DEVIEW
2019

퍼블릭 클라우드 서비스를 위한 네트워크 가상화 모델

정정화
NBP(Naver Business Platform)
NAVER

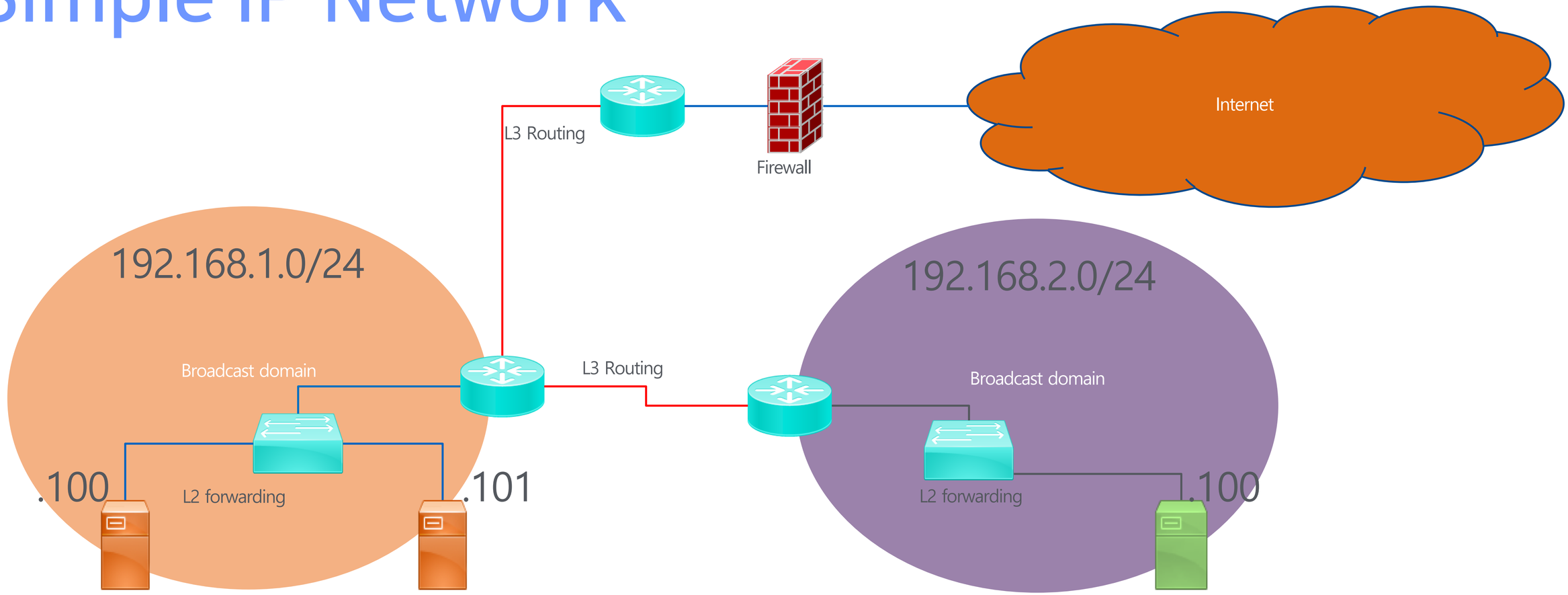
CONTENTS

DEVIEW
2019

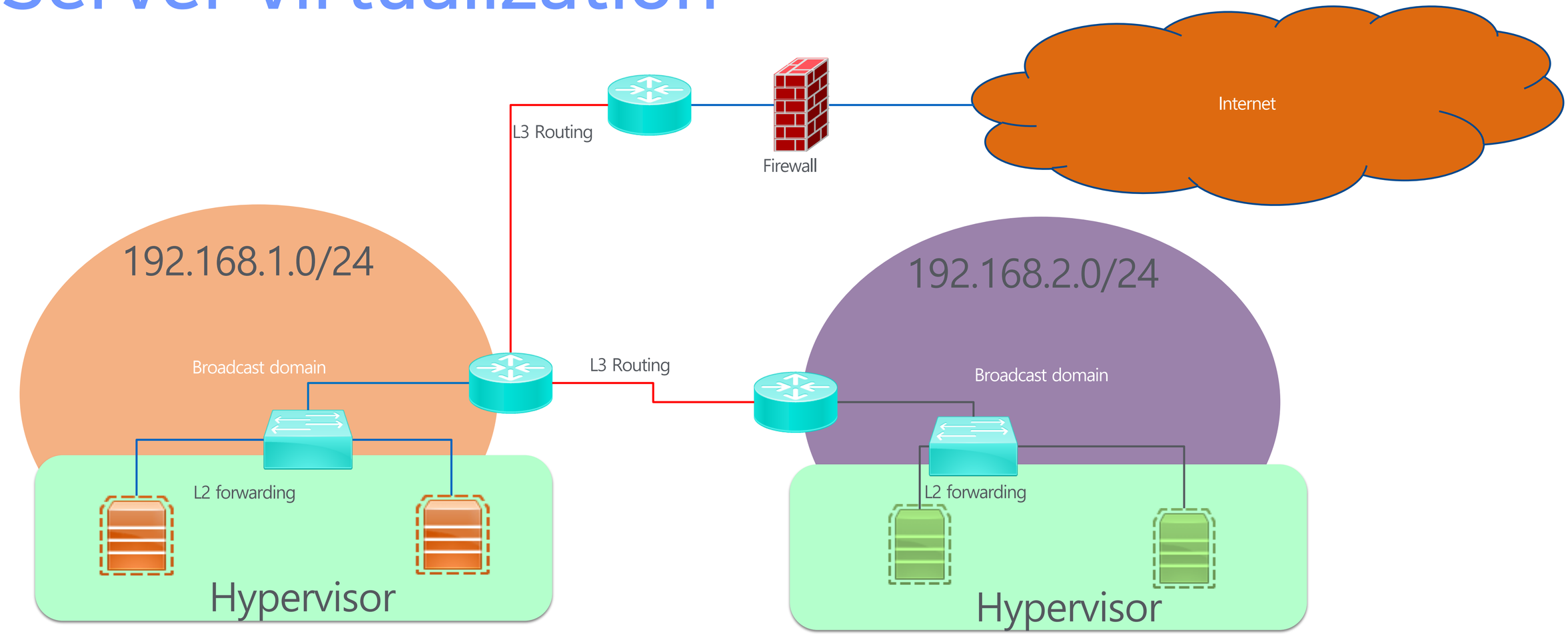
1.네트워크 가상화 필요성

Simple IP Network

DEVIEW 2019

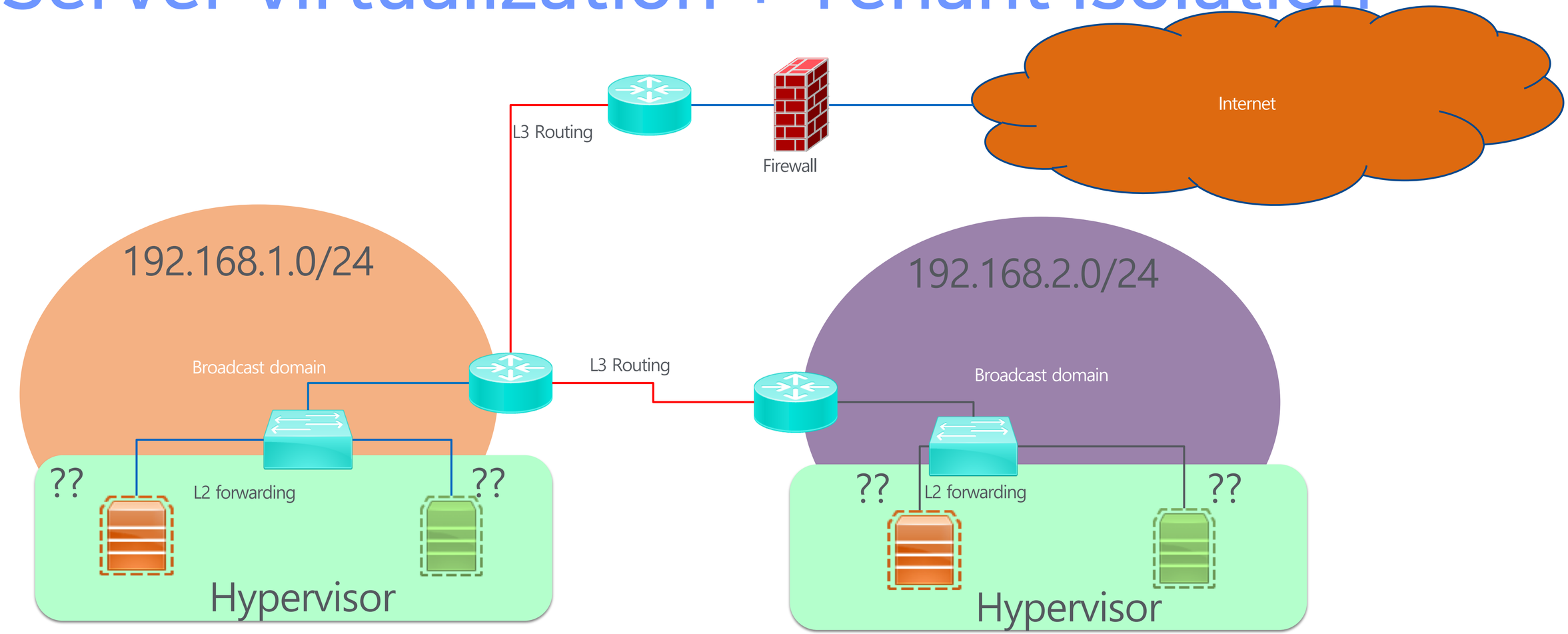


Server virtualization



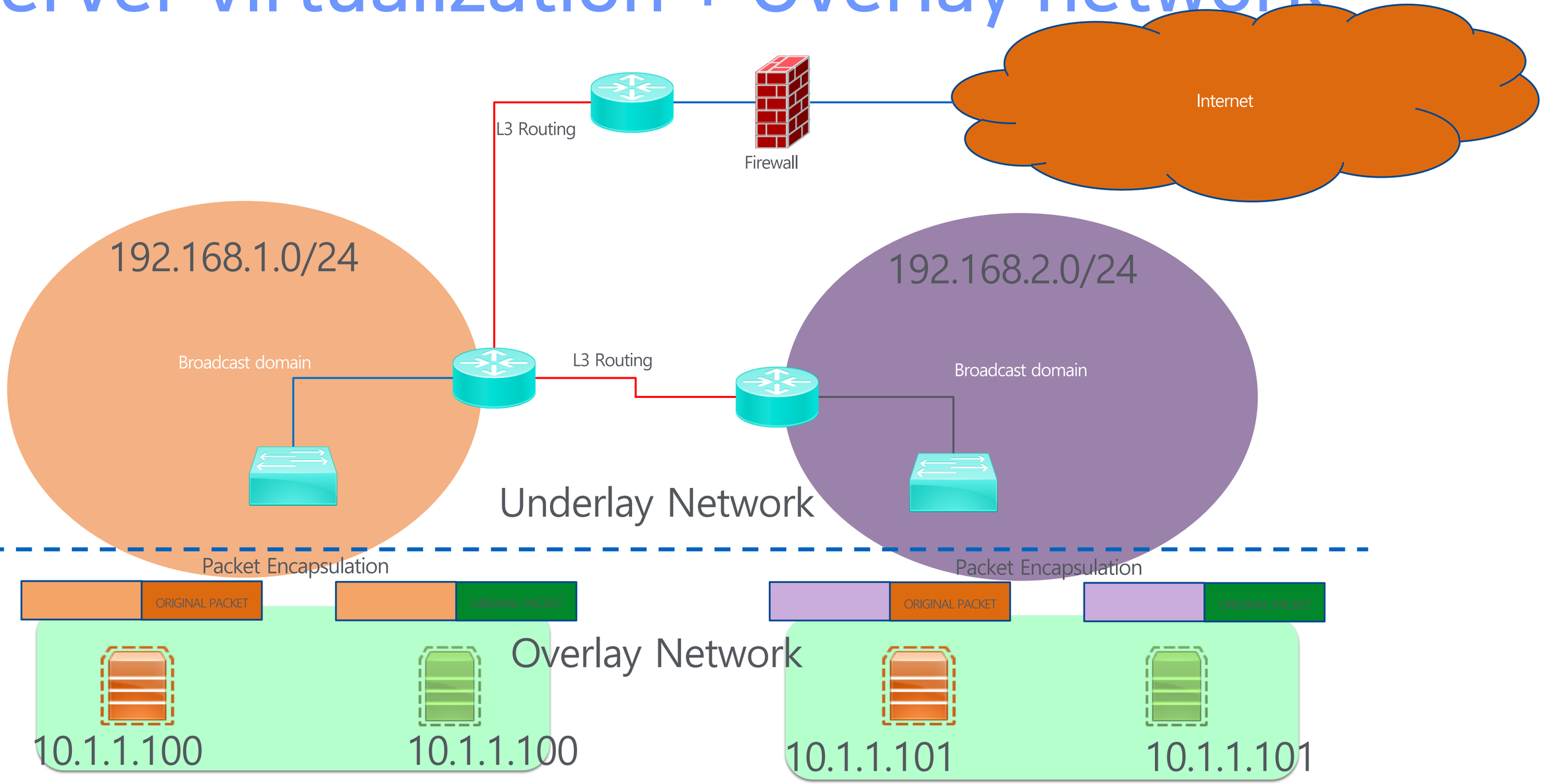
Server virtualization + Tenant isolation

DEVIEW 2019



Server virtualization + Overlay network

DEVIEW 2019



DEVIEW
2019

2.퍼블릭 클라우드를 위한 네트워크 가상화 모델

Challenge

Big 3 Public cloud service provider
- Amazon AWS, MS Azure, Google GCP
- Strong IaaS

NCP classic
- Flat + one additional subnet

NCP + VPC
- Multiple subnets
- ACG/NACL
- Public cloud service scale
- Performance(speed) + Scalability + Security

Core features

Tenant Isolation
- 고객간의 네트워크는 완전히 분리되고 서로 영향을 주면 안된다

Multiple Subnets
- 고객은 수많은 subnet을 가질 수 있다

IP Overlapping
- 고객의 네트워크는 같은 IP 대역을 문제 없이 사용해야 한다

ACL/Security Group
- VM, 네트워크 별로 세밀한 접근 제어가 가능해야 한다

Routing
- Destination 기반이 아닌 다양한 조합의 routing이 가능해야 한다

Network Endpoints
- 외부 네트워크와 연결하는 다양한 접점을 제공해야 한다
- NAT, 1:1 NAT, IPSEC, SSLVPN, Peering, Cloud connection, Tenant간 연결, ...

Private vs Public Cloud network service

~수천 tenants vs ~수백만 tenants

Small dataplane vs Big dataplane

Hypervisor based ACG, ACL vs Dataplane ACG, ACL

Simple routing vs Complex routing

Intenal connectivity vs Strong External connectivity

...

Cloud solutions

DEVIEW 2019

Openstack, Cloudstack

*SX

*CI

****rail

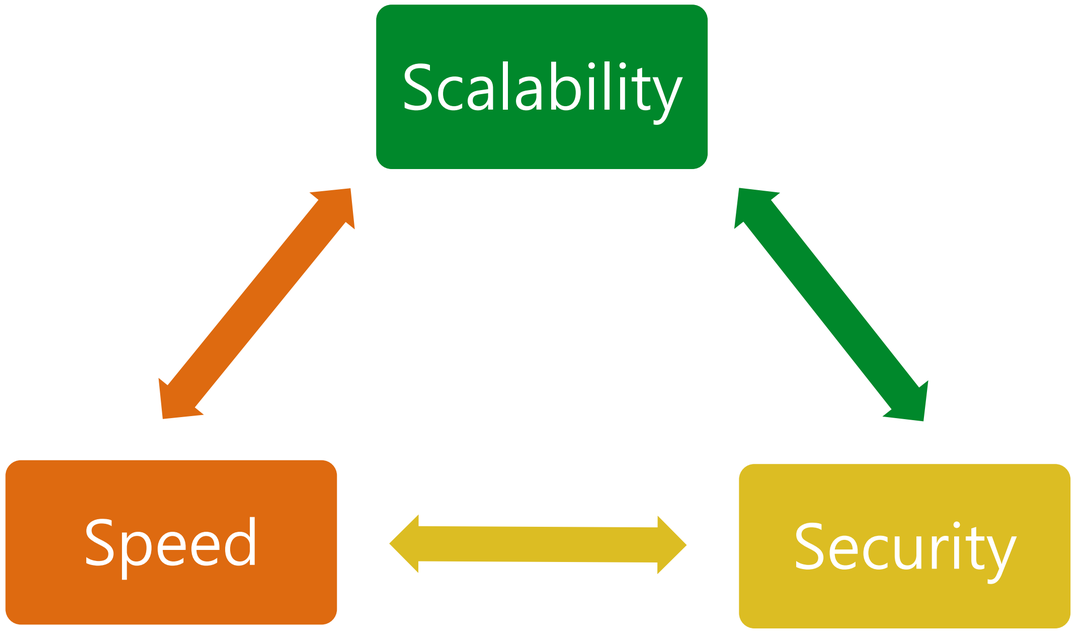
Suitable for private cloud or hybrid cloud scale

검토 결과

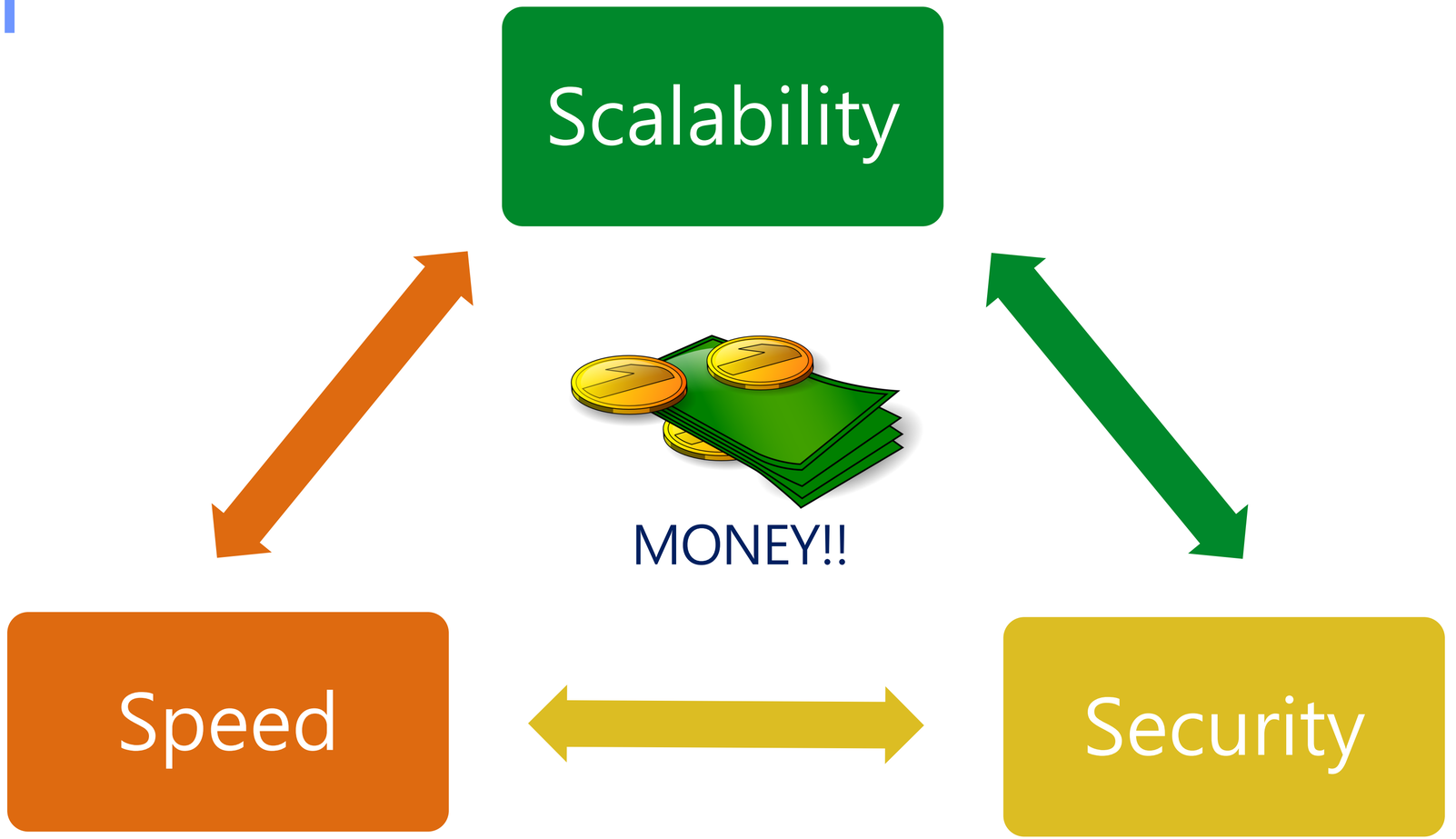
기존 솔루션은 퍼블릭 클라우드 스케일의 네트워크를 제공하기 어렵다

필요하면 만들어 보자

3S



3S+1M



3.데이터 플레인 고려 사항

Cloud Dataplane Options

1. H/W dataplane

 H/W dataplane

2. S/W dataplane

 Linux networking stack

  - TUN/TAP, VETH, SR-IOV, MACVLAN, IPVLAN, VxLAN, GRE

 Openvswitch

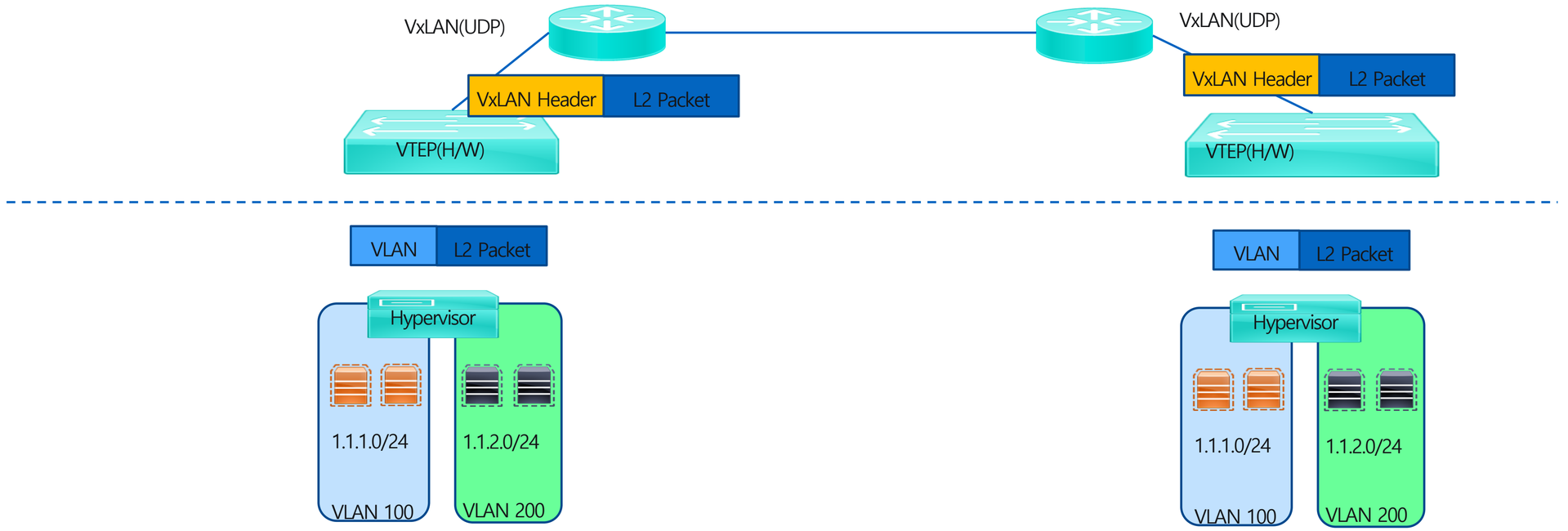
 DPDK

Choose a right overlay protocol

Protocol	Identifier	Overhead
VLAN(802.1Q)	12bit = 4K customer	16bits
VLAN-VLAN(802.1ad)	12bit + 12bits (Customer ID + Inner VLAN)	32bits
GRE	P-t-P	4bytes + 20bytes (GRE Header + IP Header)
NVGRE	24bit VSID	
STT	64bit	
GENEVE	24bit	
VxLAN	24bit(16777216)	20 + 8 + 8 = 36bytes (IPv4 + UDP + VxLAN)

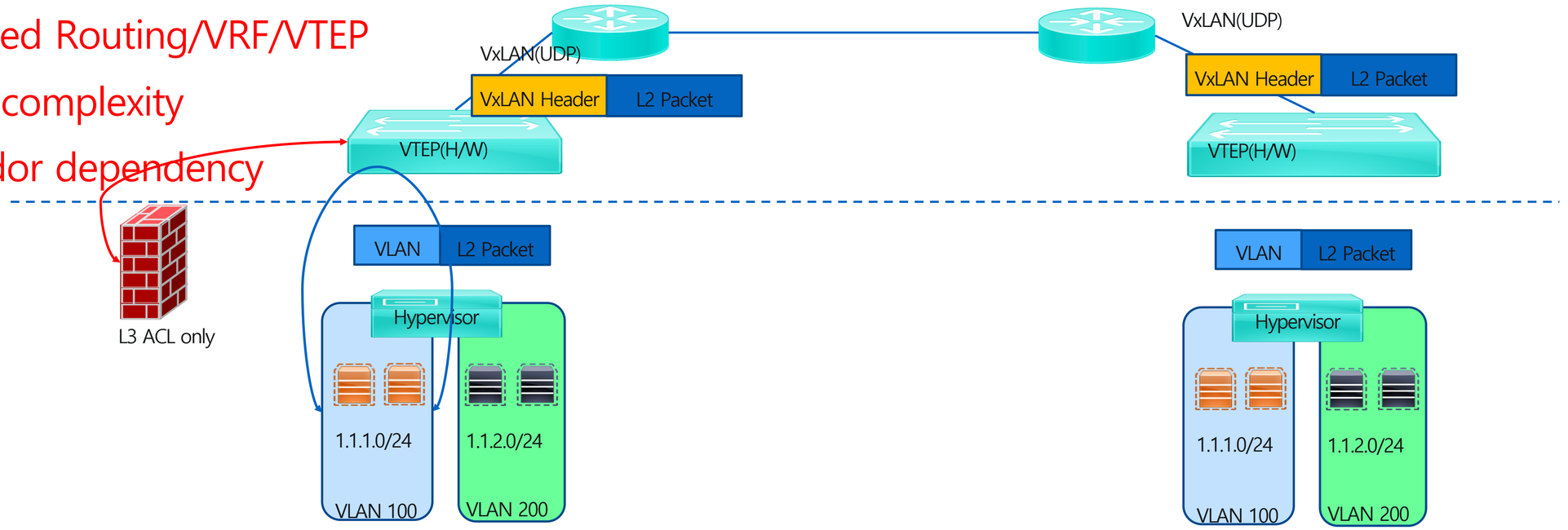
H/W dataplane

첫번째 시도

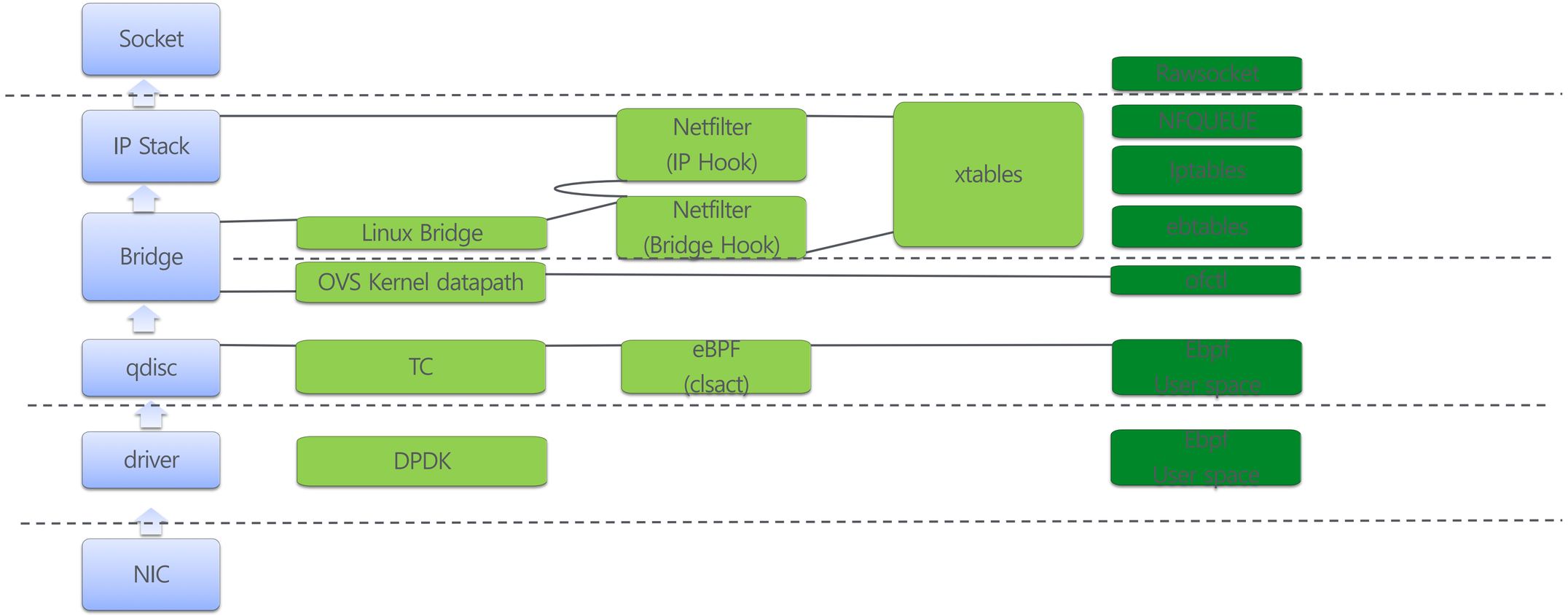


H/W dataplane

- Linerate dataplane
- NO ACG for Inter-VM(L3 ACL only)
- Limited Routing/VRF/VTEP
- BGP complexity
- Vendor dependency

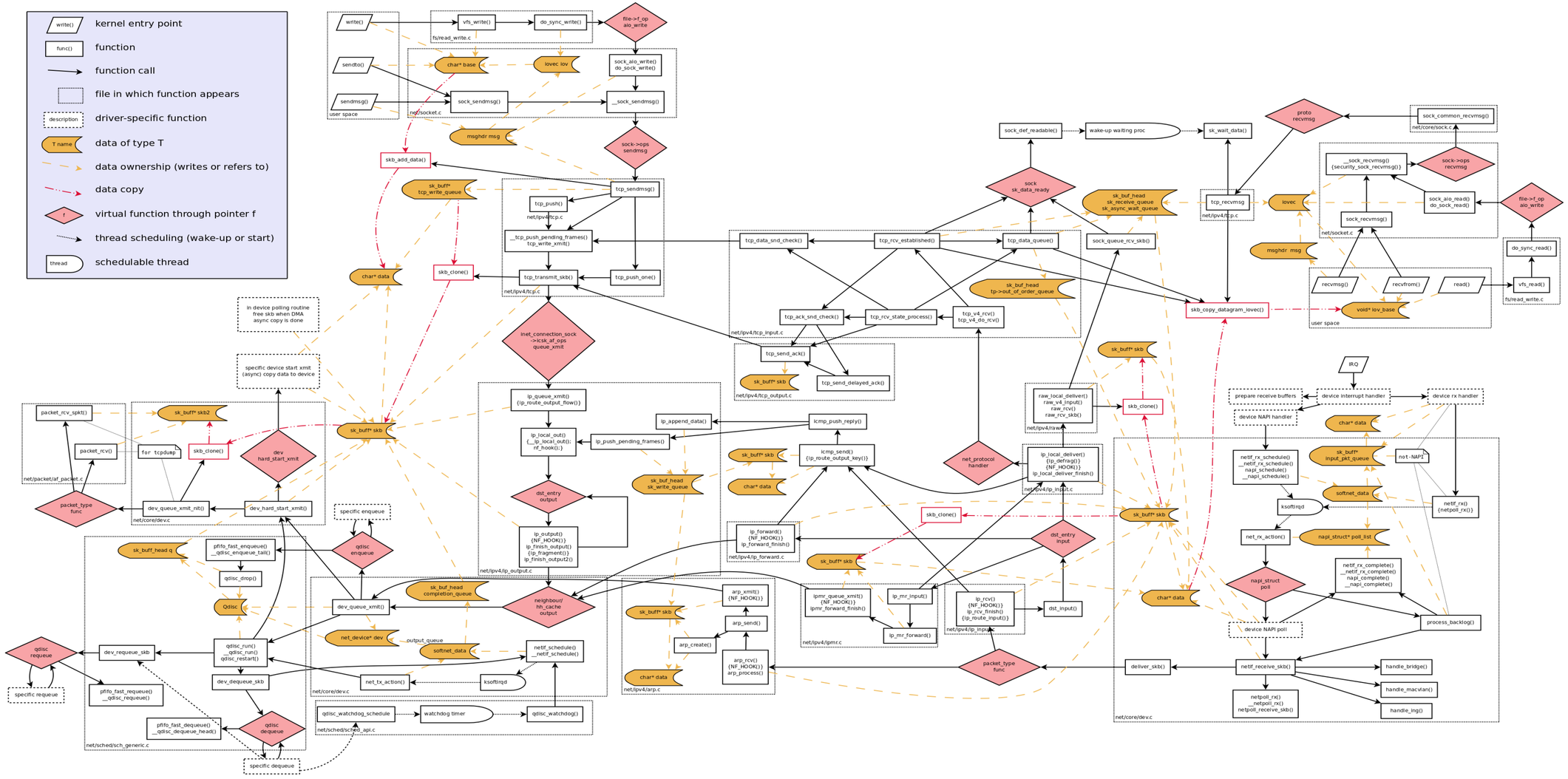


Linux packet flow



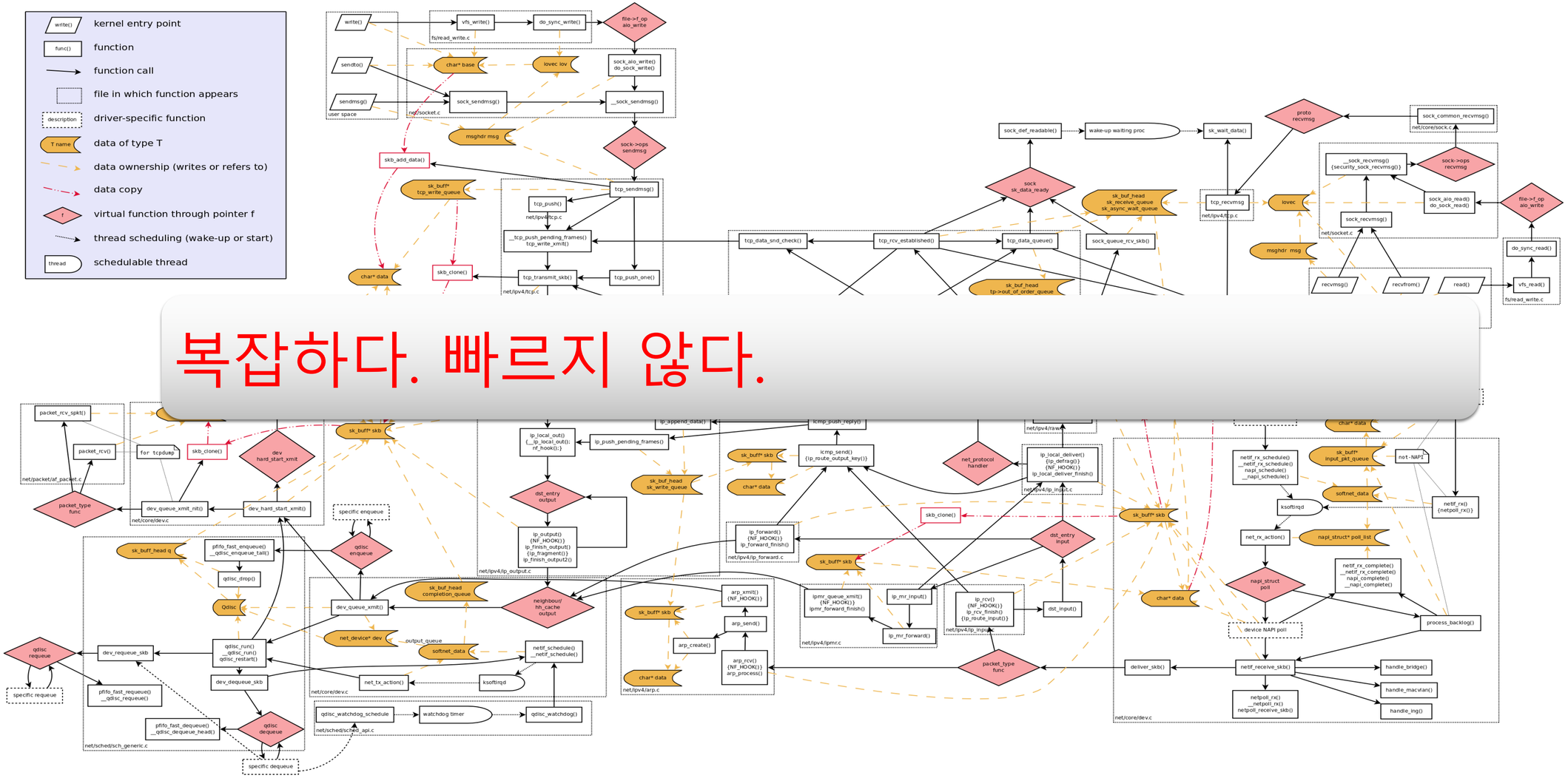
Linux networking stack

DEVIEW 2019



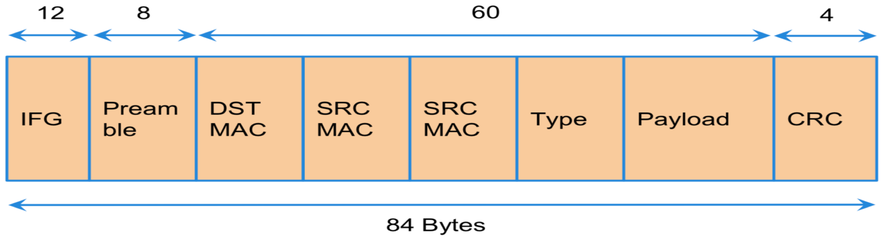
Linux networking stack

DEVIEW 2019



Linerate vs Linux

14.88 millions of 64 byte packets per second on 10G interface
1.8 GHz -> 1 cycle = 0,55 ns
1 packet -> 67.2 ns = 120 clock cycles



Function	Activity	Time (ns)
sendto	system call	96
sosend_dgram	lock sock_buff, alloc mbuf, copy in	137
udp_output	UDP header setup	57
ip_output	route lookup, ip header setup	198
ether_otput	MAC lookup, MAC header setup	162
ixgbe_xmit	device programming	220
Total		**950**

PAYLOAD	ETHER SIZE	PPS	Packet budget	CPU cycle required
46	84	14,880,952	67.2	121.0
64	102	12,254,902	81.6	146.9
128	166	7,530,120	132.8	239.0
256	294	4,251,701	235.2	423.4
512	550	2,272,727	440.0	792.0
1,024	1,062	1,177,024	849.6	1529.3
1,500	1,538	812,744	1,230.4	2214.7
4,096	4,134	302,371	3,307.2	5953.0
8,192	8,230	151,883	6,584.0	11851.2

Linux native features

Linux bridge

Netfilter framework

Namespace, VRF, VETH, MACVLAN, IPVLAN, GRE, VxLAN

다양한 기능이 있고 유연하지만

- 작은 패킷 처리 성능이 떨어짐
- 구성이 복잡함
- 서버 가상화 내부 데이터플레인으로 적합

Openvswitch

Linux IP stack 미사용

    netdev_rx_handler_register

Flow based approach(Match, Instruction, Action)

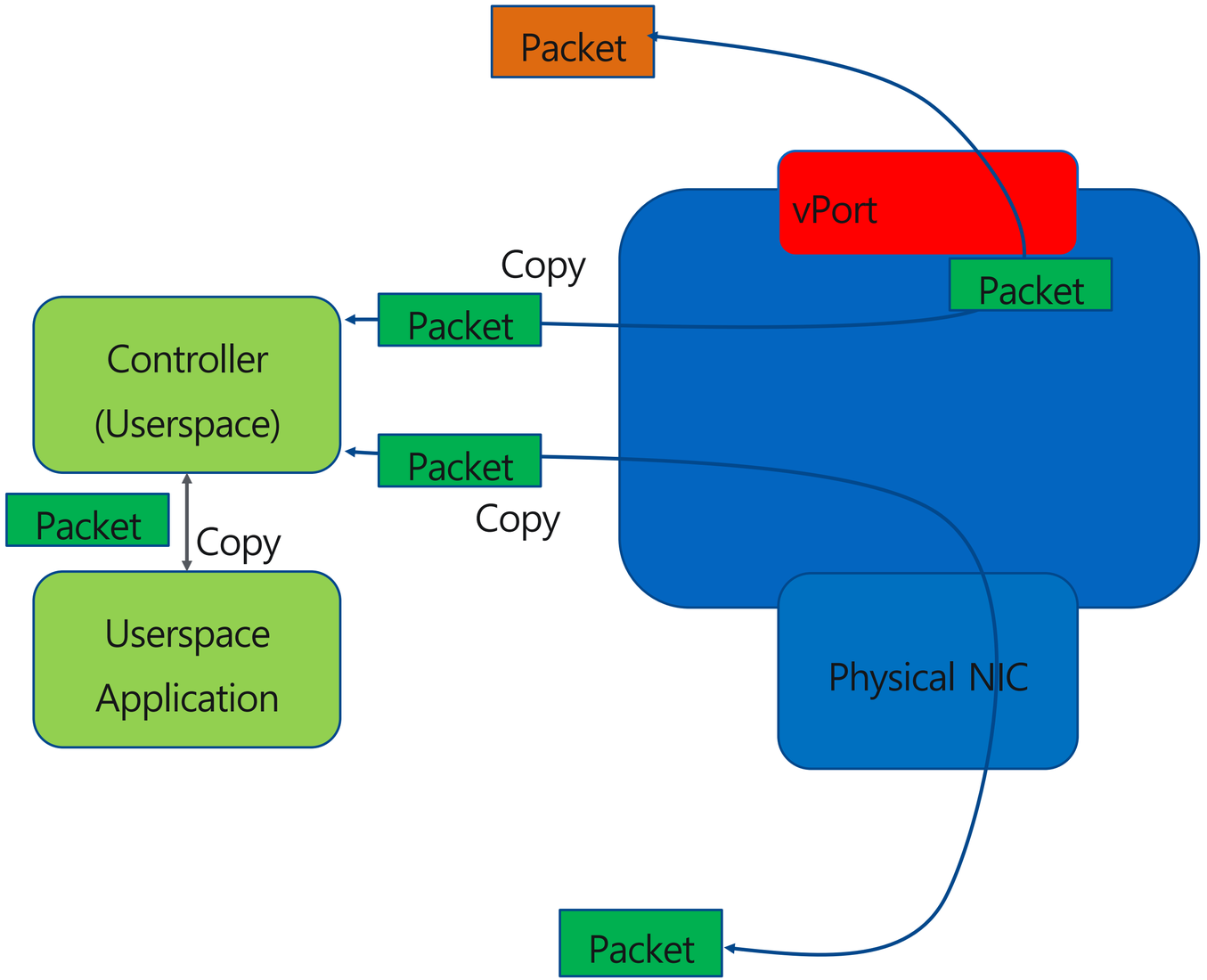
다양한 프로토콜 지원

    IP, GRE, VxLAN, GENEVE, MPLS, ...

De facto S/W dataplane, controlplane

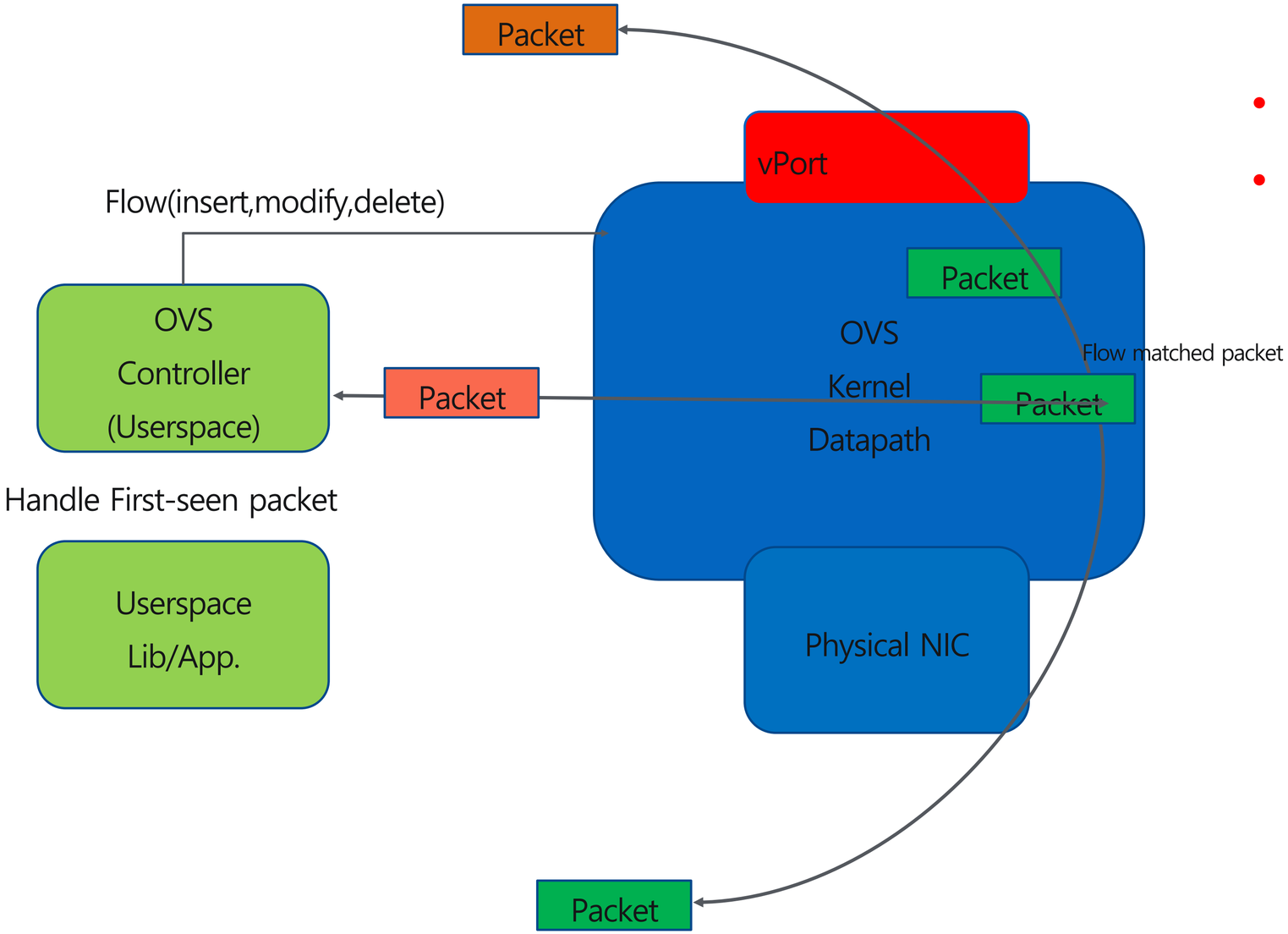
수많은 레퍼런스(Openstack!)

OVS Controller based packet flow



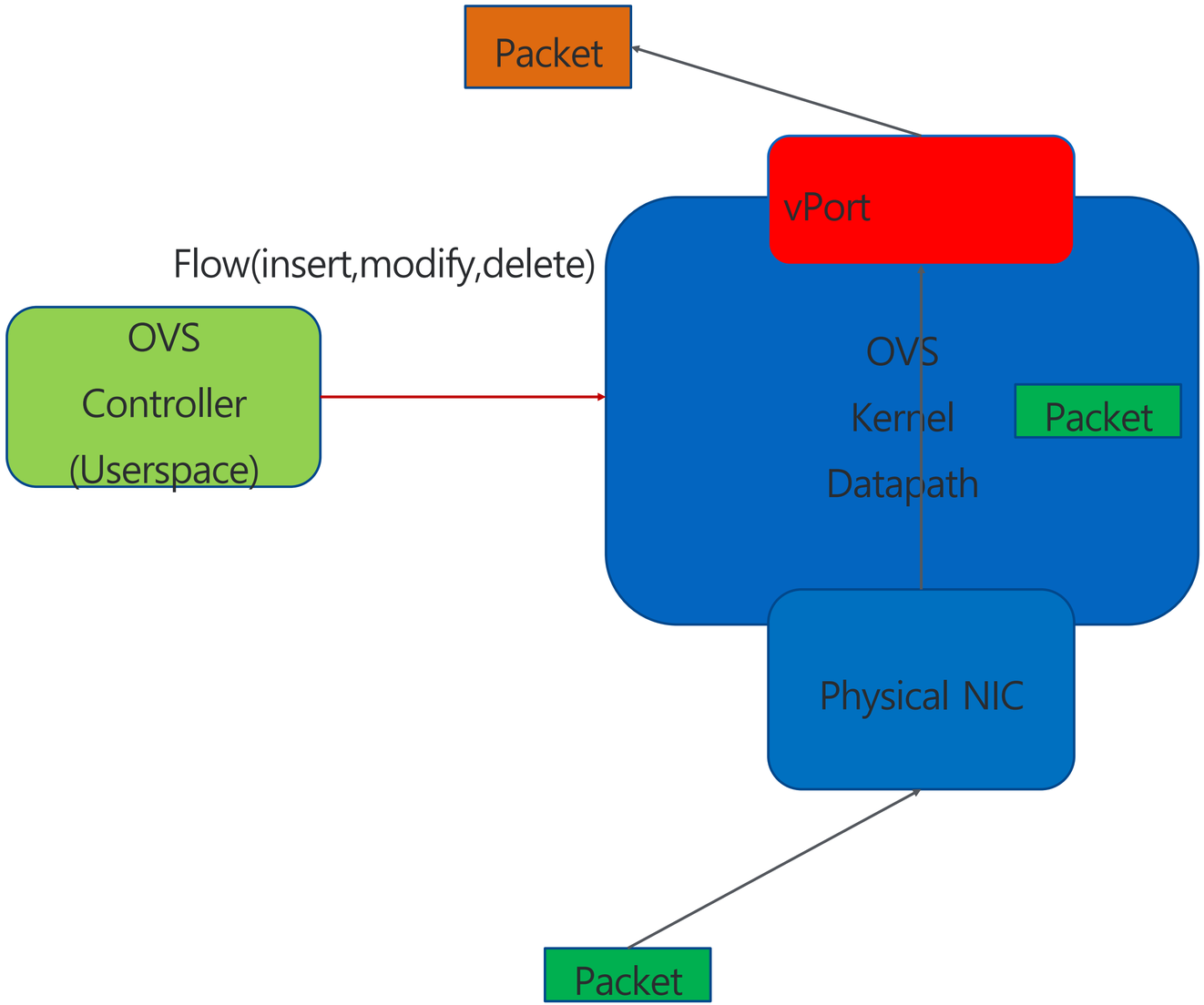
- Userspace 처리(SIowpath)
- Multiple packet copy
- CPU intensive

OVS Reactive based packet flow



- 패킷 생성 공격 취약
- Packet delay

OVS Proactive packet flow



- Fast datapath
- ~1000k flow
- No port range
- No negation(inverse match)

DPDK(Data plane development kit)

Super fast datapath

Polling(No interrupts. 100% CPU, More PPS)

Ring buffer for packets

Manipulate packets by userspace application(no kernel panic!)

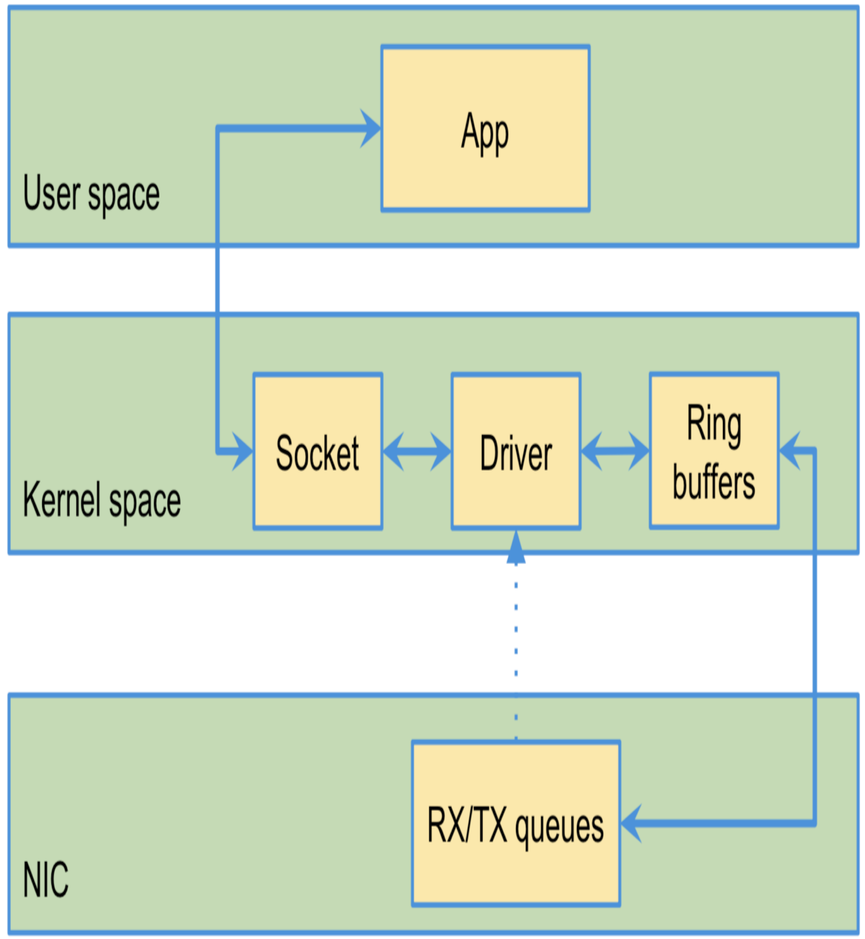
Supports various NICs, VirtIO(vhost mode)

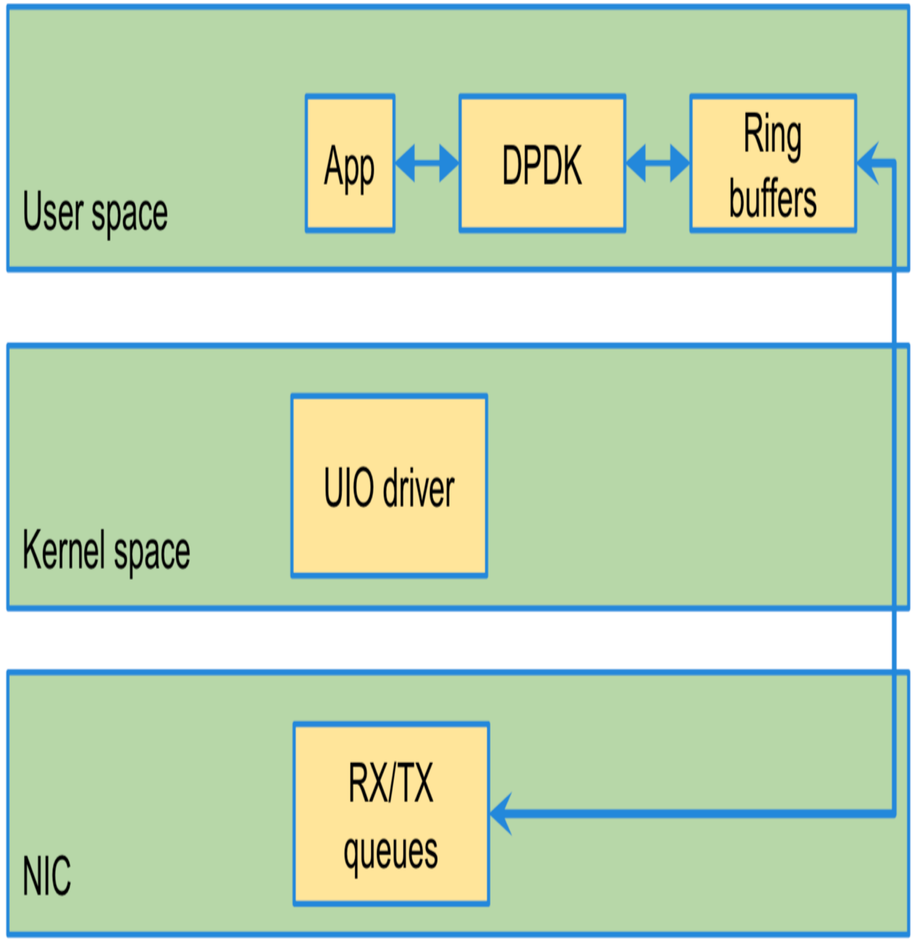
Pin thread to lcore(w/ RSS)

C, C++, GOLANG

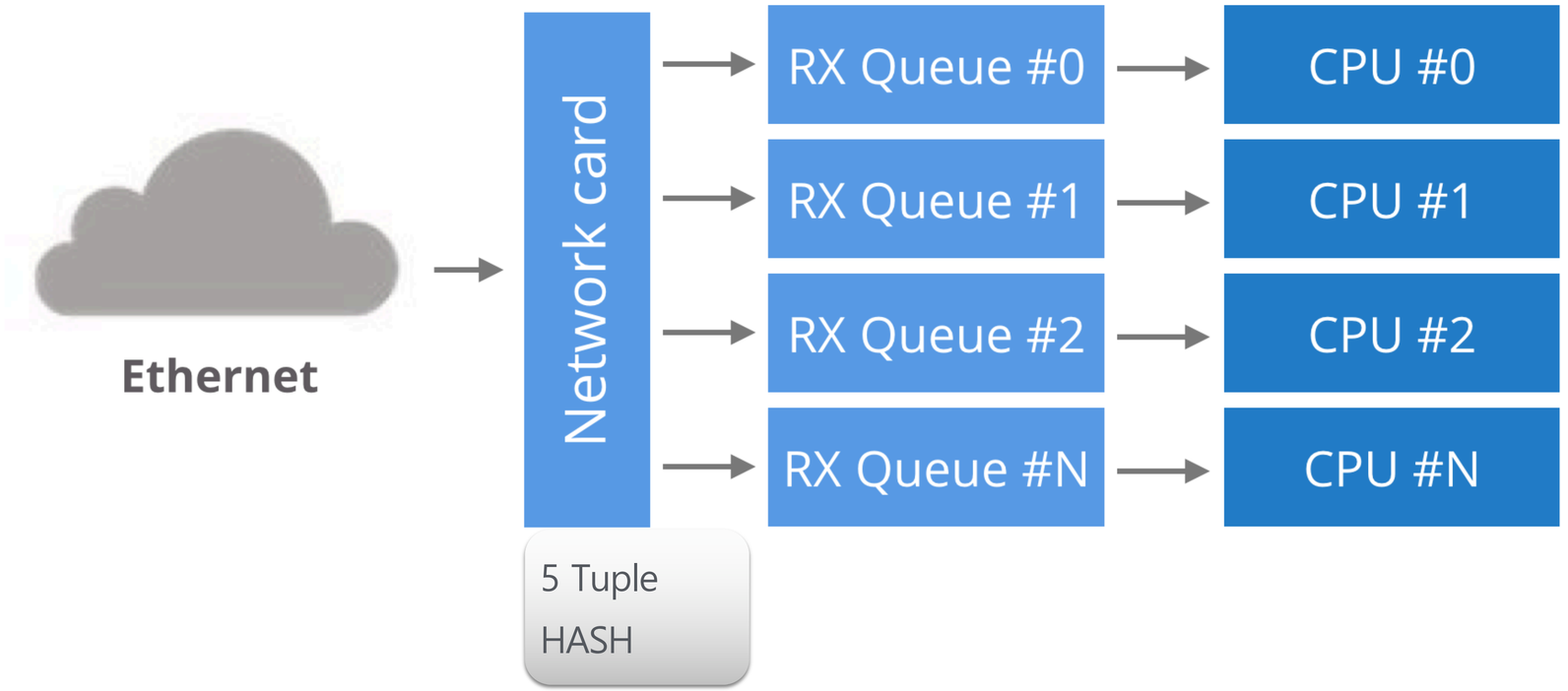
HASH, LPM, ACL, IPv6, QoS, Metering, ...

DPDK

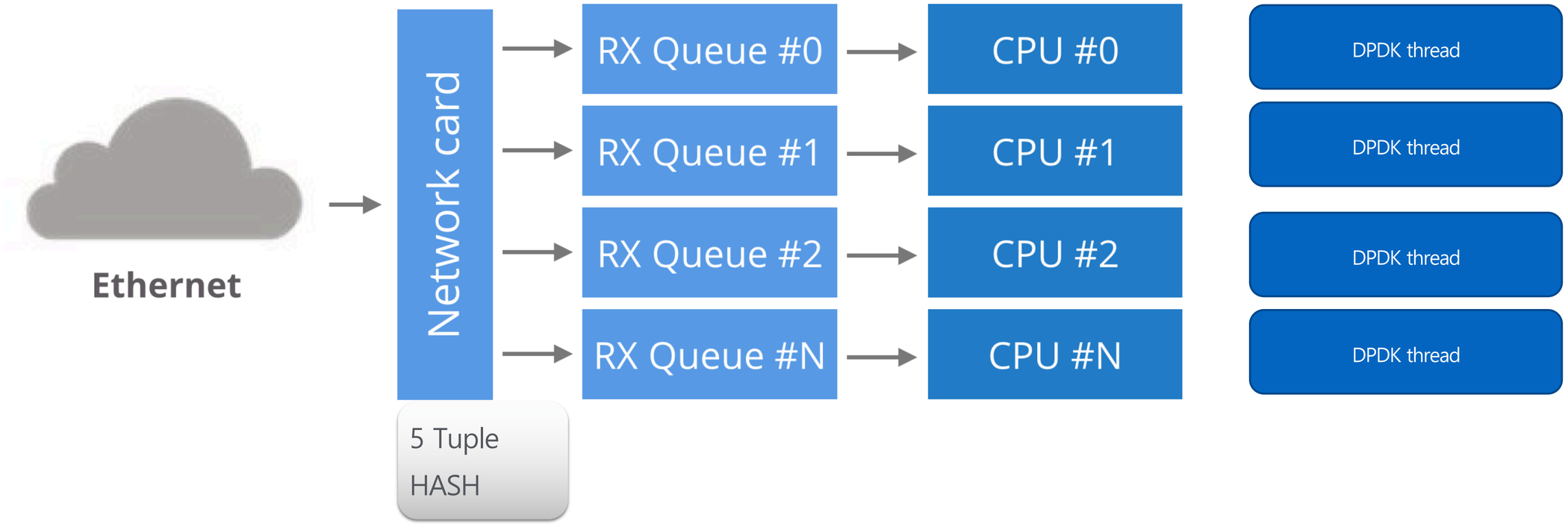

User space
App
Kernel space
Socket
Driver
Ring
buffers
NIC
RX/TX queues


User space
App
DPDK
Ring
buffers
Kernel space
UIO driver
NIC
RX/TX
queues

NIC Multiple queue + RSS



NIC Multiple queue + RSS + DPDK



DPDK problems

No interaction with kernel IP stack

No control plane, CLI, API

Build from a scratch

VxLAN encap./decap.(overhead), Forwarding/Routing, VNF, SFC, ...

개발하면 된다...

DPDK is good but...

64byte packets over 10G NIC@2.2G XEON

Per core:

L2 Forwarding only: Linerate(1400만 PPS)

Speed killer:

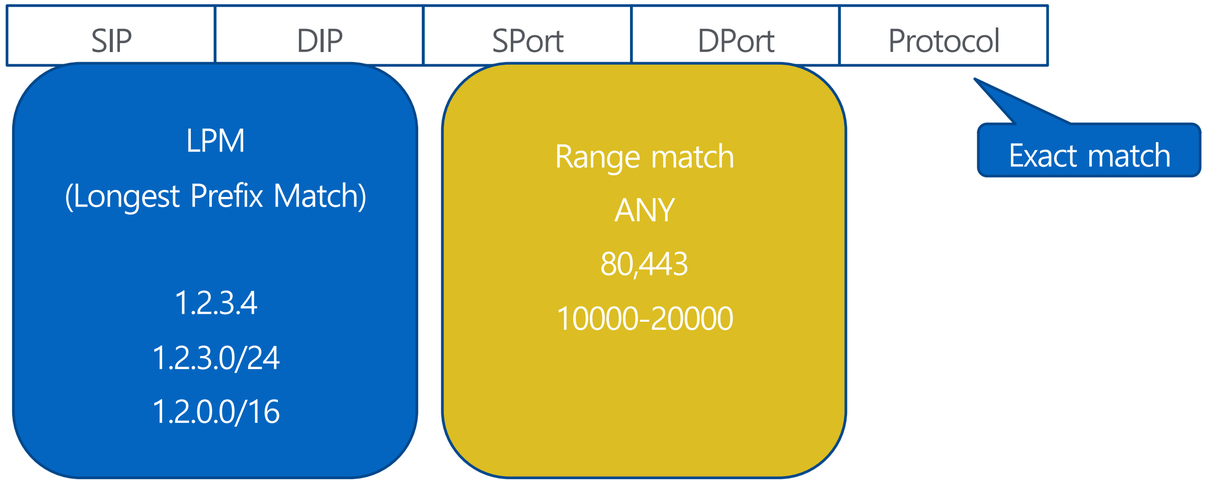
1 HASH lookup: 900만 PPS

ACL: 300만

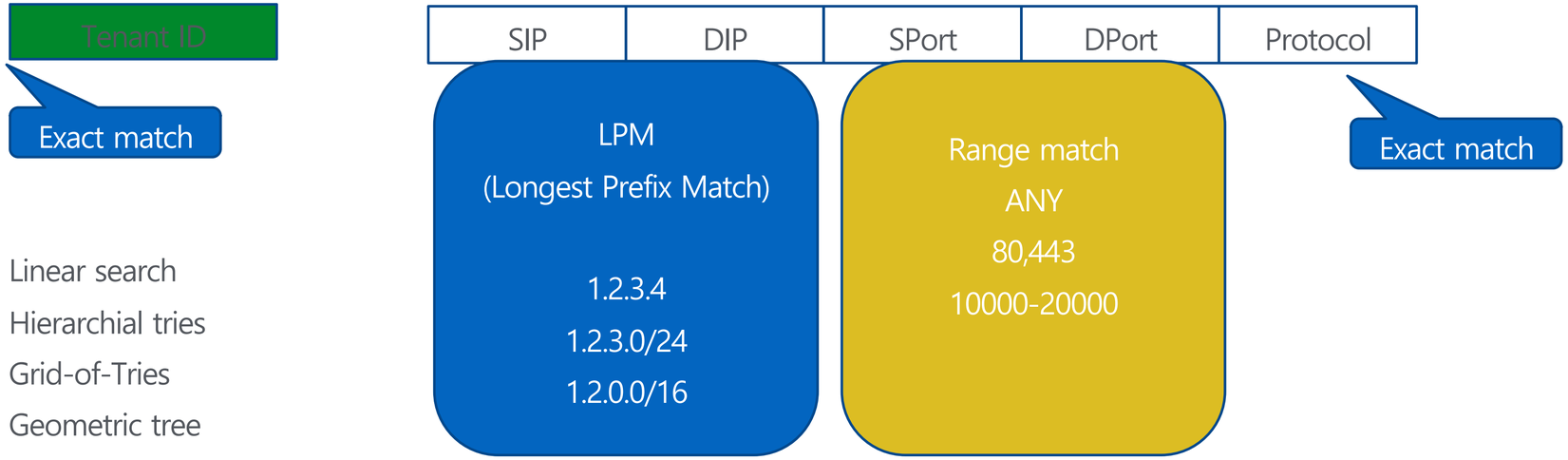
Encap./Decap: 200만(CPU only)

Can handle 25G, 40G, 100G?

Packet filter (5 tuple)



Tenant Packet filter (1+5 tuple)



Linear search

Hierarchial tries

Grid-of-Tries

Geometric tree

Cross-producting

Recursive Flow Classification

Tuple space Search

Tuple Space Pruning

Hierarchial Inelligent Cuttings(HiCut)

HyperCuts

...

O(n), O(log n), O(n log n): Multiple memory lookup per packet -> performance degrading

H/W Acceleration/Offload

Classic NIC offload

TSO, GSO, Checksum, Inner payload filter => Limited offload features

Custom H/W offload/Accelerator

- Ultra fast(Parallel processing for given packet)
- O(1) for rule match
- Linerate encapsulation, decapsulation
- Limited memory region = Rule limitation

Our Approach

ToR for underlay

- No Vendor dependency
- Simple L2, L3

H/W accelerated dataplane

- Linerate VxLAN decapsulation/encapsulation, forwarding
- Linerate VxLAN routing/SFC
- Linerate ACG/ACL

S/W dataplane

- Performance + flexibility
- Flow log, Metering,...
- Pakcet stateful table
- VNF(network endpoints)

  Public IP, NAT GW, Peering, IPSEC, ...

CONTROL PLANE

Message Queue

RPC

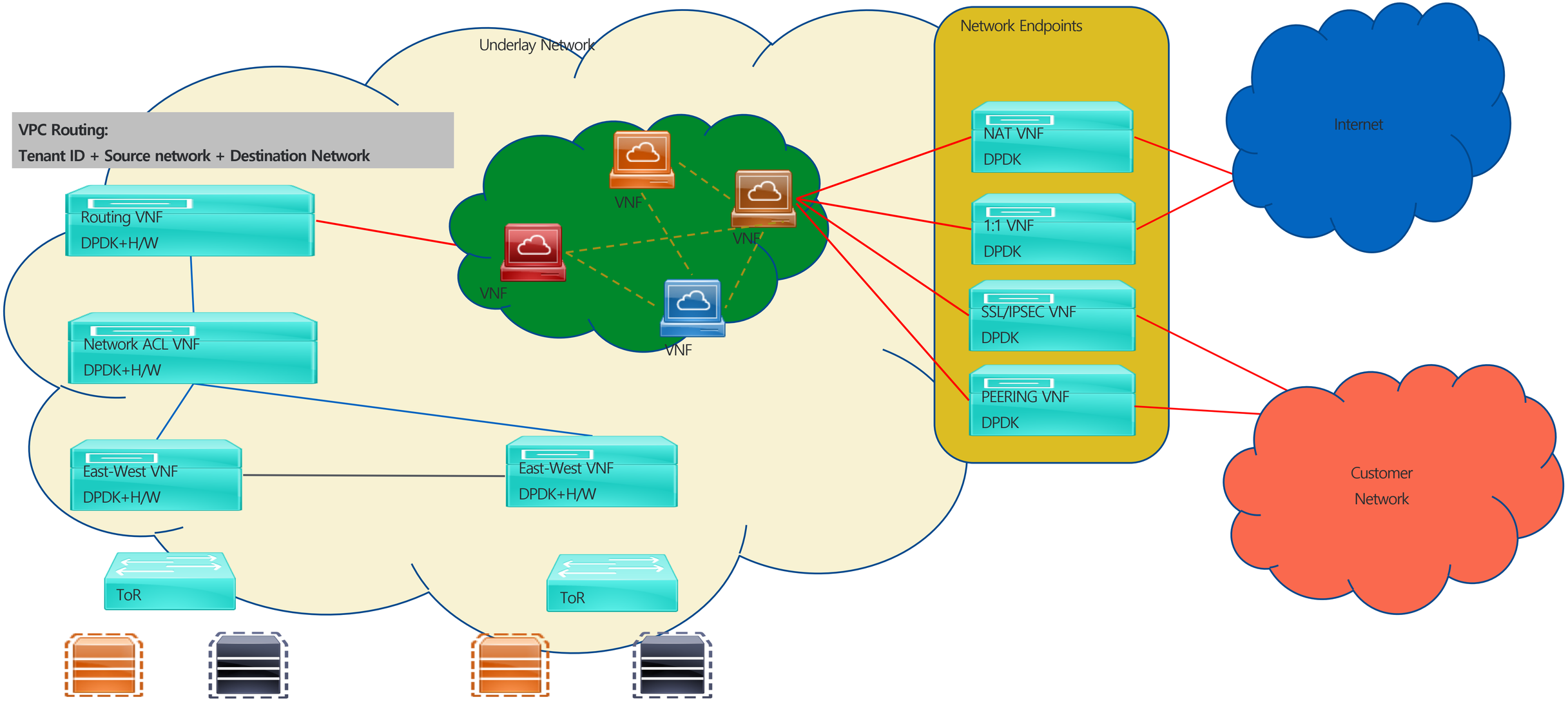
RESTful API

MP-BGP integration

(VRF, EVPN)

Network device automation

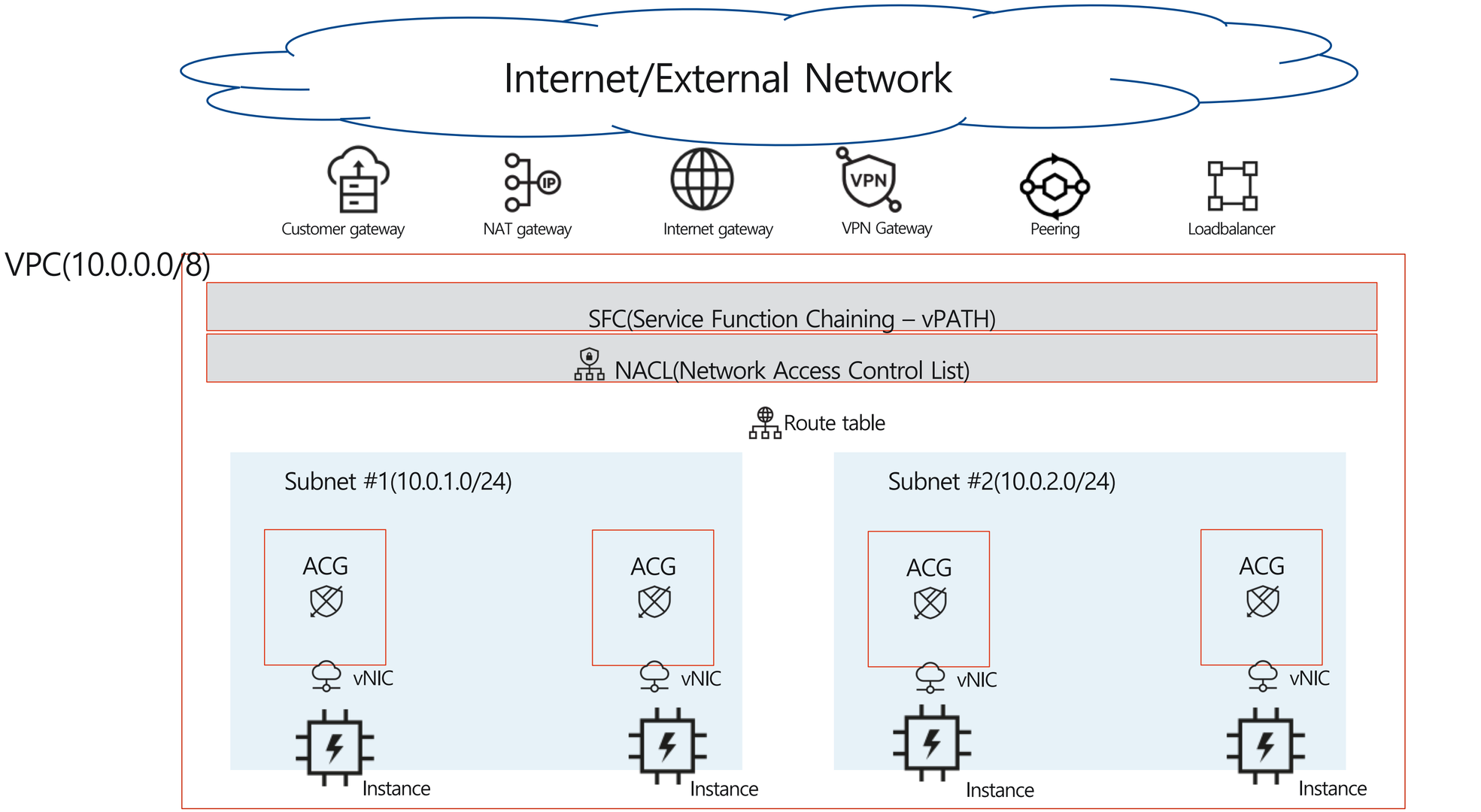
NCP VPC overview



DEVIEW
2019

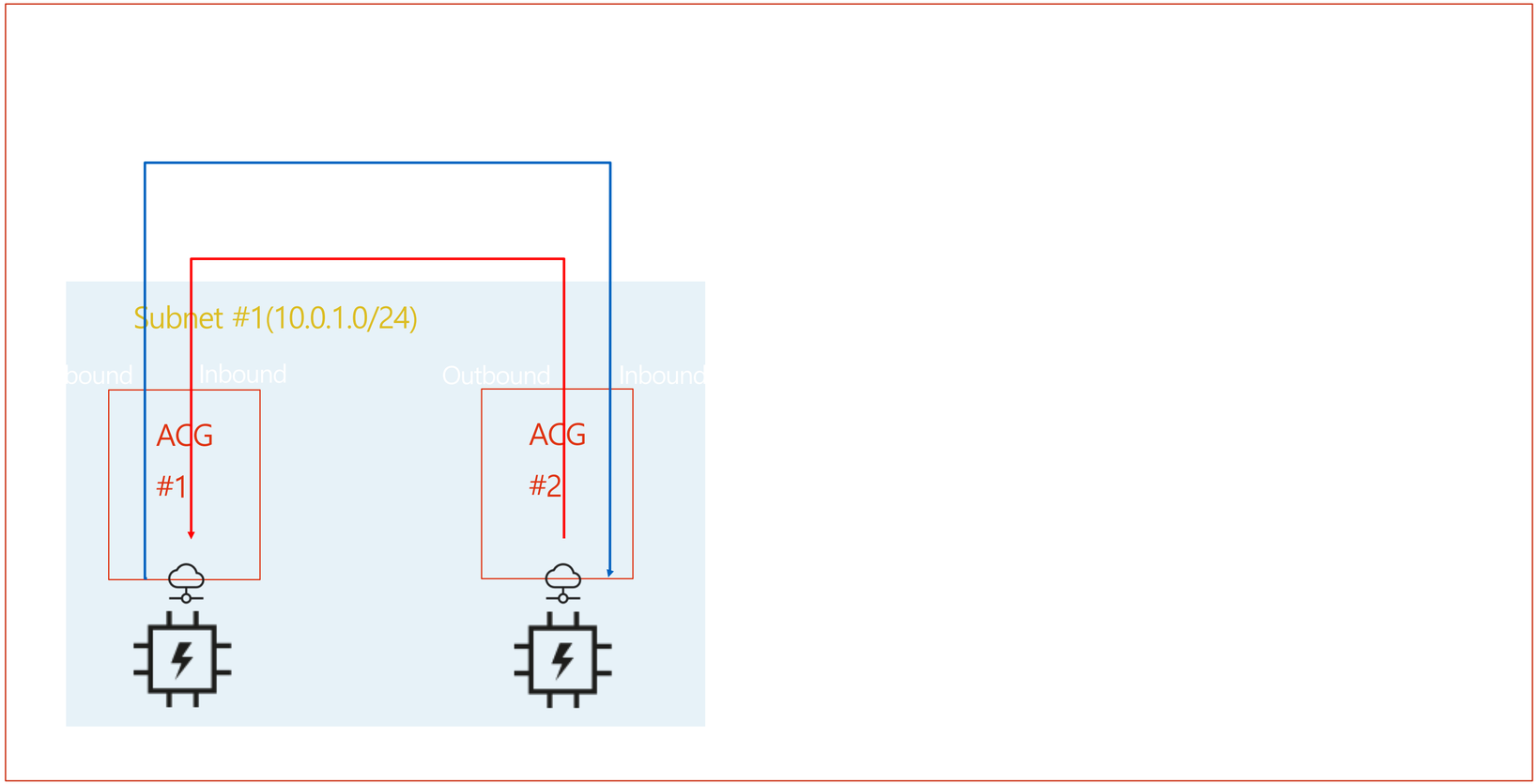
4.NCP VPC

NCP VPC

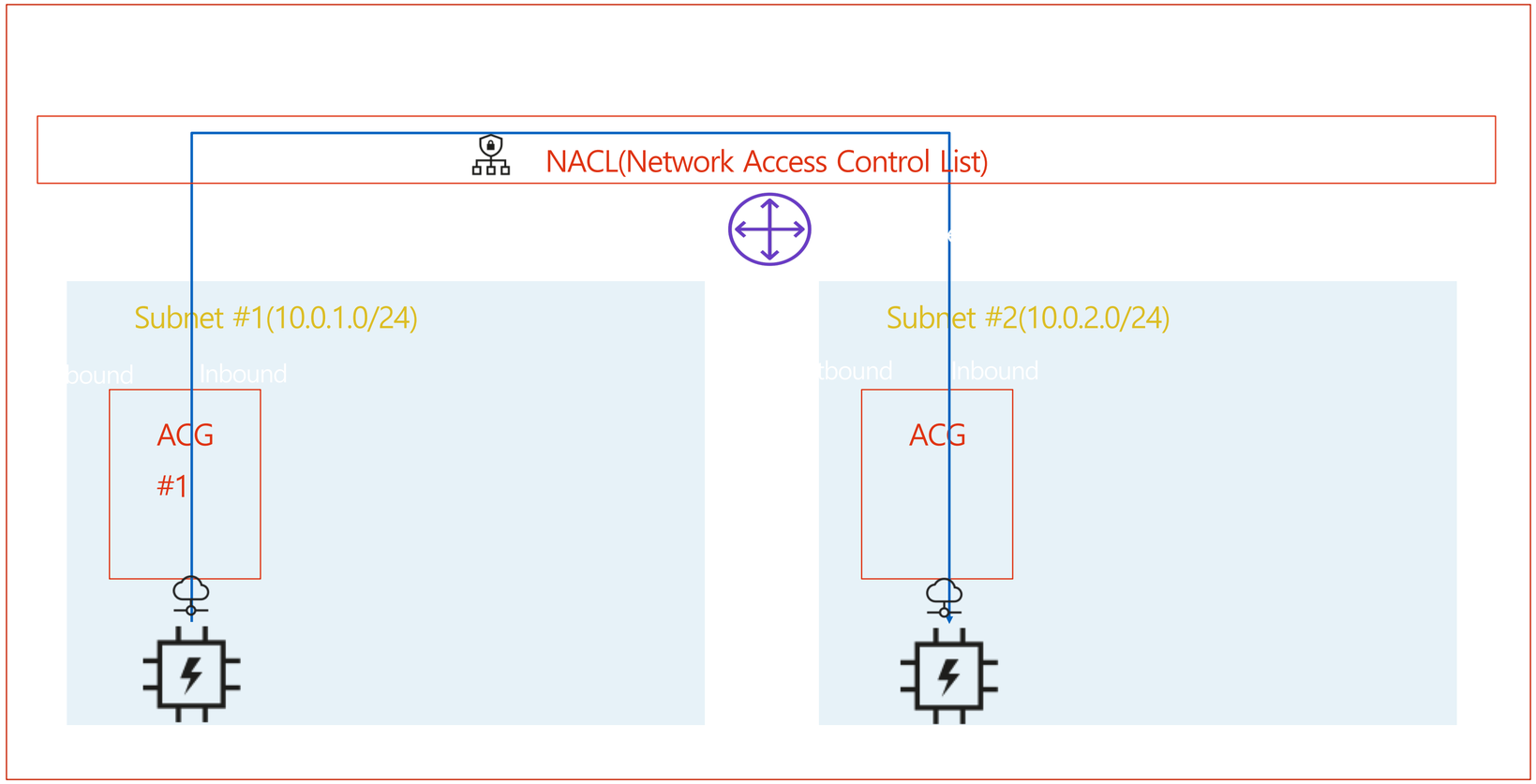

Internet/External Network
Customer gateway
NAT gateway
Internet gateway
VPN
VPN Gateway
Peering
Loadbalancer
VPC(10.0.0.0/8)
SFC(Service Function Chaining – vPATH)
NACL(Network Access Control List)
Route table
Subnet #1(10.0.1.0/24)
Subnet #2(10.0.2.0/24)
ACG
ACG
ACG
ACG
vNIC
vNIC
vNIC
vNIC
Instance
Instance
Instance
Instance

VPC Flow: Same subnet

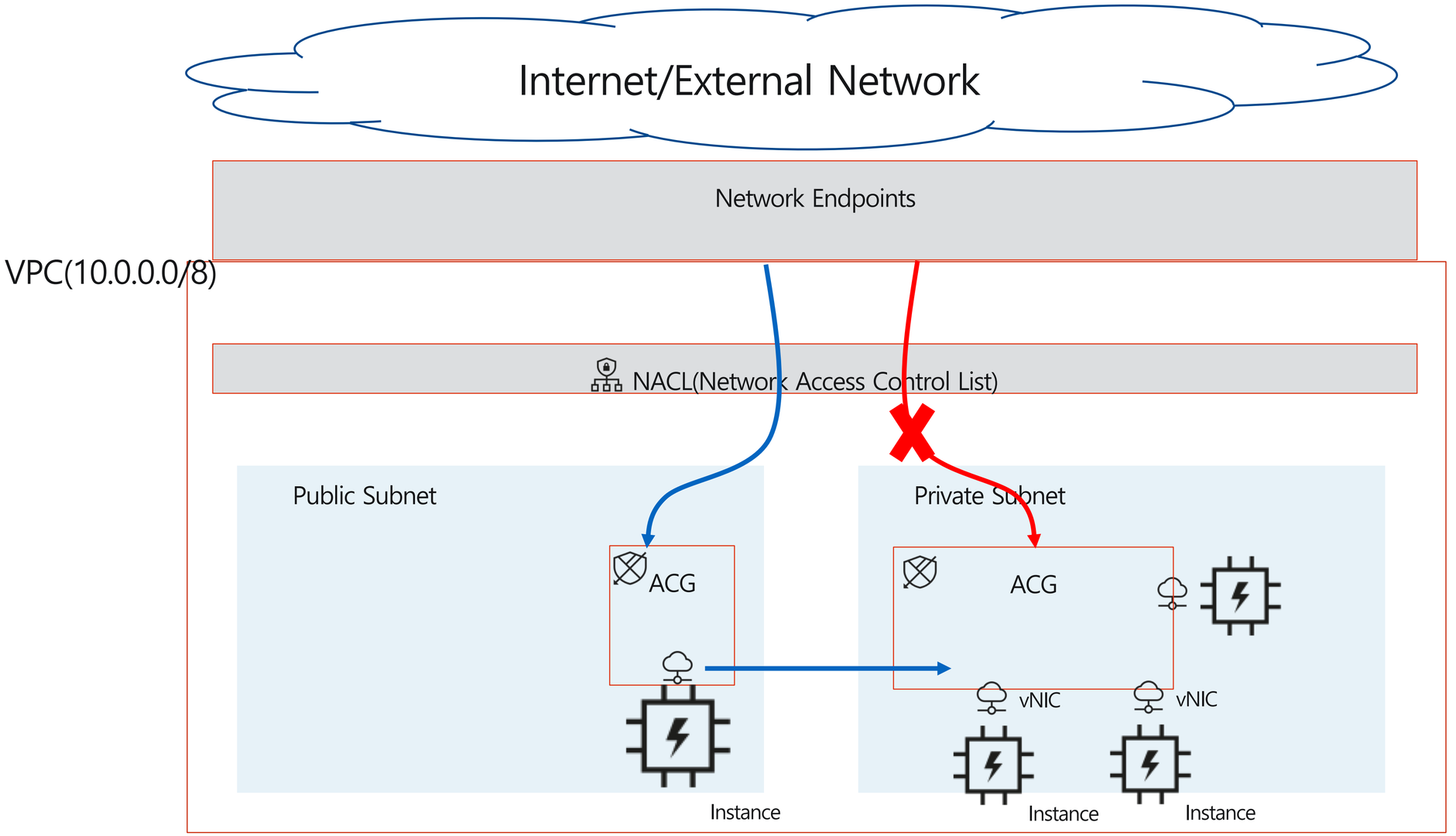
DEVIEW 2019



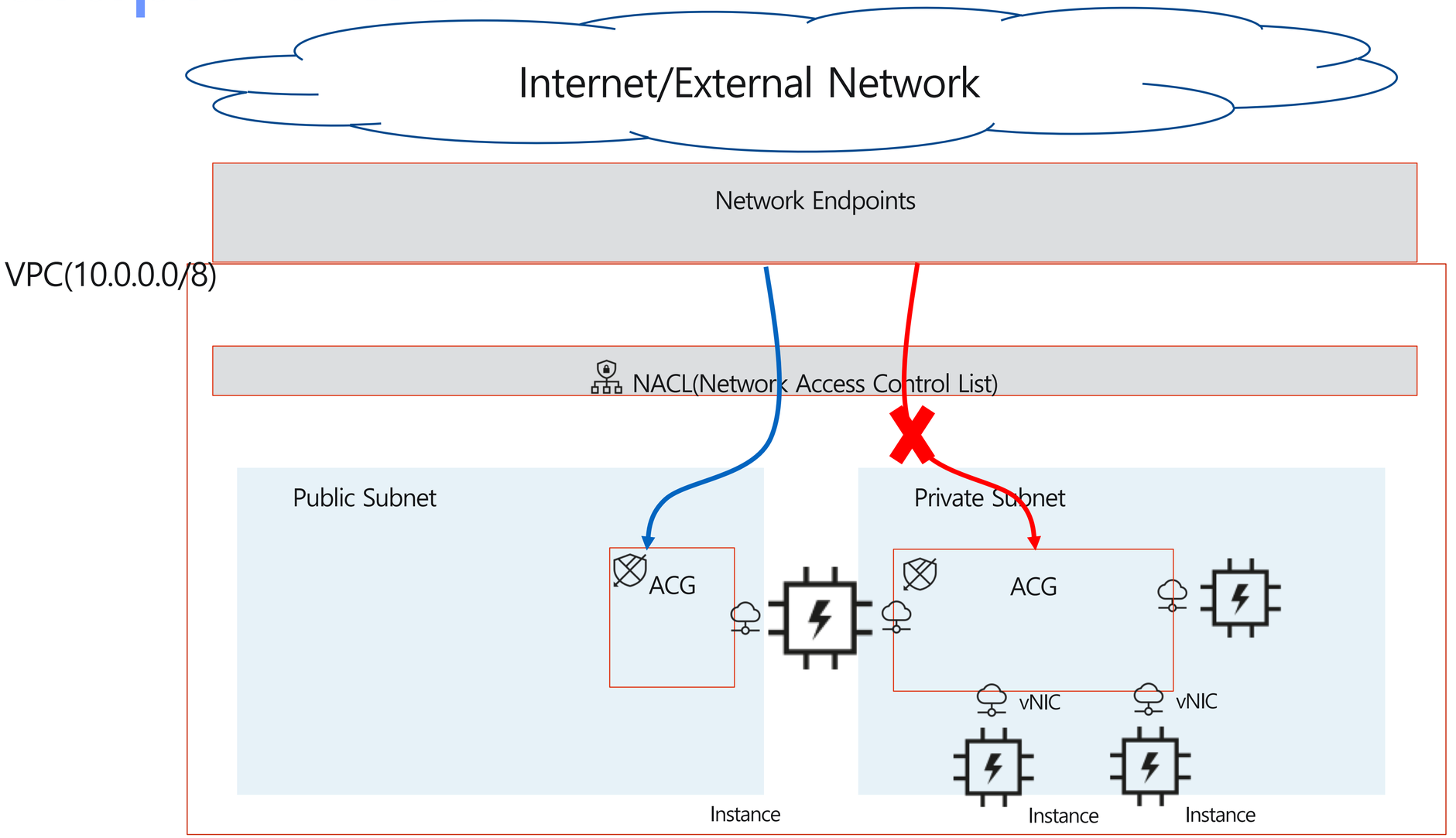
VPC Flow: Different subnet



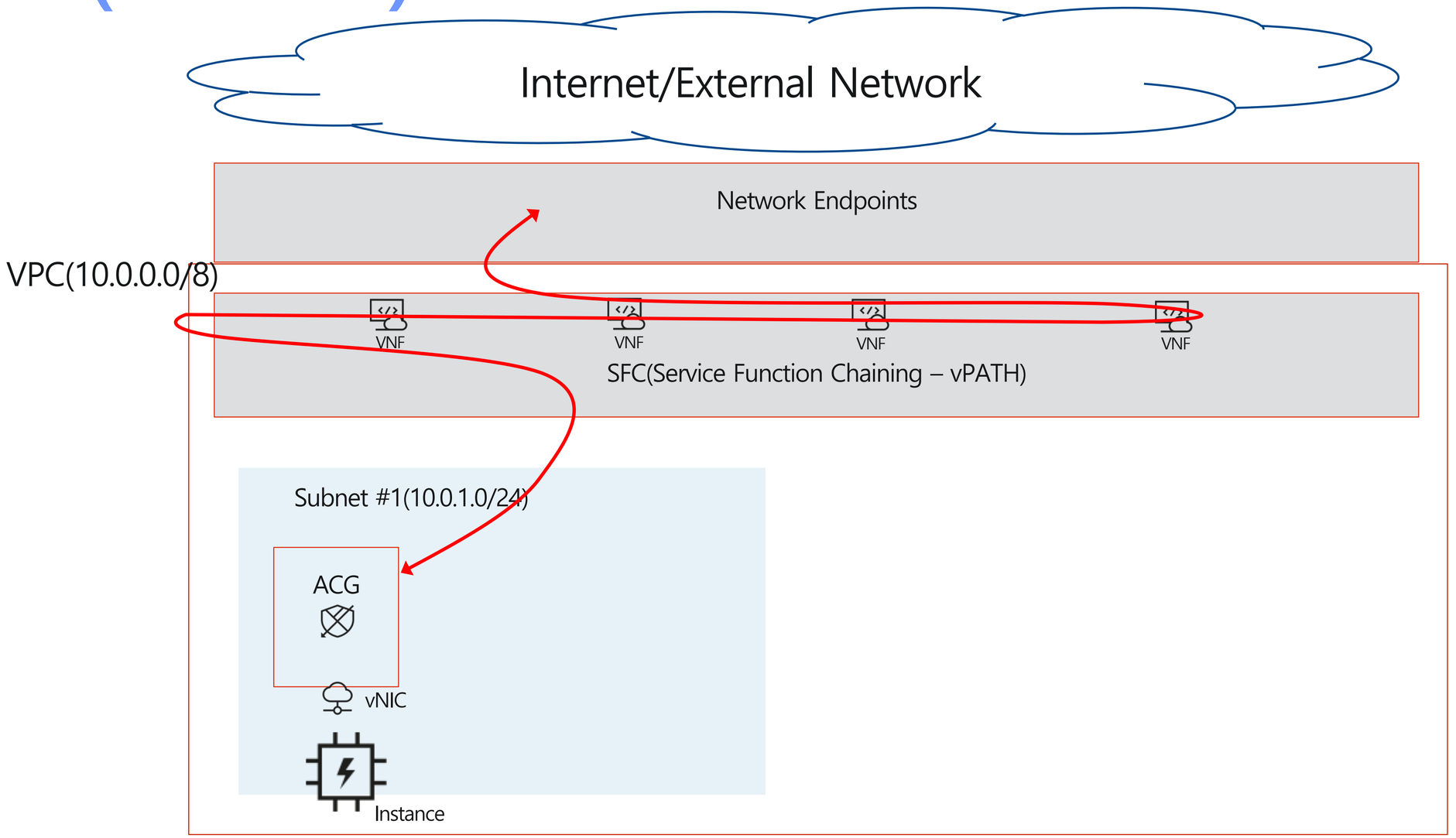
Subnet isolation



Multiple vNICs



SFC(vPATH)


Internet/External Network
Network Endpoints
VPC(10.0.0.0/8)
VNF
VNF
VNF
VNF
SFC(Service Function Chaining – vPATH)
Subnet #1(10.0.1.0/24)
ACG
vNIC
Instance

DEVIEW
2019

Q & A

DEVIEW
2019

Thank You