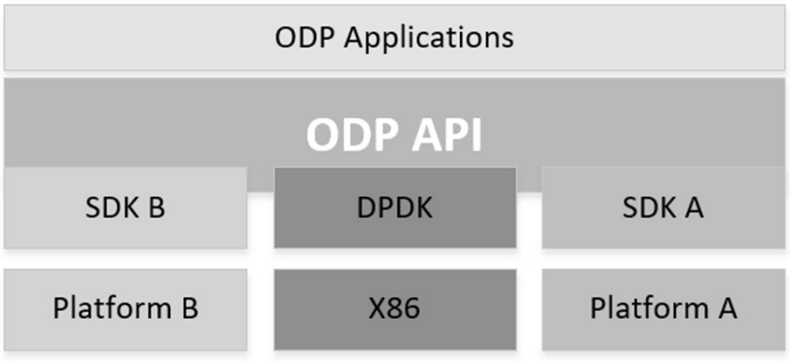
Relationship between OpenDataPlane and other Components

https://www.opendataplane.org/about/

JS Lab

IV. IO 추상화와 Datapath

- ❖ OpenDataPlane (ODP)
- ❖ ODP Implementations:

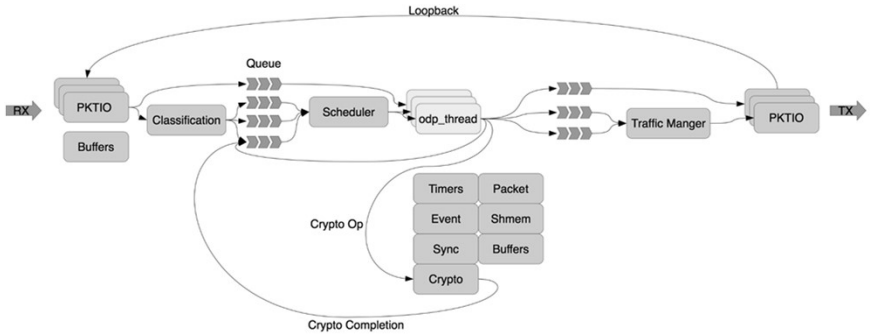


https://www.opendataplane.org/about/

JS Lab

IV. IO 추상화와 Datapath

- **OpenDataPlane (ODP)**
- **ODP Use Cases:** an example ODP flow, from receiving a packet, to processing and sending out.



https://github.com/Linaro/odp-dpdk/blob/master/doc/images/packet_flow.svg

IV. IO 추상화와 Datapath

- **Open Container Initiative (OCI)**

OCI - Quick Summary	
Name	Open Container Initiative
By	The Linux Foundation
Where it runs	Linux
What it does	Standardizes the packaging and running of containers
Features	Currently, provides specs for exporting containers and running containers
What it can do out-of-the-box	OCI includes a software called runC, which can be used to interact with containers in different container platforms, such as Docker, Kata or Linux containers.

https://www.opencontainers.org/

IV. IO 추상화와 Datapath

- Open Container Initiative (OCI)
- Two specifications defined and developed by OCI:
  - Runtime Specification (runtime-spec)
  - Image Specification (image-spec)

Opensource Networking
james@jslab.kr

IV. IO 추상화와 Datapath

- Open Container Initiative (OCI)
- OCI - Filesystem Bundle
  - The filesystem bundle specification defines a format for encoding and saving a container's set of files. This bundle includes all the files, data, and metadata required to run a container.
  - The standard container bundle contains all the files and information needed to load and run a container package. This includes a configuration file that references the locations of the container root filesystem and other files related to the container, as well as the container's main filesystem root.

Opensource Networking
james@jslab.kr

IV. IO 추상화와 Datapath

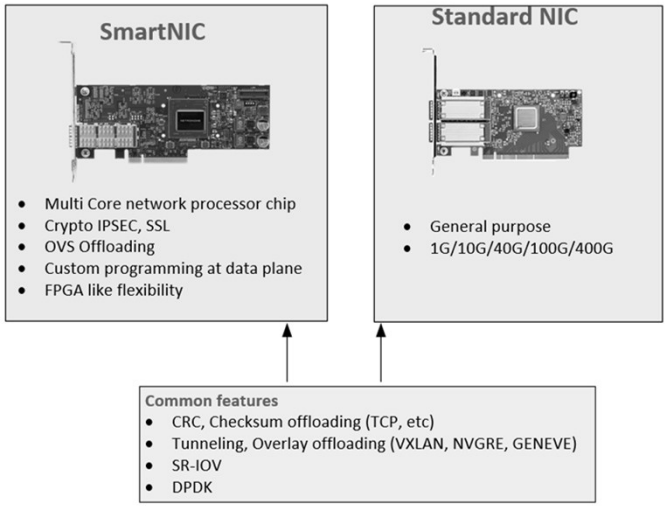
- ❖ Open Container Initiative (OCI)
- ❖ OCI - Runtime Bundle
  - The OCI runtime bundle is a standard specification for creating a bundle directory that includes all of the files required to launch an application as a container. The bundle contains an OCI configuration file where it can specify host-independent details such as which executable to launch and host-specific settings such as mount locations, hook paths, Linux namespaces and cgroups. Because the configuration includes host-specific settings, application bundle directories copied between two hosts may require configuration adjustments.

IV. IO 추상화와 Datapath

- ❖ Open Container Initiative (OCI)
- ❖ Open Container Initiative and Open Virtualization Format
  - You may find that the Open Container Initiative provides a similar function as the Open Virtualization Format (OVF) initiative did a few years ago. With OVF, you could export a virtual machine from a hypervisor (for example Xen) as an OVF file, and import the OVF file into another hypervisor, such as VMware.
    - ✓ OVF defined open standards for virtual machines and hypervisors.
    - ✓ OCI defines open standards for containers and container engines.

IV. IO 추상화와 Datapath

❖ **SmartNICs**



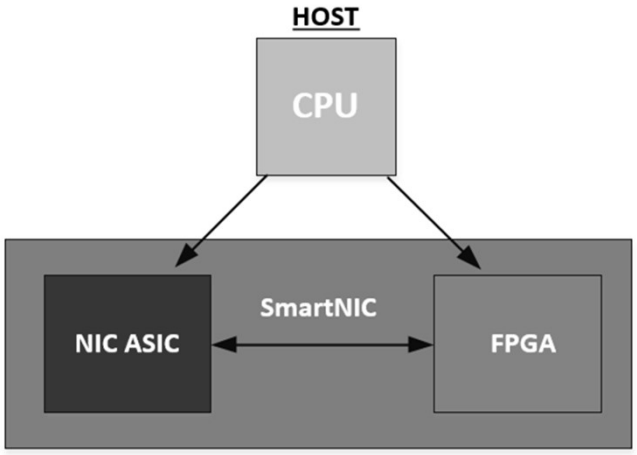
IV. IO 추상화와 Datapath

❖ **SmartNICs**

SmartNICs - Quick Summary	
Name	Smart Network Interface Cards (SmartNICs)
By	Multiple manufacturers, such as Netronome, Napatech, Mellanox Technologies, etc.
Where it runs	It's a physical PCIe card
What it does	Offloads packet processing from the host, processes packets at hi gh speed in NICs
Features	L2, L3 packet processing, custom application, etc.
What it can do o ut-of-the-box	You can build custom applications that run on SmartNIC chipsets (for example, a DDOS protector, IPS or packet encapsulators for I Psec, SSL, firewall - NGFW -, load balancing, etc.

IV. IO 추상화와 Datapath

❖ SmartNICs



https://www.opencontainers.org/

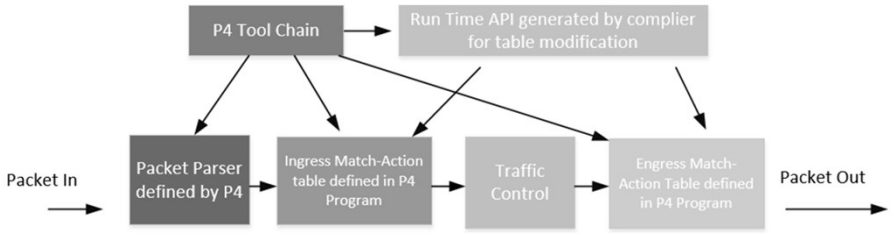
101

IV. IO 추상화와 Datapath

❖ SmartNICs

❖ SmartNICs Programmability

- SmartNICs can be programmed using their SDK or using the standard P4 language. Most of SmartNICs now support P4 programming. However, they need the vendor's compiler and tool chain in order to compile the P4 program. P4 programs are portable - you can compile the same P4 application for different SmartNIC models by compiling the P4 application using the relevant vendors' compiler.



https://www.opencontainers.org/

102

IV. IO 추상화와 Datapath

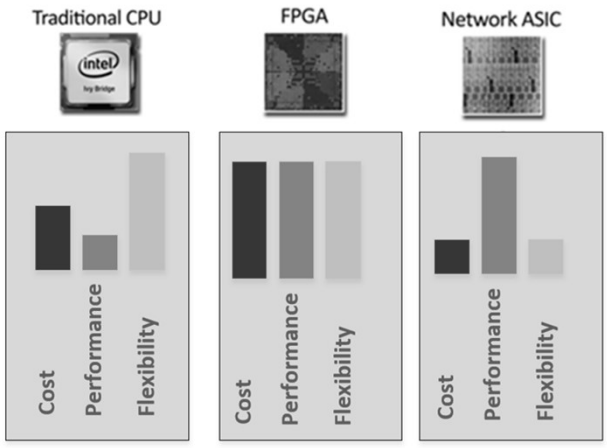
❖ **FPGAs and Xilinx SDNet**

SDNet - Quick Summary	
Name	SDNet
By	Xilinx Inc.
Where it runs	On Xilinx FPGA
What it does	SDNet is a library used to build networking applications in Xilinx FPGA to implement fast packet processing in FPGA at different speeds, such as 1Gbps, 10Gbps, 100Gbps, etc.
Features	Any packet processing function, such as switching, routing, classification, packet manipulation, ACL, etc.
What it can do out-of-the-box	You can use SDNet (which now supports P4 languages) to build your packet processing application inside a Xilinx FPGA chip. You can build a router, stateful firewall, IPs, IDs, DDOS protector, MPLS encapsulator, VXLAN encapsulator and router, etc.
Pros	You can build packet processing applications that run at very high speed, from 1Gbps to 400Gbps. Support of P4 language makes it easy to work with FPGA.
Caveats	FPGAs that can process packets at high speeds are generally expensive. Building applications for FPGA requires FPGA skills and knowledge, and the coding requires attention to numerous aspects.

IV. IO 추상화와 Datapath

❖ **FPGAs and Xilinx SDNet**

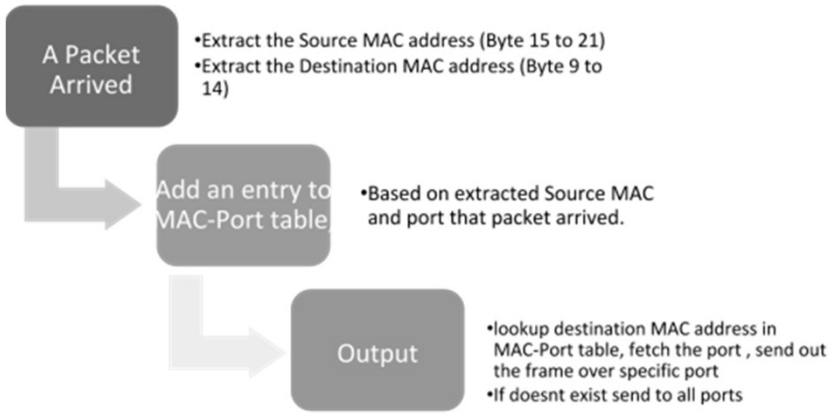
❖ **Key Differences between CPU, ASIC and FPGA**



IV. IO 추상화와 Datapath

❖ **Working with FPGAs**

❖ **Building a Basic Layer 2 Switch Using an FPGA**



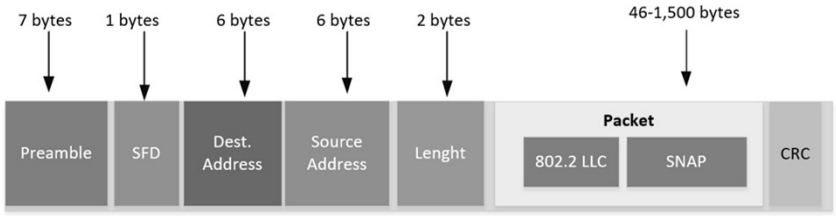
https://www.opencontainers.org/

105

IV. IO 추상화와 Datapath

❖ **Working with FPGAs**

- **To build an FPGA-based network application, you don't need to extract all fields of Ethernet, IP, etc. You only need to extract and compare what is relevant for you. For example, if you are building a DDOS protection application, you need to extract only fields such as the Source & Destination IP address and TCP ports (or, in some advanced methods, a pattern), and compare them against a predefined list in your FPGA. If the packet matches that pattern, FPGA should just drop the packet.**



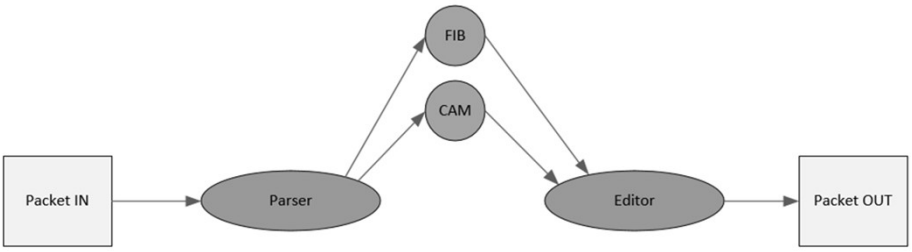
https://www.opencontainers.org/

106

IV. IO 추상화와 Datapath

❖ **Xilinx SDNet:** Xilinx has created a framework called SDNet, which allows developers to build data plane programs to run in their FPGA chips. Initially, it was only supporting the Xilinx's proprietary SDK (SDNet). However, it now supports the standard P4 language.

- **To design a data plane application that runs on an FPGA, you can start by drawing the state machine or a simple flow diagram showing how you would like your application to work. You can take a look at the following illustration of a simple routing application:**



Simple Routing Application

https://www.opencontainers.org/

107

IV. IO 추상화와 Datapath

❖ **Barefoot Networks Tofino Programmable Switch Silicon**

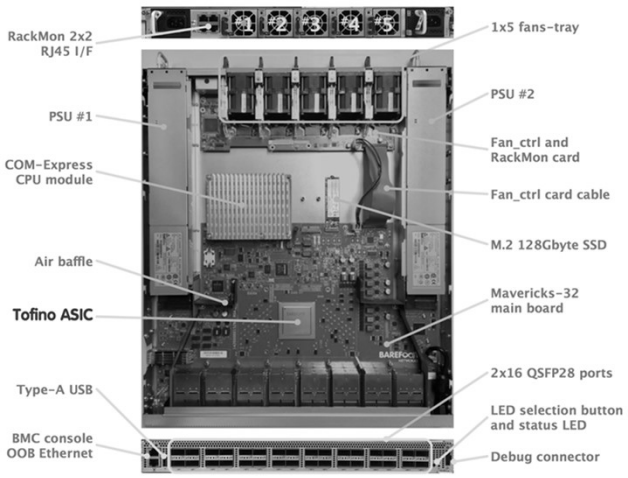
Barefoot Networks Tofino Switch Chipset - Quick Summary	
Name	Barefoot Network Tofino Switch Chipset
By	Barefoot Networks
Where it runs	A physical chipset (ASIC) for Ethernet switches
What it does	Apart from the standard features of an Ethernet switch silicon, it adds flexibility for data plane programming in a switch by supporting the P4 language and allowing to directly program the data plane
Features	Up to 6.5 Tbps; supports 32 x 100G ports; supports P4 language
What it can do out-of-the-box	Tofino is a switch chipset which can be used to build an Ethernet switch. Hardware vendors can use this chipset to build an off-the-shelf Ethernet switch. Software vendors can use the Barefoot software tools to build networking software that can run and drive the chipset, as well as use the chipset's advanced packet processing features.

https://www.opencontainers.org/

108

IV. IO 추상화와 Datapath

❖ **Barefoot Networks Tofino Programmable Switch Silicon**



https://www.opencontainers.org/

JS Lab

JS Lab

V. Network Operating Systems

- OpenSwitch (OPX)
- Disaggregated Network Operating System (DANOS)
- Stratum
- Open Network Linux (ONL)
- Free Range Routing (FRR)
- P4 Language
- SONiC (Software for Open Networking in the Cloud)
- FBOSS (Facebook Open Switching System)
- Cumulus Linux

https://www.linuxfoundation.org/projects/networking/

V. Network Operating Systems

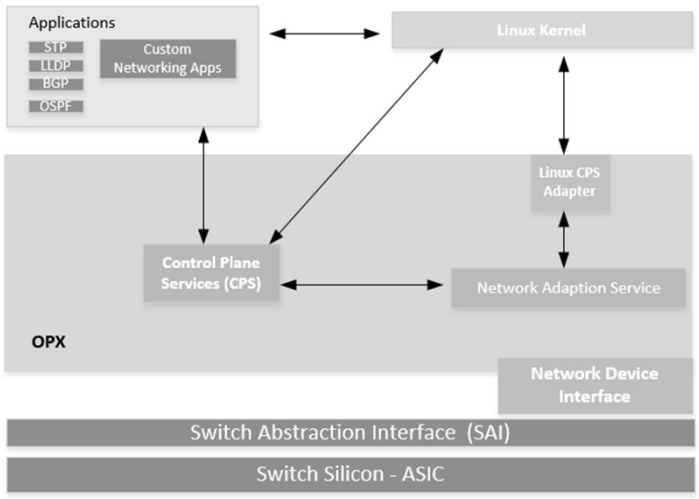
❖ **OpenSwitch (OPX) Architecture:**

OPX - Quick Summary	
Name	OpenSwitch (OPX)
By	The Linux Foundation
Where it runs	On a compatible bare metal switch, mostly Dell EMC bare metal switches
What it does	Layer 2, Layer 3 routing and switching
Features	Layer 2 and Layer 3 features, BGP, OSPFv2/3, VRRP, VRF, ACL, QoS policing, shaping, automation with Ansible, Puppet, Chef, Python
What it can do out-of-the-box	You can load and use OpenSwitch on a compatible hardware and use it as a standard L2/L3 switch. It supports automation out-of-the-box, which can help you build an infrastructure as a code platform. OpenSwitch has some command line utilities, but it uses standard Linux commands for routing configuration.

https://www.openswitch.net/

V. Network Operating Systems

❖ **OpenSwitch (OPX) Architecture:**



https://www.openswitch.net/

JS Lab

V. Network Operating Systems

❖ **OpenSwitch (OPX) Commands:**

Create VLAN 500, add Port 1 and Port 5 as tagged interfaces:

```
$ brctl addbr br500
$ ip link add link e101-001-0 name e101-001-0.500 type vlan id 500
$ ip link add link e101-005-0 name e101-005-0.500 type vlan id 500
$ brctl addif br500 e101-001-0.500
$ brctl addif br500 e101-005-0.500
```

Create a static route:

```
$ ip route add 10.0.0.0/24 via 192.168.1.2
$ ip route show
```

OpenSwitch has also its own utilities, such as:

- Enable logging and debugging
  `opx_logging_cli enable`
- Show system alarms
  `opx-show-alms`
- Show version
  `opx-show-version`
- Enable SAI-specific logging
  `opx-switch-log`

https://www.openswitch.net/

JS Lab

V. Network Operating Systems

❖ **OpenSwitch (OPX) Supports Automation:**

```
node 'Switch1.LNF.org' {
$int_enabled = true
$int_loopback = '10.0.0.1'
$int_layer3 = {
e101-001-0 => {'int'=>'e101-001-0', 'address' => '10.1.1.1', 'netmask' =>
'255.255.255.0', 'cidr_netmask' => 24},
e101-002-0 => {'int'=>'e101-002-0', 'address' => '10.2.2.2', 'netmask' =>
'255.255.255.0', 'cidr_netmask' => 24},
}
$bgp = {
myasn => 65002,
peergroupv4 => [ { name => 'Router-101', asn => 65000, peers => [ '192.168.0.2','192.168.10.2' ]
} ]
}
include ibgp::switch
}
```


V. Network Operating Systems

❖ **Disaggregated Network Operating System (DANOS):**

DANOS - Quick Summary	
Name	Disaggregated Network Operating System (DANOS)
By	The Linux Foundation
Where it runs	Not yet disclosed, but expected ti run on a compatible bare metal switch
What it does	As per the AT&T whitepaper, it supports Layer 2 and Layer 3 routing and sw itching
Features	Not yet disclosed, but expected to have Layer 2, Layer 3, BGP, OSPFv2/3, VRRP, VRF, ACL, QoS, MPLS, MP-BGP, and other BGP extensions
What it can do out-of-the-box	Not yet disclosed or available

V. Network Operating Systems

❖ **Disaggregated Network Operating System (DANOS):**

- **Supports ONIE for installation environment**
- **Supports the P4 language, and possibly, data plane programming**
- **Support SDN agents such as OpenFlow**
- **Supports automation and YANG models**
- **Supports a CLI**
- **Uses Forwarding Abstraction Layer (FAL) for communication with switch silicon**
- **Will be orchestrated with ONAP**

https://www.openswitch.net/

117

V. Network Operating Systems

❖ **Stratum:**

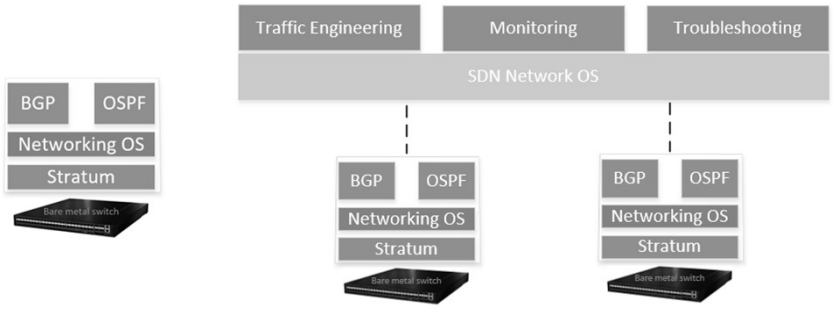
Stratum - Quick Summary	
Name	Stratum
By	The Linux Foundation - Open Networking Foundation
Where it runs	On a compatible bare metal switch
What it does	Not fully disclosed. Stratum launched in March 2018 - it is a NOS that can ru n a switch as an L2/L3 or fully managed by a network controller
Features	Traditional L2/L3, data plane programming using P4, OpenConfig, SDN age nt
What it can do out-of-the-box	Although Stratum is still in an incubation stage, it is a NOS running on bare metal switches. Stratum is considered as a NOS that you can use to slowly migrate your network to a fully SDN, traffic-engineered network. Stratum can run traditional networking protocols such as BGP, STP, OSPF, etc., while all owing you to program the switches using OpenFlow.

https://stratumproject.org/

118

V. Network Operating Systems

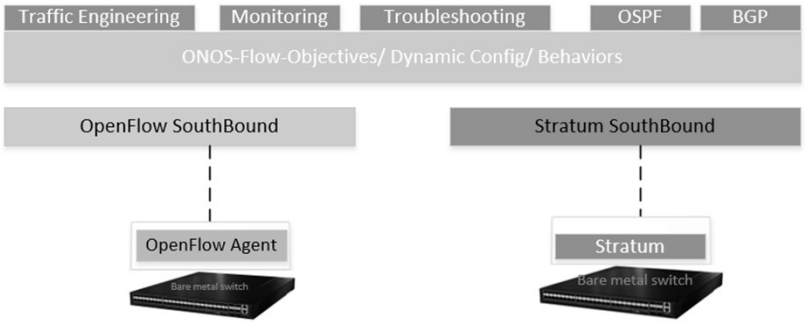
❖ **Stratum:**



Stratum Supports Hybrid Mode

V. Network Operating Systems

❖ **Stratum:**



Stratum Supports a Full SDN-based System

V. Network Operating Systems

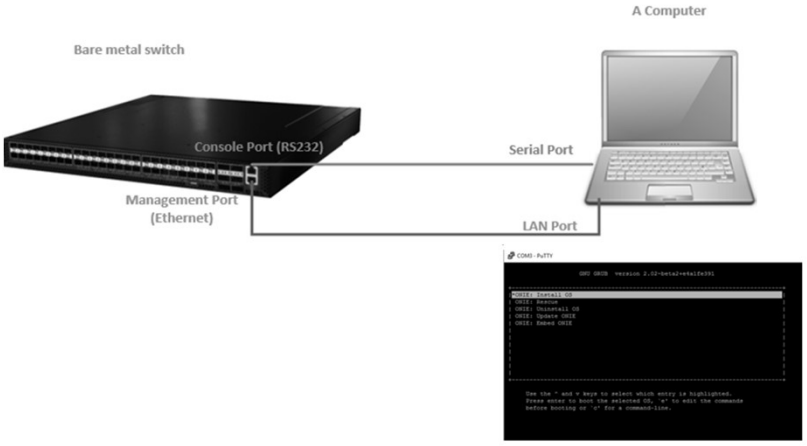
❖ Open Network Linux (ONL):

ONL - Quick Summary	
Name	Open Network Linux (ONL)
By	OCP and Open Network Linux Community, backed by Big Switch Networks, Inc.
Where it runs	On a compatible bare metal switch
What it does	ONL is a base-level operating system that can boot a full fledged Linux on a bare metal switch
Features	ONL is based on Debian Wheezy. It allows the installation of any Linux-compatible applications or the ASIC driver
What it can do out-of-the-box	ONL includes a comprehensive HCL that allows you to install it on most bare metal switches. After installing ONL, you can download the switch silicon drivers from the switch chipset manufacturer's website, and install them on ONL. This will enable ONL to control the switch ASIC. In Broadcom-based bare metal switches, a very common method is to download Broadcom's OFDPA (OpenFlow Data Plan Architecture) or OpenNSL (Open Network Switch Library) SDKs and deploy them on ONL. This is a simple way to make an OpenFlow-capable switch.

V. Network Operating Systems

❖ Installing Open Network Linux (ONL):

```
install_url tftp://172.17.172.103/latest-amd64.installer
```



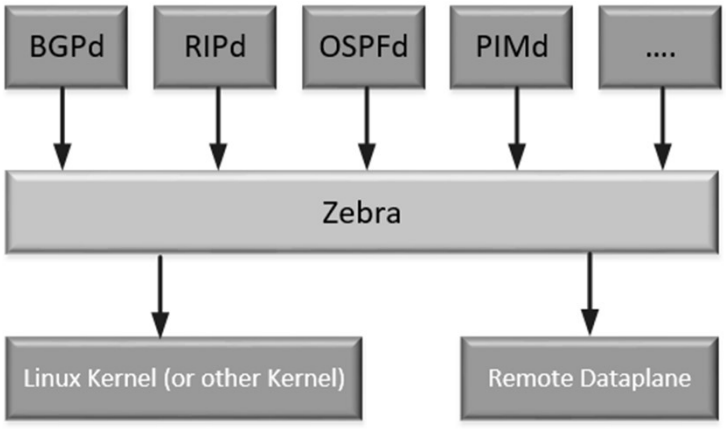
V. Network Operating Systems

❖ **Free Range Routing (FRR):**

FRRouting - Quick Summary	
Name	Free Range Routing (FRR)
By	The Linux Foundation
Where it runs	On a Linux host, or a bare metal switch that runs Linux
What it does	IP routing protocol suite for Linux
Features	In addition to the Quagga features, it provides VRF, EVPN, LSP, BFD, LDP for MP LS, CLI, support for JSON outputs
What it can do out-of-the-box	You can use FRR out-of-the-box on a Linux system to build a full-fledged router. There are other use cases as well, such as: 1. Routing on a host (on a compute node) to establish BGP with data center switches 2. Using FRR on an ONL to build an L3 Ethernet switch (requires additional components to program the switch chipset)

V. Network Operating Systems

❖ **Free Range Routing (FRR) Architecture :**



V. Network Operating Systems

❖ **P4 Language:**

P4 - Quick Summary	
Name	P4
By	P4 Community, joined the Open Networking Foundation and The Linux Foundation (March 2018)
What it does	Language to define the behavior of data planes
Features	Supports the match-and-action model with flexibility of matching different headers in a packet or a frame
What it can do out-of-the-box	P4 is a language used to describe the behavior of a data plane. You can build various applications in a data plane, such as DDOS protections, firewalls, IPS, IDs, etc.

V. Network Operating Systems

❖ **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;
 }
}
```

V. Network Operating Systems

❖ **P4 Language:**

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;
  }
}
```

Opensource Networking
james@jslab.kr

V. Network Operating Systems

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

SONiC - Quick Summary	
Name	Software for Open Networking in the Cloud (SONiC)
By	Created by Microsoft, open source NOS
Where it runs	On a supported bare metal switch
What it does	Layer 2 and Layer 3 networking on a bare metal switch
Features	Supports a variety of switch chipsets. Has a CLI, which makes the interactio n with the software easier.
What it can do out-of-the-box	You can download and install SONiC on your bare metal switch and start us ing it as a leaf, spine, or other interconnects for routing and switching.

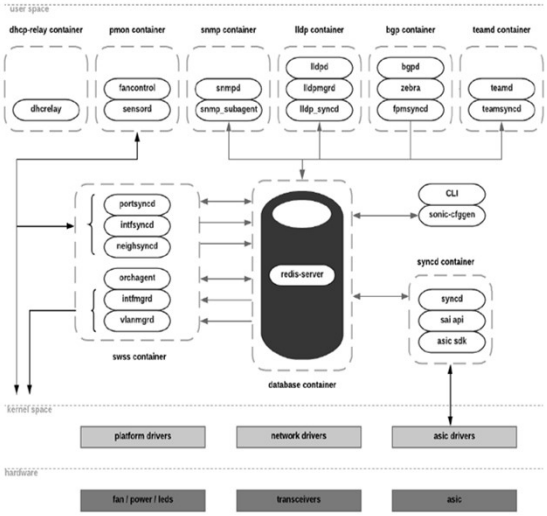
SONiC is based on Linux and uses SAI to manage and drive the switch chipset.

https://azure.github.io/SONiC/

Opensource Networking
james@jslab.kr

V. Network Operating Systems

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



V. Network Operating Systems

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

❖ **The redisDB has multiple tables. For example:**

- **BGP Table**: Stores configuration related to BGP neighbors, advertisements.
- **Port:** Contains configuration related to interfaces, speeds.
- **PortChannel:** Contains configuration of Link Aggregations (bonding/etherchannel, port channel).
- **VLAN:** Contains VLAN IDs, member ports.
- **VLAN members:** Contains configuration for individual ports for 802.1q tagging.
- **Layer 3 tables:** INTERFACE, PORTCHANNEL_INTERFACE, and VLAN_INTERFACE to store IP address details of Layer 3 interfaces.
- **ACL_RULE:** Contains configuration of access lists.

https://azure.github.io/SONiC/

V. Network Operating Systems

❖ SONiC (Software for Open Networking in the Cloud) Commands:

Show MAC:

```
admin@sonic:~$ show mac
No.    Vlan MacAddress         Port
-----  ------ -----------------  -----------
 1   1000 E2:8C:56:85:4A:CD  Ethernet12
```

Configure VLANs and member ports:

```
root@sonic:/# config vlan add 1200
root@sonic:/# config vlan member add 1200 Ethernet1
root@sonic:/# config vlan member add 1200 Ethernet9 -u   (Defining Untagged)
root@sonic:/# config vlan member add 1200 Ethernet8
```

BGP **show** commands:

```
admin@sonic:~$ show bgp neighbors 192.168.1.161
admin@sonic:~$ show bgp neighbors 192.168.1.161 advertised-routes
admin@sonic:~$ show bgp neighbors 192.168.1.161 received-routes
admin@sonic:~$ show bgp neighbors 192.168.1.161 routes
```

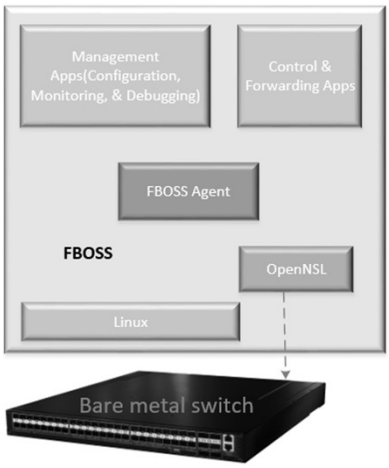

V. Network Operating Systems

❖ FBOSS (Facebook Open Switching System):

FBOSS - Quick Summary	
Name	Facebook Open Switching System (FBOSS)
By	Facebook
Where it runs	On a Linux OS that is installed on a bare metal switch
What it does	FBOSS provides an agent software that can be remotely managed via API c alls to manage the switch chipset silicon. FBOSS should be used with an ext ernal controller
Features	FBOSS abstracts and provides APIs to remotely program a hardware switch (mainly based on Broadcom Trident and Tomahawk chipsets)
What it can do out-of-the-box	You can install FBOSS on a bare metal switch that is running ONL. With FB OSS, you can control the switch silicon using a controller software. You can build your own controller software to interact with FBOSS. Your controller sof tware can be a routing protocol software such as BIRD or Quagga, that can be connected to FBOSS

V. Network Operating Systems

❖ **FBOSS Daemons:**



FBOSS mainly uses the Broadcom OpenNSL library to communicate with a Broadcom chipset.

https://github.com/facebook/fboss

V. Network Operating Systems

❖ **FBOSS Daemons:**

Configuring a port:

```
{
  "ports": [
    {
      "logicalID": 1,
      "state": 2,
      "minFrameSize": 64,
      "maxFrameSize": 1500,
      "parserType": 1,
      "routable": true,
      "ingressVlan": 1000,
      "speed": 0
    }
}
```

Configure a routed interface:

```
"interfaces": [
   {
     "intfID": 1000,
     "routerID": 0,
     "vlanID": 1000,
     "ipAddresses": [
       "169.254.0.10/16",
       "2001:db:1111:1150::a/64"
     ],
     "ndp": {
       "routerAdvertisementSeconds": 4,
       "curHopLimit": 255,
       "routerLifetime": 1800,
       "prefixValidLifetimeSeconds": 2592000,
       "prefixPreferredLifetimeSeconds": 604800,
       "routerAdvertisementManagedBit": true,
       "routerAdvertisementOtherBit": true
     }
   }
```

https://github.com/facebook/fboss

V. Network Operating Systems

❖ **FBOSS Daemons:**

Configure a VLAN IP interface (SVI):

```
{
 "intfID": 3004,
  "routerID": 0,
  "vlanID": 3004,
  "ipAddresses": [
"10.11.24.111/31",
"2001:db:3336:e01:1000::aa/127"
  ]
}
```

Configure a VLAN and 802.1q (VLAN 1000 tagged on interface 1, untagged on interface 2):

```
"vlanPorts": [
   {
     "vlanID": 1000,
     "logicalPort": 1,
     "spanningTreeState": 2,
     "emitTags": false
   },
   {
     "vlanID": 1000,
     "logicalPort": 2,
     "spanningTreeState": 2,
     "emitTags": true
   }
```


https://github.com/facebook/fboss

V. Network Operating Systems

❖ **Cumulus Linux:**

Cumulus Linux - Quick Summary	
Name	Cumulus Linux NOS
By	Cumulus Networks
Where it runs	On supported bare metal switches, Linux hosts
What it does	Layer 2, Layer 3 automation
Features	Layer 2 and Layer 3 features, VRF, compatibility with automation tools such as Puppet and Ansible. Also, it provides a CLI tool.
What it can do out-of-the-box	Cumulus Linux is a commercial network operating system that can run on a supported bare metal switch. Cumulus use cases include mainly datacenter networks (CLOS-based leaf, spine).

https://www.opencontainers.org/

V. Network Operating Systems

- Cumulus Linux:
- To configure and manage Cumulus Linux, you need to use pure Linux networking commands and configuration. For example:
  - Use ifupdown2 to manage interfaces, bonds (Ethernetchannels)
  - Use brctl to configure and manage bridges (VLANs, tagging)
  - Use cl-acltool (Cumulus Access List Tool) to manage access lists
  - Use vtysh to manage routing daemons such as bgpd, ospfd, etc.

Opensource Networking
james@jslab.kr

V. Network Operating Systems

- Cumulus Linux:
- To create VLANs:
  - Create VLAN 100 to 110:
  - switch# net add vlan 100-110
  - Add port 10 to VLAN 100 as untagged:
  - switch1# net add int swp10 bridge access 100
  - Add port 18 to VLAN 100 as 802.1q tagged:
  - net add int swp18 bridge trunk vlans 100
  - Create a bridge IP interface (AKA Switch Virtual Interface SVI or vlan interface):
  - net add vlan 100 ip address 172.16.17.1/24

Opensource Networking
james@jslab.kr

I. 오픈소스 네트워킹 개요
II. 오픈소스와 SDN Landscape
III. 소프드웨어/하드웨어 분리
IV. IO 추상화와 Data Path
V. **NOS** (Network Operating systems)
VI. 네트워크 제어 (Network Control)
VII. 클라우드와 가상화 관리
VIII. 네트워크 가상화
IX. **NFV** (Network Function Virtualization)
X. 네트워크 자동화
XI. 네트워크 데이터 분석
XII. Use Case
❖ 실습교재 (별도)

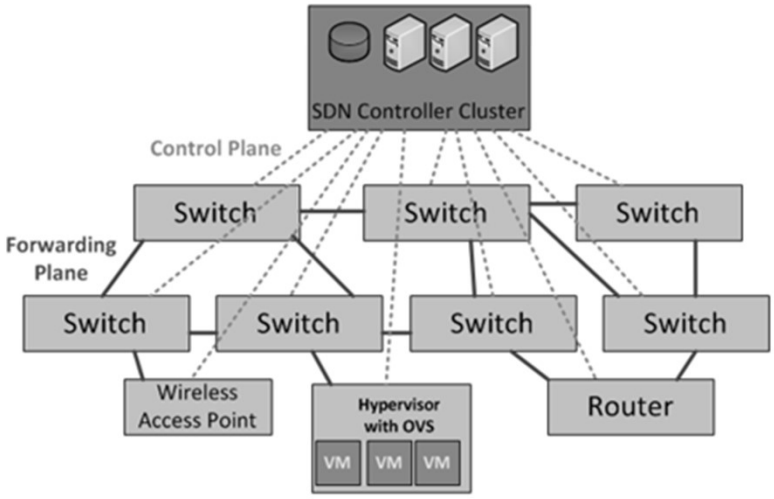
VI. 네트워크 제어(Network Control)

- **Introduction to SDN Controllers**
- **OpenDaylight** (ODL)
- **Open Network Operating System** (ONOS)
- **Tungsten Fabric** (formerly OpenContrail)
- **Project Calico**
- **Cisco Application-Centric Infrastructure** (ACI)
- **Floodlight**
- **Big Cloud Fabric** (BCF)

https://www.linuxfoundation.org/projects/networking/

VI. 네트워크 제어 (Network Control)

❖ **Introduction to SDN Controllers:**



141

VI. 네트워크 제어 (Network Control)

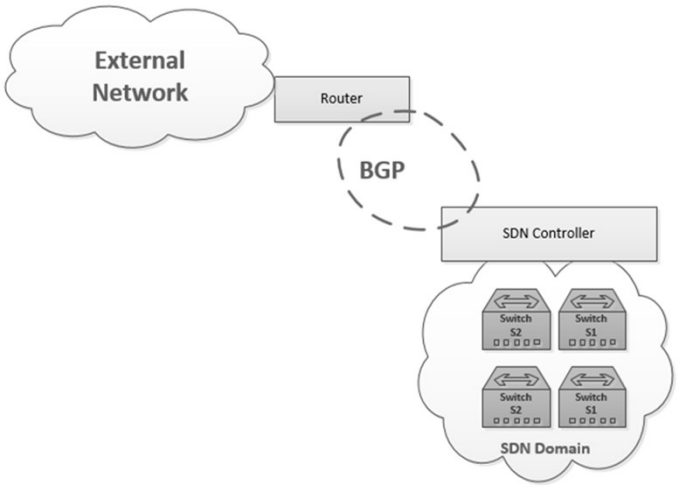
❖ **Introduction to SDN Controllers:**

Feature	Direct Fabric Programming	Overlay
Can work and co-exist with existing IP network	NO	YES
Required to encapsulate packets	NO	YES
Scalable	YES	YES

142

VI. 네트워크 제어 (Network Control)

❖ **Introduction to SDN Controllers:**



Relationship between a Network Managed by an SDN Controller and External Networks

143

VI. 네트워크 제어 (Network Control)

❖ **OpenDaylight (ODL):**

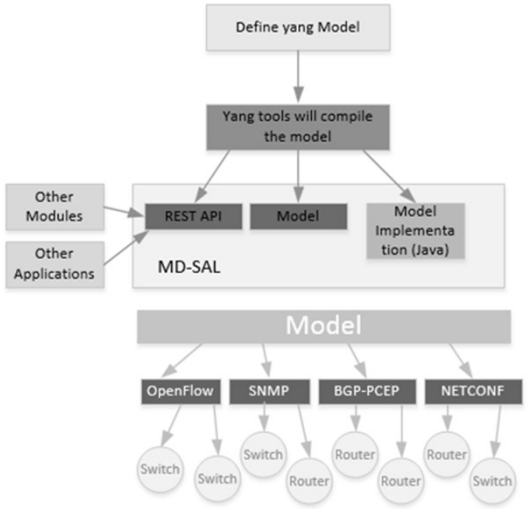
ODL - Quick Summary	
Name	OpenDaylight (ODL) SDN controller
By	The Linux Foundation
Where it runs	As an SDN controller, it runs on a separate host or a cluster of nodes to manage the underlying network
What it does	Manages and operates underlying network devices using southbound protocols
Features	It comes with a comprehensive library of protocols, ready made applications. It is a modular and flexible system, and it allows you to develop any networking application.
What it can do out-of-the-box	You can use ODL for networking with your cloud orchestration platform such as OpenStack. ODL can manage OpenFlow-based switches (it can be bare metal). You can develop a custom event-driven application on top of ODL. Normally, applications get activated on a specific event (for example, a packet arrived) and perform some actions.

https://www.opendaylight.org/what-we-do/odl-platform-overview

144

VI. 네트워크 제어 (Network Control)

❖ **OpenDaylight (ODL):**



https://www.opendaylight.org/what-we-do/odl-platform-overview

VI. 네트워크 제어 (Network Control)

❖ **OpenDaylight (ODL):**

https://www.opendaylight.org/what-we-do/odl-platform-overview

VI. 네트워크 제어 (Network Control)

❖ **Open Network Operating System (ONOS):**

ONOS - Quick Summary	
Name	Open Network Operating System (ONOS)
By	Hosted by The Linux Foundation, maintained by the Open Networking Foundation
Where it runs	As an SDN controller platform, it runs on a separate host to manage the underlying network
What it does	Manages and operates the underlying network devices using southbound protocols
Features	Modular software that allows building plugins and applications, built-in northbound APIs such as REST and gRPC, built-in southbound protocols such as P4, OpenFlow, NETCONF, TL1, SNMP, CLI, BGP, RESTCONF. GUI, YANG tool chain
What it can do out-of-the-box	ONOS can be used as a standard SDN controller to manage underlying networking devices such as routers and switches. ONOS supports different southbound protocols, such as OpenFlow, BGP, and NETCONF, which can configure and manage underlying routers and switches.

https://onosproject.org/

147

VI. 네트워크 제어 (Network Control)

❖ **ONOS vs ODL:**

	ODL	ONOS
Applications for datacenters, cloud environment	ODL provides applications to integrate with OpenStack (such as VTN Virtual Tenant Manager) or southbound protocols that are used in datacenters, such as OVSDB	SONA (Simplified Overlay Network Architecture) is an ONOS application which integrates with OpenStack and provides network virtualization and tenant isolation
Applications for service providers, telcos	Limited to some use cases and southbound protocols, such as BGP-PCEP	Has many use cases and applications for telcos, such as CORD, Packet Optical, IP RAN, SDN IP, VPLS, Carrier Ethernet, etc.
Performance and high availability	ODL supports clustering	ONOS clustering is mature and comprehensive

148

VI. 네트워크 제어 (Network Control)

❖ **Tungsten Fabric (formerly OpenContrail):**

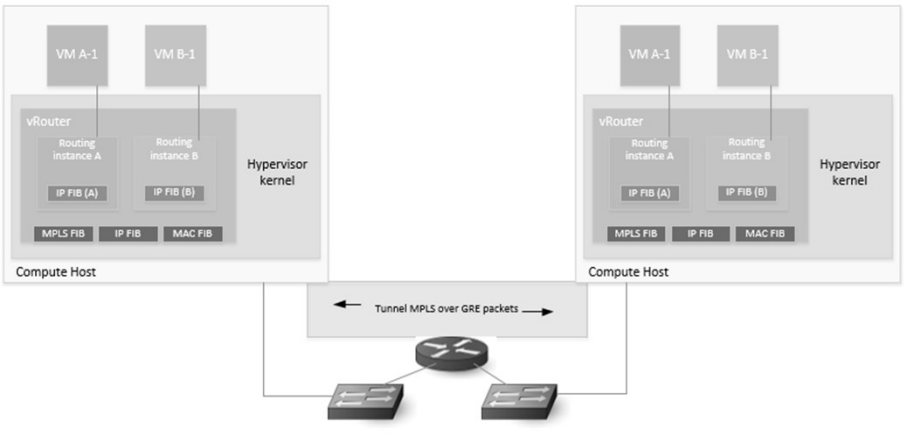
Tungsten Fabric - Quick Summary	
Name	Tungsten Fabric (formerly OpenContrail)
By	The Linux Foundation
Where it runs	On multiple nodes; it can be deployed on each compute host of a virtual environment
What it does	Creates and manages virtual overlay networks over any underlying fabric
Features	Layer2 and Layer3 overlays over a Layer3 underlay. Uses MPLS over GRE/MPLS over UDP. The underlying switches do not need to support MPLS
What it can do out-of-the-box	Tungsten Fabric works well in a cloud environment, such as OpenStack. You can deploy and integrate Tungsten Fabric with your cloud orchestration software to provide services such as service function chaining and tenant network isolation.

https://tungstenfabric.io/

149

VI. 네트워크 제어 (Network Control)

❖ **Tungsten Fabric (formerly OpenContrail):**



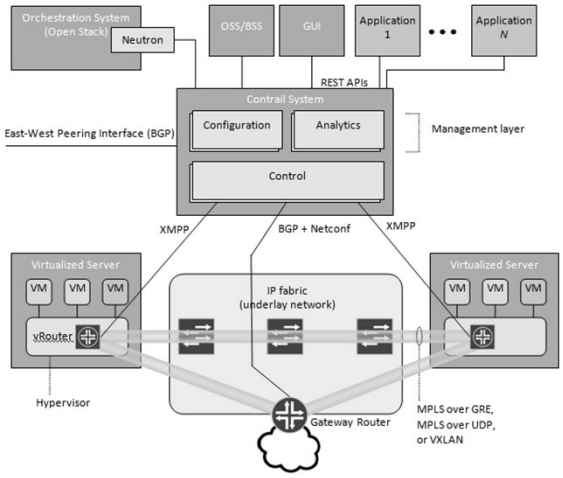
Virtual Overlay Networks Over A Standard Legacy L2/L3 Network

https://tungstenfabric.io/

150

VI. 네트워크 제어 (Network Control)

❖ **Tungsten Fabric (formerly OpenContrail):**



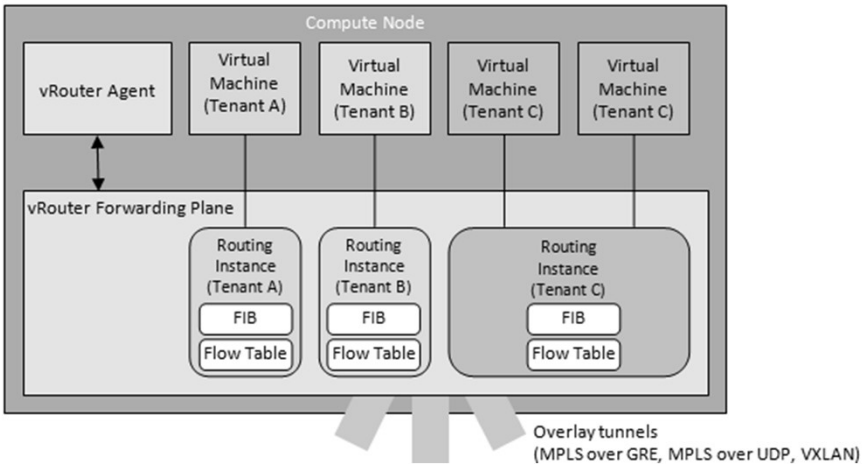
Tungsten Fabric Architecture

https://tungstenfabric.io/

JS Lab

VI. 네트워크 제어 (Network Control)

❖ **Tungsten Fabric (formerly OpenContrail):**



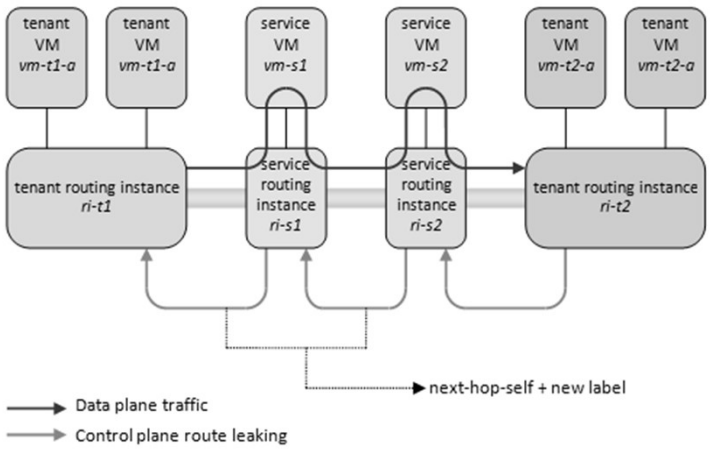
Tungsten Fabric vRouter Architecture within a Compute Host

https://tungstenfabric.io/

JS Lab

VI. 네트워크 제어 (Network Control)

❖ **Tungsten Fabric (formerly OpenContrail):**



Service Chaining Policy Applied for Traffic between the Two Virtual Machine Workloads

https://tungstenfabric.io/

153

VI. 네트워크 제어 (Network Control)

❖ **Project Calico:**

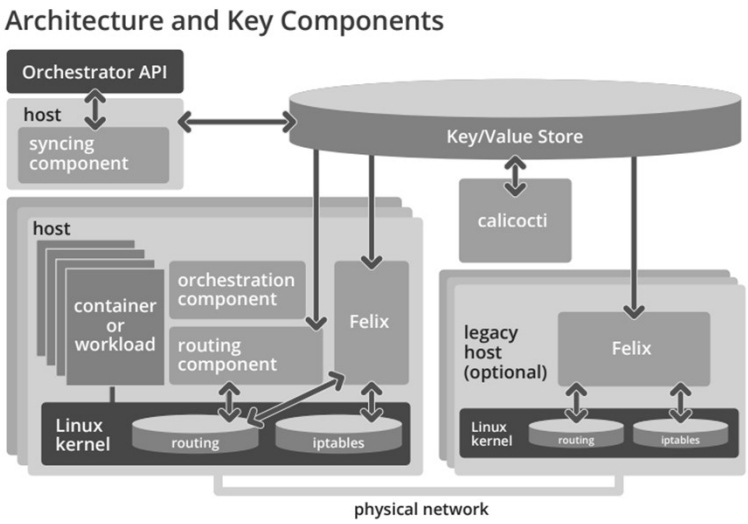
Project Calico - Quick Summary	
Name	Project Calico
By	Calico open source community, backed by Tigera, Inc.
Where it runs	The Calico controller runs on multiple nodes, and gets deployed on each compute host of a virtual environment
What it does	Creates and manages virtual networks for VMs, containers and bare metal
Features	Supports overlay and encapsulation, uses a built-in IPAM for allocation and assigning IP addresses to workloads
What it can do out-of-the-box	Project Calico can be used in OpenStack, Kubernetes, and other orchestration environments to provide networking

https://www.projectcalico.org/

154

VI. 네트워크 제어 (Network Control)

❖ **Project Calico:**



https://www.projectcalico.org/wp-content/uploads/2018/04/ProjectCalico.v3.1.datasheet.pdf

155

VI. 네트워크 제어 (Network Control)

❖ **Cisco Application-Centric Infrastructure (ACI):**

ACI - Quick Summary	
Name	Cisco ACI SDN Controller
By	Cisco Systems
Where it runs	ACI/APIC runs on a cluster of servers to manage the underlying network of Nexus 9000 switches
What it does	Manages the SDN domain and its switches. Provides traffic isolation, security, and virtual networks.
Features	Manages single and multiple sites
What it can do out-of-the-box	You can integrate the ACI with your virtualization platform or cloud environment and manage your physical and virtual network switches. ACI provides automation features that can help enhance the processes and reduce the time for changes in the network.

https://www.opendaylight.org/what-we-do/odl-platform-overview

156

VI. 네트워크 제어 (Network Control)

❖ **Floodlight:**

Floodlight - Quick Summary	
Name	Floodlight Controller
By	Sponsored by Big Switch Networks
Where it runs	On a cluster of hosts to manage an underlay network
What it does	Floodlight is a modular SDN controller capable of having SDN applications. It manages the underlying physical and virtual switches via the OpenFlow protocol.
Features	Integrates with OpenStack; open source
What it can do out-of-the-box	Floodlight documentation is not fancy and might seem old. However, you can use Floodlight to manage any OpenFlow-capable physical switch.

https://www.bigswitch.com/tags/floodlight-controller

VI. 네트워크 제어 (Network Control)

❖ **Big Cloud Fabric (BCF):**

Big Cloud Fabric - Quick Summary	
Name	Big Cloud Fabric (BCF)
By	Big Switch Networks
Where it runs	On a cluster of servers. To manage the data plane, Big Cloud Fabric provides a lightweight NOS called SwitchLight (to be installed on a compatible hardware)
What it does	Manages a datacenter or tenant-based cloud environment
Features	Commercial SDN controller, works with bare metal switches, integrates with cloud orchestration tools such as OpenStack, VMware vSphere, and Kubernetes.
What it can do out-of-the-box	By deploying Big Cloud Fabric, you can integrate a network with your cloud orchestration platform. For example, BCF integrates with VMware vSphere. Any VLAN you create on VMware vSphere will be automatically presented in your underlying network managed by BCF. BCF can apply network policies and service chaining on your network traffic to enforce traffic to pass through a network analytics or IPS when it is being routed to the Internet.

https://www.bigswitch.com/products/big-cloud-fabric

VII. 클라우드와 가상화 관리

- **Open Network Automation Platform** (ONAP)
- **M-CORD** (Mobile CORD)
- **R-CORD** (Residential CORD)
- **E-CORD** (Enterprise CORD)
- **Trellis**
- **Open Source MANO**
- **Open Platform for NFV** (OPNFV)
- **Open Security Controller** (OSC)
- **Akraino Edge Stack**

https://www.linuxfoundation.org/projects/networking/

VII. 클라우드와 가상화 관리

❖ **Open Network Automation Platform (ONAP):**

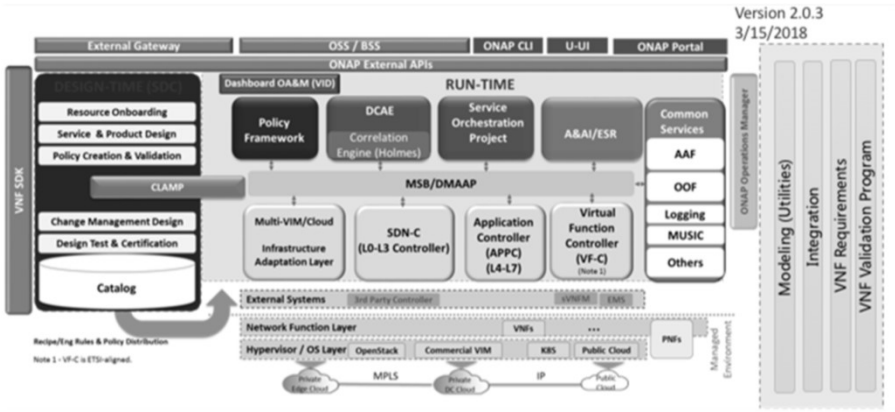
ONAP - Quick Summary	
Name	Open Network Automation Platform (ONAP)
By	The Linux Foundation
Where it runs	On separate hosts, recommended to run on Kubernetes or OpenStack
What it does	ONAP is a platform that orchestrates the lifecycle of virtual network services in a software defined networking environment
Features	Open source; creates services that consist of VNFs and policies. Runs and executes the services. Has a closed-loop.
What it can do out-of-the-box	ONAP relies on many other components and platforms; it will not be able to deliver the real functions without the other components out-of-the-box. There are multiple use cases for ONAP, such as uCPE, edge networking services, Voice-over-LTE, virtual firewall, virtual DHCP server, etc. You can deploy a full or minimum ONAP environment using OpenStack. A full ONAP environment requires numerous virtual machines and gigabytes of RAM.

https://www.onap.org/ https://onap.readthedocs.io/en/dublin/

161

VII. 클라우드와 가상화 관리

❖ **Open Network Automation Platform (ONAP)**
❖ **ONAP Architecture**



https://www.onap.org/

162

VII. 클라우드와 가상화 관리

❖ **ONAP Design Time Environment:**

- **Service Design and Creation (SDC)**
  - ✓ To define system assets and their policies.
- **VNF Software Development Kit (VNFSDK) and VNF Validation Program (VVP)**
  - ✓ For VNF packaging and validation of VNFs.
- **Policy Creation (POLICY)**
  - ✓ To define policies that need to be maintained or enforced.
- **Closed Loop Automation Management Platform (CLAMP)**
  - ✓ To design and manage closed control loops.
- **Optimization Framework (OOF)**
  - ✓ To optimize the application and services.

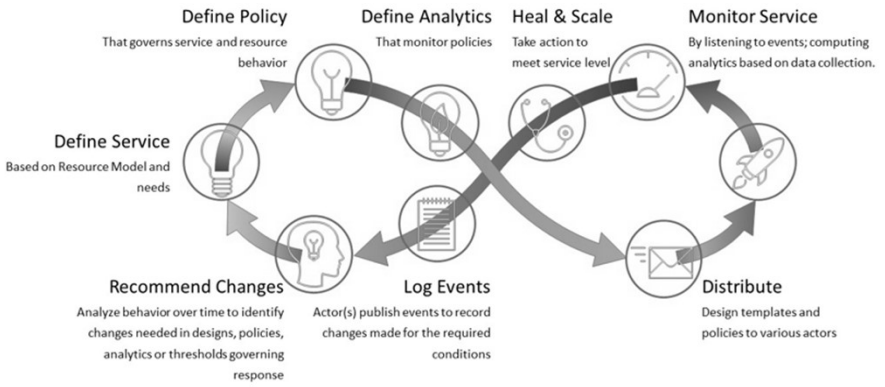
VII. 클라우드와 가상화 관리

❖ **ONAP Run Time Environment:**

- **Service Orchestrator (SO)**
  - ✓ The Service Orchestrator is an automation engine in the ONAP Run Time environment.
- **Software Defined Network Controller (SDNC)**
  - ✓ It is responsible for executing the network configuration.
- **Application Controller (APPC)**
  - ✓ It is responsible for executing and configuring the Virtual Network Functions (VNF).
- **Virtual Function Controller (VF-C)**
  - ✓ It is responsible for the lifecycle management of the Virtual Network Functions (VNF) which are run by the VNF manager.
- **Active and Available Inventory (A&AI)**
  - ✓ It is responsible for the real-time view of system resources, products and their relationships.

VII. 클라우드와 가상화 관리

❖ **ONAP: Closed Loop Automation:**

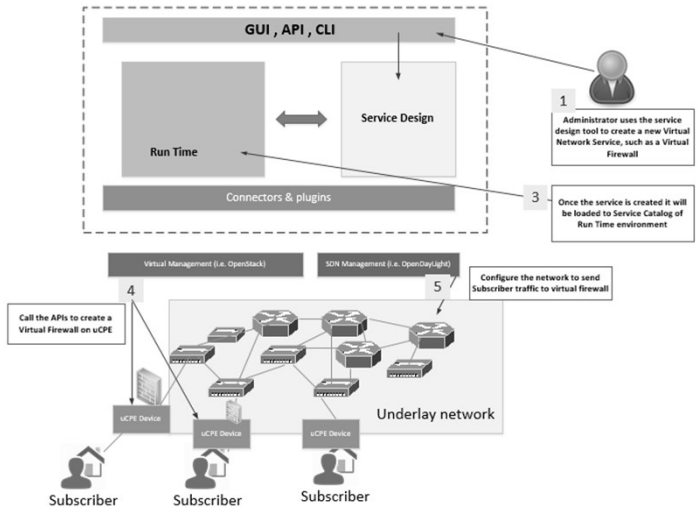


https://www.onap.org/

165

VII. 클라우드와 가상화 관리

❖ **ONAP Example:**

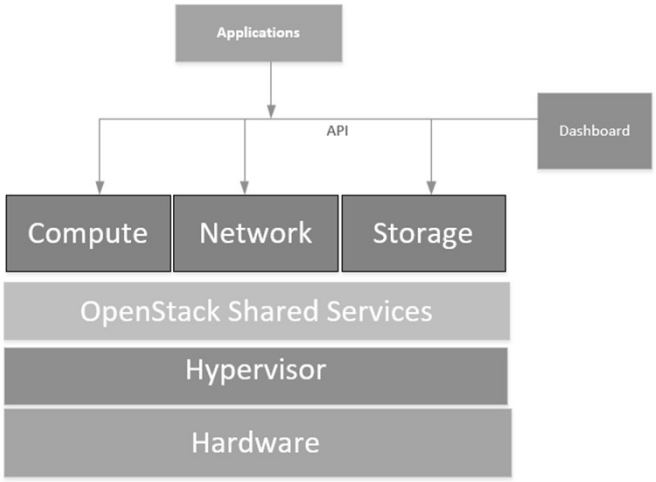


https://www.onap.org/ https://onap.readthedocs.io/en/beijing/guides/onap-developer/settingup/onap_oom.html

166

VII. 클라우드와 가상화 관리

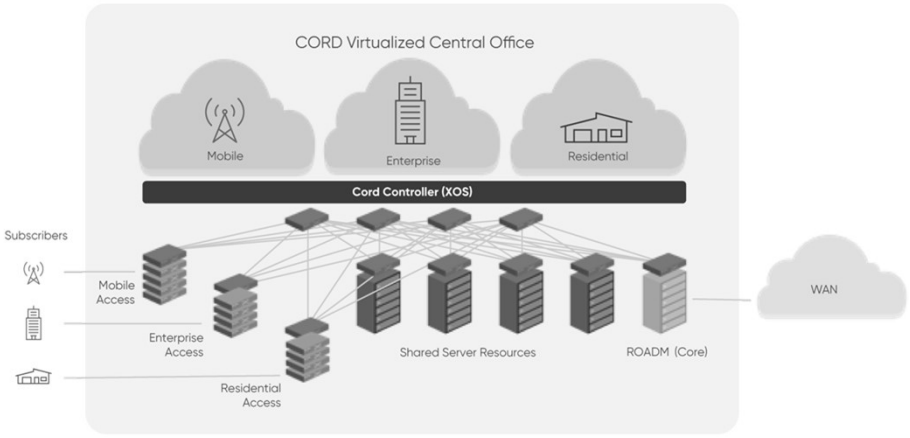
❖ **ONAP and OpenStack:**



https://www.onap.org/ https://onap.readthedocs.io/en/amsterdam/guides/onap-developer/settingup/fullonap.html

VII. 클라우드와 가상화 관리

❖ **ONAP and CORD:**

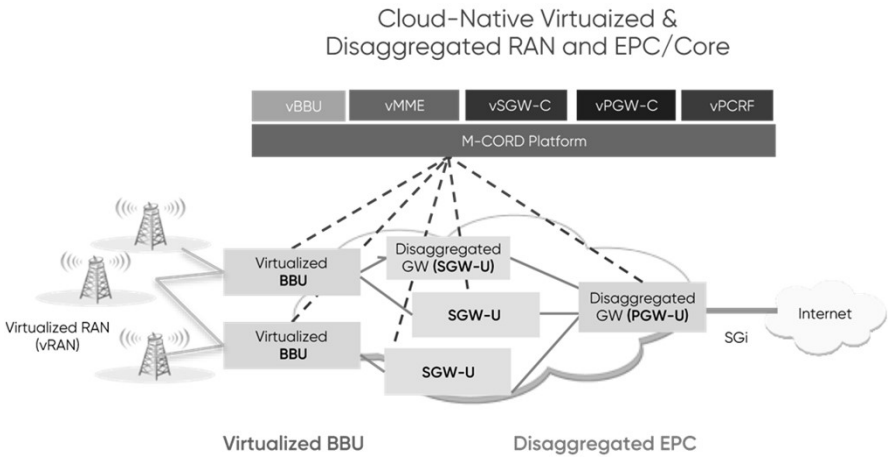


CORD Hardware Architecture

https://www.opennetworking.org/cord/

VII. 클라우드와 가상화 관리

❖ **M-CORD (Mobile CORD):**

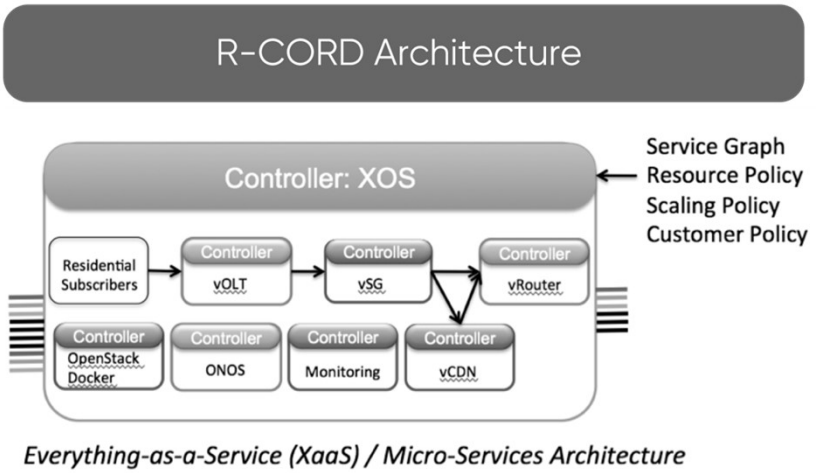


The Role of M-CORD in a Mobile Service Provider Infrastructure

https://www.bigswitch.com/products/big-cloud-fabric

VII. 클라우드와 가상화 관리

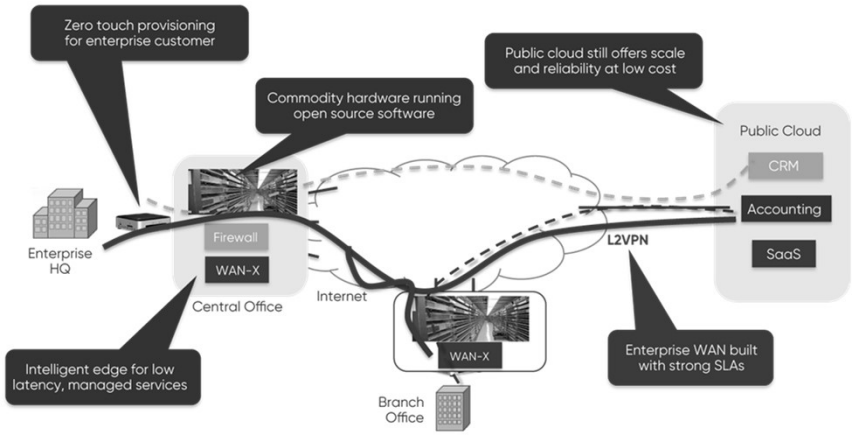
❖ **R-CORD (Residential CORD):**



https://www.opennetworking.org/r-cord/

VII. 클라우드와 가상화 관리

❖ **E-CORD (Enterprise CORD):**



https://www.opennetworking.org/e-cord/

171

VII. 클라우드와 가상화 관리

❖ **E-CORD (Enterprise CORD):**

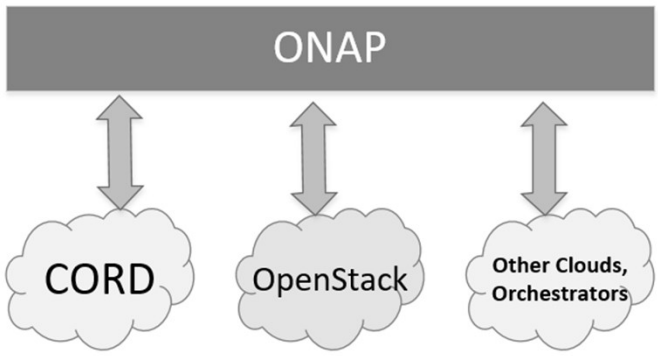
- **VPN**
- **Internet Access**
- **Firewall and border protection**
- **CDN (Content Delivery Network)**
- **Network core functions such as DNS, DHCP, etc.**
- **SD-WAN**
- **Traffic optimization and enhanced QoS**
- **Zero Touch Provisioning of commodity hardware at customer premises and sites**
- **Correctly measured monitoring services to deliver an outcome-based SLAs and KPIs**
- **A platform that enables creation and delivery of innovative services.**

https://www.bigswitch.com/products/big-cloud-fabric

172

VII. 클라우드와 가상화 관리

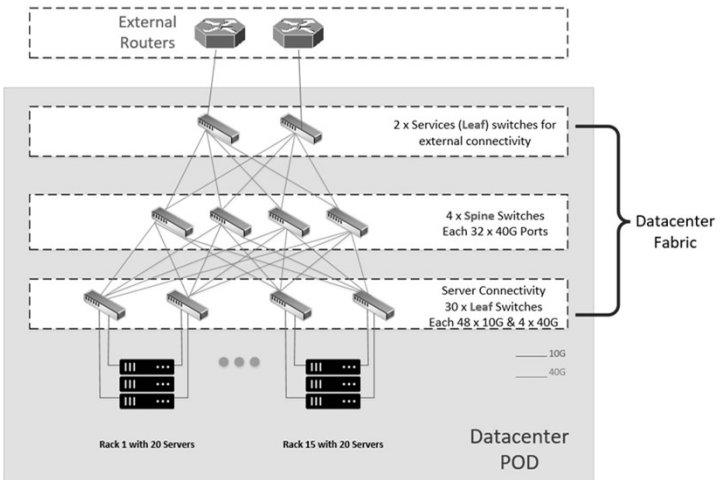
❖ **ONAP Communication with CORD:**



https://www.opennetworking.org/e-cord/

VII. 클라우드와 가상화 관리

❖ **Trellis:**

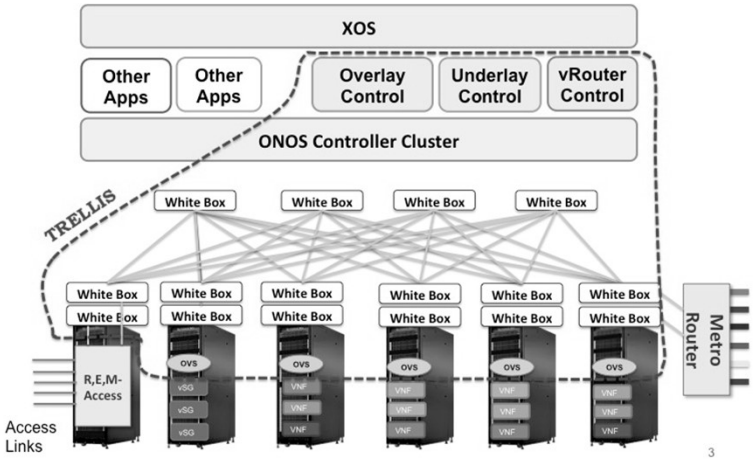


Leaf-Spine Design in a Datacenter

https://www.opennetworking.org/trellis/

VII. 클라우드와 가상화 관리

❖ **Trellis:**

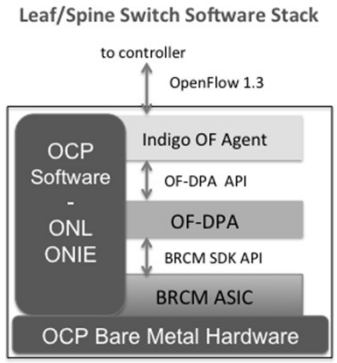


High-Level Architecture of Trellis in a Datacenter

https://wiki.opencord.org/display/CORD/Trellis%3A+CORD+Network+Infrastructure

VII. 클라우드와 가상화 관리

❖ **Trellis Architecture:**



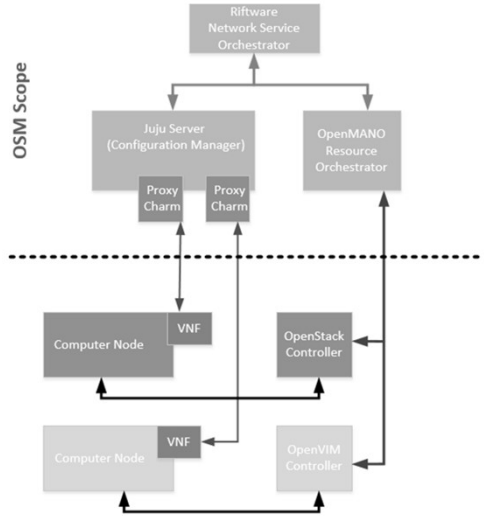
OCP: Open Compute Project
ONL: Open Network Linux
ONIE: Open Network Install Environment
BRCM: Broadcom Merchant Silicon ASICs
OF-DPA: OpenFlow Datapath Abstraction

Trellis Components on a White Box Switch

https://wiki.opencord.org/display/CORD/Trellis+Underlay+Fabric

VII. 클라우드와 가상화 관리

❖ **Open Source MANO Architecture:**



https://osm.etsi.org/

177

VII. 클라우드와 가상화 관리

❖ **Open Source MANO:**

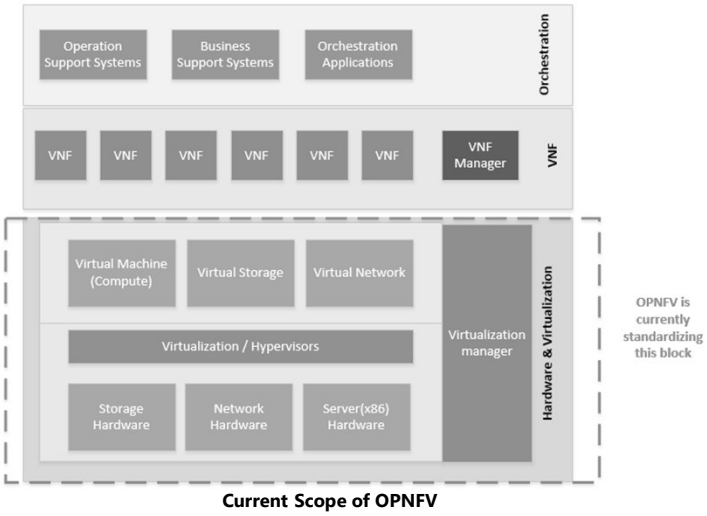
- **Service Orchestrator (SO)**
  SO is responsible for the end-to-end service orchestration and provisioning of VNFs and service chaining. SO manages the automation workflow for service deployment. OSM uses RIFT.io as orchestration engine.
- **Resource Orchestrator (RO)**
  RO is responsible to communicate with virtualization platforms such as OpenStack and VMware. RO provisions the NFV virtual workloads on virtualization platforms.
- **VNF Configuration and Abstraction (VCA)**
  VCA is responsible for the configuration of VNFs that are provisioned by the Resource Orchestrator. OSM uses Canonical's Juju Charms as an automation engine to apply the required configuration to the provisioned VNFs.

https://www.bigswitch.com/products/big-cloud-fabric

178

VII. 클라우드와 가상화 관리

❖ **Open Platform for NFV (OPNFV):**



Current Scope of OPNFV

https://www.opnfv.org/

JS Lab

VII. 클라우드와 가상화 관리

❖ **Open Security Controller (OSC):**

Open Security Controller - Quick Summary	
Name	Open Security Controller (OSC)
By	The Linux Foundation
Where it runs	On a Linux host
What it does	Provision virtual firewall, IPS, and other components. Communicate with SDN controllers to create service chaining and ensure traffic is routed via the provisioned security functions.
What it can do out-of-the-box	OSC has built-in capabilities to integrate with OpenStack and Kubernetes to provision firewalls and other virtual security services.

https://www.opensecuritycontroller.org/

JS Lab

VII. 클라우드와 가상화 관리

❖ **Open Security Controller (OSC):**

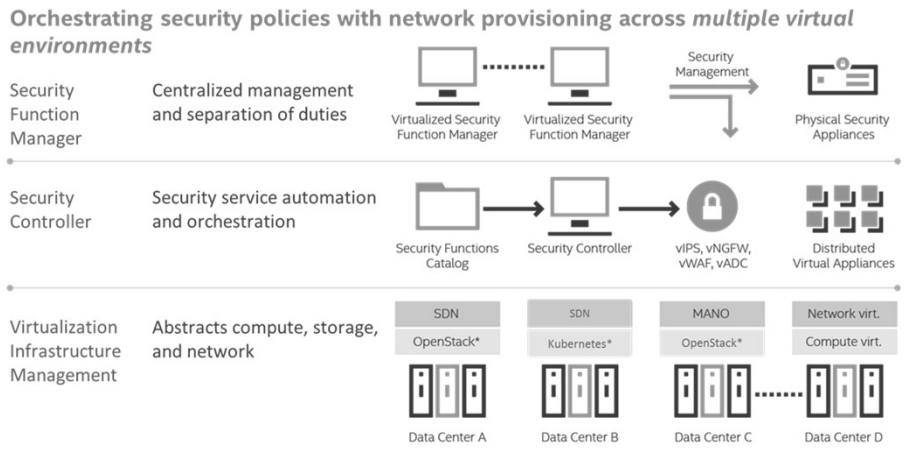
- **Call an API on virtual machine management platform (i.e. OpenStack) to provision a virtual machine using the base image of virtual firewall, along with the networks that it needs to connect to.**
- **Virtual machine management platform (i.e. OpenStack) provisions the virtual machine, reports back the Virtual Machine details, such as IP address, MAC address allocated for this VNF, etc.**
- **OSC calls the API to the network control layer (i.e OpenDaylight or Tungsten Fabric) to create a service chain that sends all traffic from VM A to this newly created VNF.**

https://www.opensecuritycontroller.org/

181

VII. 클라우드와 가상화 관리

❖ **OSC Architecture:**



Open Security Controller Conceptual Architecture

https://www.opensecuritycontroller.org/

182

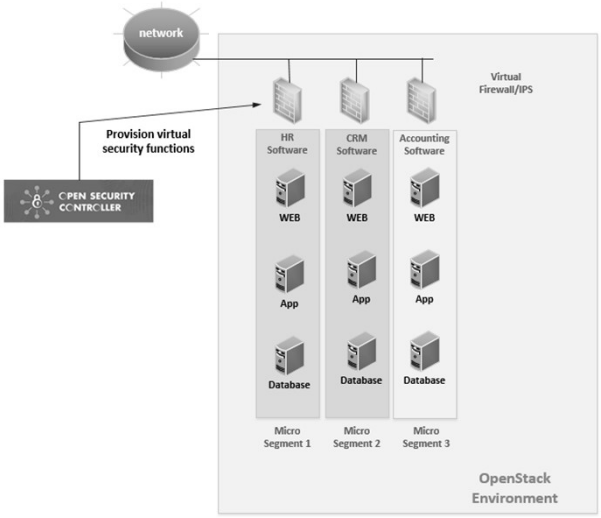
VII. 클라우드와 가상화 관리

❖ **OSC Interactions:**

- **Virtualization management systems**
  OSC includes a connector to communicate with virtualization management systems such as OpenStack and Kubernetes. This connector directly calls the Virtual Infrastructure Manager (VIM) APIs in order to provision virtual network security functions. OSC Virtualization Management plugin also subscribes to notification events from the VIM system in order to receive information and status related to provisioned virtual network security workloads (virtual machines or containers).
- **SDN controllers**
  OSC supports communication with multiple networking and SDN controllers. OSC has a built-in connector that works with multiple SDN controllers. OSC uses the SDN controller plugin to implement traffic redirection or Service Function Chaining (SFC) to send specific traffic to the newly created network security function.
- **Security Function Managers**
  OSC uses this connector to communicate with security function systems such as IPS manager, firewall manager, security policy manager, etc. Using this plugin, OSC will be able to call the Security Function Manager APIs to apply specific policy updates, device group membership settings, etc., to the newly created virtual network function

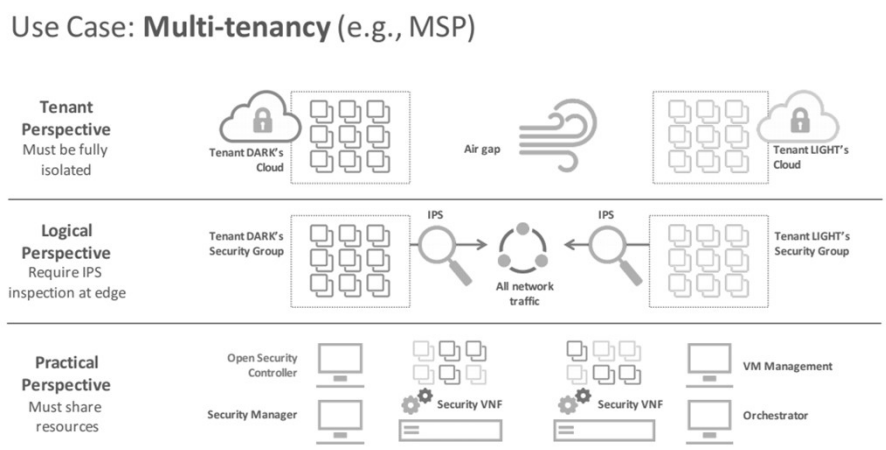
VII. 클라우드와 가상화 관리

❖ **OSC Use Case: Microsegmentation**



VII. 클라우드와 가상화 관리

OSC Use Case: Segmentation within a Multi-Tenant Environment

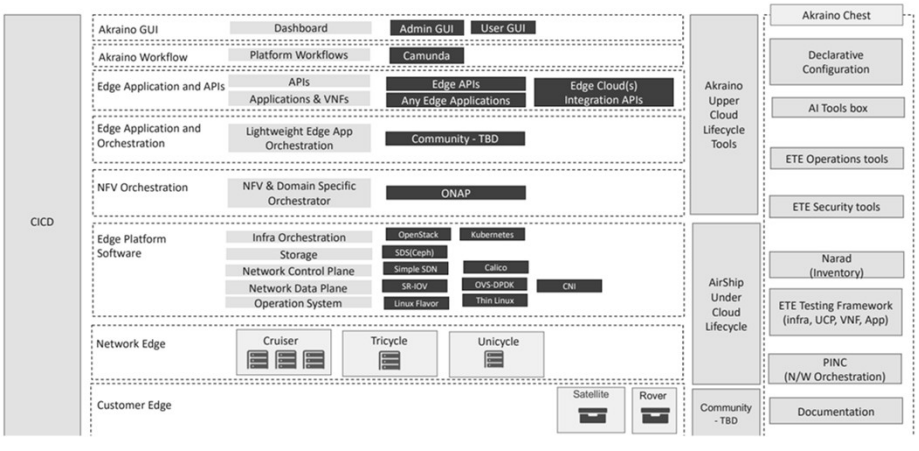


https://www.opensecuritycontroller.org/

JS Lab

VII. 클라우드와 가상화 관리

❖ **Akraino Edge Stack:**



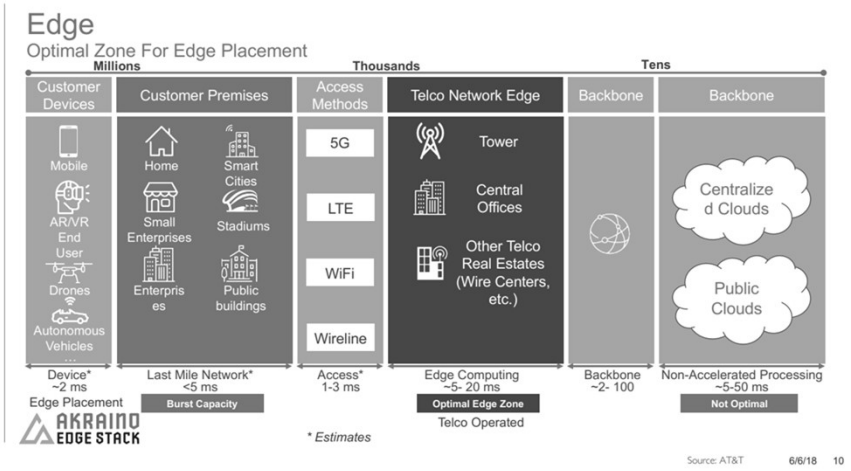
Akraino's Proposed Architecture

https://www.akraino.org/

JS Lab

VII. 클라우드와 가상화 관리

❖ **Akraino Edge Stack:**



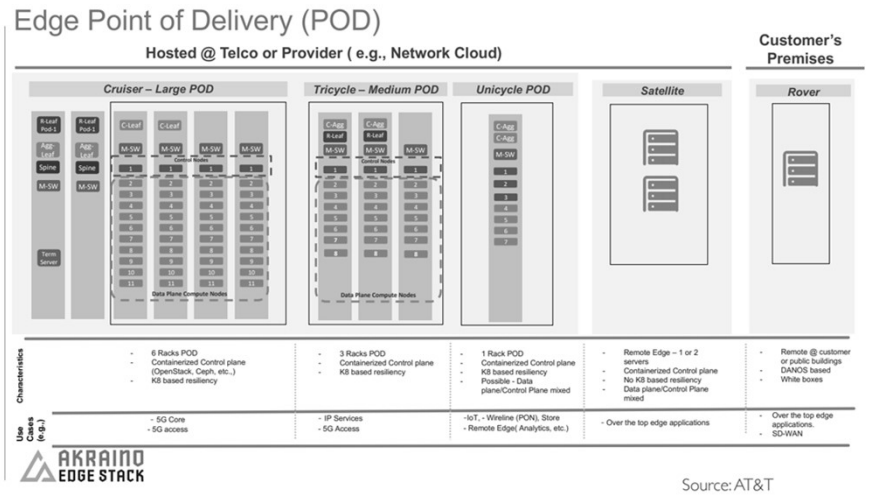
Optimal Zone for Edge Placement

https://www.akraino.org/

JS Lab

VII. 클라우드와 가상화 관리

❖ **Akraino Edge Stack:**



Edge Point of Delivery

https://www.akraino.org/

JS Lab

VIII.네트워크 가상화

- **Introduction to Network Virtualization**
- **Network Virtualization vs. Network Function Virtualization**
- **Vendor-Specific Virtualization: VMware NSX**
- **Overlay Networks**
- **OpenStack Networking**
- **Hardware Acceleration**
- **Containers and Networking**
- **Docker Networking for Containers**
- **Kubernetes** (K8s)
- **Service Mesh** (Istio)

https://www.linuxfoundation.org/projects/networking/

VIII. 네트워크 가상화

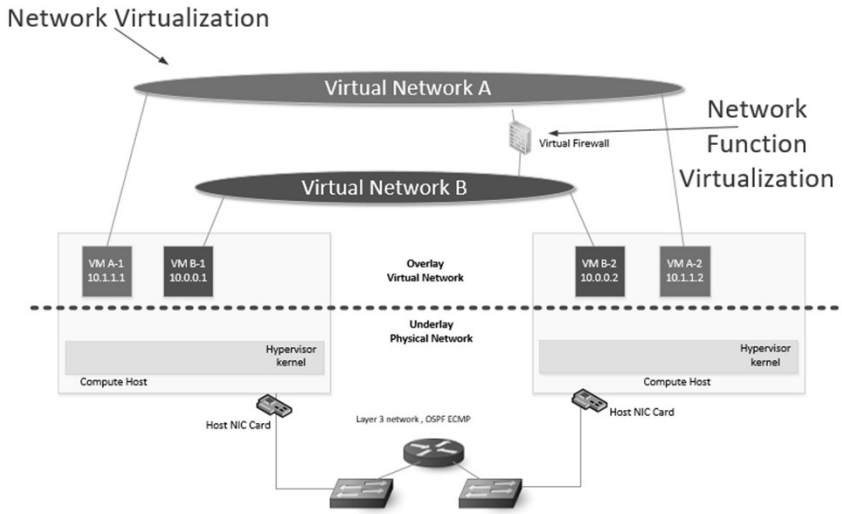
❖ **Introduction to Network Virtualization:**

Network virtualization is a key concept for both open source networking, as well as new cloud technologies. Virtualization technologies use network virtualization to allow communication between virtual machines or containers within a compute host, or across multiple compute hosts.
Network virtualization includes virtual networks that only exist within a host (such as Linux bridge, IO Visor, etc.), as well as technologies that allow communication between Linux bridges of multiple hosts (encapsulation and overlay networks).
Virtual networks that have been connecting the virtual machines within a host have existed for many years. Most of them use a host-based virtual bridge to connect the virtual interface of the virtual machines (a Linux bridge is a virtual Layer 2 switch). Virtual machines have been the main use case for Linux bridges for a long time. In recent years, with the introduction of containers, virtual switches started being used to connect containers, as well as virtual machines.
Container systems such as Docker use Linux bridge as their main method of connectivity.

191

VIII. 네트워크 가상화

❖ **Network Virtualization vs. Network Function Virtualization:**



192

VIII. 네트워크 가상화

❖ **Vendor-Specific Virtualization: VMware NSX:**

VMware NSX - Quick Summary	
Name	VMware NSX
By	VMware
Where it runs	Over a virtualized environment, in VMware vSphere
What it does	Creates and manages virtual networks, firewalls, load balancers, routers. Secures the East-West traffic by protecting VM-to-VM traffic at the host level. Provides security compliance and auditing. Provides a platform to implement microsegmentation using its ready-made firewall, load balancer, router and networking features. NSX has its own virtual firewall and virtual load balancers, which can be used for any workload.
What you can do out-of-the-box	You can deploy VMware NSX in your existing VMware vSphere environment (version should be supported). After installation, you will get a new section in your VMware vCenter to manage your network and security. You can create virtual networks, provision firewalls and load balancers, create virtual routers and peers with external networks, create traffic filtering between VMs and many other features out-of-the-box.

VIII. 네트워크 가상화

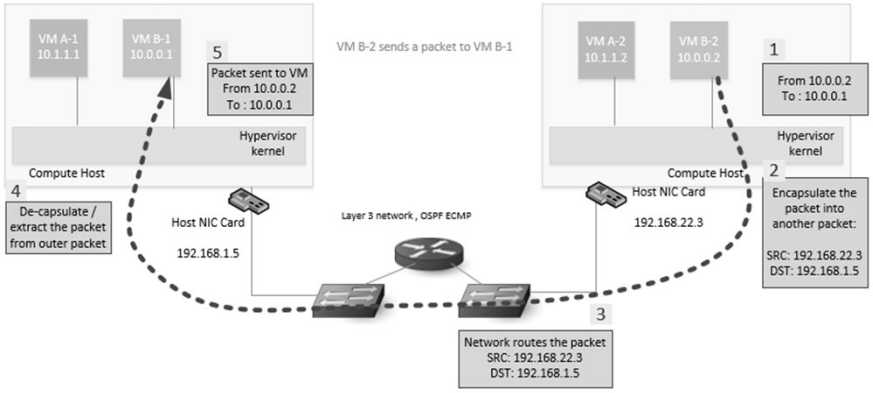
❖ **Vendor-Specific Virtualization: VMware NSX:**

❖ **VMware NSX enhances the VMware's distributed virtual switches and adds many networking and security features. VMware NSX provides the following virtual functions:**

- **Switching**
- **Routing**
- **Firewalling**
- **Load balancing**
- **VPN**
- **Access control**
- **Quality of Service management**

VIII. 네트워크 가상화

❖ **Vendor-Specific Virtualization: VMware NSX:**



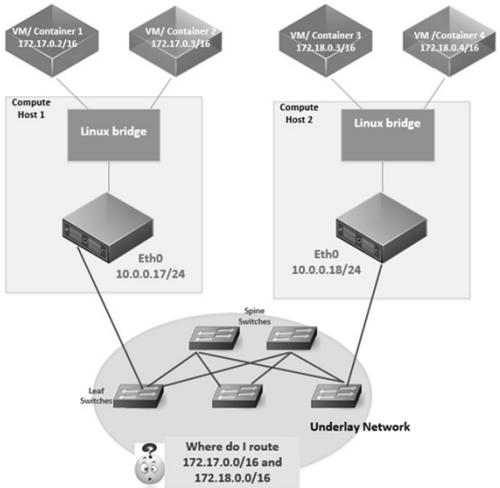
Basic Protocol-Independent Encapsulation and Overlay

https://www.vmware.com/products/nsx.html

JS Lab

VIII. 네트워크 가상화

❖ **Overlay Networks:**

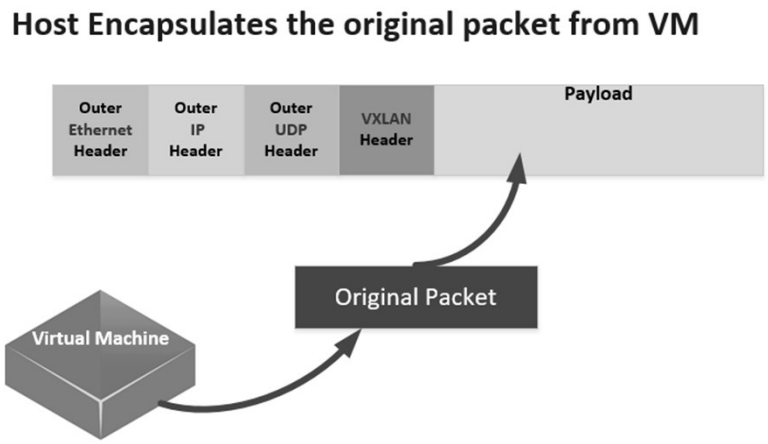


Underlay Network Unaware of IP Addresses and Virtual Networks Existing on Compute Hosts

JS Lab

VIII. 네트워크 가상화

❖ **Overlay Networks:**

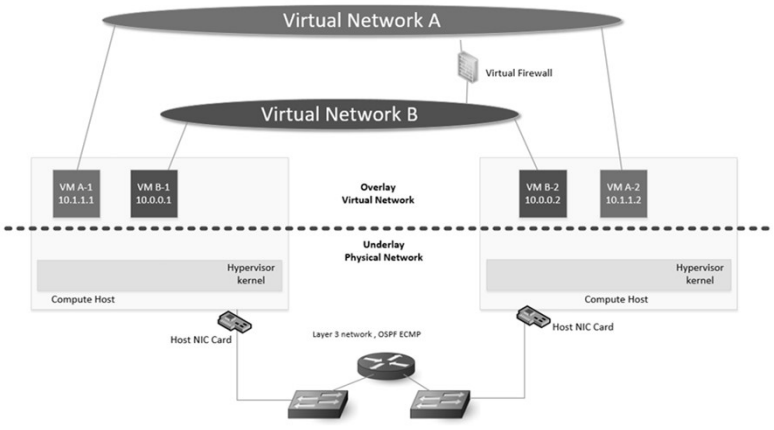


Host Encapsulates the Original Packet from the VM

197

VIII. 네트워크 가상화

❖ **Overlay Networks:**



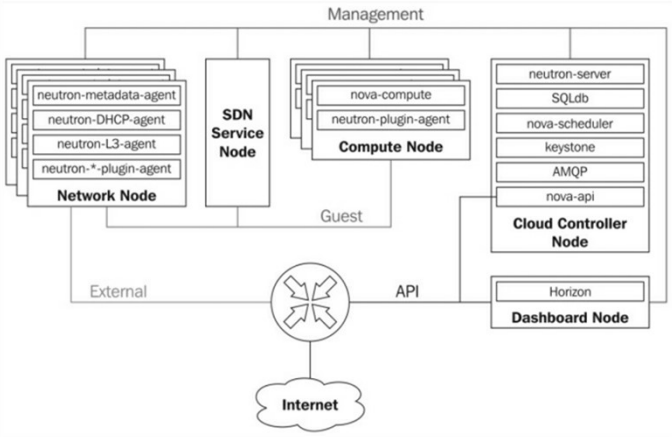
Overlay Network

198

VIII. 네트워크 가상화

❖ 오픈스택(OpenStack) 네트워크

- 네트워크를 분리 (관리/Guest/외부/API)
- VLAN 사용시 트렁크 모드 사용을 피하거나 네트워크를 물리적으로 나누는 것이 필요



Locati, Fabio Alessandro. OpenStack Cloud Security

VIII. 네트워크 가상화

❖ **Hardware Acceleration:**

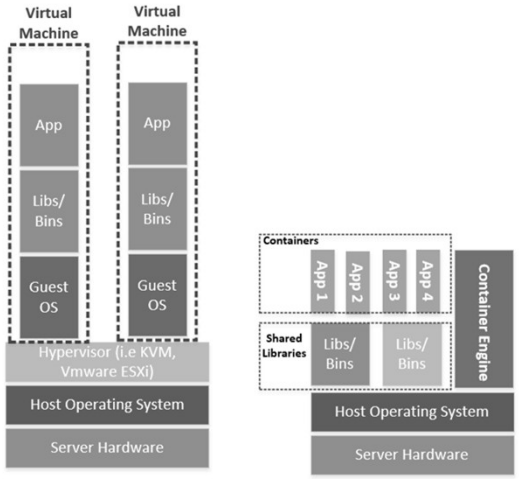
Encapsulation and decapsulation of traffic add additional work on the host's CPU to process each packet from and to virtual workloads. This process adds extra overhead to the system, reducing the host's CPU time which should be used for running applications.

The good news is that, these days, most NIC cards support VXLAN offloading, which means the work of encapsulation and decapsulation can be offloaded to the NIC card chipset, instead of the CPU.

When we talked about DPDK, NIC cards and SmartNICs, the process of traffic encapsulation and decapsulation can be hardware-accelerated. Most NIC cards have built-in capabilities to offload the VXLAN Encapsulation/Decapsulation works from the CPU, without requiring special configuration. If a NIC card doesn't support native VXLAN offloading, you can use DPDK to build an application to accelerate the VXLAN encapsulation and decapsulation process.

VIII. 네트워크 가상화

❖ **Containers and Networking:**



Virtual Machines VS Containers

201

VIII. 네트워크 가상화

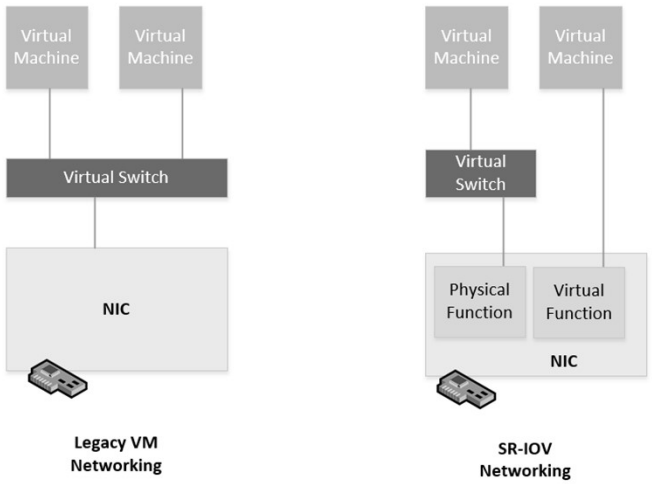
❖ **Containers and Networking:**

	Virtual Machines	Containers
Running Engine	Hypervisor	Container Engine
Workload's OS/Kernel	Each VM has its own OS and kernel.	All containers use the host machine's OS and kernel.
Density	Host machine needs to run multiple full operating systems.	Host machines run multiple containers using a common kernel and groups of common binaries and libraries. Less memory and CPU utilization than VMs. A host can run more containers compared to VMs. Higher density.
Networking	Relies on host virtual switches or direct NIC connectivity using SR-IOV* (Single Root Input Output Virtualization).	Relies on host virtual switches or direct NIC connectivity using SR-IOV.
Storage	Host presents a virtual block storage to the VM.	Container Engine manages the container storage.
Guests	Can be any OS (such as Linux, BSD, Windows, etc.).	Only the same as the host's kernel. You can still run an Ubuntu container on a CentOS host.

202

VIII. 네트워크 가상화

❖ **Containers and Networking:**

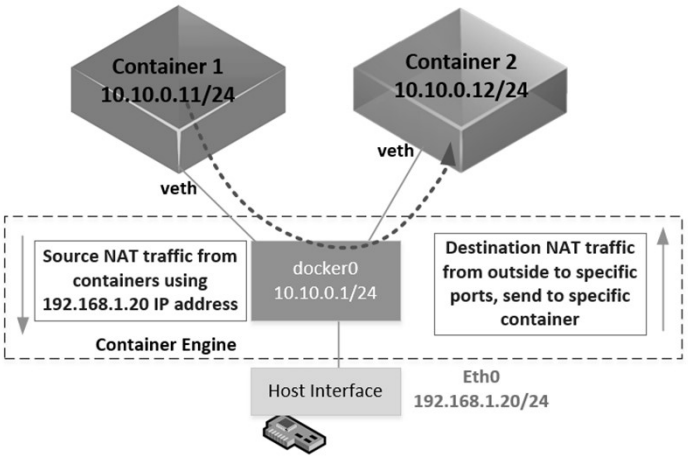


Legacy vs SR-IOV Networking

203

VIII. 네트워크 가상화

❖ **Docker Networking for Containers:**

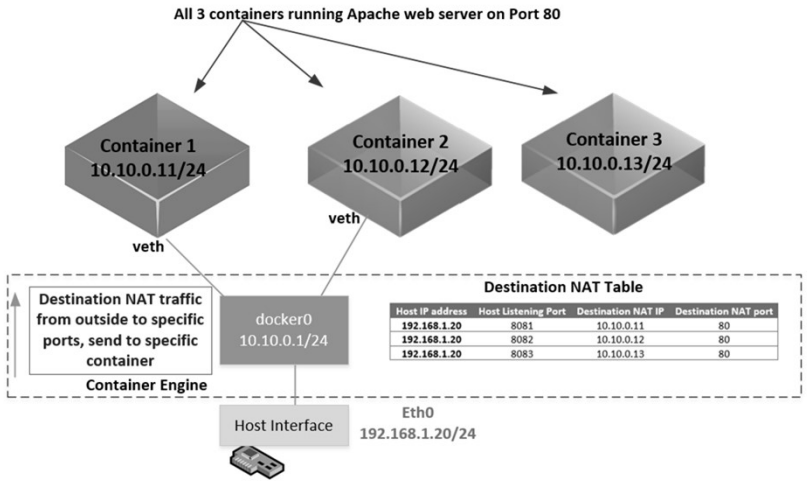


Docker Networking Function and NATing

204

VIII. 네트워크 가상화

❖ **Connecting to Containers from the Outside:**

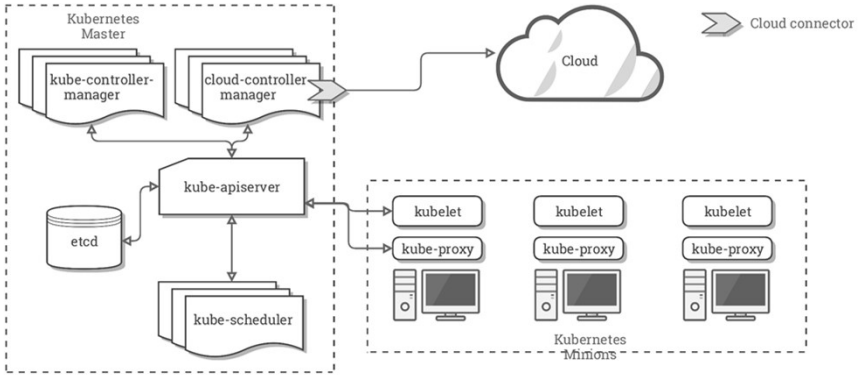


Destination NAT for Containers

205

VIII. 네트워크 가상화

❖ **Kubernetes:**



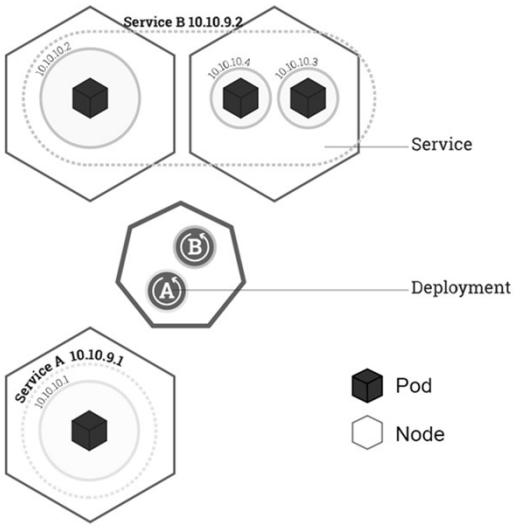
Kubernetes Architecture

https://kubernetes.io/

206

VIII. 네트워크 가상화

❖ **Kubernetes Services:**



https://kubernetes.io/

VIII. 네트워크 가상화

❖ **Kubernetes Pod: Istio-telemetry**(예)

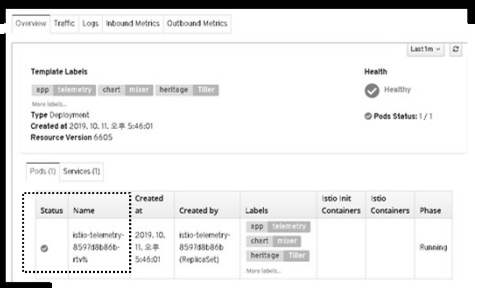
VIII. 네트워크 가상화

❖ **Kubernetes Pod: Istio-telemetry and Proxy @ Docker** (예)

- **docker ps**
- **docker exec CONTAINER ifconfig**

```
root@worker72:/home/jslab# docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED       STATUS       PORTS               NAMES
9ac429cc8e5a        f31df53bffed                   "/usr/local/bin/mixs…"   25 hours ago  Up 25 hours                      k8s_mixer_istio-telemetry-8597d8b86b-rtvfs_istio-system_8d5af400-7c2e-4cab-8fee-64050ac3de76_3
385fecb3572d        rancher/pause:3.1              "/pause"                 25 hours ago  Up 25 hours                      k8s_POD_istio-telemetry-8597d8b86b-rtvfs_istio-system_8d5af400-7c2e-4cab-8fee-64050ac3de76_0
4d87223a4af2        rancher/istio-proxyv2          "/usr/local/bin/pilo…"   25 hours ago  Up 25 hours
root@worker72:/home/jslab#
```

```
root@worker72:/home/jslab# docker exec 4d87223a4af2 ifconfig
eth0      Link encap:Ethernet  HWaddr b2:2d:11:40:1c:03
          inet addr:10.42.1.9  Bcast:0.0.0.0  Mask:255.255.255.255
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:174489 errors:0 dropped:0 overruns:0 frame:0
          TX packets:167359 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:20185982 (20.1 MB)  TX bytes:582220802 (582.2 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:30864 errors:0 dropped:0 overruns:0 frame:0
          TX packets:30864 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:551008609 (551.0 MB)  TX bytes:551008609 (551.0 MB)

root@worker72:/home/jslab#
```



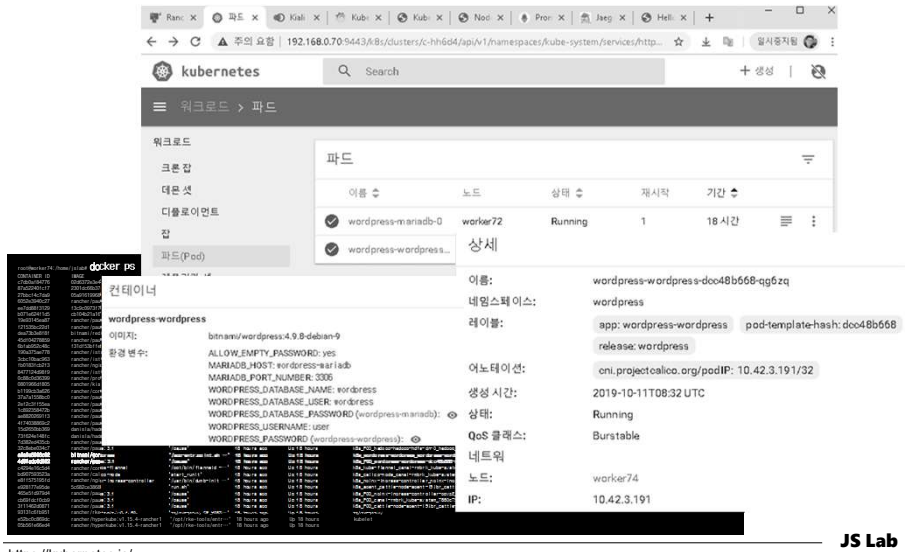
VIII. 네트워크 가상화

❖ **Kubernetes Pod: Istio-telemetry and Proxy @ iptables** (예)

```
root@worker72:/home/jslab# iptables -t nat -L -n

Chain KUBE-NODEPORTS (1 references)
target     prot opt source               destination
KUBE-MARK-MASQ  tcp  --  0.0.0.0/0            0.0.0.0/0            /* kubeflow/ambassador:ambassador */ tcp dpt:30138
KUBE-SVC-STYH4L3MNMUUEPBQ  tcp  --  0.0.0.0/0            0.0.0.0/0            /* kubeflow/ambassador:ambassador */ tcp dpt:30138
KUBE-MARK-MASQ  tcp  --  0.0.0.0/0            0.0.0.0/0            /* docker-registry/docker-registry:registry */ tcp dpt:31486
KUBE-SVC-5PSZEDW4UNLDD4A6  tcp  --  0.0.0.0/0            0.0.0.0/0            /* docker-registry/docker-registry:registry */ tcp dpt:31486
KUBE-MARK-MASQ  tcp  --  0.0.0.0/0            0.0.0.0/0            /* wordpress/wordpress-wordpress:http */ tcp dpt:31904
KUBE-SVC-2FUFTE55KQIFOMSM  tcp  --  0.0.0.0/0            0.0.0.0/0            /* wordpress/wordpress-wordpress:http */ tcp dpt:31904
KUBE-MARK-MASQ  tcp  --  0.0.0.0/0            0.0.0.0/0            /* wordpress/wordpress-wordpress:https */ tcp dpt:30526
KUBE-SVC-2YER72GNBESPQKXP  tcp  --  0.0.0.0/0            0.0.0.0/0            /* wordpress/wordpress-wordpress:https */ tcp dpt:30526

Chain KUBE-SEP-G4VQDTVXSGZWTZOE (1 references)
target     prot opt source               destination
KUBE-MARK-MASQ  all  --  10.42.1.9            0.0.0.0/0
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp to:10.42.1.9:9091

Chain KUBE-SEP-UVXKT3JO7F7QKUSS (1 references)
target     prot opt source               destination
KUBE-MARK-MASQ  all  --  10.42.1.9            0.0.0.0/0
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp to:10.42.1.9:42422

Chain KUBE-SEP-WI4Q7OH56TS2COK3 (1 references)
target     prot opt source               destination
KUBE-MARK-MASQ  all  --  10.42.1.9            0.0.0.0/0
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp to:10.42.1.9:15014

Chain KUBE-SEP-Z6QXVYMQAAERW6O5 (1 references)
target     prot opt source               destination
KUBE-MARK-MASQ  all  --  10.42.1.9            0.0.0.0/0
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp to:10.42.1.9:15004

Chain KUBE-SERVICES (2 references)
KUBE-MARK-MASQ  tcp  -- !10.42.0.0/16        10.43.7.130          /* istio-system/istio-telemetry:grpc-mixer-mtls cluster IP */ tcp dpt:15004
KUBE-SVC-POFVSRWRNLJ5KKAQ  tcp  --  0.0.0.0/0            10.43.7.130          /* istio-system/istio-telemetry:grpc-mixer-mtls cluster IP */ tcp dpt:15004
KUBE-SVC-TUZA6CVEP5VUF2XG  tcp  --  0.0.0.0/0            10.43.24.19          /* istio-system/istio-policy:grpc-mixer-mtls cluster IP */ tcp dpt:15004
KUBE-SVC-U2XTOAGDXOJP3ONI  tcp  --  0.0.0.0/0            10.43.24.19          /* istio-system/istio-policy:grpc-mixer cluster IP */ tcp dpt:9091
KUBE-MARK-MASQ  tcp  -- !10.42.0.0/16        10.43.7.130          /* istio-system/istio-telemetry:grpc-mixer cluster IP */ tcp dpt:9091
KUBE-SVC-LTQKVKL3D46WIGR3  tcp  --  0.0.0.0/0            10.43.7.130          /* istio-system/istio-telemetry:grpc-mixer cluster IP */ tcp dpt:9091
```

VIII. 네트워크 가상화

❖ **Kubernetes Pod: Wordpress @ K8s Dashboard**



https://kubernetes.io/

VIII. 네트워크 가상화

❖ **Kubernetes Pod: Wordpress @ K8s Dashboard**

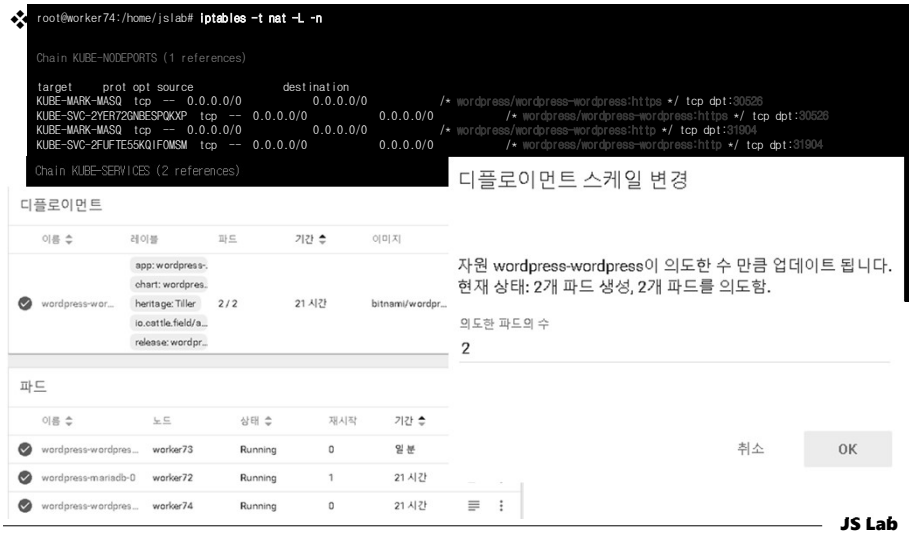
- **Pod Information @ iptables**
- **docker ps**

```
root@worker74:/home/jslab# iptables -t nat -L -n

Chain KUBE-NODEPORTS (1 references)
target     prot opt source               destination
KUBE-MARK-MASQ  tcp  --  0.0.0.0/0            0.0.0.0/0            /* wordpress/wordpress-wordpress:https */ tcp dpt:30526
KUBE-SVC-2YER72GNBESPQKXP  tcp  --  0.0.0.0/0            0.0.0.0/0            /* wordpress/wordpress-wordpress:https */ tcp dpt:30526
KUBE-MARK-MASQ  tcp  --  0.0.0.0/0            0.0.0.0/0            /* wordpress/wordpress-wordpress:http */ tcp dpt:31904
KUBE-SVC-2FUFTE55KQIFOMSM  tcp  --  0.0.0.0/0            0.0.0.0/0            /* wordpress/wordpress-wordpress:http */ tcp dpt:31904

Chain KUBE-SERVICES (2 references)
target     prot opt source               destination
KUBE-MARK-MASQ  tcp  -- !10.42.0.0/16         10.43.0.91           /* wordpress/wordpress-wordpress:https cluster IP */ tcp dpt:443
KUBE-SVC-2YER72GNBESPQKXP  tcp  --  0.0.0.0/0            10.43.0.91           /* wordpress/wordpress-wordpress:https cluster IP */ tcp dpt:443
KUBE-MARK-MASQ  tcp  -- !10.42.0.0/16         10.43.205.38         /* wordpress/wordpress-mariadb:mysql cluster IP */ tcp dpt:3306
KUBE-SVC-DEV3U2PZRMWXYVGH  tcp  --  0.0.0.0/0            10.43.205.38         /* wordpress/wordpress-mariadb:mysql cluster IP */ tcp dpt:3306
KUBE-MARK-MASQ  tcp  -- !10.42.0.0/16         10.43.0.91           /* wordpress/wordpress-wordpress:http cluster IP */ tcp dpt:80
KUBE-SVC-2FUFTE55KQIFOMSM  tcp  --  0.0.0.0/0            10.43.0.91           /* wordpress/wordpress-wordpress:http cluster IP */ tcp dpt:80
```

```
root@worker74:/home/jslab# docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS               NAMES
a3a3a5386c92        bitnami/wordpress      "/app-entrypoint.sh …"   18 hours ago        Up 18 hours                             k8s_wordpress-wordpress_wordpress-wordpress-dcc48b668-qg6zq_wordpress_8d0edcc3-f572-4230-b148-af24c4fe8ae8_0
4d8fadc6d822        rancher/pause:3.1      "/pause"                 18 hours ago        Up 18 hours                             k8s_POD_wordpress-wordpress-dcc48b668-qg6zq_wordpress_8d0edcc3-f572-4230-b148-af24c4fe8ae8_0
```

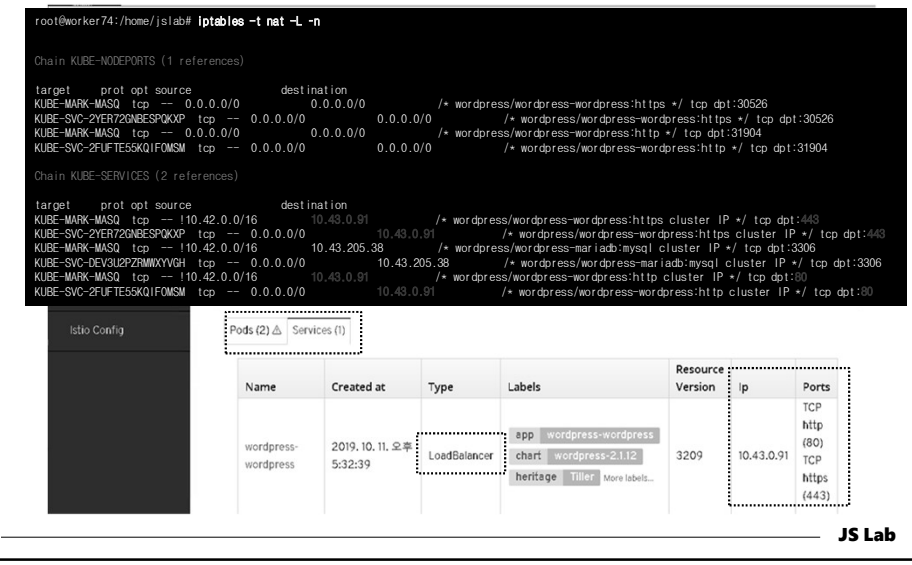
VIII. 네트워크 가상화

❖ Kubernetes Pod: Scaleout for Wordpress @ K8s Dashboard



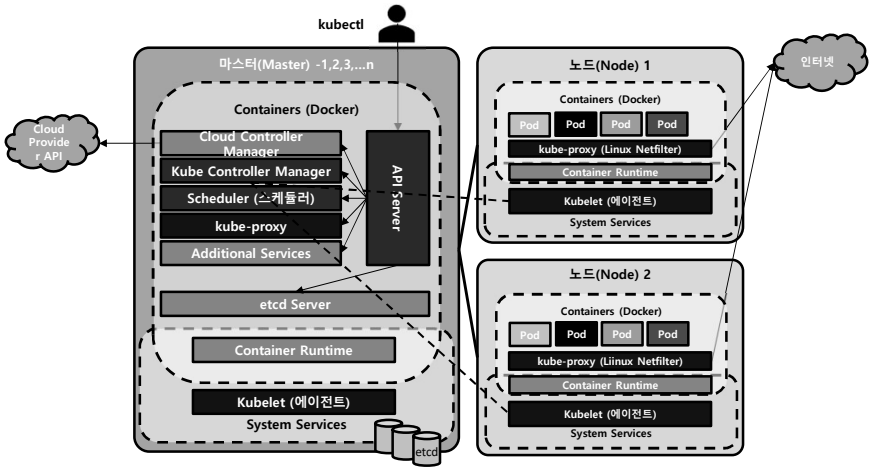
VIII. 네트워크 가상화

❖ Kubernetes Pod: Wordpress w/ LB @ iptables



VIII. 네트워크 가상화

❖ **Kubernetes Architecture:**



https://kubernetes.io/

JS Lab

215

VIII. 네트워크 가상화

❖ **Kubernetes Networking:**

❖ **Master Server Components:**

- **kube-apiserver**
  Provides frontend and controllable APIs to control the Kubernetes environment via an orchestration or management platform.
- **etcd**
  Kubernetes data store.
- **kube-scheduler**
  Responsible for allocating worker nodes for newly created pods with no nodes assigned.
- **kube-controller-manager**
  Includes four controllers as:
  - **Node Controller**: Responsible for finding out when a node goes down
  - **Replication Controller**: Responsible for replication in the system
  - **Endpoints Controller**: Populates the endpoints objects (i.e, joins services & pods)
  - **Service Account & Token Controllers**: Create default accounts and API access tokens for new namespaces
- **cloud-controller-manager**
  Responsible to run controllers that interact with the underlying cloud providers, such as public cloud (AWS, Azure) or on-prem cloud (such as OpenStack).

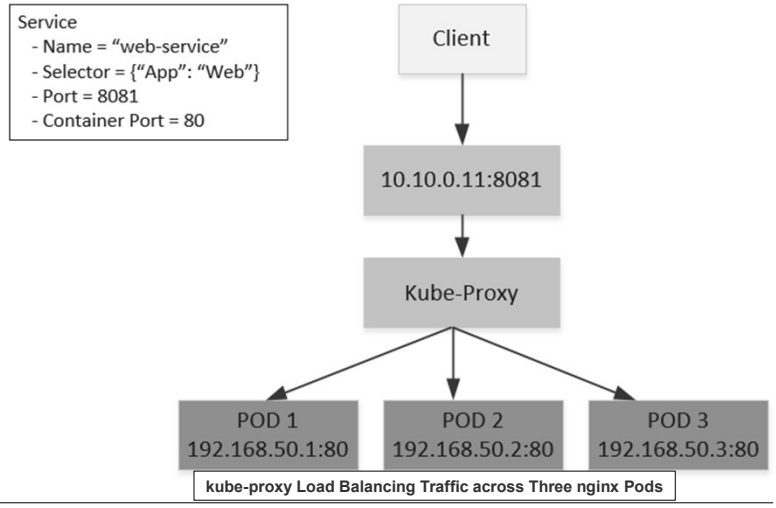
https://kubernetes.io/

JS Lab

216

VIII. 네트워크 가상화

❖ **Kubernetes Networking:**



https://kubernetes.io/

JS Lab

VIII. 네트워크 가상화

❖ **Kubernetes Networking:**

Kubernetes is compatible with the following networking solutions to create networking clusters:

- **Cisco ACI**
- **Big Switch Big Cloud Fabric**
- **Tungsten Fabric (OpenContrail)**
- **VMware NSX-T**
- **OpenVSwitch (OVS)**
- **Project Calico**

https://kubernetes.io/

JS Lab

VIII. 네트워크 가상화

❖ **Service Mesh (서비스 메시):**

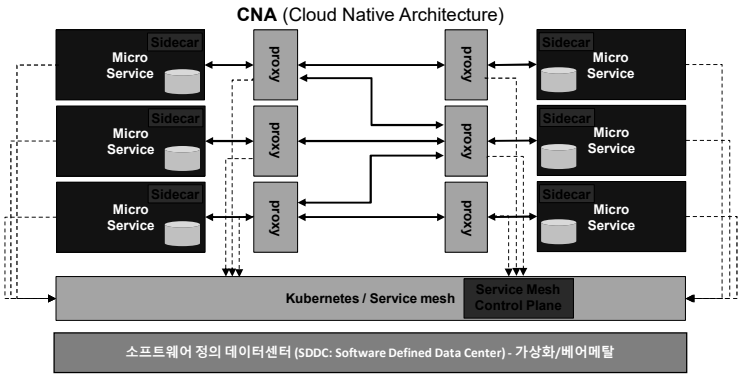
분산 시스템 및 클라우드 네이티브 애플리케이션에 주로 사용되는 서비스 간 통신에 중점을 둔 인프라 계층

- 서비스 또는 마이크로 서비스의 모음으로 컨테이너형 애플리케이션이 구축되면 서비스 메시가 형성
- 서비스간 연결하고 서비스의 상호 작용을 관리하기 위해 IP 주소와 포트 위에 계층을 만들고 로드 밸런싱, 모니터링, 서비스 간 인증 등을 제공
- 예: Istio, linkerd

VIII. 네트워크 가상화

❖ 서비스 메쉬 관리 체계

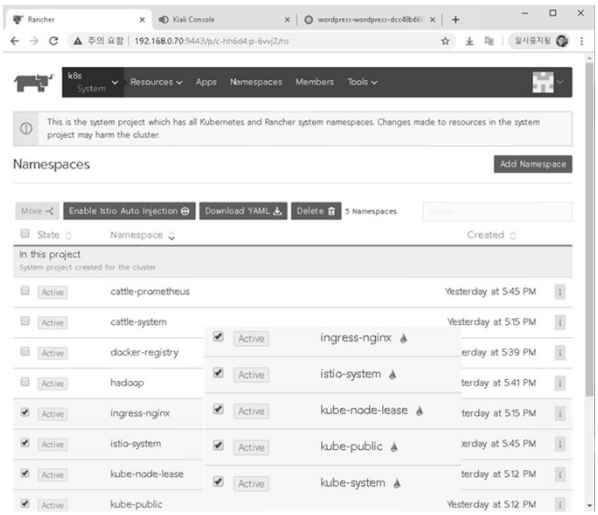
- Sidecar Design Pattern: 라우터 내장 CB, LB, SD 내장
- 오픈소스 CNCF의 'Istio'는 정책 강화 Telemetry 제공



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

VIII. 네트워크 가상화

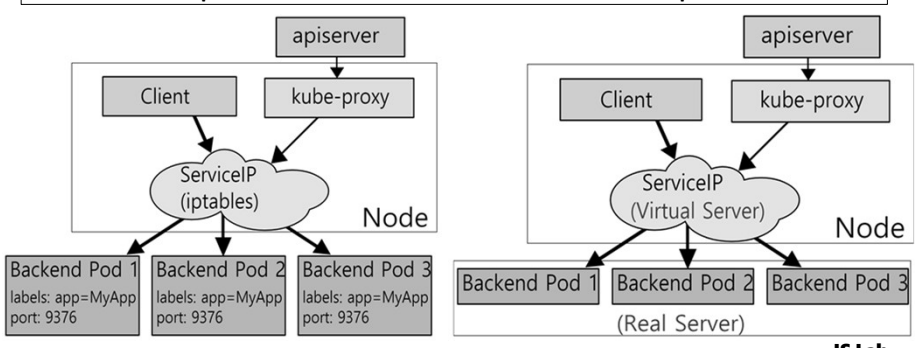
❖ 서비스 메시 관리 체계



221

VIII. 네트워크 가상화

❖ **IP Virtual Server** (IPVS)
❖ 리눅스 기반에 설치되는 서버로 **L4** 기능을 대체
❖ **ipvs(IP Virtual Server)** 모드는 리눅스 커널에 있는 **L4** 로드밸런싱 기술로 **Netfilter**에 포함
❖ **ipvs**는 커널스페이스에서 작동하고 데이터 구조를 해시테이블로 저장해서 가지고 있기 때문에 **iptables**보다 빠르고 좋은 성능



출처: https://arisu1000.tistory.com/27839

222

I. 오픈소스 네트워킹 개요
II. 오픈소스와 SDN Landscape
III. 소프드웨어/하드웨어 분리
IV. IO 추상화와 Data Path
V. **NOS** (Network Operating systems)
VI. 네트워크 제어 (Network Control)
VII. 클라우드와 가상화 관리
VIII. 네트워크 가상화
IX. **NFV** (Network Function Virtualization)
X. 네트워크 자동화
XI. 네트워크 데이터 분석
XII. Use Case
❖ 실습교재 (별도)

IX. **NFV**(Network Function Virtualization)

- **Introduction to Network Function Virtualization**
- **Virtual Firewalls**
- **pfSense Open Source Virtual Firewall**
- **Snort Open Source Virtual IPS/IDS**
- **Virtual Load Balancers**
- **Katran Open Source Load Balancer**
- **HAproxy Open Source Virtual Load Balancer**
- **Virtual Routers**
- **VyOS Open Source Virtual Router/Firewall**
- **Service Chaining**
- **uCPE** (Universal Customer Premises Equipment)

https://www.linuxfoundation.org/projects/networking/

IX. NFV(Network Function Virtualization)

❖ **Introduction to Network Function Virtualization:**

❖ **Some important NFV use cases are:**

- **Service Providers**
  - ✓ At the core, virtualizing routers
  - ✓ At the edge, virtualizing CPE (Customer Premises Equipment)
- **Enterprises**
  - ✓ Decentralizing network functions, such as firewall and load balancer
  - ✓ Building micro-segments and microservices
- **Cloud and Datacenter Providers**
  - ✓ Building a virtualized infrastructure for tenants
  - ✓ Decentralizing network functions, such as firewall and load balancer.

IX. NFV(Network Function Virtualization)

❖ **Introduction to Network Function Virtualization:**

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

IX. NFV(Network Function Virtualization)

❖ **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

IX. NFV(Network Function Virtualization)

❖ **pfSense Open Source Virtual Firewall:**

pfSense - Quick Summary	
Name	pfSense
By	Rubicon Communications, LLC (Netgate)
Where it runs	On a dedicated hardware appliance or on a virtual machine
What it does	pfSense is a stateful firewall with industry standard capabilities and features
Features	Firewalling, logging, Layer 2 transparent firewalling, state table control, NAT, high availability clustering, multi-WAN load balancing, server load balancing, IP Sec VPN, SSL VPN, PPPOE Server, reporting and graphs, captive portal, DHCP server
What it can do out-of-the-box	You can install pfSense on a hypervisor, assign virtual interfaces, and start using it as a firewall. pfSense can be used as a virtual firewall in a microsegmentation environment, or can be used as CPE, or for NAT configuration.

IX. NFV(Network Function Virtualization)

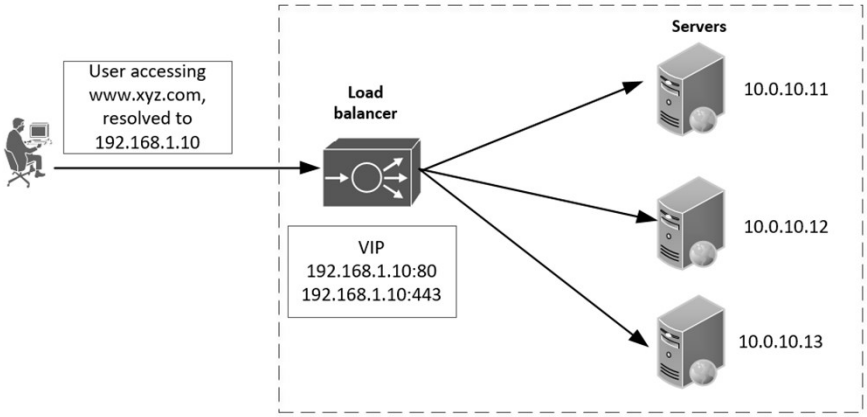
❖ **Snort Open Source Virtual IPS/IDS:**

Snort - Quick Summary	
Name	Snort
By	Currently hosted and developed by Cisco
Where it runs	On a dedicated hardware appliance or on a virtual machine
What it does	Snort is a network Intrusion Detection/Prevention System
Features	Traffic logging, detecting and matching packet header information (L2-L7), finds patterns and executes actions such as Alert, Block, Replace, etc. Ha s flexible rules and policies
What it can do out-of-the-box	You can install Snort on a virtual machine and have it connected to monit or and check the traffic of a network segment or even the traffic that is go ing to a specific host. Snort comes with a predefined attack signature data base; you can register to receive the updated attack signature database re gularly, which is a paid subscription service.

229

IX. NFV(Network Function Virtualization)

❖ **Virtual Load Balancers:**



230

IX. NFV(Network Function Virtualization)

❖ **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

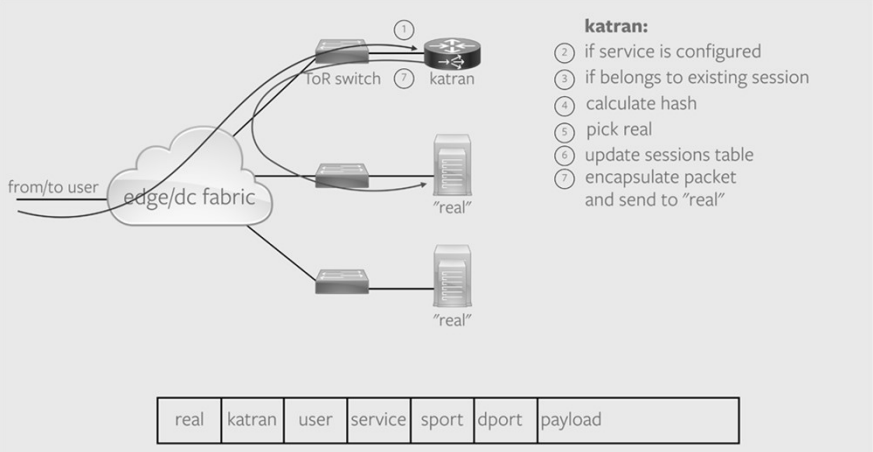
IX. NFV(Network Function Virtualization)

❖ **Katran Open Source Load Balancer:**

Katran - Quick Summary	
Name	Katran
By	Facebook
Where it runs	On dedicated virtual machines
What it does	Hy performance load balancing
Features	Open source, fast (especially with XDP in the driver mode), performance scales linearly with a number of NIC RX queu es, RSS (Received Side Scaling) friendly encapsulation.
What it can do out-of-the-box	You can use it for load balancing in high volume environme nts

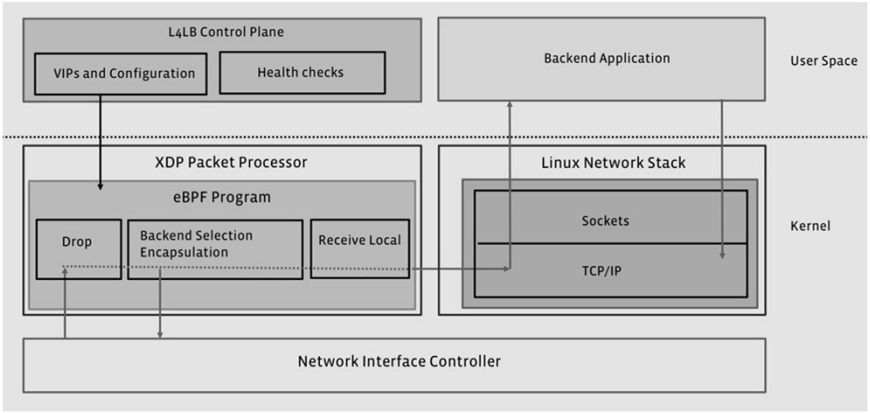
IX. NFV(Network Function Virtualization)

❖ **Katran Open Source Load Balancer:**



IX. NFV(Network Function Virtualization)

❖ **Katran Open Source Load Balancer:**



Katran Architecture

IX. NFV(Network Function Virtualization)

❖ **HAproxy Open Source Virtual Load Balancer:**

HAProxy - Quick Summary	
Name	HAProxy
By	HAPROXY Community
Where it runs	On a host
What it does	High performance L4-L7 load balancing
Features	L4-L7 load balancing, SSL
What it can do out-of-the-box	High performance load balancing, used in high profile websites such as Git Hub, Vimeo, Stack Overflow, etc.

IX. NFV(Network Function Virtualization)

❖ **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

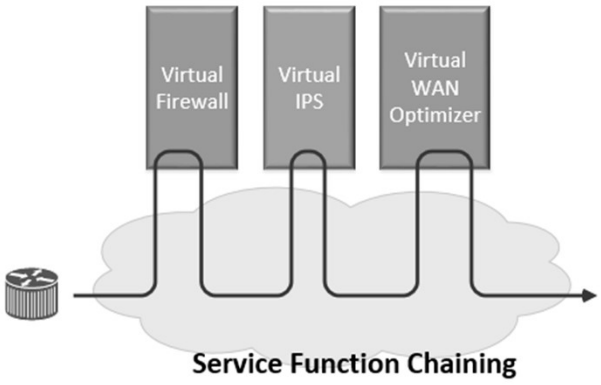
IX. NFV(Network Function Virtualization)

❖ **VyOS Open Source Virtual Router/Firewall:**

VyOS - Quick Summary	
Name	VyOS
By	Open Source
Where it runs	As a virtual appliance or on a x86 server
What it does	Routing and firewalling
Features	Layer 2, VLANs, 802.1q, QinQ, Layer 3, BGP, OSPF, RIP, PBR, ECMP, zone-based firewalling, tunneling, PPPOE, GRE, L2TP, VXLAN, IPSec VPN, SSL VPN, NAT, DHCP server, VRRP, sFlow, web proxy, QoS and traffic shaping. Uses a CLI for configuration without GUI
What it can do out-of-the-box	You can just load VyOS on a virtual machine and use it as a router to connect to an ISP or route between networks, or use it as VPN server or a firewall within your network.

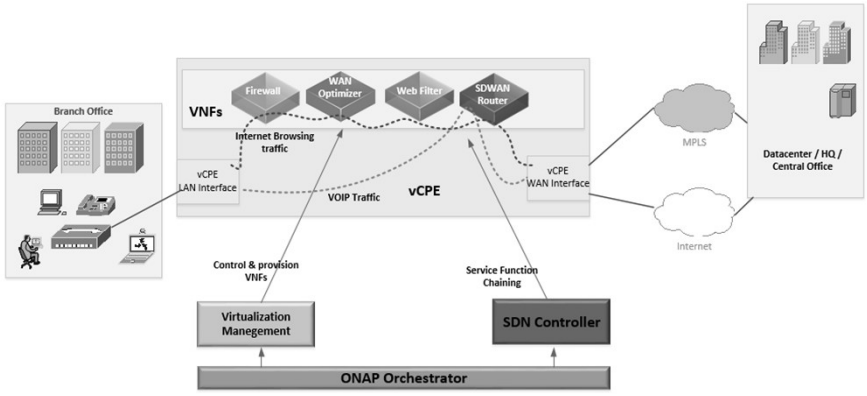
IX. NFV(Network Function Virtualization)

❖ **Service Chaining:**



IX. NFV(Network Function Virtualization)

❖ **Service Chaining:**

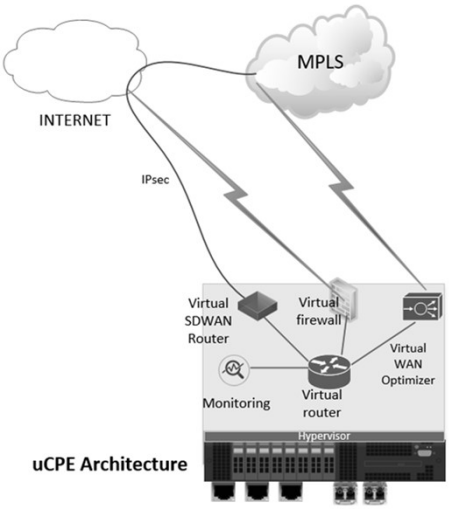


NFV and Service Chaining in a Service Provider Network

239

IX. NFV(Network Function Virtualization)

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



uCPE Architecture

240

IX. NFV(Network Function Virtualization)

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

•Virtual routers

These are standard packet-forwarding systems. They can route and run routing protocols and other features, such as NAT, Policy-Based Routing, and so on.

•Virtual firewalls

These are standard stateful or stateless firewalls with L3-L7 filtering capabilities. They may be equipped with deep packet inspection engines to provide features such as IDS and IPS.

•Virtual load balancers

L4 to L7 load balancers with capability of hosting virtual IPs (VIP) and forwarding (and NAT) the traffic to real servers. They may be equipped with Web Application Firewall (WAF) features.

•Virtual WAN optimizers

These include caching, TCP optimization, and protocol acceleration.

•SD-WAN routers

They are used to logically bound multiple WAN and Internet connections and build VPN tunnels back to the SD-WAN head-end units in data centers. They are used for intelligent link measurement and application-based routing.

X. 네트워크 자동화

- **Introduction to Network Automation**
- **Ansible**
- **Puppet**
- **Chef**
- **Python**
- **Using Netmiko to Connect to a Networking Device**
- **NETCONF**

Opensource Networking
james@jslab.kr

X. 네트워크 자동화

❖ **Introduction to Network Automation:**

Next, we will explore the network automation tools that can help you build automated workflows to execute tasks in a network. Network automation works on both next-generation SDN and legacy networks. Even if you have an aging network equipment, you are still able to use automation tools to make faster and reliable changes.

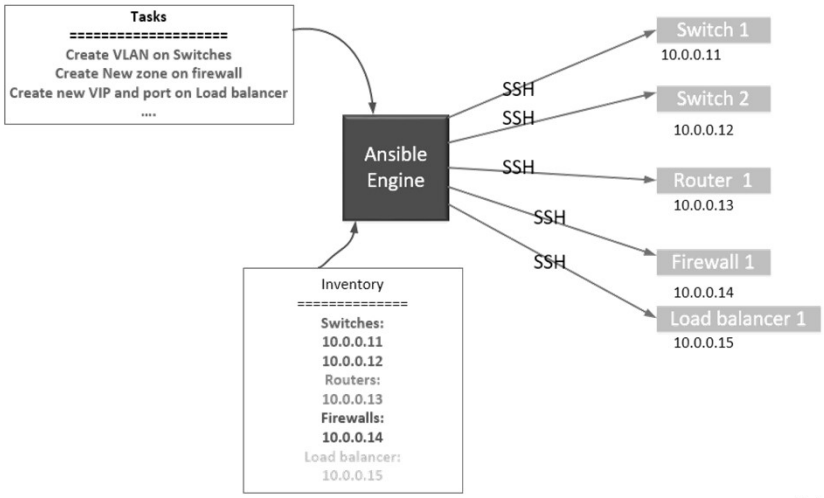
Automation tools that we are exploring in this chapter are:

- Ansible
- Puppet
- Chef.

Opensource Networking
james@jslab.kr

X. 네트워크 자동화

❖ **Ansible:**



245

X. 네트워크 자동화

❖ **Ansible:**

Ansible - Quick Summary	
Name	Ansible
By	Red Hat
Where it runs	On a workstation or a server
What it does	You can build automation playbooks to execute repeatable tasks on multiple devices
Features	Supports SSH/telnet access to networking devices. Ansible has multiple ready-made plugins for networking products, such as Cisco IOS, Arista, F5, Juniper, Cumulus, etc.
What it can do out-of-the-box	You can simply create Ansible scripts to tell Ansible to execute specific tasks on your equipment.

246

X. 네트워크 자동화

❖ **Puppet:**

Puppet - Quick Summary	
Name	Puppet
By	PuppetLabs
Where it runs	On a server. Requires agents to be installed on all managed devices (except Cisco IOS devices)
What it does	Manages the configuration of network devices and other servers
What it can do out-of-the-box	You can use Puppet to manage your Cisco IOS-based catalyst switches without the need of agents (as of June 2018). Puppet also supports other products, from Juniper, Cumulus Networks, OpenSwitch, Cisco ACI, etc.

247

X. 네트워크 자동화

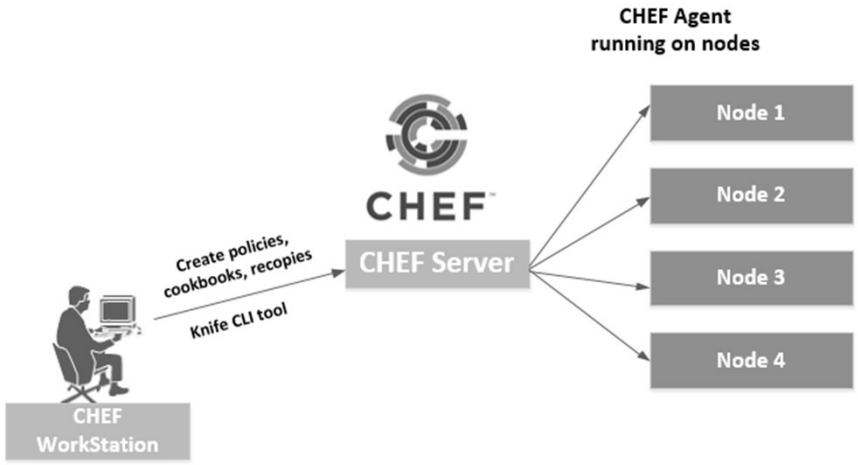
❖ **Chef:**

Chef - Quick Summary	
Name	Chef
By	Chef
Where it runs	On a separate server. Requires Chef agents to be installed on all managed nodes
What it does	Automates tasks for managing the infrastructure network, servers, application servers, as well as building and deploying applications
What it can do out-of-the-box	You can manage and automate the configuration of networking devices that come with a Chef agent or allow you to install a Chef agent. This includes all networking tools running on a host-based such as OVS, as well as other networking appliances such as Cisco Nexus switches.

248

X. 네트워크 자동화

❖ **Chef Architecture:**



249

X. 네트워크 자동화

❖ **Python:**

Python is a very popular modern programming language. Python is used for network automation, especially when you need to write complex, customized rules that are needed to perform specific sets of tasks. Python has many networking libraries that you can use to manage your network. Using Python, you can build programs that can connect to a network device, execute commands, grab the outputs, and show you the results.

Python supports multiple protocols such as SSH, SNMP, Telnet and APIs to communicate with a networking device. If you have an aging device that only supports older versions of SSH, you will be able to use Python's SSH library (Paramiko) to manage that device.

When dealing with legacy networking devices that only support CLI (via SSH or Telnet), the main issue is to parse the outputs. In a CLI environment, the device always returns a stream of characters which is formatted for human reading. For example, when you issue a "show run interface Gigabit 1" command, the output is a set of characters that define the interface speed, its mode (Layer 2 without VLANs, Layer 2 with 802.1q, or Layer 3), VLANs, etc.:

https://www.python.org/

250

X. 네트워크 자동화

❖ Using Netmiko to Connect to a Networking Device:

The following example shows how to define a networking device and use Netmiko to connect to the device and execute some configurations.

```
from netmiko import ConnectHandler
cisco_881 = {
    'device_type': 'cisco_ios',
    'ip':   '10.10.10.10',
    'username': 'test',
    'password': 'password',
    'port' : 8022,          # optional, defaults to 22
    'secret': 'secret',     # optional, defaults to ''
    'verbose': False,       # optional, defaults to False
}
```

Establish an SSH connection to the device by passing in the device dictionary.

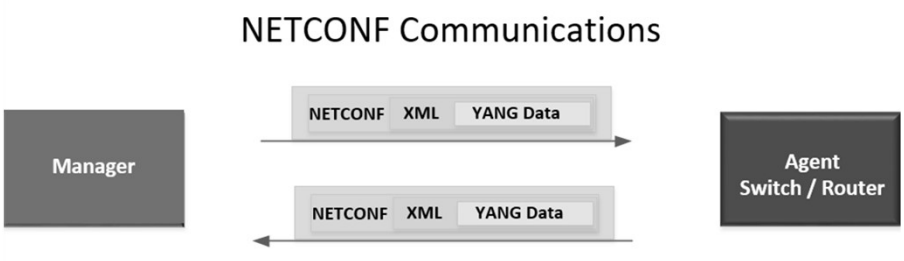
```
net_connect = ConnectHandler(**cisco_881)
```

Execute **show** commands.

```
output = net_connect.send_command('show ip int brief')
print(output)
Interface                  IP-Address OK? Method Status                Protocol
FastEthernet0              unassigned YES unset down                  down
FastEthernet1              unassigned YES unset down                  down
FastEthernet2              unassigned YES unset down                  down
FastEthernet3              unassigned YES unset down                  down
FastEthernet4              10.10.10.10 YES manual up                    up
Vlan1
```


X. 네트워크 자동화

❖ NETCONF:



X. 네트워크 자동화

❖ **NETCONF:**

NETCONF Command	Description
<get>	Retrieve running configuration and device state information
<get-config>	Retrieve all or part of a specified configuration datastore
<edit-config>	Edit a configuration datastore by creating, deleting, merging, or replacing content
<copy-config>	Copy an entire configuration datastore to another configuration datastore
<delete-config>	Delete a configuration datastore
<lock>	Lock an entire configuration datastore of a device
<unlock>	Release a configuration datastore lock previously obtained with the <lock> com mand
<close-session>	Request graceful termination of a NETCONF session
<kill-session>	Force the termination of a NETCONF session

Opensource Networking

james@jslab.kr

XI. Network Data Analytics

- SNAS (Streaming Network Analytics System)
- PNDA
- PNDA Principles and Benefits
- BGP Analytics Application (Example)

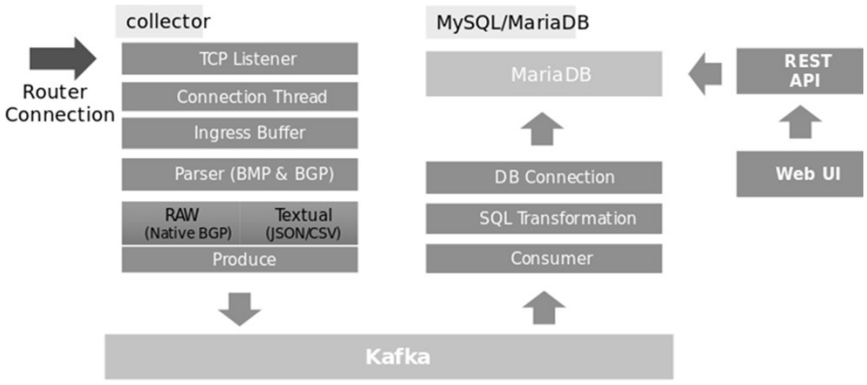
https://www.linuxfoundation.org/projects/networking/

Opensource Networking

james@jslab.kr

XI. Network Data Analytics

❖ SNAS (Streaming Network Analytics System):

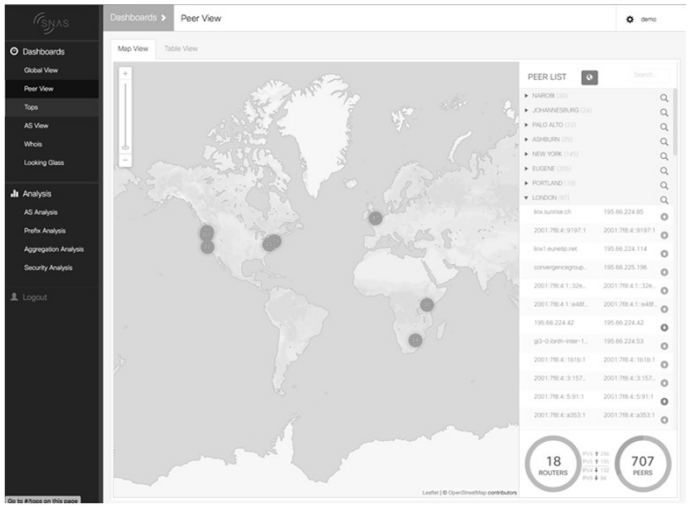


SNAS Architecture

https://www.snas.io/

XI. Network Data Analytics

❖ **SNAS (Streaming Network Analytics System):**



SNAS Browser UI

http://www.snas.io/demo/demo-ui-rv/

JS Lab

XI. Network Data Analytics

❖ **SNAS (Streaming Network Analytics System):**

Using Grafana with SNAS

http://www.snas.io/demo/demo-grafana/

JS Lab

XI. Network Data Analytics

❖ **PNDA:**

PNDA features:

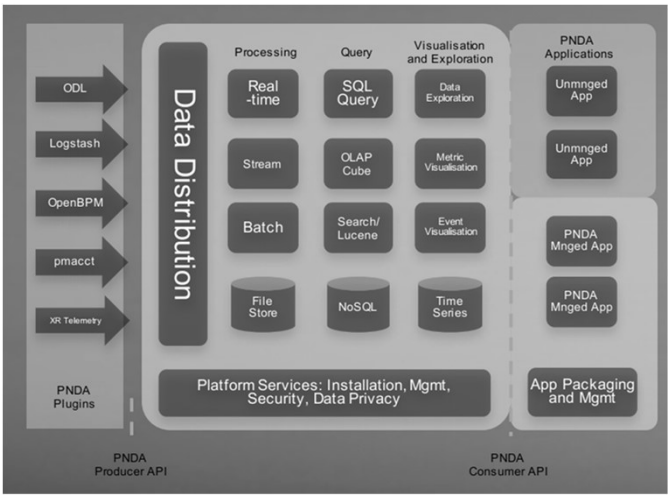
- Open source platform for Network Data Analytics
- Aggregates data like logs, metrics and network telemetry
- Scales up to consume millions of messages per second
- Efficiently distributes data with a publisher and a subscriber model
- Processes bulk data in batches, or streams data in real-time
- Manages the lifecycle of applications that process and analyze data
- Lets you explore data using interactive notebooks.

PNDA has a 3-tier architecture:

- Log ingestion plugin to get data into PNDA
- Analysis engine, which includes data distribution, parsers, storage, big data queries, and data visualization
- Consumer application - PNDA applications that utilize the PNDA analytical information to produce specific use cases for the user.

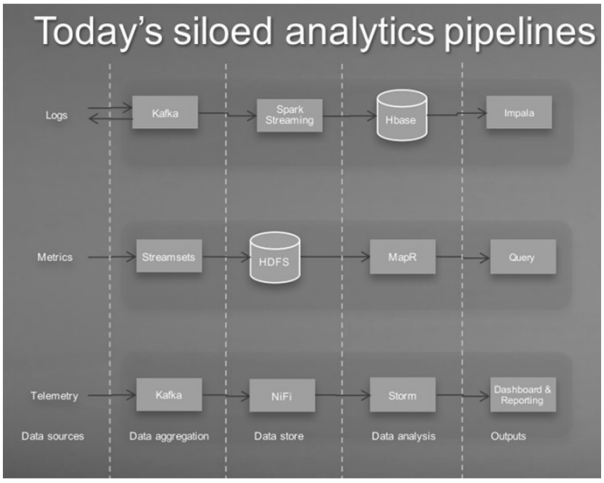
XI. Network Data Analytics

❖ **PNDA:**



XI. Network Data Analytics

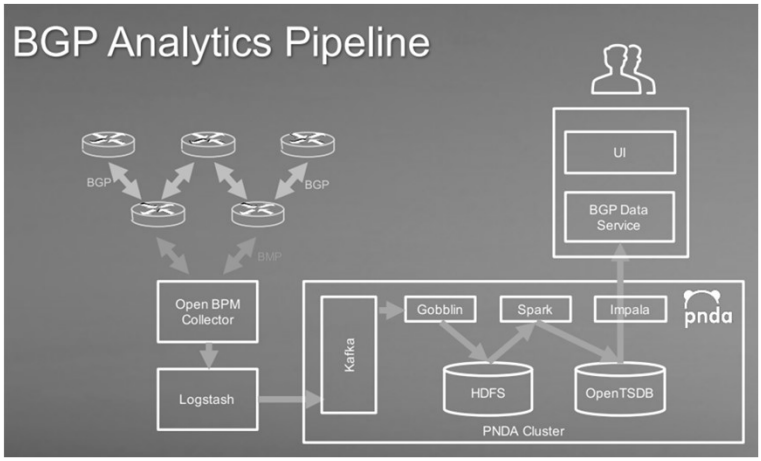
❖ **PNDA Principles and Benefits:**



Siloed Analytics Pipelines

XI. Network Data Analytics

❖ **BGP Analytics Application (Example):**



BGP Analytics Pipeline

Opensource Networking
james@jslab.kr

XII. Use Case

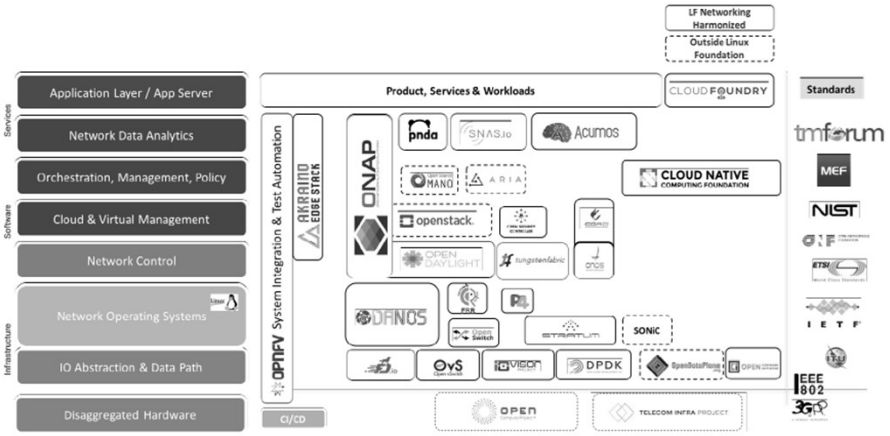
- All Projects on One Page
- Use Case: Service Provider Transit Network
- Use Case: Service Provider Core Network
- Use Case: Service Provider uCPE
- Use Case: Cloud Providers and Enterprise Datacenters

Opensource Networking
james@jslab.kr

https://www.linuxfoundation.org/projects/networking/

XII. Use Case

❖ **All Projects on One Page**



XII. Use Case

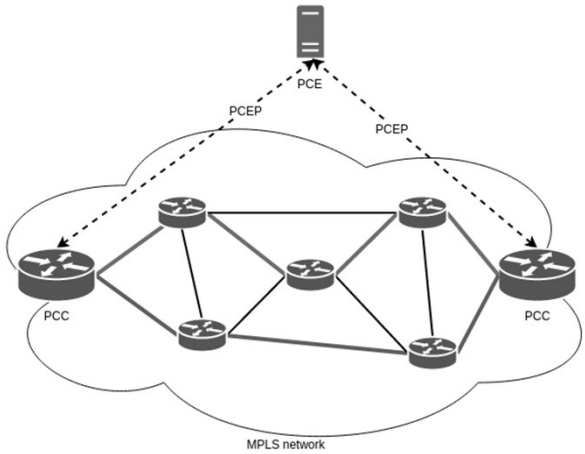
❖ **Use Case: Service Provider Transit Network**

Example: Using open source technologies, service providers can build a software defined transit network:

- Bare metal switches
  Bare metal (1G/10G/40G/100G/400G) switches can be used to build the transit network connected to a distributed fiber infrastructure.
- Bare metal optical
  Using new bare metal ODTN (Open Disaggregated Transport Network), service providers can leverage the software defined optical networking.
- Switch operating system
  Service providers can load open source networking operating system such as ONL (Open Network Linux), OpenSwitch or Trellis.
- SDN Controller
  Service providers can use open source SDN controllers such as OpenDaylight or ONOS as a platform to manage and control the transit network. The SDN controllers will be able to automatically populate the flow tables of the transit switches in order to deliver packets to customers and subscribers.
- Orchestration
  Using ONAP as an orchestration platform can help service providers to create an automated orchestration system that can manage not only the network, but also interact with other OSS and BSS services.
- Analytics
  Leveraging PNDA and SNAS can help service providers build a monitoring and analytics platform for their network.

XII. Use Case

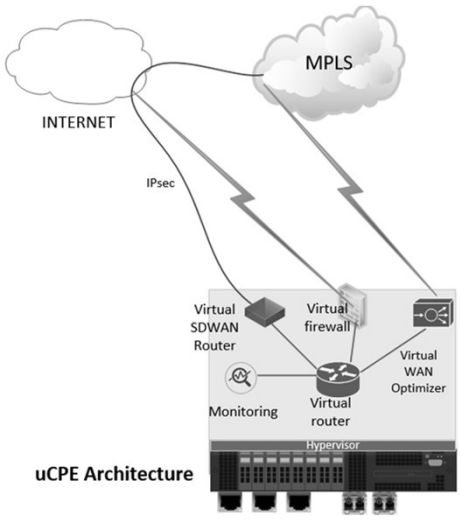
❖ **Use Case: Service Provider Core Network**



PCEP-Based Architecture

XII. Use Case

❖ **Use Case: Service Provider uCPE**



uCPE Architecture

XII. Use Case

❖ **Use Case: Cloud Providers and Enterprise Datacenters**

•Bare metal switches

Bare metal (1G/10G/40G/100G/400G) switches can be used to deploy a Clos-based leaf-spine switch architecture.

•Switch operating system

Service providers can load open source networking operating systems such as ONL (Open Network Linux), OpenSwitch or Trellis.

•SDN Controller

Service providers can use open source SDN controllers such as OpenDaylight or ONOS as a platform to manage and control the transit network. The SDN controllers will be able to automatically populate the flow tables of the transit switches in order to deliver packets to customers and subscribers.

Opensource Networking james@jslab.kr

Opensource Networking james@jslab.kr

Opensource Networking
james@jslab.kr

별첨

1. 리눅스 네트워킹
2. 오픈스택
3. 컨테이너 네트워킹
4. 쿠버네티스 네트워킹
5. OPNFV
6. 5G 코어네트워킹
7. AI 네트워킹 인프라

JS Lab

271

Opensource Networking
james@jslab.kr

별첨

1. 리눅스 네트워킹
2. 오픈스택
3. 컨테이너 네트워킹
4. 쿠버네티스 네트워킹
5. OPNFV
6. 5G 코어네트워킹
7. AI 네트워킹 인프라

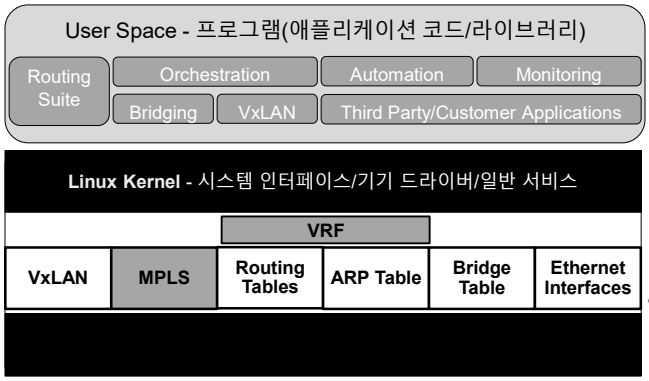
JS Lab

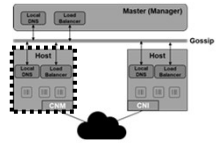
272

❖ 별첨 1. 리눅스 네트워킹

❖ Linux Kernel 발전 - 4.x의 MPLS 지원

- **MPLS LSR 지원: v4.1**
- **LWT / MPLS IP tunnel 지원: v4.3**
- **MPLS multipath 지원: v4.5**
- **MPLS VRF의 IP 명령어 지원**



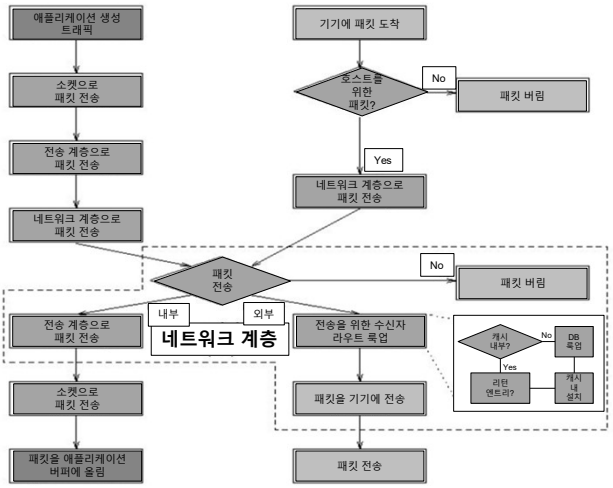



```
jslab@ubuntu60:/istio-playground$ uname -a
Linux ubuntu60 4.4.0-116-generic #140-
Ubuntu SMP Mon Feb 12 21:23:04 UTC 2018
x86_64 x86_64 x86_64 GNU/Linux
```

273

❖ 별첨 1. 리눅스 네트워킹

❖ Linux의 네트워크 개요



274

❖ 별첨 1. 리눅스 네트워킹

❖ **Linux Kernel의 TCP/IP 처리**

❖ **EGRESS - 서브루틴과 시스템 콜** (L5, L4, L3, L2)

- **Layer 5 - write(), sendto(), sendmsg()** -- 네트워크 상의 데이터 전송과 시스템콜(syscall)
- **Layer 4 - tcp_sendmsg (see tcp.c kernel source code)** -- 데이터 프레임의 적절한 시간 내 전송(emit)
- **Layer 3**
  - ✓ **Ip_queue_xmit()** - 라우팅과 IPv4 헤더 생성
  - ✓ **Nf_hook()** - 네트워크 필터링
  - ✓ **ip_output()** - post-routing 필터링
- **Layer 2** - 패킷 큐잉 확인 통제 discipline (qdisc)

Opensource Networking
james@jslab.kr

❖ 별첨 1. 리눅스 네트워킹

❖ **Linux Kernel의 TCP/IP 처리**

❖ **INGRESS - 서브루틴과 시스템 콜**

- **Layer 2 - netif_receive_skb() -- 커널에 패킷 제공**
- **Layer 3**
  - ✓ **ARP - arp_rcv()**
  - ✓ **IP - ip_rcv()**
- **Layer 4**
  - ✓ **TCP - tcp_v4_rcv()**
- **Layer 5 - read(), rcvfrom(), recvmsg() -** 네트워크 상의 데이터 수신과 시스템콜(syscall)

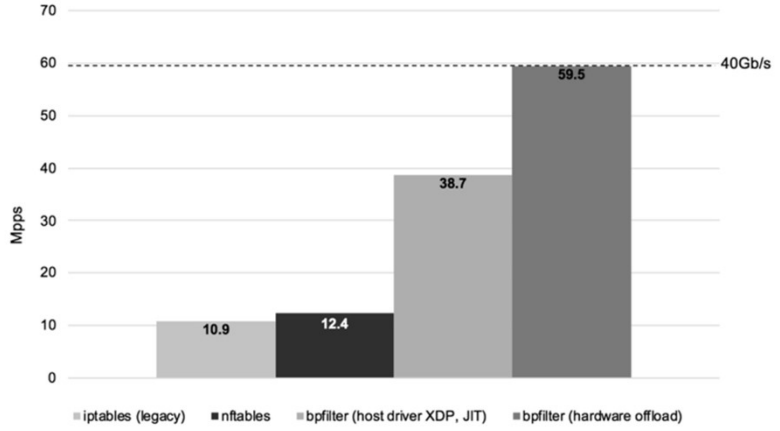
Opensource Networking
james@jslab.kr

❖ 별첨 1. 리눅스 네트워킹

❖ **eBPF XDP Linux 4.8+**

❖ **소프트웨어 기반 성능 가속**

❖ **성능 가속 전용 하드웨어 발전** (스마트 NIC, HCI, 컨테이너 가속 Storage 등등)



https://www.netronome.com/blog/frnog-30-faster-networking-la-francaise/

❖ 별첨 1. 리눅스 네트워킹

❖ **네트워크 관리**

- **Network Information**
  - ✓ `ip route show` displays host-based routing tables
  - ✓ `ip address show` displays L3 information
  - ✓ `ip link show` displays L2 information
- **Socket Information**
  - ✓ `ss -tanup` displays socket information
- **Others***
  - ✓ `route, netstat -rn` displays host-based routing tables
  - ✓ `ifconfig -a` displays all available network interfaces
  - ✓ `netstat -tulpn` displays socket information

❖ 별첨 1. 리눅스 네트워킹

❖ 네트워크 관리

- **Static Network Configuration** (임시)
  - ✓ **ip route add default via <ip_addr>** add default route
  - ✓ **ip address add <ip_addr> dev <dev>** add l3 ip address
- **Static Network Configuration** (영구, RHEL 계열)
  - ✓ **/etc/sysconfig/network** global nic configuration
  - ✓ **/etc/sysconfig/network-scripts/ifcfg-*** per-nic configuration
- **Static Network Configuration** (영구, Debian 계열)
  - ✓ **/etc/network/interfaces global** nic configuration
  - ✓ **/etc/network/interfaces.d/<nic>.cfg** per-nic configuration
- 기타
  - ✓ **route add default via <ip_addr>**
  - ✓ **ifconfig <dev> <ip_addr>**

❖ 별첨 1. 리눅스 네트워킹

❖ 네트워크 관리

- **DNS Configuration**
  - ✓ **/etc/resolv.conf** resolver configuration, getnameinfo()
  - ✓ **/etc/nsswitch.conf** service provider -name service switch per category defs
  - ✓ **/etc/hosts** service provider - for instance consumed by dnsmasq for A records

❖ 별첨 1. 리눅스 네트워킹

❖ **Iptables**
- **iptables 룰세트(Rule set) 정의를 위한 테이블 구조**
- **IP table 의 각 룰은**
  - ✓ **매칭을 위한 Classifiers** (iptables matches)
  - ✓ **연결 된 액션을 위한 connected action** (iptables target)

❖ **커널 모듈은 netfilter**
- **반드시 커널에서 실행** (2.4.x 이상)
- **'stateless' / 'stateful' 네트워크 필터링**

❖ **기본 3개의 테이블로 구성**
- **Mangle** - 특별 패킷을 처리
- **NAT** - 네트워크주소 변환을 실행
- **Filter** - 패킷 필터링을 실행

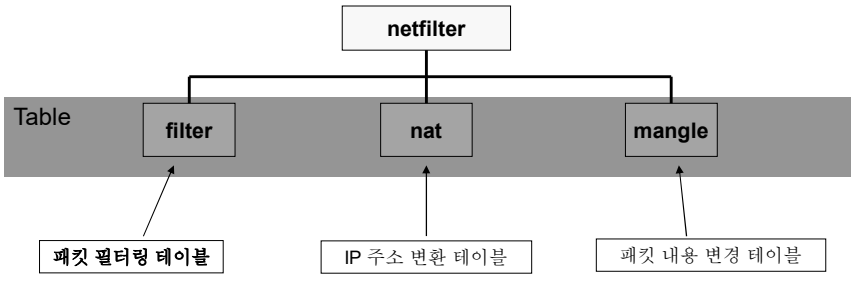
❖ **각 테이블은 1개 이상의 chain**

❖ **IP Tables 패킷을 전송 할 수 있고 NAT를 실행**
- **Linux/UNIX 기반 제조사 기기들은 추가기증 제공** (i.e. Arista, Cumulus Networks)

❖ 별첨 1. 리눅스 네트워킹

❖ **iptables – Tables and Chains**

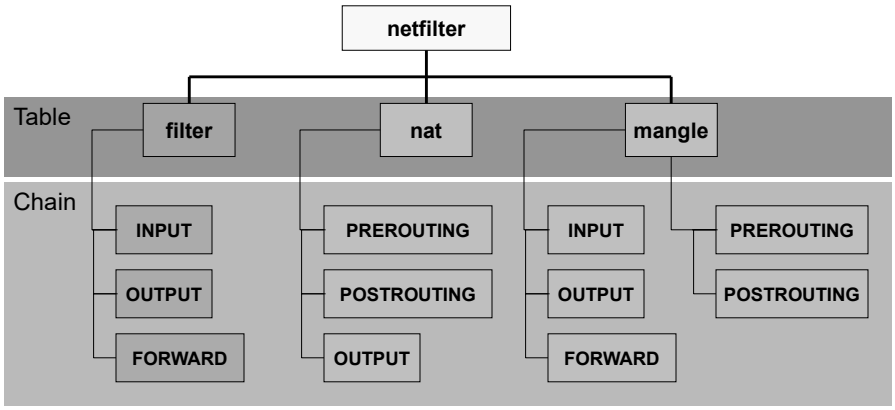
❖ **각 기능은 netfilter architecture에 테이블로 준비되어 제공**
- **filter**
- **nat**
- **mangle**



❖ 별첨 1. 리눅스 네트워킹

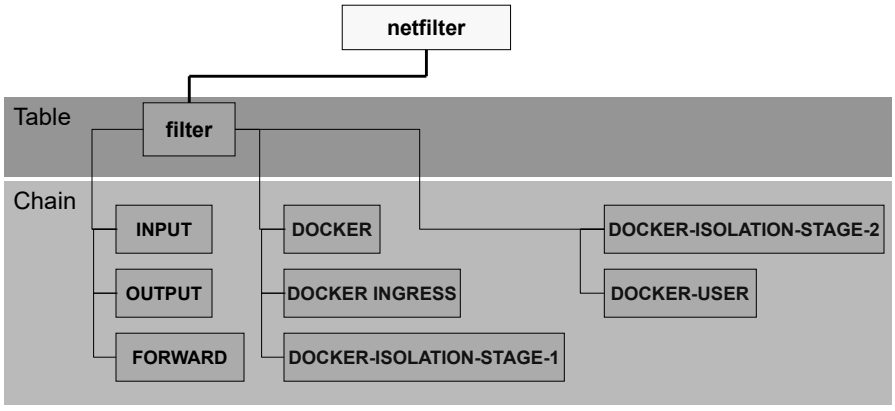
❖ **iptables – Tables and Chains** (3 built -in tables: Filter, NAT, Mangle)

❖ **각 테이블은 체인 세트가 있으며, 각체인에는 룰 세트를 할당 할 수 있음**
(Tables → Chains → Rules)



❖ 별첨 1. 리눅스 네트워킹

❖ **Docker Chains:**
DOCKER, DOCKER INGRESS, DOCKER-ISOLATION-STAGE-1, DOCKER-ISOLATION-STAGE-2, DOCKER-USER



❖ 별첨 1. 리눅스 네트워킹

❖ sudo iptables –t filter –L

```
root@ubuntu60:~# sudo iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     udp  --  anywhere             anywhere             udp dpt:domain
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:domain
ACCEPT     udp  --  anywhere             anywhere             udp dpt:bootps
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:bootps

Chain FORWARD (policy DROP)
target     prot opt source               destination
DOCKER-USER  all  --  anywhere             anywhere
DOCKER-INGRESS  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-1  all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             192.168.122.0/24     ctstate RELATED,ESTABLISHED
ACCEPT     all  --  192.168.122.0/24     anywhere
ACCEPT     all  --  anywhere             anywhere
REJECT     all  --  anywhere             anywhere             reject-with icmp-port-unreachable
REJECT     all  --  anywhere             anywhere             reject-with icmp-port-unreachable
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
DROP       all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     udp  --  anywhere             anywhere             udp dpt:bootpc
```

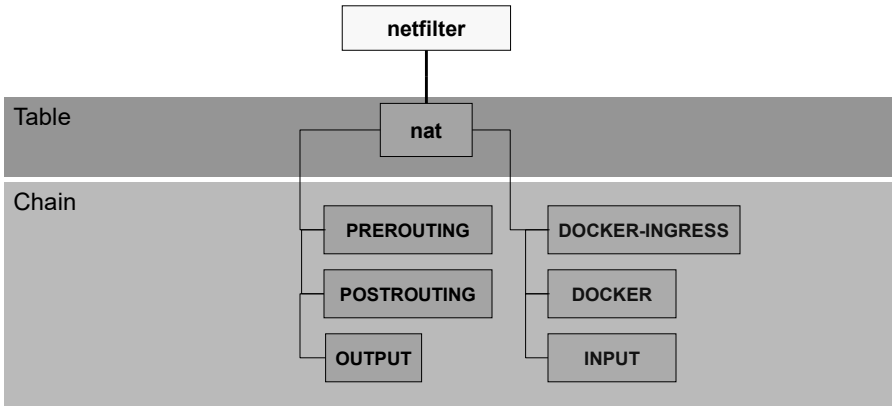

❖ 별첨 1. 리눅스 네트워킹

```
Chain DOCKER (2 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             172.17.0.2           tcp dpt:6653
ACCEPT     tcp  --  anywhere             172.17.0.2           tcp dpt:8181
ACCEPT     tcp  --  anywhere             172.17.0.2           tcp dpt:8101

Chain DOCKER-INGRESS (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:8282
ACCEPT     tcp  --  anywhere             anywhere             state RELATED,ESTABLISHED tcp spt:8282
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:30000
ACCEPT     tcp  --  anywhere             anywhere             state RELATED,ESTABLISHED tcp spt:30000
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:5001
ACCEPT     tcp  --  anywhere             anywhere             state RELATED,ESTABLISHED tcp spt:5001
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:5000
ACCEPT     tcp  --  anywhere             anywhere             state RELATED,ESTABLISHED tcp spt:5000
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:9090
ACCEPT     tcp  --  anywhere             anywhere             state RELATED,ESTABLISHED tcp spt:9090
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http-alt
ACCEPT     tcp  --  anywhere             anywhere             state RELATED,ESTABLISHED tcp spt:http-alt
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:3001
ACCEPT     tcp  --  anywhere             anywhere             state RELATED,ESTABLISHED tcp spt:3001
RETURN     all  --  anywhere             anywhere

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target     prot opt source               destination
DOCKER-ISOLATION-STAGE-2  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-2  all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-ISOLATION-STAGE-2 (2 references)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
DROP       all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-USER (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

❖ 별첨 1. 리눅스 네트워킹

❖ **iptables – Tables and Chains** (3 built -in tables: Filter, NAT, Mangle)

❖ **각 테이블은 체인 세트가 있으며, 각체인에는 룰 세트를 할당 할 수 있음**
(Tables → Chains → Rules)



❖ 별첨 1. 리눅스 네트워킹

❖ **sudo iptables –t nat –L**

```
jslab@ubuntu60:~$ sudo iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER-INGRESS  all  --  anywhere             anywhere             ADDRTYPE match dst-type LOCAL
DOCKER     all  --  anywhere             anywhere             ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER-INGRESS  all  --  anywhere             anywhere             ADDRTYPE match dst-type LOCAL
DOCKER     all  --  anywhere            !loopback/8           ADDRTYPE match dst-type LOCAL
```

❖ 별첨 1. 리눅스 네트워킹

❖ sudo iptables –t nat –L

```
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere             ADDRTYPE match src-type LOCAL
MASQUERADE  all  --  172.18.0.0/16        anywhere
RETURN     all  --  192.168.122.0/24     base-address.mcast.net/24
RETURN     all  --  192.168.122.0/24     255.255.255.255
MASQUERADE  tcp  --  192.168.122.0/24     !192.168.122.0/24     masq ports: 1024-65535
MASQUERADE  udp  --  192.168.122.0/24     !192.168.122.0/24     masq ports: 1024-65535
MASQUERADE  all  --  192.168.122.0/24     !192.168.122.0/24
MASQUERADE  all  --  172.17.0.0/16        anywhere
MASQUERADE  tcp  --  172.17.0.2           172.17.0.2           tcp dpt:6653
MASQUERADE  tcp  --  172.17.0.2           172.17.0.2           tcp dpt:8181
MASQUERADE  tcp  --  172.17.0.2           172.17.0.2           tcp dpt:8101

Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere
DNAT       tcp  --  anywhere             anywhere             tcp dpt:1653 to:172.17.0.2:6653
DNAT       tcp  --  anywhere             anywhere             tcp dpt:1181 to:172.17.0.2:8181
DNAT       tcp  --  anywhere             anywhere             tcp dpt:1101 to:172.17.0.2:8101

Chain DOCKER-INGRESS (2 references)
target     prot opt source               destination
DNAT       tcp  --  anywhere             anywhere             tcp dpt:3001 to:172.18.0.2:3001
DNAT       tcp  --  anywhere             anywhere             tcp dpt:8282 to:172.18.0.2:8282
DNAT       tcp  --  anywhere             anywhere             tcp dpt:5000 to:172.18.0.2:5000
DNAT       tcp  --  anywhere             anywhere             tcp dpt:30000 to:172.18.0.2:30000
DNAT       tcp  --  anywhere             anywhere             tcp dpt:5001 to:172.18.0.2:5001
DNAT       tcp  --  anywhere             anywhere             tcp dpt:9090 to:172.18.0.2:9090
DNAT       tcp  --  anywhere             anywhere             tcp dpt:http-alt to:172.18.0.2:8080
RETURN     all  --  anywhere             anywhere
jslab@ubuntu60:~$
```


❖ 별첨 1. 리눅스 네트워킹

❖ brctl show: 브릿지 ID, STP 상태,

```
jslab@ubuntu60:/$ brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242cb6a0d24       no              vetha47e861
                                                        vethd5f4b06
docker_gwbridge         8000.024239101cd2       no              veth50c3c19
                                                        vethb177b4d
                                                        vetho0042bc
                                                        vethc38bae7
                                                        vethd4f0617
virbr0          8000.525400cbd4ad       yes             virbr0-nic
jslab@ubuntu60:/$
```

❖ 별첨 1. 리눅스 네트워킹

❖ ip link show (L2)

```
jslab@ubuntu60:/$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:f8:f7:62 brd ff:ff:ff:ff:ff:ff
3: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1
    link/ether de:d7:e6:2a:8e:82 brd ff:ff:ff:ff:ff:ff
4: ovs60-3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1
    link/ether a6:41:b4:7b:d3:42 brd ff:ff:ff:ff:ff:ff
5: ovs60-1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1
    link/ether ce:80:c4:2b:f4:4c brd ff:ff:ff:ff:ff:ff
6: ovs60-2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1
    link/ether e6:d1:6c:71:07:49 brd ff:ff:ff:ff:ff:ff
7: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 02:42:cb:6a:0d:24 brd ff:ff:ff:ff:ff:ff
8: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default qlen 1000
    link/ether 52:54:00:cb:d4:ad brd ff:ff:ff:ff:ff:ff
9: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN mode DEFAULT group default qlen 1000
    link/ether 52:54:00:cb:d4:ad brd ff:ff:ff:ff:ff:ff
14: dooker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 02:42:39:10:1c:d2 brd ff:ff:ff:ff:ff:ff
16: veth50c3c19@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether 2e:d9:68:c5:7f:34 brd ff:ff:ff:ff:ff:ff link-netnsid 1
23: vethc38bae7@if22: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether d6:b4:54:90:f9:82 brd ff:ff:ff:ff:ff:ff link-netnsid 3
36: vethb177b4d@if35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether 1e:5a:3d:98:80:ce brd ff:ff:ff:ff:ff:ff link-netnsid 6
40: vethd5f4b06@if39: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP mode DEFAULT group default
    link/ether 1e:d2:02:7b:ba:e5 brd ff:ff:ff:ff:ff:ff link-netnsid 5
42: vetha47e861@if41: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP mode DEFAULT group default
    link/ether 9e:d2:b1:33:f9:6d brd ff:ff:ff:ff:ff:ff link-netnsid 4
49: vetho0042bc@if48: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether ce:49:6e:c2:72:92 brd ff:ff:ff:ff:ff:ff link-netnsid 8
53: vethd4f0617@if52: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether f6:e6:ae:1d:31:30 brd ff:ff:ff:ff:ff:ff link-netnsid 9
jslab@ubuntu60:/$
```


❖ 별첨 1. 리눅스 네트워킹

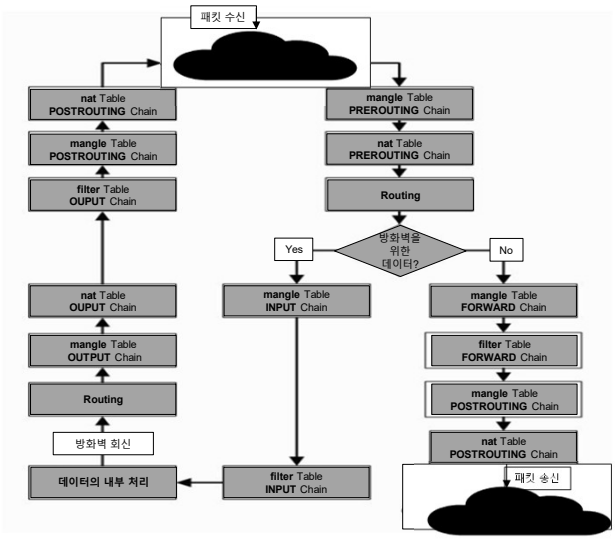
❖ ifconfig

```
jslab@ubuntu60:/$ ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:cb:6a:0d:24
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:cbff:fe6a:d24/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3693195 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3631616 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:723534375 (723.5 MB)  TX b

dooker_gwbridge Link encap:Ethernet  HWaddr 0
          inet addr:172.18.0.1  Bcast:172.18.
          inet6 addr: fe80::42:39ff:fe10:1cd2
          UP BROADCAST RUNNING MULTICAST  MTU
          RX packets:30 errors:0 dropped:0 ov
          TX packets:336 errors:0 dropped:0 o
          collisions:0 txqueuelen:0
          RX bytes:2070 (2.0 KB)  TX bytes:32

ens33     Link encap:Ethernet  HWaddr 00:0c:2
          inet addr:192.168.0.60  Bcast:192.1
          inet6 addr: fe80::20c:29ff:fef8:f76
          UP BROADCAST RUNNING MULTICAST  MTU
          RX packets:9455294 errors:0 dropped
          TX packets:6079694 errors:0 dropped
          collisions:0 txqueuelen:1000
          RX bytes:2836591041 (2.8 GB)  TX by

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Met
          RX packets:278 errors:0 dropped:0 o
          TX packets:278 errors:0 dropped:0 o
          collisions:0 txqueuelen:1
          RX bytes:24842 (24.8 KB)  TX bytes:

veth50c3c19 Link encap:Ether
          inet6 addr: fe80::
          UP BROADCAST RUNNI
          RX packets:30 erro
          TX packets:494 err
          collisions:0 txque
          RX bytes:2490 (2.4

vetha47e861 Link encap:Ether
          inet6 addr: fe80::
          UP BROADCAST RUNNI
          RX packets:3434269
          TX packets:3467311
          collisions:0 txque
          RX bytes:577053210

vethb177b4d Link encap:Ether
          inet6 addr: fe80::
          UP BROADCAST RUNNI
          RX packets:0 error
          TX packets:398 err
          collisions:0 txque
          RX bytes:0 (0.0 B)

vetho0042bc Link encap:Ether
          inet6 addr: fe80::
          UP BROADCAST RUNNI
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:198 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:20236 (20.2 KB)

vethc38bae7 Link encap:Ethernet  HWaddr d6:b4:54:90:f9:82
          inet6 addr: fe80::d4b4:54ff:fe90:f982/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:406 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:41139 (41.1 KB)

vethd4f0617 Link encap:Ethernet  HWaddr f6:e6:ae:1d:31:30
          inet6 addr: fe80::f4e6:aeff:fe1d:3130/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:196 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:20236 (20.2 KB)

vethd5f4b06 Link encap:Ethernet  HWaddr 1e:d2:02:7b:ba:e5
          inet6 addr: fe80::1cd2:2ff:fe7b:bae5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:258636 errors:0 dropped:0 overruns:0 frame:0
          TX packets:164453 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:198032819 (198.0 MB)  TX bytes:37512927 (37.5 MB)

virbr0    Link encap:Ethernet  HWaddr 52:54:00:cb:d4:ad
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

❖ 별첨 1. 리눅스 네트워킹

패킷 수신

❖ Linux의 IPTable 개요



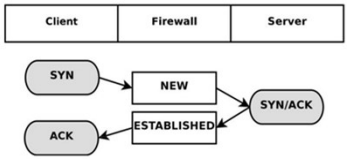
❖ 별첨 1. 리눅스 네트워킹

❖ Linux IPTables 방화벽

❖ Client Firewall (iptables)

#iptables –L INPUT (for input chain type or Inbound)
#iptables –L OUTPUT (for output chain type or outbound)

```
jslab@ubuntu60:/$ sudo iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     udp  --  anywhere             anywhere            udp dpt:domain
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:domain
ACCEPT     udp  --  anywhere             anywhere            udp dpt:bootps
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:bootps
jslab@ubuntu60:/$ sudo iptables -L OUTPUT
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     udp  --  anywhere             anywhere            udp dpt:bootpc
jslab@ubuntu60:/$
```



❖ 별첨 1. 리눅스 네트워킹

❖ Linux IPTables 방화벽

❖ Append INBOUND Rules in iptables (Client Firewall)

```
# iptables –A INPUT –s 192.168.0.1 –d 192.168.0.254 –p ICMP –j DROP
# iptables –A INPUT –s 192.168.0.0/24 –d 192.168.0.254/32 –p ICMP –j DROP
# iptables –L INPUT
# service iptables save
# service iptables start
```

```
iptables –A INPUT –s 192.168.0.1 –d 192.168.0.254 –p ICMP –j DROP
```

Append | Source | Destination | Protocol | Jump

```
# iptables –D INPUT 1
```

Delete | Line Number

❖ 별첨 1. 리눅스 네트워킹

❖ Linux의 IPTable 명령어

- 방화벽 룰의 우선 순위

Rule Name	Action
Rule 1	ACCEPT
Rule 2	ACCEPT
Rule 3	ACCEPT
Rule 4	DROP
All Traffic	Deny

❖ 별첨 1. 리눅스 네트워킹

❖ NAT with IPTables

- Post Routing
  - ✓ snat
- Pre Routing
  - ✓ MASQUERADE Port (1 to 65535)
- Masquerade (Port Address Translation (PAT))
  - ✓ Port Address Table
- IP Translation

❖ 별첨 1. 리눅스 네트워킹

❖ SNAT (POSTROUTING) with IPTables

- One to One

  # iptables -t nat -A POSTROUTING -s 192.168.1.2(LAN) -j SNAT --to 200.200.200.1(WAN-Public IP)
- Many to One

  # iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j SNAT --to 200.200.200.1
- Many to Many

  # iptables -t nat -A POSTROUTING -s 192.168.1.0/255.255.255.0 -j SNAT --to 200.200.200.1-200.200.200.6
- Many to One(PAT)

  # iptables -t nat -A POSTROUTING -s 192.168.1.0/255.255.255.0 -o eth0(WAN) -j MASQUERADE

  # iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE Port (1 to 65535)

iptables –t nat –L
iptables –t nat –L POSTROUTING
iptables –t nat –F

❖ 별첨 1. 리눅스 네트워킹

❖ DNAT (PREROUTING) with IPTables

❖ Port Forwardig

```
# iptables -t nat -A PREROUTING -p tcp --dport 3389 -i eth0(WAN-Public IP) -j
DNAT --to 192.168.0.1:3389
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport $srcPortNumber
REDIRECT --to-port $dstPortNumber
```

Opensource Networking
james@jslab.kr

❖ 별첨 1. 리눅스 네트워킹

❖ DNAT (PREROUTING) with IPTables

❖ Port Forwardig

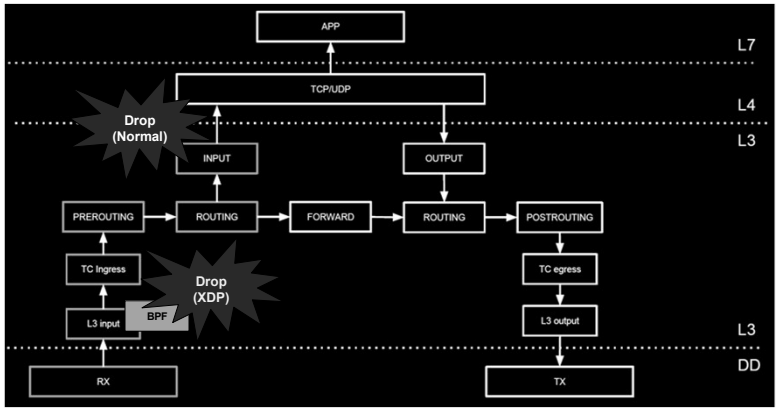
```
# iptables -t nat -A PREROUTING -p tcp --dport 3389 -i eth0(WAN-Public IP) -j
DNAT --to 192.168.0.1:3389
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport $srcPortNumber
REDIRECT --to-port $dstPortNumber
```

Opensource Networking
james@jslab.kr

JS Lab

❖ 별첨 1. 리눅스 네트워킹

❖ XDP (eXpress Data Path)

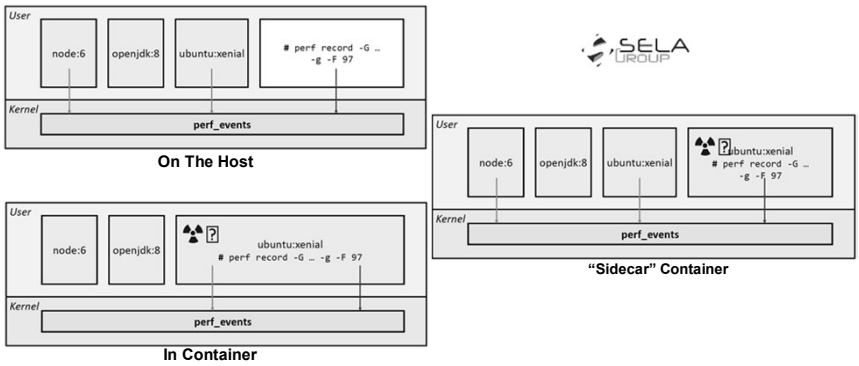


301

❖ 별첨 1. 리눅스 네트워킹

❖ Tool Deployment

- **호스트 내 설치** (On The Host)
- **컨테이너 내 설치** (In Container)
- **도구 전용 컨테이너** (“Sidecar” Container)

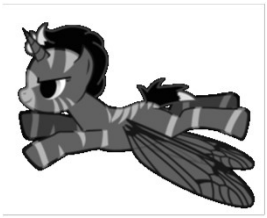


302

❖ 별첨 1. 리눅스 네트워킹

❖ **eBPF** (enhanced Berkeley Packet Filter)

- **Lead developer: Alexei Starovoitov** (Facebook)
- **사용 분야**
  - ✓ **가상 네트워킹** (Virtual networking)
  - ✓ **보안** (Security)
  - ✓ **프로그램 사용 추적** (Programmatic tracing)
- **프론트엔드 확장** (Different front-ends)
  - ✓ **C, perf, bcc, ply, … BPF**



❖ 별첨 1. 리눅스 네트워킹

❖ **BPF Enhancements by Linux Version**

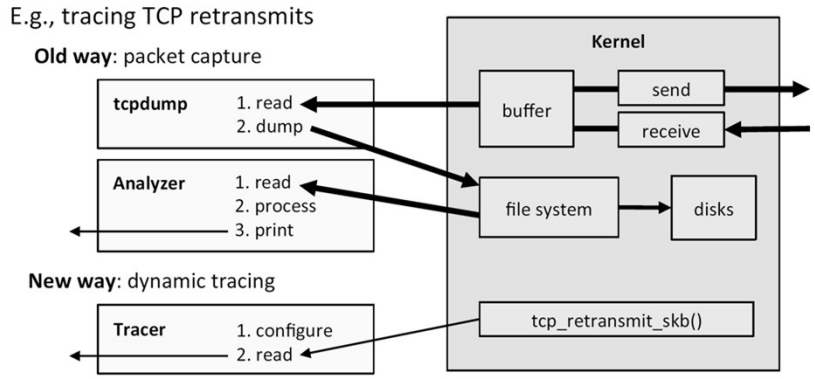
- **3.18: bpf syscall**
- **3.19: sockets**
- **4.1: kprobes**
- **4.4: bpf_perf_event_output** (ubuntu 16.06)
- **4.6: stack traces**
- **4.7: tracepoints** (ubuntu 16.10)
- **4.9: profiling**

```
jslab@ubuntu60:/istio-playground$ uname -a
Linux ubuntu60 4.4.0-116-generic #140-Ubuntu SMP Mon Feb 12 21:23:04 UTC 2018 x86_64 x86_64
x86_64 GNU/Linux
```

❖ 별첨 1. 리눅스 네트워킹

❖ **BPF Enhancements by Linux Version**

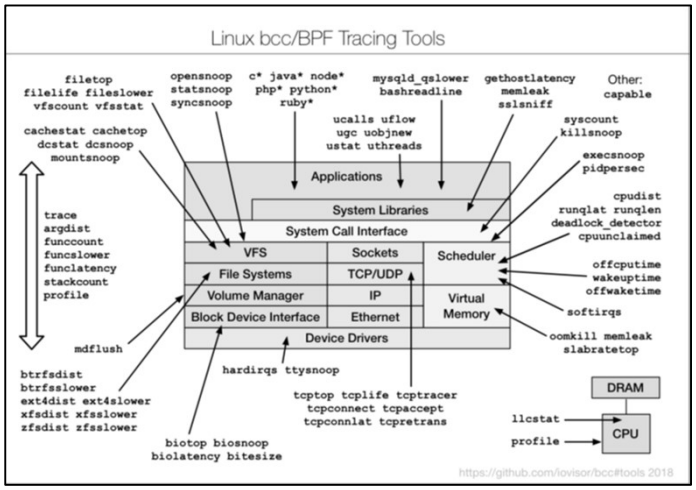
❖ **Event Tracing Efficiency**



❖ 별첨 1. 리눅스 네트워킹

❖ **eBPF bcc Linux 4.4+**

- **Enhanced BPF**



https://github.com/iovisor/bcc

Opensource Networking
james@jslab.kr

별첨

1. 리눅스 네트워킹
2. 오픈스택
3. 컨테이너 네트워킹
4. 쿠버네티스 네트워킹
5. OPNFV
6. 5G 코어네트워킹
7. AI 네트워킹 인프라

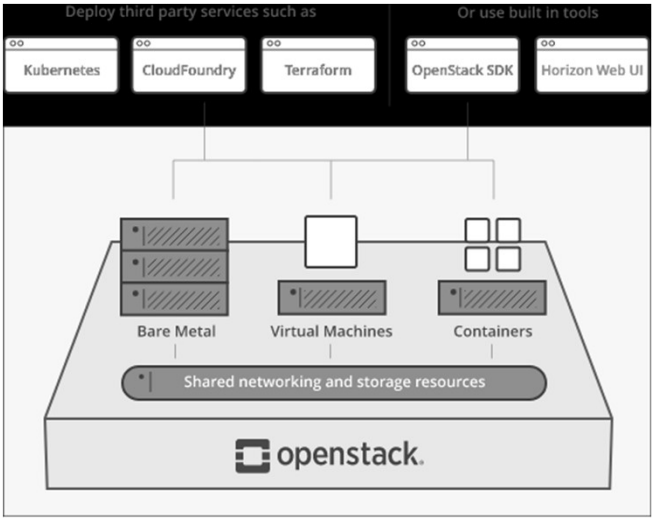
JS Lab

307

Opensource Networking
james@jslab.kr

❖ 별첨 2. 오픈스택

❖ WHAT IS OPENSTACK?



https://www.openstack.org/software/

JS Lab

308

❖ 별첨 2. 오픈스택

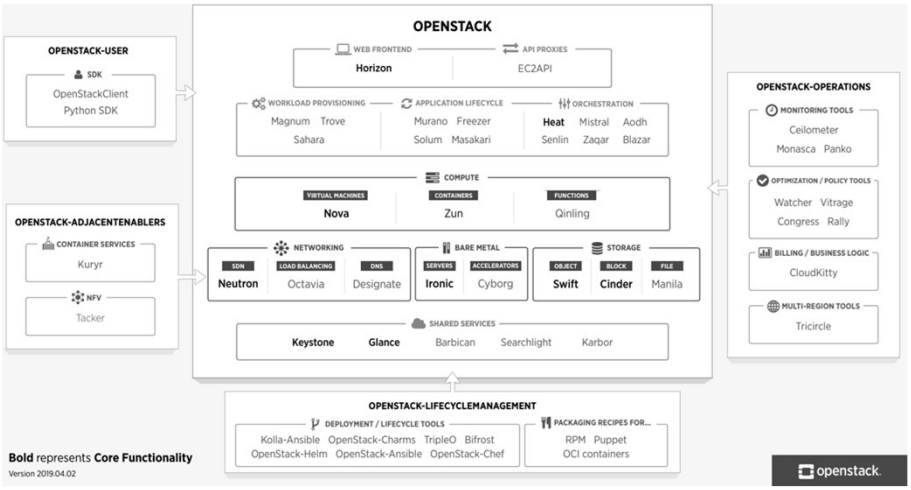
❖ WHAT IS OPENSTACK?

- 클라우드 환경에서 컴퓨팅 자원과 스토리지 인프라를 셋업하고 구동하기 위해 사용하는 오픈 소스 소프트웨어 프로젝트의 집합
- OpenStack은 공용 (Public) 클라우드와 사설 (Private) 클라우드 구축을 가능하게 하는 오픈 소스 소프트웨어
- OpenStack은 서버, 스토리지, 네트워크들과 같은 자원들을 모두 모아, 이들을 제어하고 운영하기 위한 클라우드 Operating System
- OpenStack은 오픈 소스를 기반으로 클라우드를 구축하고 운용하고자 하는 오픈 소스 개발자, 회사, 사용자들이 주축이 되어 발전하는 커뮤니티
- IaaS 형태의 클라우드 컴퓨팅 오픈 소스 프로젝트로 컴퓨팅, 스토리지, 네트워킹 자원을 관리하는 여러 개의 하위 프로젝트들로 이루어짐

https://www.openstack.org/software/

❖ 별첨 2. 오픈스택

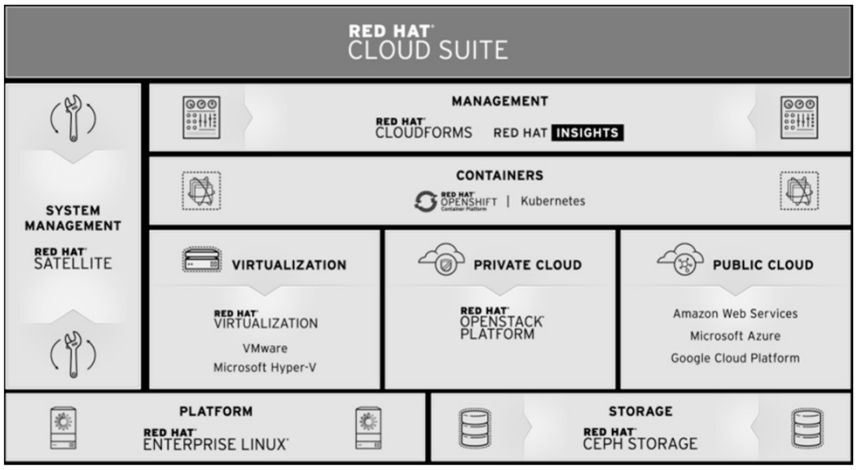
❖ THE OPENSTACK LANDSCAPE



https://www.openstack.org/software/

❖ 별첨 2. 오픈스택

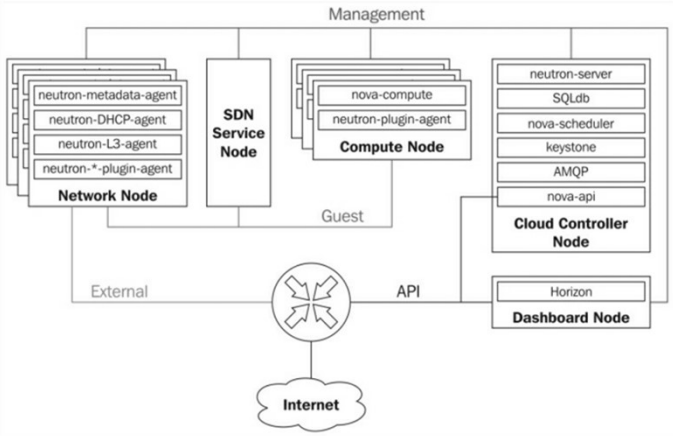
❖ 상용화 (예: Redhat)



❖ 별첨 2. 오픈스택

❖ 오픈스택(OpenStack) 네트워크 설계 (예)

- 네트워크를 분리 (관리/Guest/외부/API)
- VLAN 사용시 트렁크 모드 사용을 피하거나 네트워크를 물리적으로 나누는 것이 필요



Locati, Fabio Alessandro. OpenStack Cloud Security

Opensource Networking

james@jslab.kr

별첨

1. 리눅스 네트워킹
2. 오픈스택
3. 컨테이너 네트워킹
4. 쿠버네티스 네트워킹
5. OPNFV
6. 5G 코어네트워킹
7. AI 네트워킹 인프라

JS Lab

313

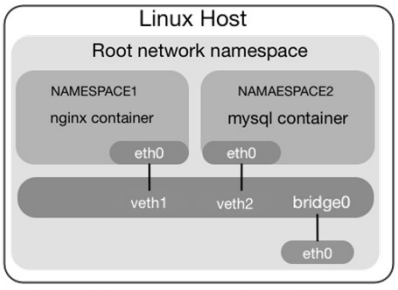
Opensource Networking

james@jslab.kr

❖ 별첨 3. 컨테이너 네트워킹

❖ 컨테이너 네트워킹(Container Networking) 종류

- **Container Network Model (CNM):** Docker libnetwork에서 사용하며 Cisco Contiv / Kuryr, OVN, Project Calico / VMware / Vwave에서 사용
- **Container Network Interface (CNI):** CoreOS에서 사용하며 kuberntes / Kurma / rkt / Apache Mesos / Cloud Foundry / Cisco Contiv / Project Calico / Weave



CNI 의 주요 플러그인은 Main과 IPAM;

1) **Main 플러그인:** Network 네임스페이스로 'veth pair'를 생성하여 내부에서 컨테이너와 브릿지등을 연결

2) **IPAM(ip management) 플러그인**: IP세팅과 할당

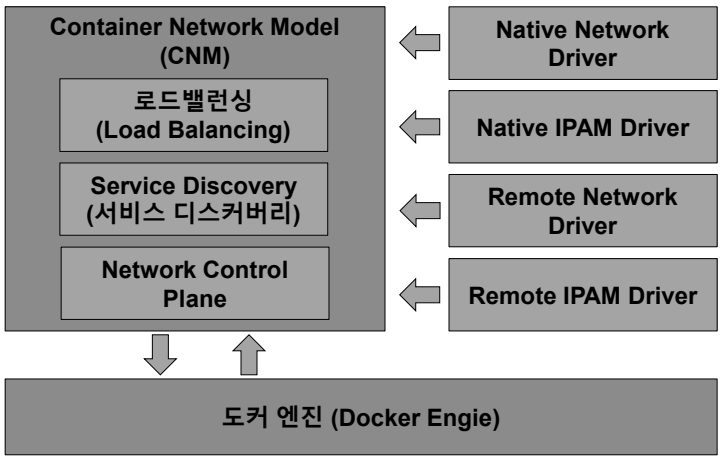
- https://github.com/containernetworking/cni/blob/master/scripts/docker-run.sh#L8-L20
- http://murat1985.github.io/kubernetes/cni/2016/05/14/netns-and-cni.html

JS Lab

314

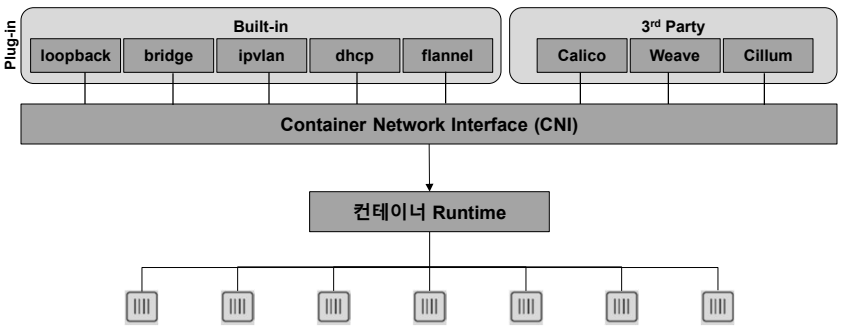
❖ 별첨 3. 컨테이너 네트워킹

❖ Container Network Model (CNM)



❖ 별첨 3. 컨테이너 네트워킹

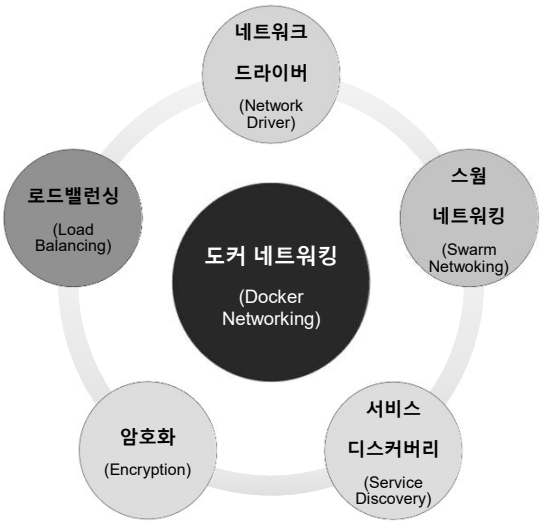
❖Container Network Interface (CNI)



https://github.com/containernetworking/cni

❖ 별첨 3. 컨테이너 네트워킹

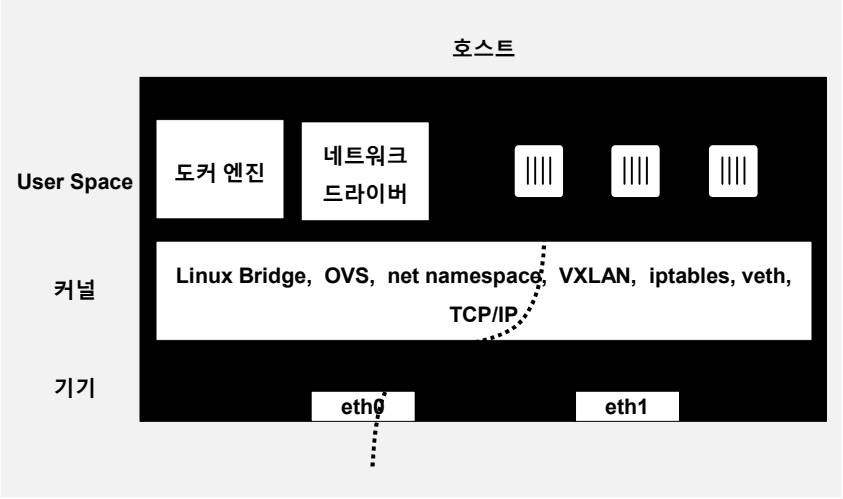
❖ 도커 네트워킹



317

❖ 별첨 3. 컨테이너 네트워킹

❖ 도커 네트워킹은 리눅스 네트워킹



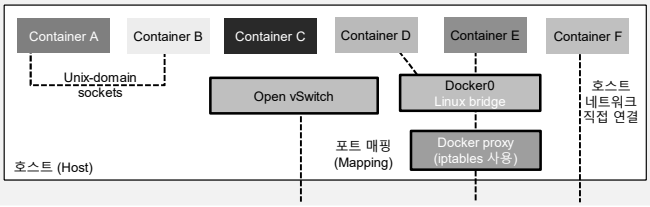
IPAM: IP Address Management

318

❖ 별첨 3. 컨테이너 네트워킹

❖ 하이레벨 (High-level) 기능

Namespace	/proc에서 프로세스 수준 관리의 컨테이너 네트워킹
Linux Bridge	커널에서 포워딩에 사용하는 L2/MAC을 인식하는 스위치
Open vSwitch	프로그램 가능하고 터널링을 지원하는 개선한 브릿지 (SDN 스위치)
NAT	네트워크 주소 변화 IP address + Ports (Types: SNAT, DNAT)
iptables	커널 내의 정책 엔진으로 패킷전송, 방화벽, NAT를 관리함
Unix domain sockets	단일 호스트 내 통신 기반의 File descriptor, FIFO 파이프로 동작
User-space vs Kernel-space	자원과 성능을 정상화 제어하는 애플리케이션도메인 • 컨테이너(Container) 애플리케이션(applications)은 user-space 에서 실행 • 네트워크 전송은 kernel space에서 실행



❖ 별첨 3. 컨테이너 네트워킹

❖ 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**

	오버레이	브릿지 / 포트맵핑	언더레이
멀티 호스트 연결	Yes	No (native support)	No (native support)
서비스 발견 (Service Discovery)	클러스터 간의 글로벌 SD	호스트 네트워크 상의 로컬 SD	호스트 네트워크 상의 로컬 SD
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

❖ 별첨 3. 컨테이너 네트워킹

❖ 도커 네트워크 드라이버 종류

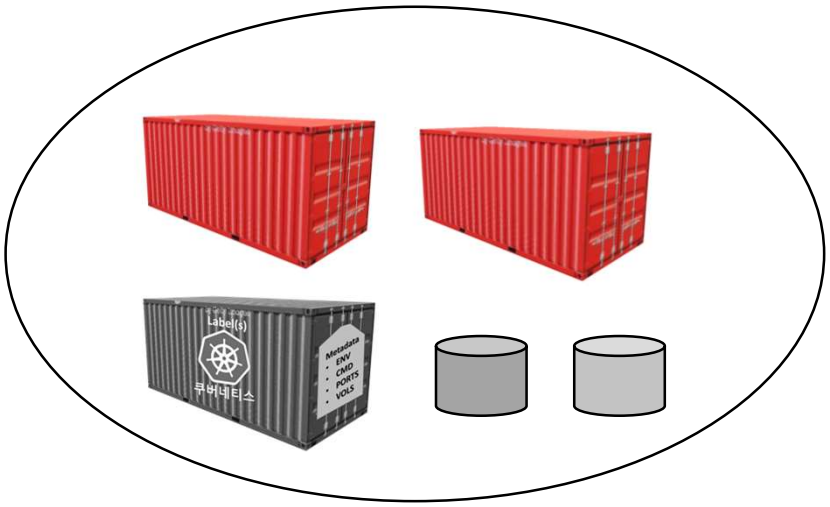
드라이버/ 기능	브릿지 (Bridge)	User Defined Bridge	호스트 (Host)	오버레이 (Overlay)	Macvlan/ ipvlan
연결	동일 호스트	동일 호스트	동일 호스트	멀티 호스트	멀티 호스트
서비스 디스커버리 / DNS	'links' 사용, DNS 사용 /etc/hosts	도커엔진에서 DNS 서버 사용	도커엔진에서 DNS 서버 사용	도커엔진에서 DNS 서버 사용	도커엔진에서 DNS 서버 사용
외부 접속	NAT	NAT	호스트 게이트웨이 사용	외부 접속 없음	언더레이 게이트웨이 사용
Namespace	분리	분리	동일 호스트	분리	분리
스웜 모드	미지원	미지원	미지원	지원	미지원
캡슐화	이중 캡슐화 없음	이중 캡슐화 없음	이중 캡슐화 없음	VxLAN 사용 더블 캡슐화	이중 캡슐화 없음
애플리케이션	노스(North), 사우스(South) 외부 접속	노스(North), 사우스(South) 외부 접속	모든 네트워킹 제어, 분리 필요 없음	호스트간 컨테이너 연결	컨테이너는 언더레이 네트워킹 직접 연결 필요

별첨

1. 리눅스 네트워킹
2. 오픈스택
3. 컨테이너 네트워킹
4. 쿠버네티스 네트워킹
5. OPNFV
6. 5G 코어네트워킹
7. AI 네트워킹 인프라

❖ 별첨 4. 쿠버네티스 네트워킹

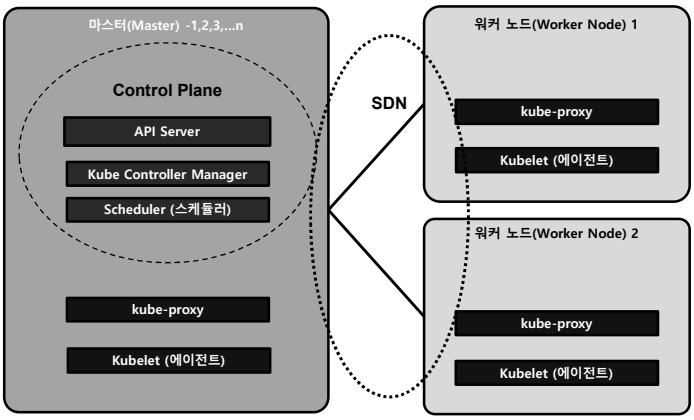
❖ **Pod:** 쿠버네티스 관리의 기본 단위



323

❖ 별첨 4. 쿠버네티스 네트워킹

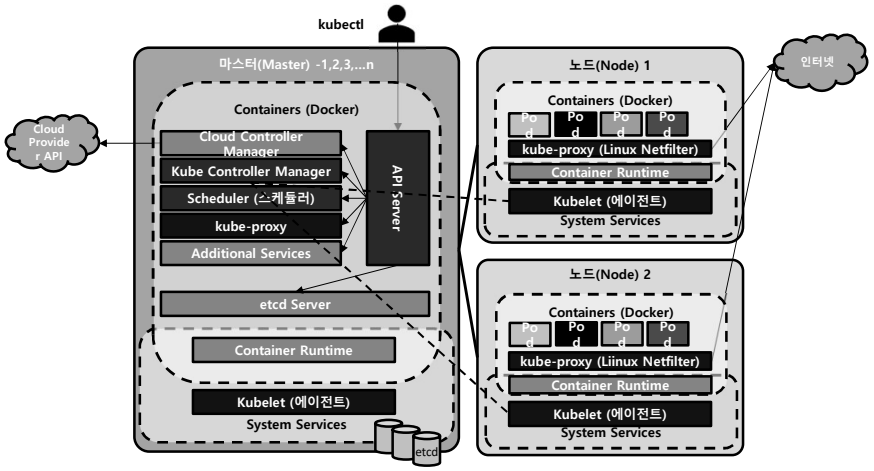
❖ **기본 구성:** HA 가능 마스터 노드(Master Node)와 실제 앱이 구동하는 컨테이너들의 Pod를 호스팅하는 1개 이상의 워커 노드(Worker Node)로 클러스터(Cluster) 구성



324

❖ 별첨 4. 쿠버네티스 네트워킹

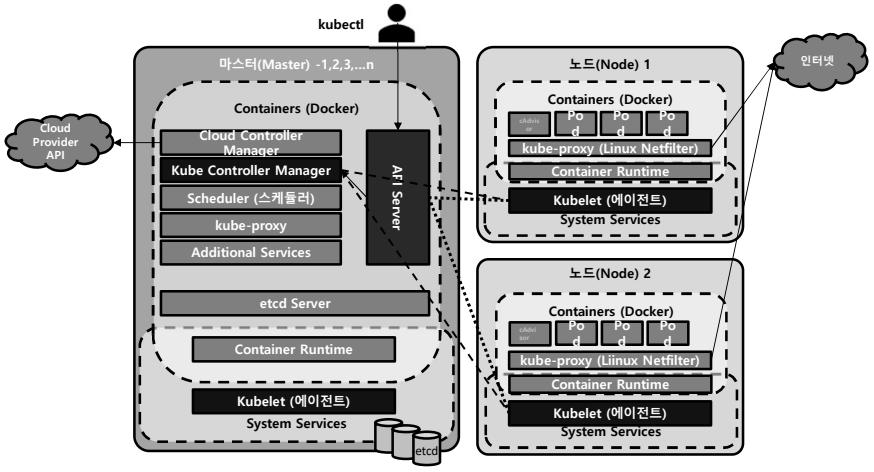
❖ **아키텍처 구성:** 마스터 노드(Master Node)와 1개 이상의 워커 노드(Worker Node)로 이뤄진 가상/물리 머신의 꾸러미(Set)를 통해 클러스터(Cluster) 구성



325

❖ 별첨 4. 쿠버네티스 네트워킹

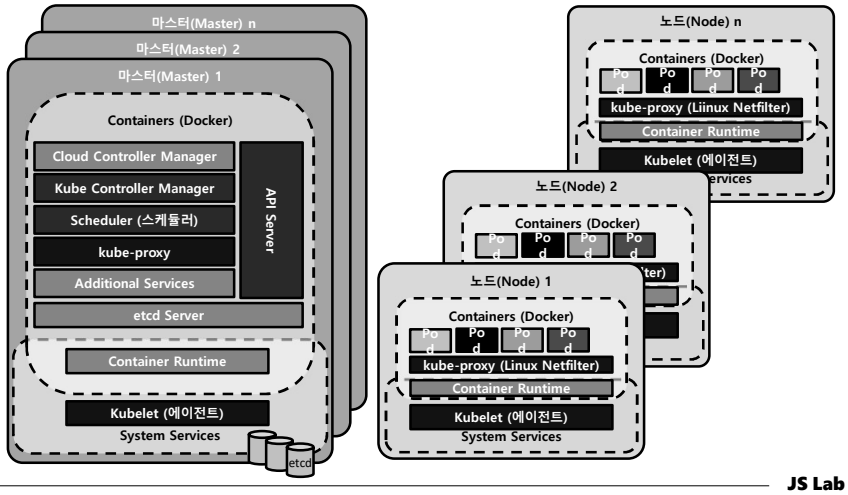
❖ **kubelet**



326

❖ 별첨 4. 쿠버네티스 네트워킹

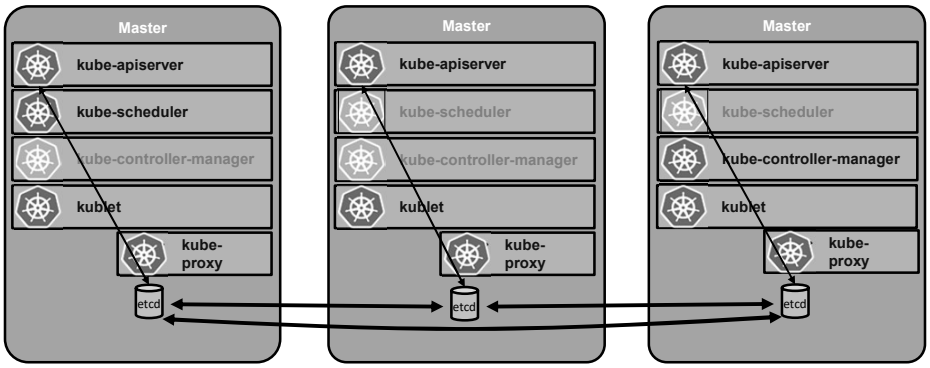
❖ **kube-proxy:** Master 노드는 agent(Kubelet)를 포함한 모든 기능을 포함하고 워커 노드 와 1개 이상의 Worker 노드는 agent와 외부 통신을 위한 kube-proxy 등으로 구성 가능



327

❖ 별첨 4. 쿠버네티스 네트워킹

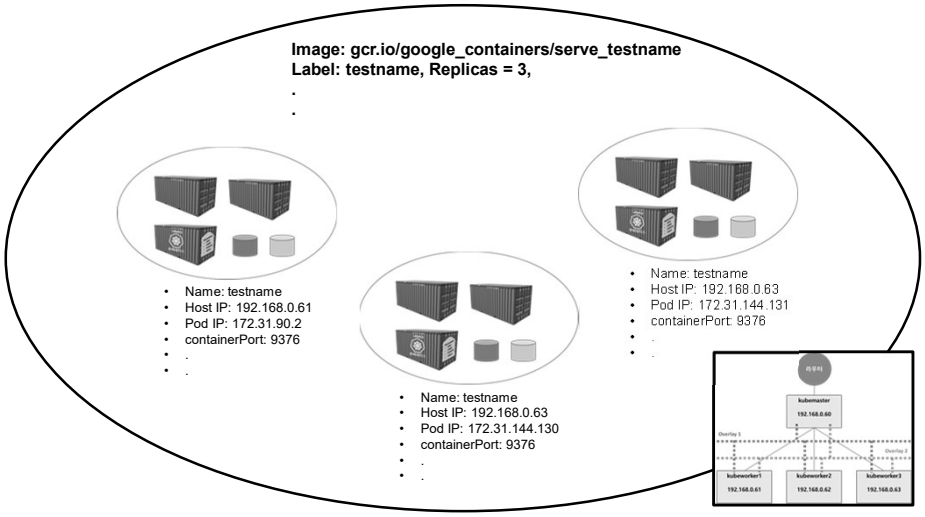
❖ **Master 노드의 HA:** Kube-scheduler와 kube-controller-manager는 Master의 HA 구성에서도 1개만 Active



328

❖ 별첨 4. 쿠버네티스 네트워킹

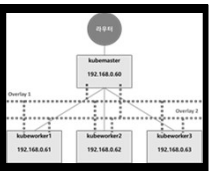
❖ **Pod 구성(예): 3**개 노드(Worker Node)에 3개 Pod 복제(Replica) 생성 구성 (예)



❖ 별첨 4. 쿠버네티스 네트워킹

❖ **설치 구성(예):** 쿠버네티스(K8s) 설치 호스트에서 노드와 Pod 구성 확인

```
[root@kubemaster k8s-test]# kubectl get nodes
NAME          STATUS    ROLES     AGE       VERSION
kubemaster    Ready     master    5h        v1.9.4
kubeworker1   Ready     <none>    5h        v1.9.4
kubeworker2   Ready     <none>    5h        v1.9.4
kubeworker3   Ready     <none>    5h        v1.9.4
[root@kubemaster k8s-test]#

[root@kubemaster kube-cni-calico]# kubectl get pods -n kube-system -o wide
NAME                                       READY     STATUS    RESTARTS   AGE       IP              NODE
calico-etcd-fxt29                          1/1       Running   0          3h        192.168.0.60    kubemaster
calico-kube-controllers-6cfcf554b9-wbkdg   1/1       Running   0          3h        192.168.0.61    kubeworker1
calico-node-657t5                          2/2       Running   0          3h        192.168.0.60    kubemaster
calico-node-6p6xs                          2/2       Running   1          3h        192.168.0.61    kubeworker1
calico-node-dscx6                          2/2       Running   1          3h        192.168.0.63    kubeworker3
calico-node-hvddc                          2/2       Running   0          3h        192.168.0.62    kubeworker2
etcd-kubemaster                            1/1       Running   2          3h        192.168.0.60    kubemaster
kube-apiserver-kubemaster                  1/1       Running   0          3h        192.168.0.60    kubemaster
kube-controller-manager-kubemaster         1/1       Running   0          3h        192.168.0.60    kubemaster
kube-dns-6f4fd4bdf-6c9kp                   3/3       Running   0          3h        172.31.90.1     kubeworker1
kube-proxy-2cbqv                           1/1       Running   0          3h        192.168.0.61    kubeworker1
kube-proxy-7vd7b                           1/1       Running   0          3h        192.168.0.63    kubeworker3
kube-proxy-hcdbh                           1/1       Running   0          3h        192.168.0.60    kubemaster
kube-proxy-pr9dp                           1/1       Running   0          3h        192.168.0.62    kubeworker2
kube-scheduler-kubemaster                  1/1       Running   0          3h        192.168.0.60    kubemaster
[root@kubemaster kube-cni-calico]#
```



❖ 별첨 4. 쿠버네티스 네트워킹

❖ **K8s를 위한 도커 네트워크 구성(예)**

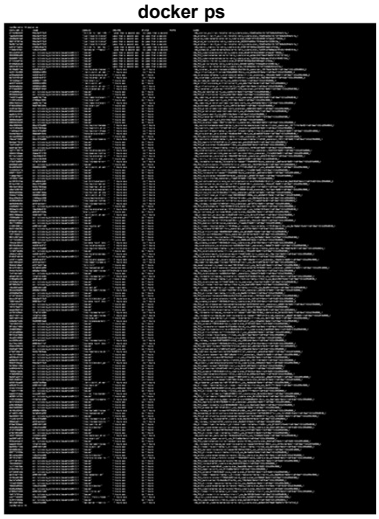
❖ 별첨 4. 쿠버네티스 네트워킹

❖ **K8s를 위한 도커 컨테이너 (예):** 도커 호스트에서 K8s Master/Worker 구성 확인

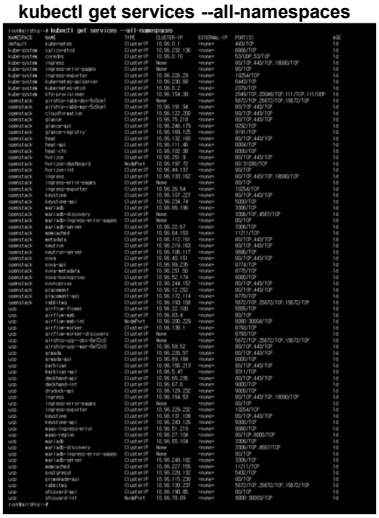
❖ 별첨 4. 쿠버네티스 네트워킹

❖ Docker and Kubernetes for Airship

docker ps



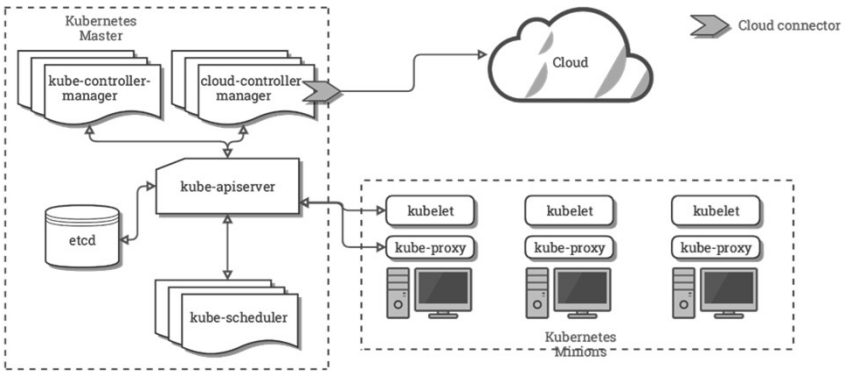
kubectl get services --all-namespaces



333

❖ 별첨 4. 쿠버네티스 네트워킹

❖ **Controller Manager :** Kubernetes controller manager (KCM)에서 Cloud controller manager (CCM)을 분리하여 별도의 프로세스로 구동하여 클라우드에 의존적 이던 KCM 환경을 개선



https://kubernetes.io/docs/concepts/architecture/cloud-controller/

334

❖ 별첨 4. 쿠버네티스 네트워킹

❖ **kube-scheduler:** 새로 생성한 Pod가 구동 하는 노드를 선택하고 감시하는 마스터의 구성 요소이며, 슈케듈링의 결정을 위해 자원 요구, 하드웨어 / 소프트웨어 / 정책의 억제(Constraint), 친화(affinity)와 비친화(anti-affinity)의 명시, 데이터의 위치, 워크로드간 간섭(interference)과 데드라인 요소 수집

❖ **스케듈링 방법:**

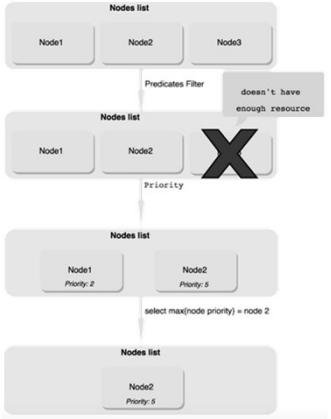
- **예측 가능:** 노드의 자원이나 성격으로 지정하며, Pod의 포트, Pod의 자원, 노드 지정 매치, 호스트 이름, 서비스 친화, 라벨
- **가중치 우선 순위(Priority) 지정:** 가중치에 최적인 노드를 사용 (최저 요구 우선, Balanced Resource Allocation, Service Spreading Priority, Equal Priority)

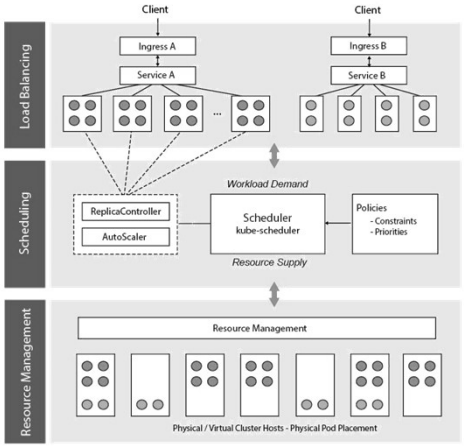
affinity / anti-affinity
nodeSelector
taints / tolerations
reservation / limits

http://blog.kubernetes.io/2017/03/advanced-scheduling-in-kubernetes.html

❖ 별첨 4. 쿠버네티스 네트워킹

❖ **스케듈링 (예):** 컨테이너가 실행 할 노드를 결정하며, CPU/메모리/동작중인 컨테이너 수를 기반으로 고려하여 배치

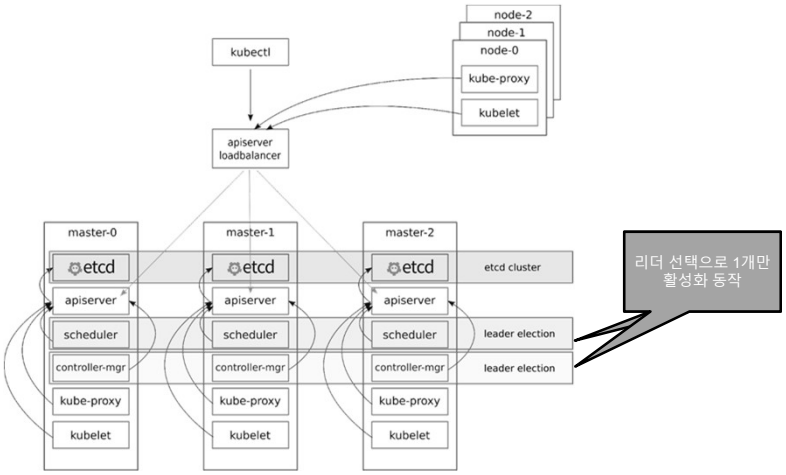




http://dockone.io/article/2885 http://rancher.com/three-pillars-kubernetes-container-orchestration/

❖ 별첨 4. 쿠버네티스 네트워킹

❖ **etcd (Distributed key-value store):** 컨피규레이션, 네임스페이스, 레플리케이션 등의 pod/service등의 상태 저장과 DNS 데이터 저장에 사용



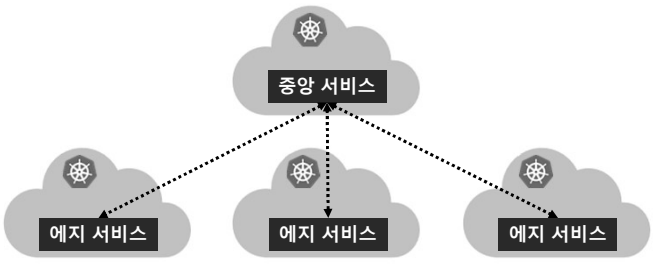
https://elastisys.com/2018/01/25/setting-highly-available-kubernetes-clusters/

337

❖ 별첨 4. 쿠버네티스 네트워킹

❖ **Kubernetes:**

- **K8s는 워크로드 Agnostic** (컨테이너, VM, Function)
- **K8s는 다양한 요구의 하드웨어 플랫폼 지원**
- **K8s는 역동적 서비스를 위한 앱의 이동과 처리 증가와 감소의 장점**
- **K8s는 상용 적용 확장을 위한 일관된 플랫폼의 검증**
- **K8s는 신개발이 필요하지 않은 수준의 많은 레퍼런스로 개발자가 익숙함**
- **기기 관리 필요** (PXE, DHCP, IPMI, TFTP, Discovery)



338

별첨

1. 리눅스 네트워킹
2. 오픈스택
3. 컨테이너 네트워킹
4. 쿠버네티스 네트워킹
5. OPNFV
6. 5G 코어네트워킹
7. AI 네트워킹 인프라

Opensource Networking
james@jslab.kr

❖ 별첨 5. OPNFV

❖ **OPNFV** (**O**pen **P**latform for **NFV**)

- **통신 사업자 주도로 2014년 10월 출범**
- **NFV (Network Function Virtualization)를 사용하는 신제품이나 서비스를 빠르게 도입 할 수 있도록 하기 위한 캐리어급 통합 오픈 소스 플랫폼** (Carrier-Grade, Integrated, Open Source Platform)
- **Linux 재단의 협업 프로젝트** (Linux Foundation Collaborative Project)

https://www.opnfv.org/

Opensource Networking
james@jslab.kr

❖ 별첨 5. OPNFV

❖ OPNFV Mission

- 다양한 오픈 소스 생태계 전반의 NFV 구성 요소 개발 및 발전을 촉진
- 시스템 레벨 통합, 배포 및 테스트를 통해 엔터프라이즈 및 서비스 사업자 네트워크의 변환을 가속화하는 NFV 플랫폼 레퍼런스를 생성

https://www.opnfv.org/about/mission

❖ 별첨 5. OPNFV

❖ Goals for OPNFV

- **오픈 소스 플랫폼 개발:** NFV 기능 구축에 사용할 수 있는 통합적이고 검증 된 오픈 소스 플랫폼 개발 (새로운 제품 및 서비스의 도입 가속화)
- **사용자의 참여:** OPNFV가 사용자 커뮤니티의 요구 사항을 충족하는지 검증하기 위해 선도적 인 최종 사용자의 참여 포함
- **운용성 확보:** OPNFV 플랫폼에서 활용 될 관련 오픈 소스 프로젝트 구성 요소 간의 일관성, 성능 및 상호 운용성 확보
- **생태계 구축:** 최종 사용자의 요구를 충족시키기 위한 개방형 표준 및 소프트웨어에 기반한 NFV 솔루션을 위한 생태계 구축
- **홍보:** 오픈 소스 NFV를 위한 선호 플랫폼 및 커뮤니티로서 OPNFV 홍보

https://www.opnfv.org/about/mission

❖ 별첨 5. OPNFV

❖ OPNFV Scope

- 초기 OPNFV 프로젝트에서는 Upstream Project 들과 통합을 통하여 NFV Infrastructure (NFVI) 와 Virtualized Infrastructure Manager (VIM) 구현을 목표로 함
- NFVI, VIM 의 API (Application Programmable Interface)를 제공하여 다른 NFV Elements (e.g. VNF Manager, VNFs, Orchestrator) 와 연동이 가능하도록 함

❖ 별첨 5. OPNFV

❖ OPNFV 는 4개의 하부 프로젝트 분야로 구성

- 요구사항 프로젝트 (Requirements)
  - ✓ NFV Reference Platform 개발을 위해 필요한 요구사항에 대해 수집 및 문서화를 위한 프로젝트
  - ✓ 정의된 요구사항은 OPNFV Community 나 Upstream Project 에서 구현
- 통합 및 검증 프로젝트 (Integration & Testing)
  - ✓ NFV Reference Platform 구축에 사용되는 다양한 Open Source Project (e.g. OpenStack, OpenDaylight)에 대한 통합 및 검증 수행을 위한 프로젝트
- 협력개발 프로젝트 (Collaborative Development)
  - ✓ NFV Reference Platform 개발을 위해 다른 Open Source Project 그룹 및 표준화 단체등과 협력 개발을 위한 프로젝트
- 문서화 프로젝트 (Documentation)
  - ✓ OPNFV 와 관련된 문서화 및 문서 작성 도구 개발을 위한 프로젝트

https://wiki.opnfv.org/display/PROJ

❖ 별첨 5. OPNFV

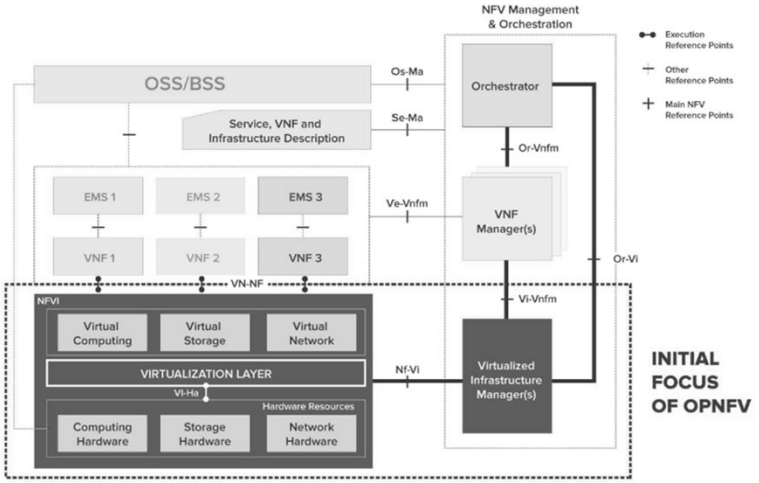
❖ 업스트림(Upstream) OPNFV 커뮤니티(Community)

- OPNFV has an “upstream first” philosophy.
- Do not look to fork upstream projects or create OPNFV specific versions
- Analysis of NFV requirements or testing activities, we discover gaps, desired features, or bugs, we will participate in the appropriate community process of the upstream organization to incorporate blueprints, patches, and other changes.

https://www.opnfv.org/community/upstream-projects

❖ 별첨 5. OPNFV

❖ NFV reference architecture framework



‘부록 a. NFV Basic’ 참고

ETSI GS NFV 002 v1.2.1 (2014-12)

❖ 별첨 5. OPNFV

❖ **OPNFV provides consumable releases every six months**

- **Hunter** - Released May 14, 2019
- **Gambia** - Updated Aug 15, 2018
- **Fraser** - Updated on Oct. 10, 2017
- **Euphrates** - Updated Oct. 10, 2017
- **Danube** - Released April 4, 2017
- **Colorado** - Released September 26, 2016
- **Brahmaputra (EOL)** - Released March 1, 2016
- **Arno (EOL)** - Launched on June 4, 2015

https://www.opnfv.org/community/upstream-projects

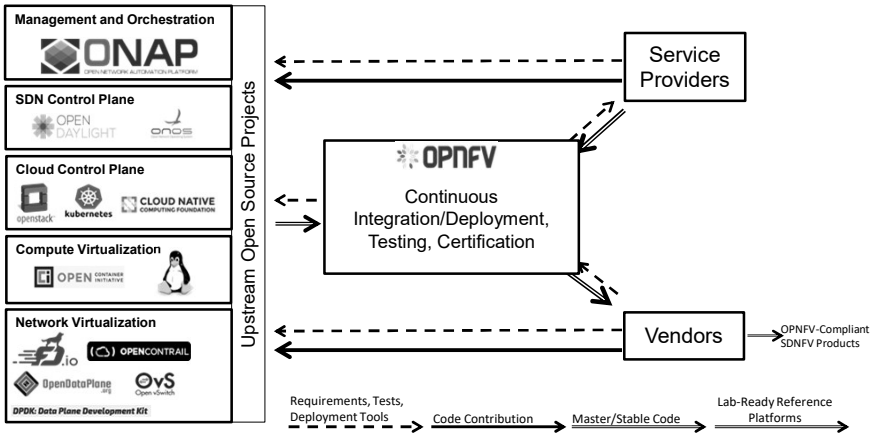
❖ 별첨 5. OPNFV

❖ **OPNFV Interaction with Upstream Communities and Downstream Projects/Users**

- **OPNFV Member 주로 전세계 주요 통신사와 통신장비 제조사들로 구성되어 있는 것이 특징**

❖ 별첨 5. OPNFV

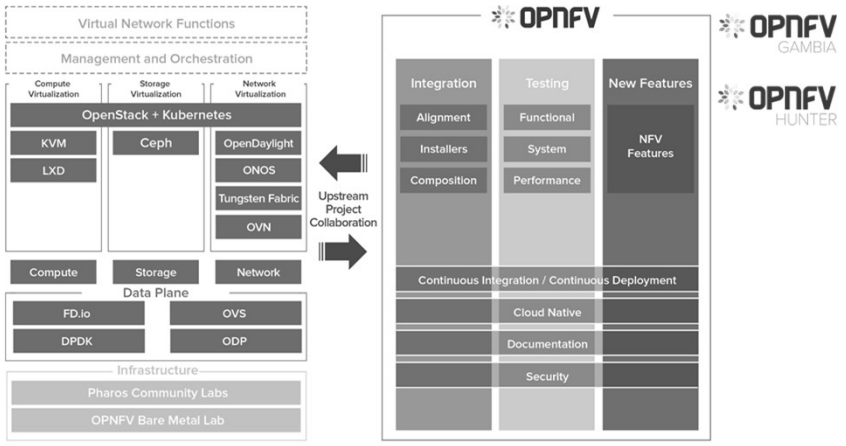
❖ OPNFV Interaction with Upstream Communities and Downstream Projects/Users



❖ 별첨 5. OPNFV

❖ OPNFV Current Release: Gambia 7.2 / Hunter 8.0

- **Architectural view of Gambia and Hunter**



https://www.opnfv.org/software

Opensource Networking

james@jslab.kr

별첨

1. 리눅스 네트워킹
2. 오픈스택
3. 컨테이너 네트워킹
4. 쿠버네티스 네트워킹
5. OPNFV
6. 5G 코어네트워킹
7. AI 네트워킹 인프라

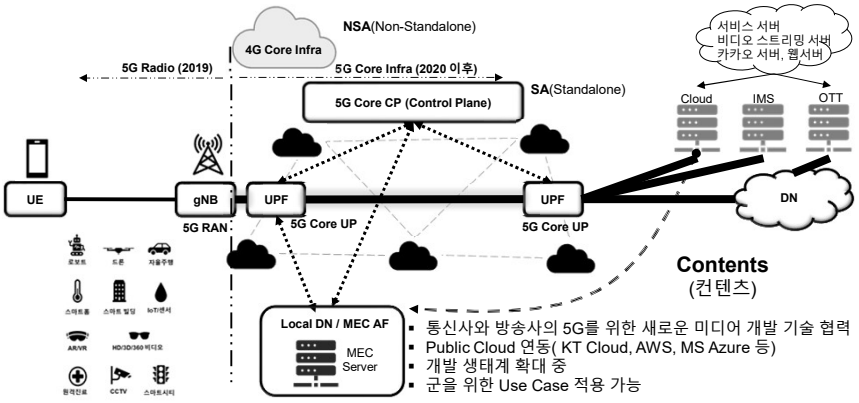
JS Lab

351

❖ 별첨 6. 5G 코어 네트워킹

❖ **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



- 통신사와 방송사의 5G를 위한 새로운 미디어 개발 기술 협력
- Public Cloud 연동(KT Cloud, AWS, MS Azure 등)
- 개발 생태계 확대 중
- 군을 위한 Use Case 적용 가능

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

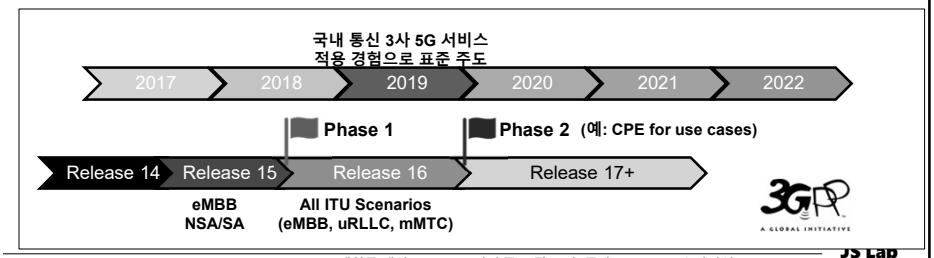
JS Lab

352

❖ 별첨 6. 5G 코어 네트워킹

❖ 5G 표준 Roadmap 고려

- **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)

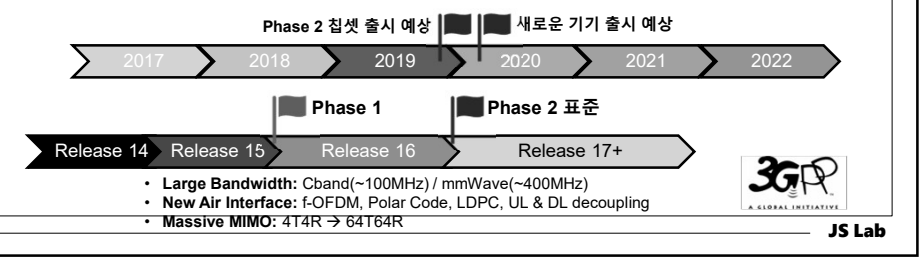
❖ 별첨 6. 5G 코어 네트워킹

❖ 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 (900MHz)	
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

❖ 별첨 6. 5G 코어 네트워킹

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

❖ **무선 등의 서비스 개선과 새로운 기능 추가**

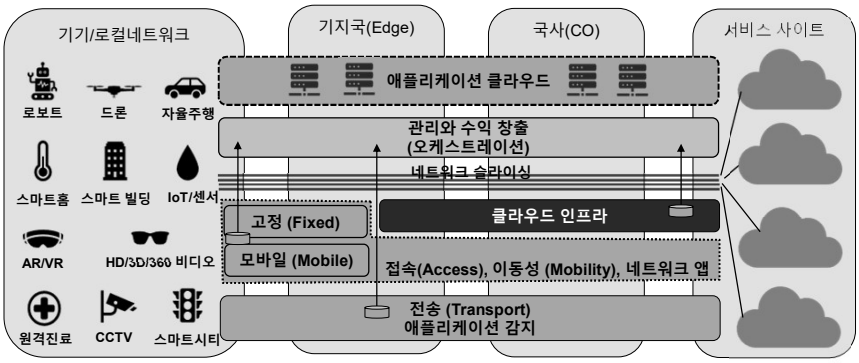
- **NR Light** – NR evolution
- **Small data transfer optimization**
- **Sidelink enhancements** – NR evolution
- **NR above 52.6 GHz (60GHz unlicensed)** – NR evolution
- **Multi SIM Operation**
- **NR multicast broadcast** (예: 재난망)
- **Coverage enhancements**
- **NB-Iot and eMTC enhancememts**
- **IIoT 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

❖ 별첨 6. 5G 코어 네트워킹

❖ **5G 코어 인프라(Core Infra)**

- **Edge(기지국)와 Central Office(국사)의 데이터센터 화 진행**
- **클라우드 네이티브화** (오픈소스 기반: 애플리케이션 서비스, 관리, 인프라)
- **네트워크 슬라이싱** (종단간 Network Slicing)
- **클러스터링 확장성 고려** (갯수등)



JS Lab

❖ 별첨 6. 5G 코어 네트워킹

❖ 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

357

❖ 별첨 6. 5G 코어 네트워킹

❖ Cloud Edge Computing: 단순 데이터센터 보다 큰 의미

❖ Akraino, Airship, StalingX (협력)

❖ Killer Service Solution 탑재 필요

OpenStack (코드 개발) — airship, STARLINGX → Linux Foundation (Use Case 정의, Integration, 검증) — AKRAINO EDGE STACK

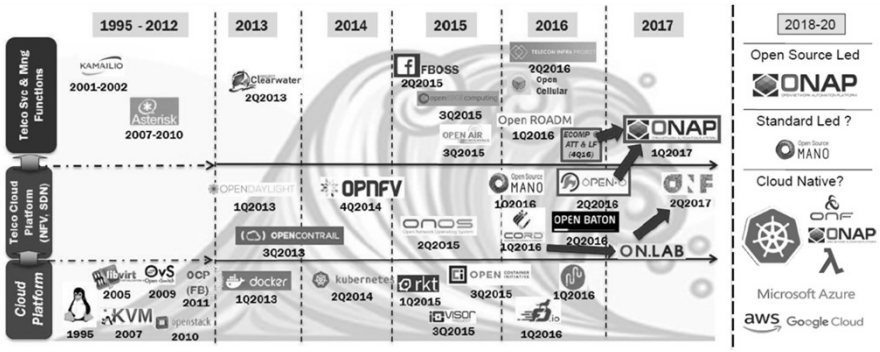
출처: https://wiki.openstack.org/wiki/Edge_Computing_Group?fbclid=IwAR3GNTB5_2lOJO-SvaGsmhCC2jhLxG9X-lSt021v-mIfG-TxsR7jiPtrM80

358

❖ 별첨 6. 5G 코어 네트워킹

❖ 운영지원 시스템(OSS)의 미래

- **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)

Opensource Networking james@jslab.kr

❖ 별첨 6. 5G 코어 네트워킹

❖ 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/

❖ 별첨 6. 5G 코어 네트워킹

❖ 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

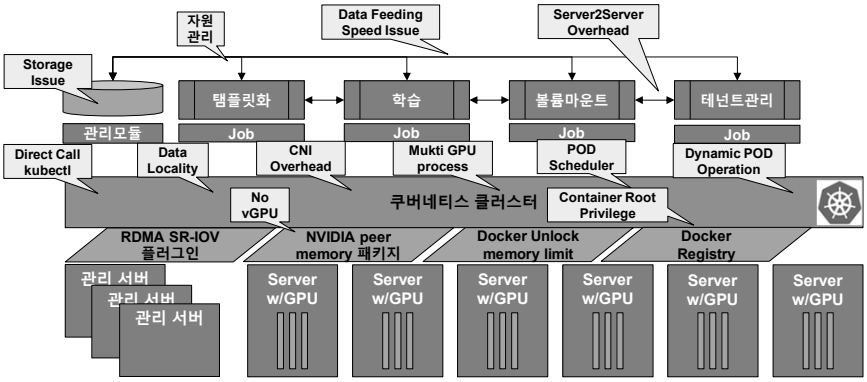
별첨

1. 리눅스 네트워킹
2. 오픈스택
3. 컨테이너 네트워킹
4. 쿠버네티스 네트워킹
5. OPNFV
6. 5G 코어네트워킹
7. AI 네트워킹 인프라

❖ 별첨 7. AI 네트워킹 인프라

❖ MSA 수용 클라우드 네이티브 아키텍처 'CNA' 체계

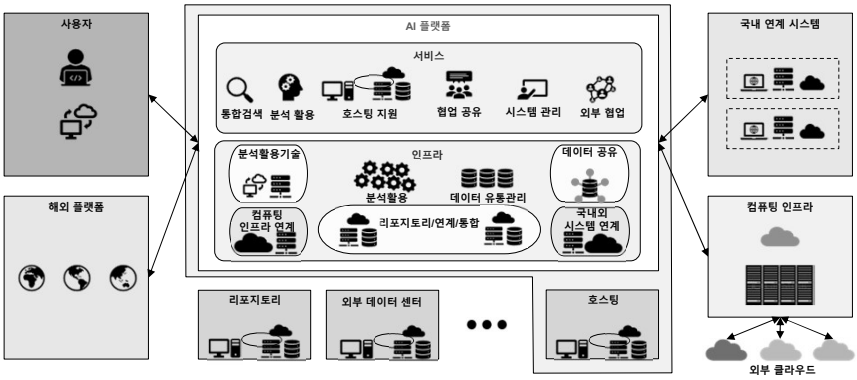
- K8s Based ML Platform



❖ 별첨 7. AI 네트워킹 인프라

❖ 클라우드 네이디브 기반 AI 플랫폼 서비스

- 플랫폼 서비스 제공
- 클라우드 서비스 인프라 기반 서비스
- 클라우드 서비스를 위한 SDDC 인프라 기반 추상화 계층 제공




Q
&
A
James Ahn @ james@jslab.kr
Community for KOREN AI Network Lab
JS Lab