L.T.H.

kakao

Scalable VM and Container Networking using /32bit subnets and BGP routing

Andrew Yongjoon Kong

DaumKakao

A Mobile Lifestyle Platform

Social Platform

- KakaoTalk
- KakaoStory
- KakaoGroup
- Daum Cafe
- Zap
- Sol Group

Contents Platform

- Media Daum
- KakaoGame
- Digital Item Store
- KakaoTopic
- KakaoPage
- KakaoMusic
- Daum tvPot
- Daum Webtoon

Commerce Platform

- KakaoPick
- Gift Shop
- KakaoStyle

Marketing Platform

- Yellow ID
- Plus Friend
- Story Plus

Local Platform

- Daum Map
- KakaoPlace

Personal Platform

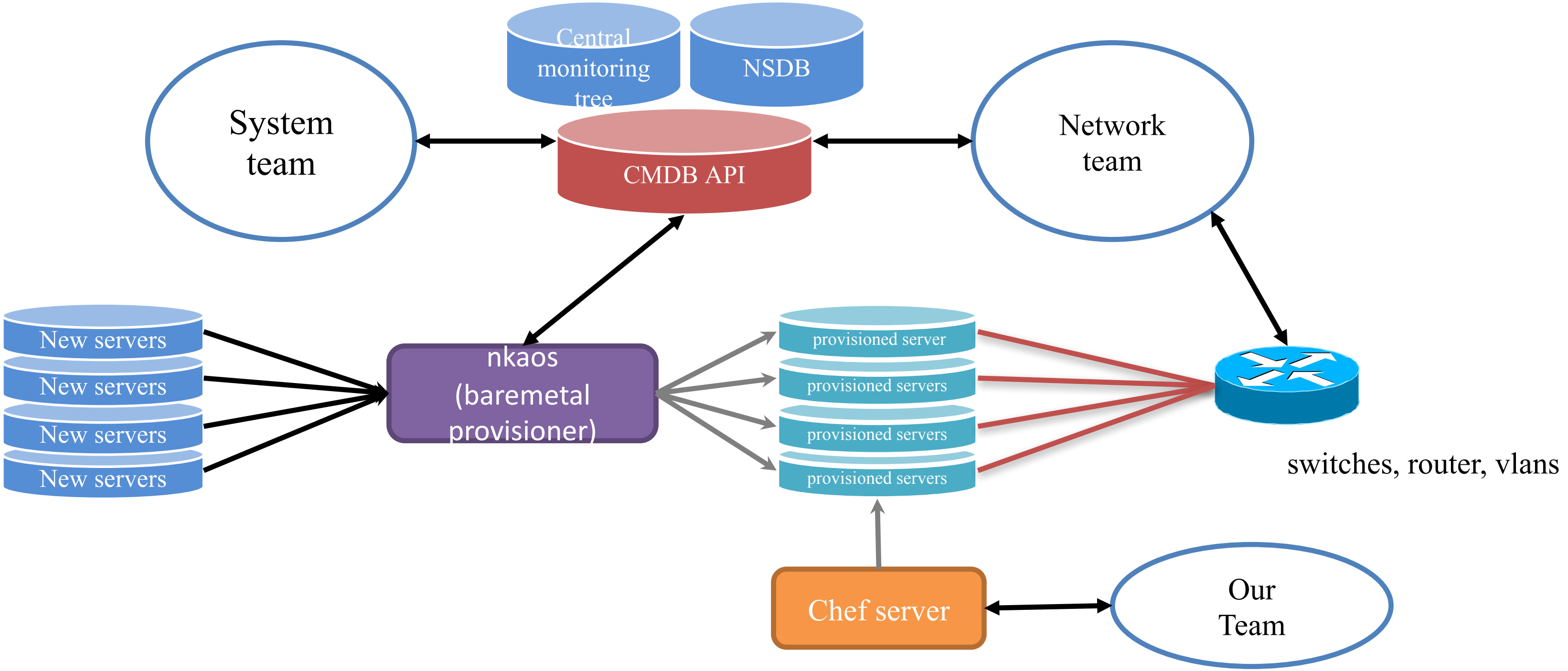
- KakaoHome
- Sol calendar
- Sol Mail
- Daum Cluod

96% of Korean smartphone users are using KakaoTalk messenger, 170 million users worldwide)

2nd largest search and portal

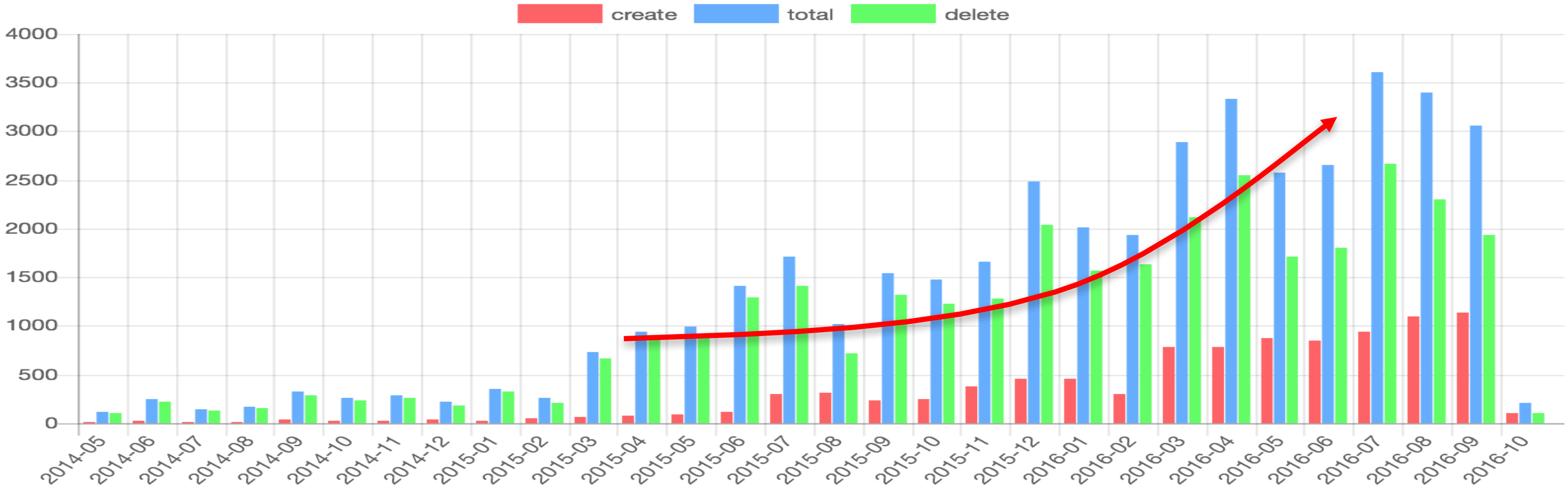
The Peaceful operation

When we’re running out of resources (cpu, memory, disk),
Just add new(or additional) resources to existing one.



The Growth(I)

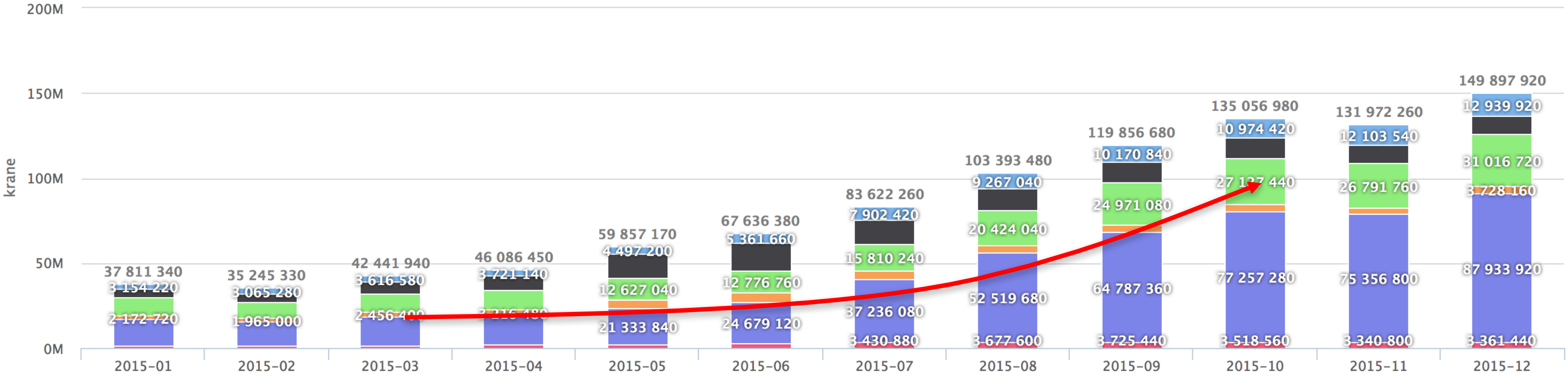
VM creation speed is accelerating



The Growth(II)

Spend more than 45M krane ($45,000) per month

– this also increased.



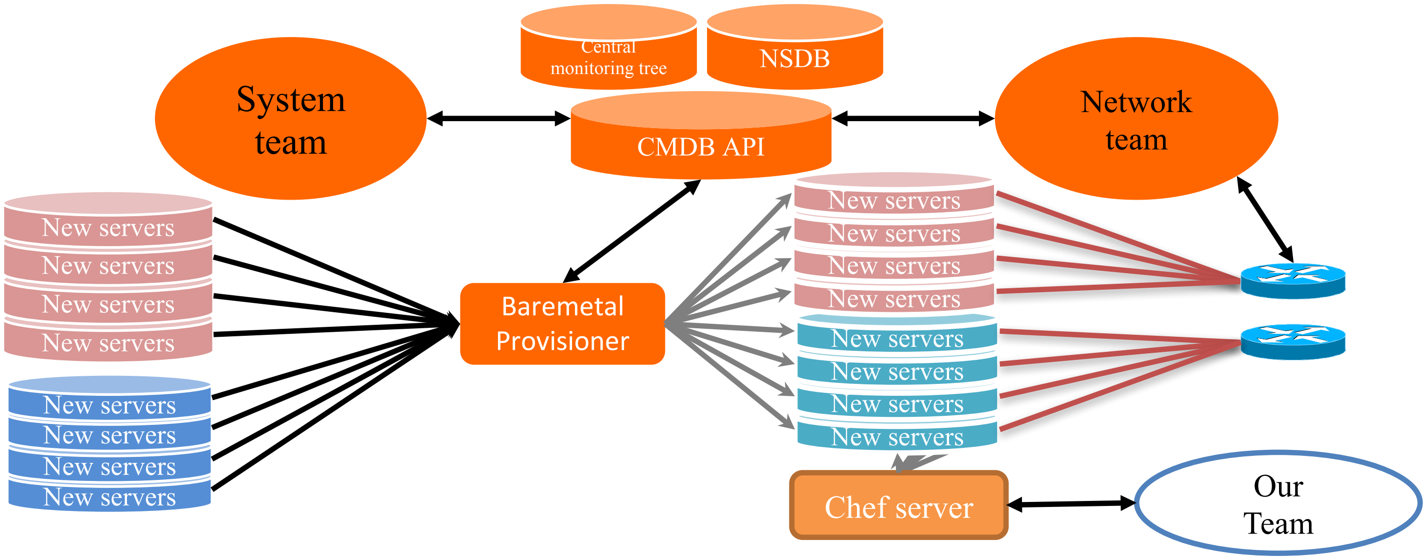
1 krane = 1 Won ($0.001)

- Using similar pricing with AWS EC2
- Network/Disk usage not included

The Growth(III)

Growth is accelerating

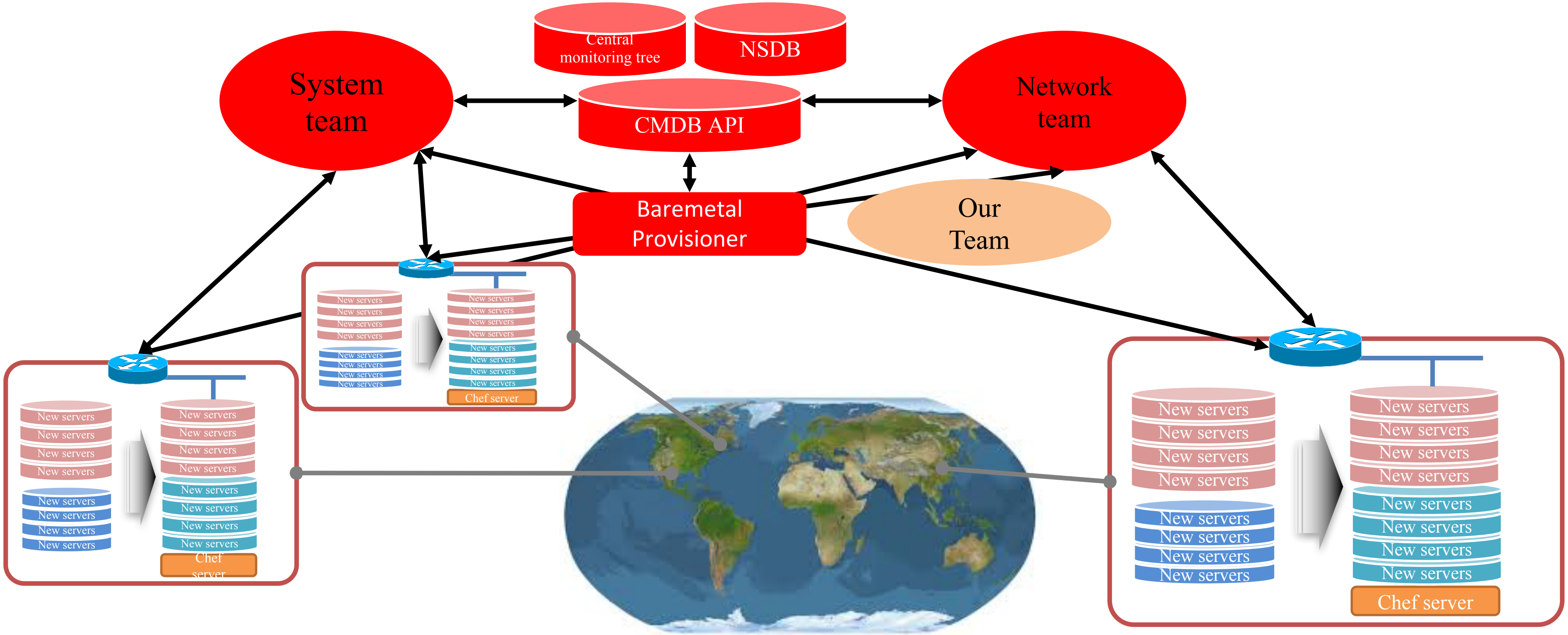
- No. of Engineer is growing
- New Pilot services or experiments are growing.
- The resources depletion speed is accelerating → this simply make more work to resource management teams



The Growth(IV)

kakao

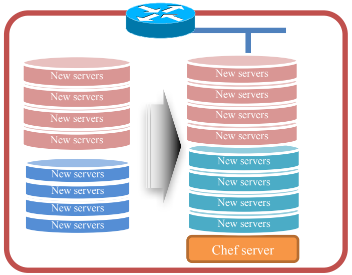
Scale, The only driving force disrupt everything.



The Growth – Lesson learned

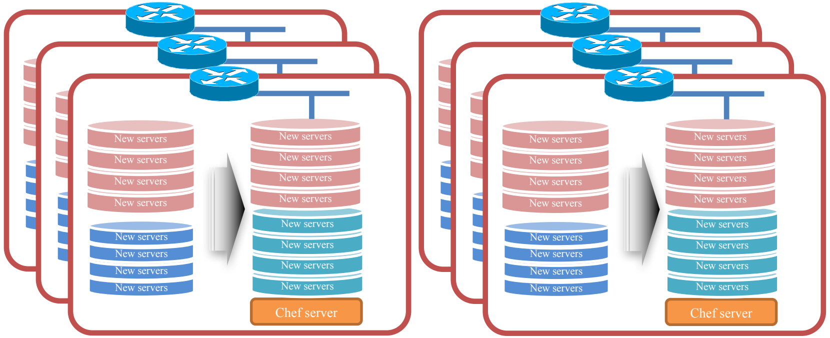
Growth doesn’t come alone

- Infra growth includes scale-up , scale-out as well
- Scale-up includes these
  - Add Server, Storage, Switches
  - Add more power facility to supply juice fluently
  - This is not that difficult.
- Scale-out include these
  - Add New Datacenters, New Availability Zones
  - This is nightmare!



This leads radical changes over everything

- The way of preparing, provisioning
- The way of monitoring, logging, developing



Some Numbers

kakao

1021 tenants

662 pull request since 2014.9

136 VMs are created/deleted per day

Some information about kakao Openstack

openstack upgraded from grizzly to **Liberty**

total **4Region**

additional service **Heat/Trove/Sahara/Octavia**

The Growth – Lesson learned, Openstack (2)

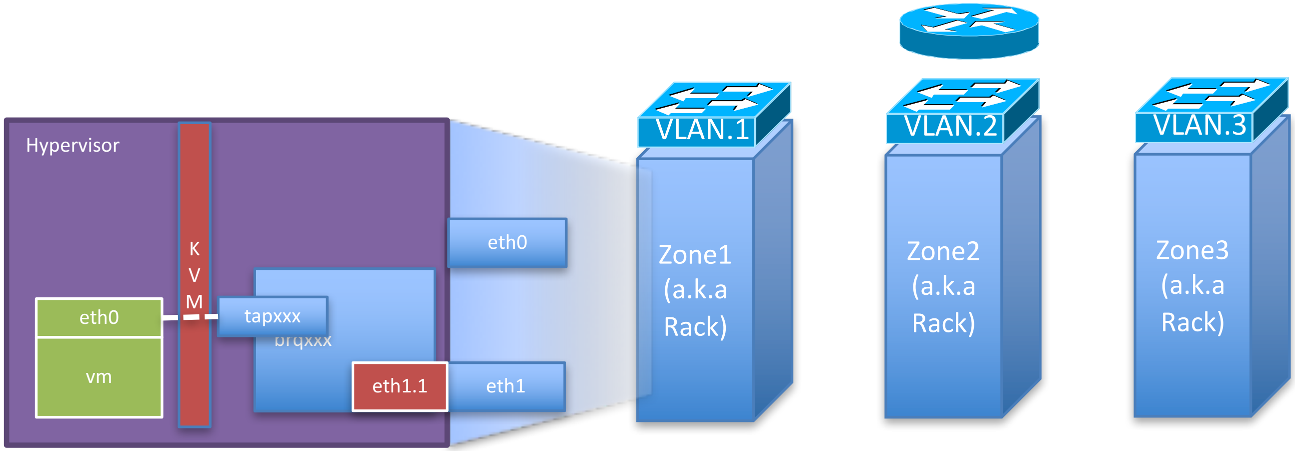
Resources for Openstack finally comes to be exhausted

- CPU, Memory, Storage always experience shortages.
- They have skewness.
- Sometimes, CPU depleted. Sometimes, Storage depleted.
  - All resources are able to be re-balanced.
  - you can migration clients’ VM (image , volume)
- IP is also Resources.
  - Very limited than our expectations
    - No of IP counts is limited.
    - Location of IP also is limited.
  - Managing these Resources is getting tougher issue.

OpenStack Neutron Network

We’ve been using Provider Network (VLAN)

- ML2 plugin
- From OVS → LinuxBridge.
- Network Team plan/setup networks (the VLAN, IP[subnet], Gateways)
- Mapping availability zone / Neutron Network to that Physical networks

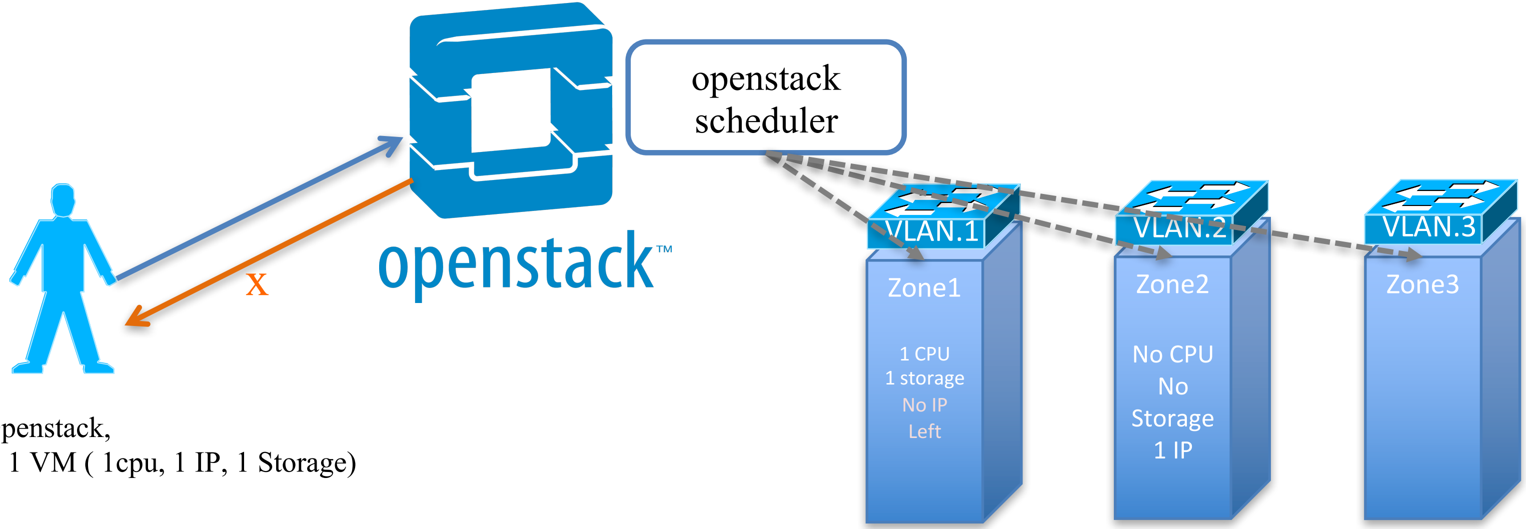


Resource Imbalance

kakao

After Running multiple Available Zones

- Experiencing resource imbalance between zones, naturally
- Filter Scheduling won’t helpful.
- Migration is a proper solution. (add extra resource is better If possible)



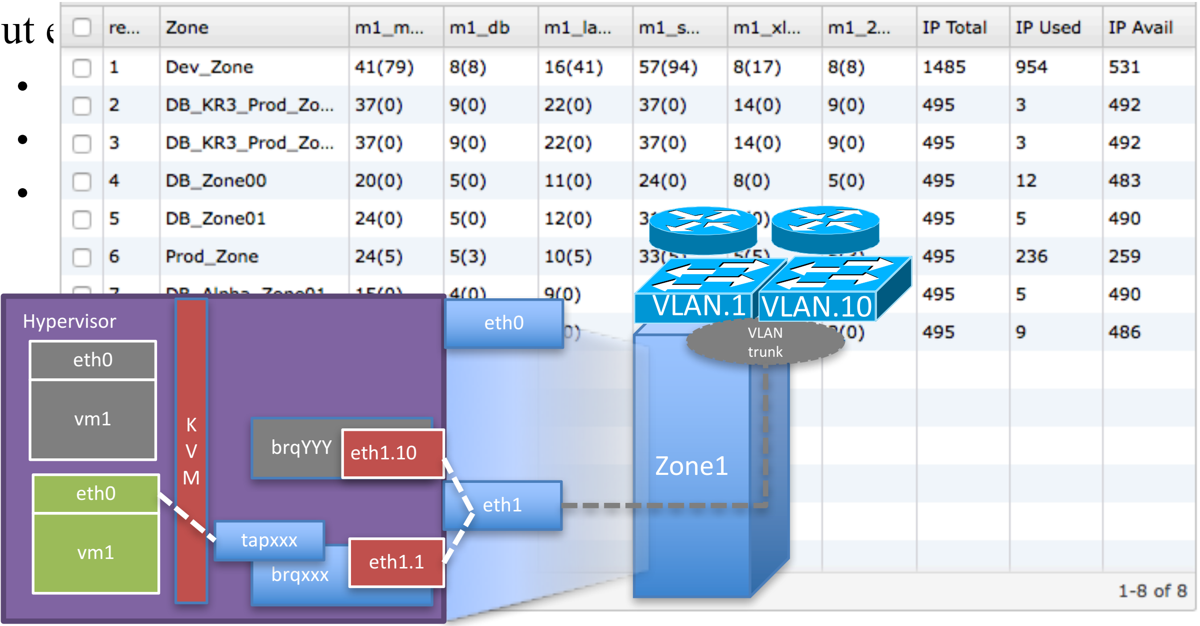
Hey Openstack,
Create 1 VM (1cpu, 1 IP, 1 Storage)

Resource Imbalance & Remedies

kakao

Develop Network Count filter

- Check Remaining IP count for each zone, treat ip count as resource
- Select the zone which have more ip count
- but e

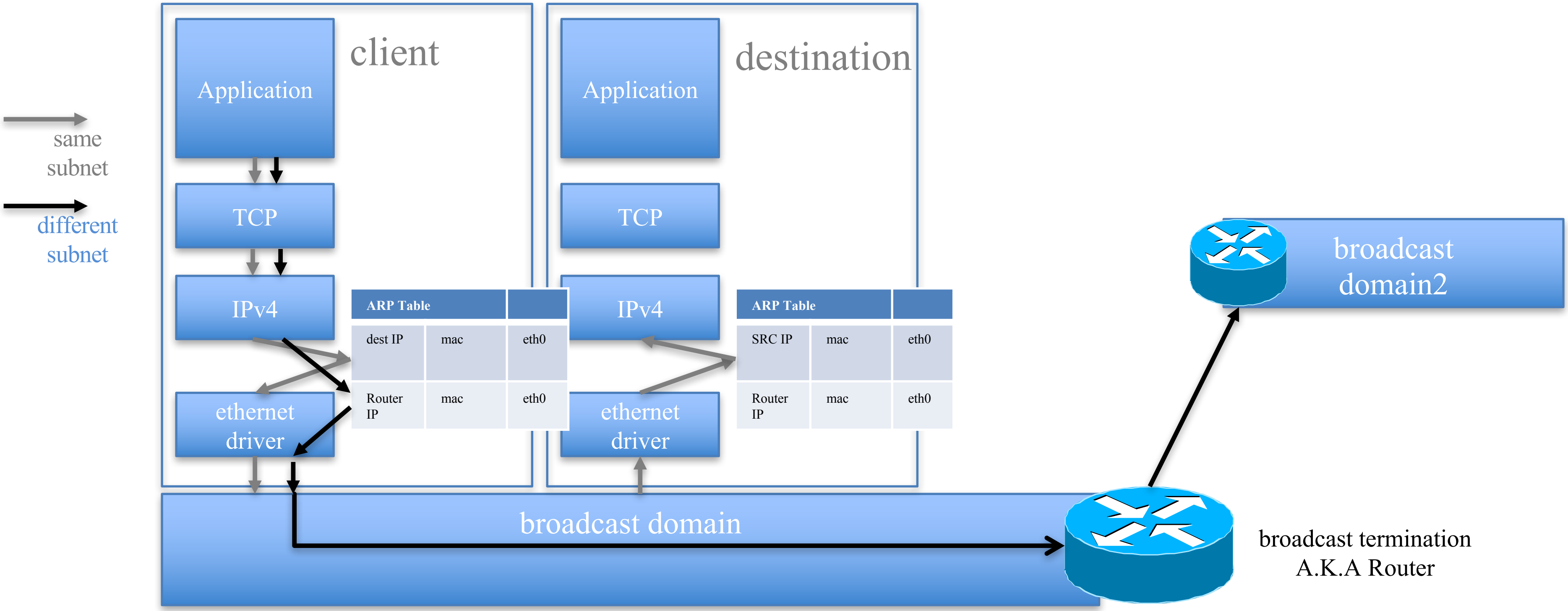


□	re...	Zone	m1_m...	m1_db	m1_la...	m1_s...	m1_xl...	m1_2...	IP Total	IP Used	IP Avail
□	1	Dev_Zone	41(79)	8(8)	16(41)	57(94)	8(17)	8(8)	1485	954	531
□	2	DB_KR3_Prod_Zo...	37(0)	9(0)	22(0)	37(0)	14(0)	9(0)	495	3	492
□	3	DB_KR3_Prod_Zo...	37(0)	9(0)	22(0)	37(0)	14(0)	9(0)	495	3	492
□	4	DB_Zone00	20(0)	5(0)	11(0)	24(0)	8(0)	5(0)	495	12	483
□	5	DB_Zone01	24(0)	5(0)	12(0)	3	0)		495	5	490
□	6	Prod_Zone	24(5)	5(3)	10(5)	33(	5(5)		495	236	259
□	7	DB_Alpha_Zone01	15(0)	4(0)	9(0)				495	5	490
					0)			(0)	495	9	486

1-8 of 8

Rationale

Rethinking about Connectivity



ARP Table		
dest IP	mac	eth0
Router IP	mac	eth0

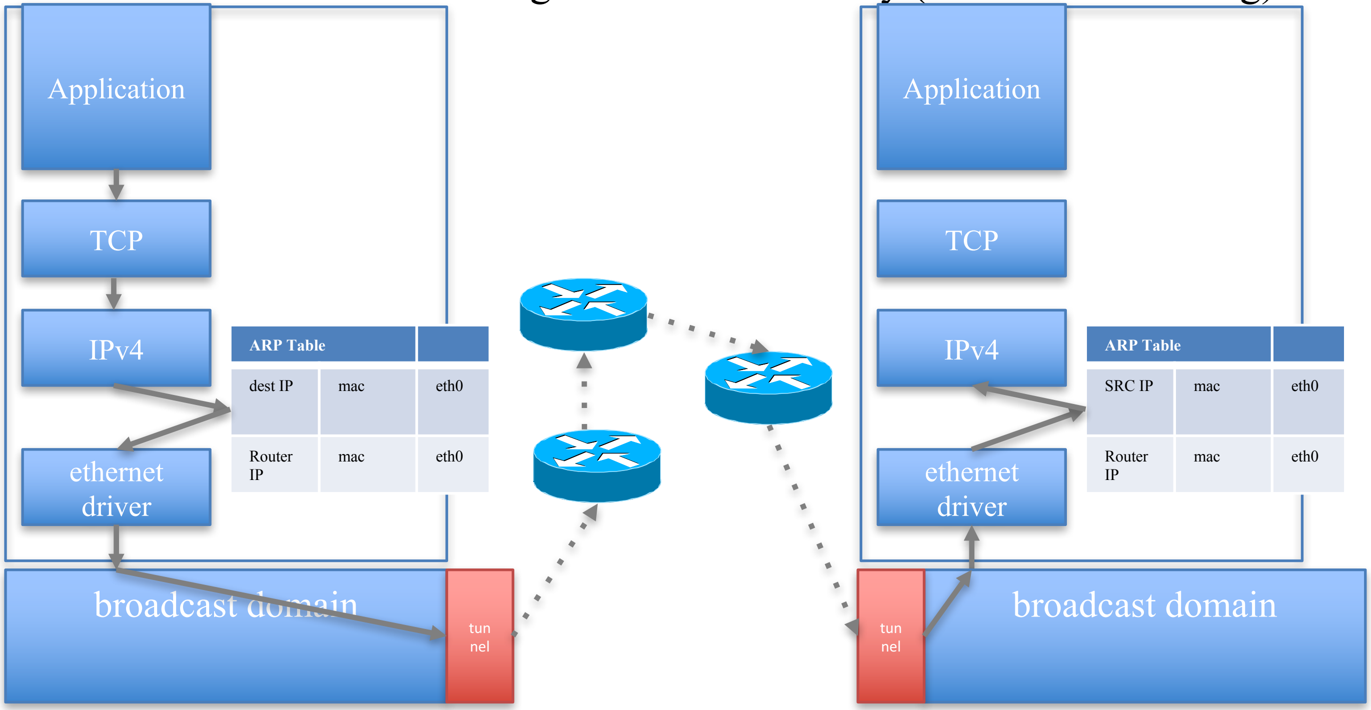
ARP Table		
SRC IP	mac	eth0
Router IP	mac	eth0

Rationale

kakao

Rethinking about Connectivity (Overlay)

- it solve remote link layer separation issue.
- Still have issue with IP management. and Gateway (Packet Forwarding)



ARP Table		
dest IP	mac	eth0
Router IP	mac	eth0

ARP Table		
SRC IP	mac	eth0
Router IP	mac	eth0

Remedy , Version 2.0

we need to thinks of those requirement

- IP movement inter-rack, inter-zone, inter-dc(?)
- IP resource imbalance
- Fault Resilience
- Dynamically check status of network
- Simple IP Resource Planning and Management

Router

We thinks Router as best candidate

- It dynamically detects and exchanges changes. (via dynamic routring protocol)
- It is highly distributed.
- It have HA (e.g. VRRP)
- the issue is that most of time routing is done in ranges (a.k.a Subnet)
  - Because of Memory and CPU issue

Finally, Come to route only IP

Generally, Known as /32 network.

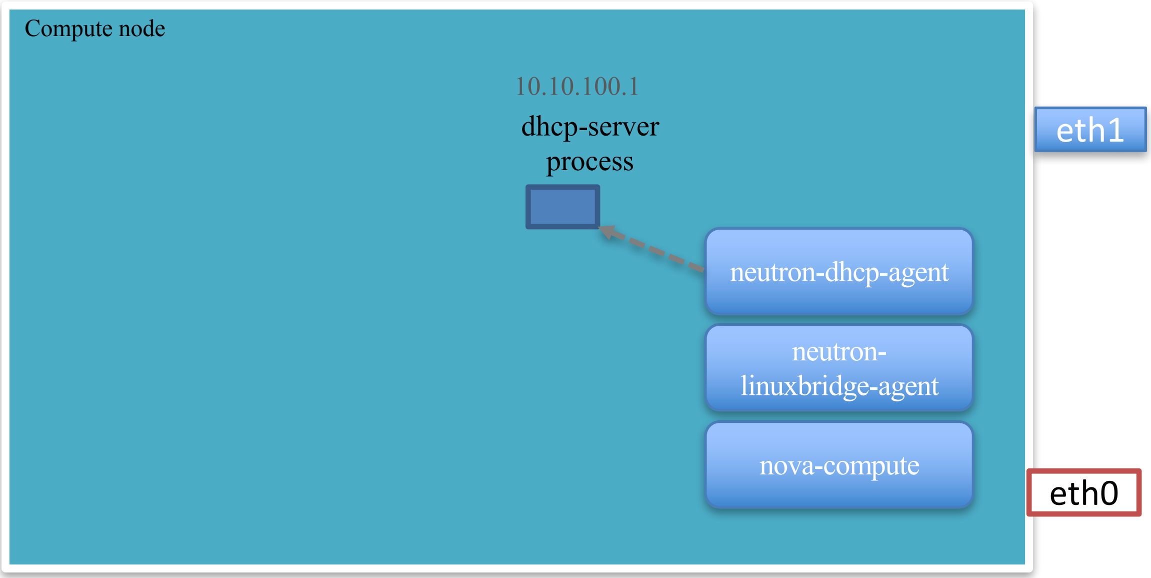
10.0.0.1 / 32 or

IP 10.0.0.1 netmask 255.255.255.255

- No L2 (link) consideration needed anymore (no subnet)
- With Dynamic Routing Protocol, it move every where.
- Simple IP planning (Just think of IP ranges)
- It’s very Atomic Resource, it keeps its IP after migration through zones

How it setup

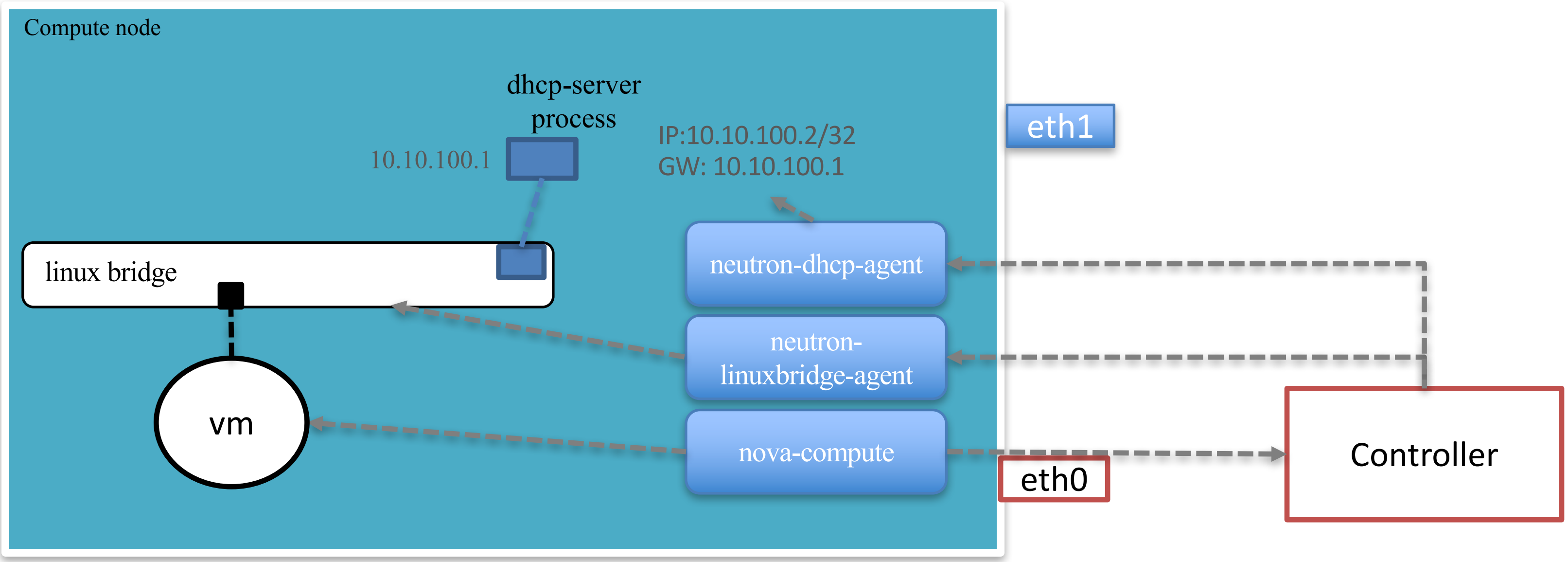
1. install nova/neutron agent.
2. create neutron network (name: freenet, subnet: 10.10.100.0/24)



How it setup

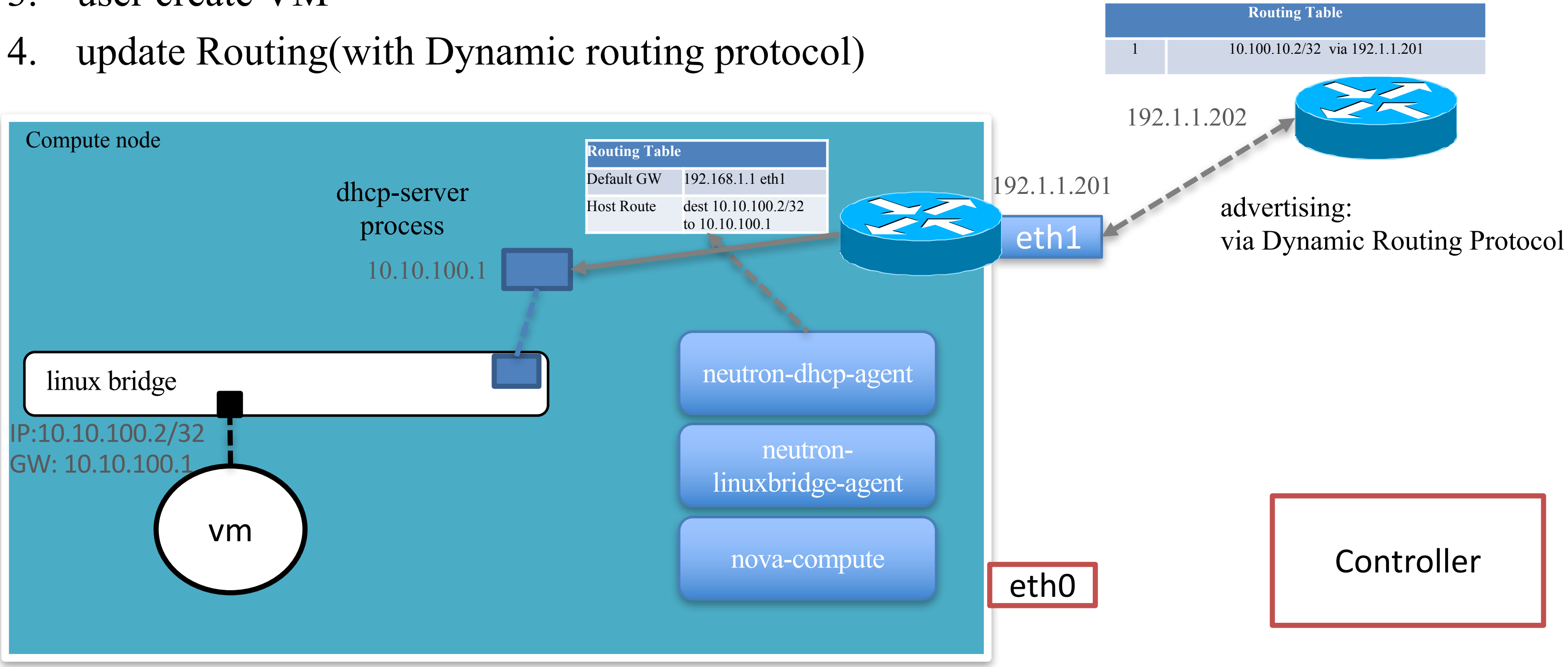
kakao

1. install nova/neutron agent.
2. create neutron network (name: freenet, subnet: 10.10.100.0/24)
3. user create VM



How it works

1. install nova/neutron agent.
2. create neutron network (name: freenet, subnet: 10.10.100.0/24)
3. user create VM
4. update Routing(with Dynamic routing protocol)



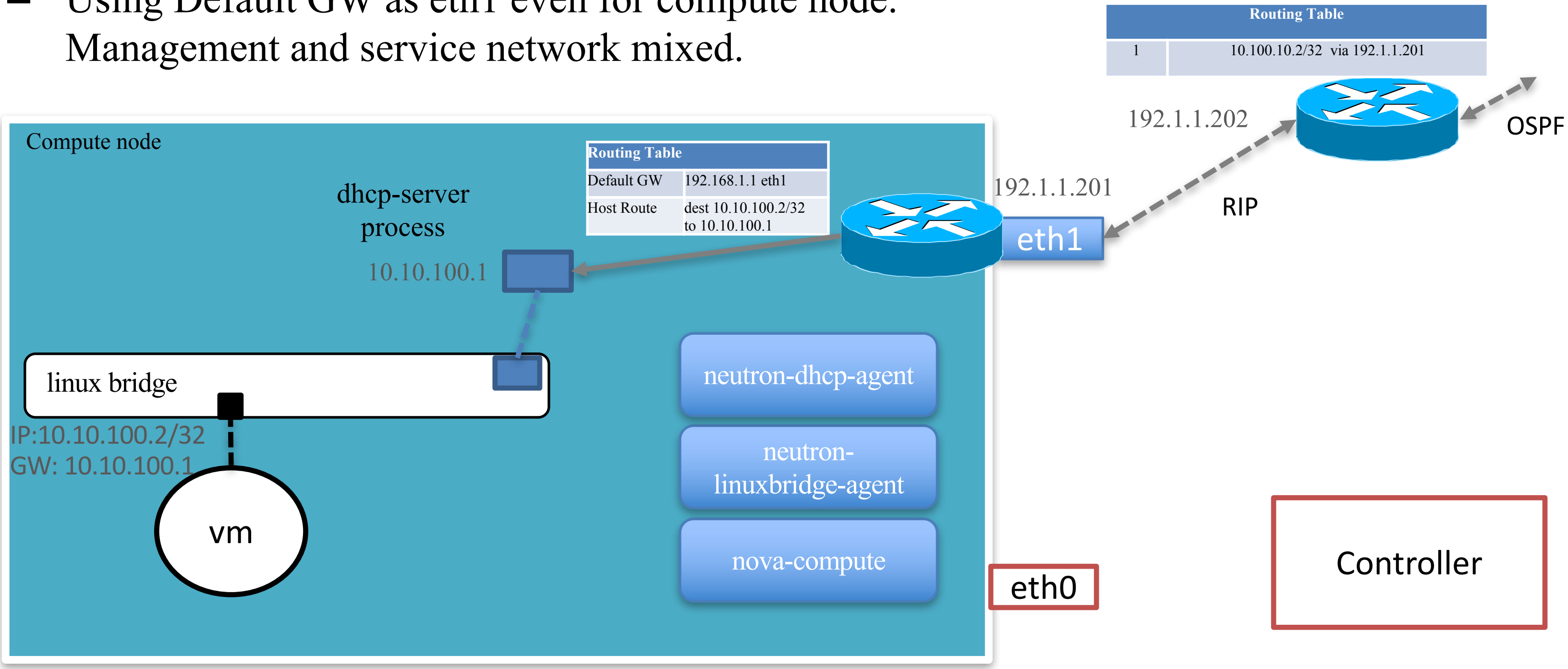
Routing Table	
1	10.100.10.2/32 via 192.1.1.201

Routing Table	
Default GW	192.168.1.1 eth1
Host Route	dest 10.10.100.2/32 to 10.10.100.1

Phase 1

Use RIP and OSPF

- Heterogeneous setting will be burden
- Using Default GW as eth1 even for compute node. Management and service network mixed.



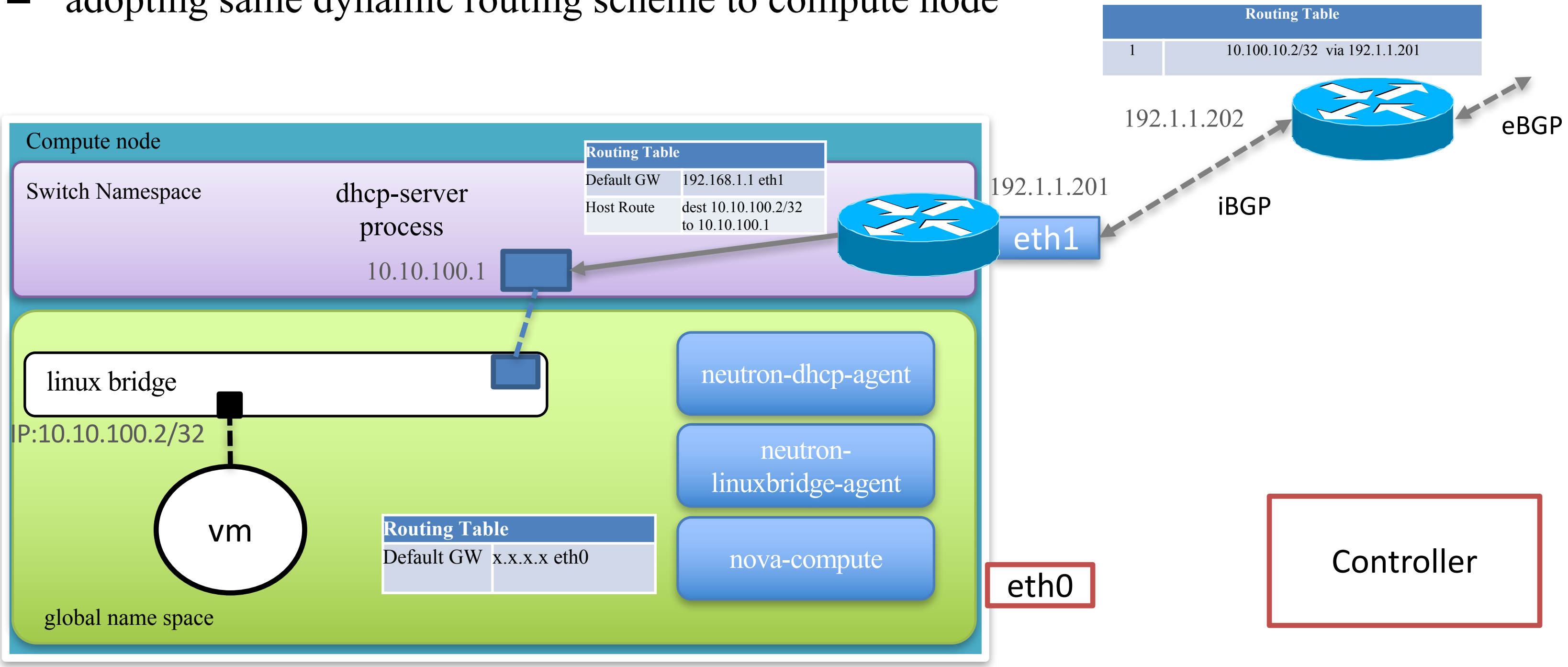
Routing Table	
1	10.100.10.2/32 via 192.1.1.201

Routing Table	
Default GW	192.168.1.1 eth1
Host Route	dest 10.10.100.2/32 to 10.10.100.1

Phase 2

Use BGP and switch namespace

- Isolating vm’s traffic using switch namespace.
- adopting same dynamic routing scheme to compute node

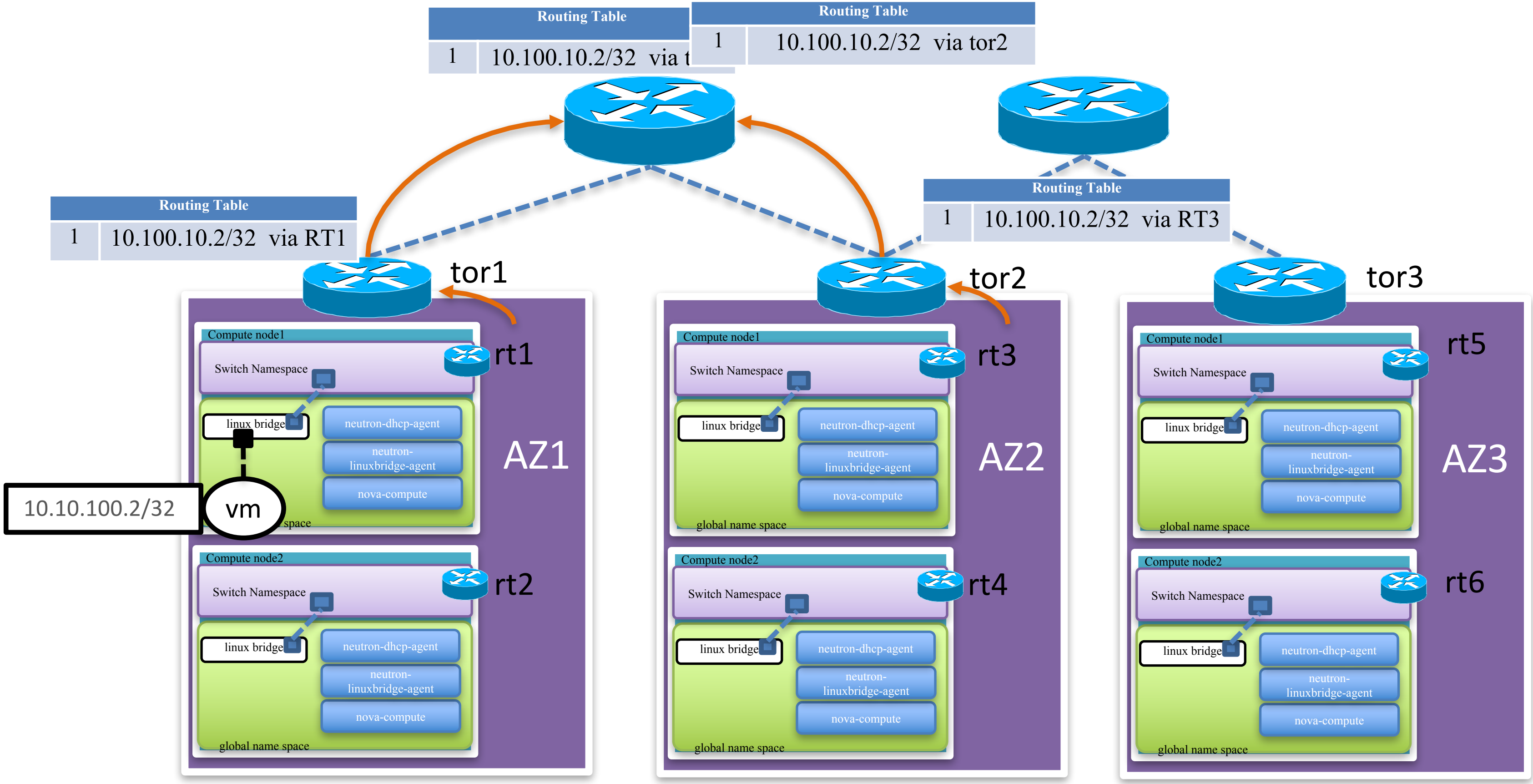


Routing Table	
1	10.100.10.2/32 via 192.1.1.201

Routing Table	
Default GW	192.168.1.1 eth1
Host Route	dest 10.10.100.2/32 to 10.10.100.1

Routing Table	
Default GW	x.x.x.x eth0

What we solved?



Routing Table	
1	10.100.10.2/32 via t

Routing Table	
1	10.100.10.2/32 via tor2

Routing Table	
1	10.100.10.2/32 via RT1

Routing Table	
1	10.100.10.2/32 via RT3

What we solve?

Simple IP planning

- only IP ranges matter. (no more VLAN, IP subnet, Router planning)

Resource imbalancing

- No chance of IP imbalancing.

Fault Resilience

- If one router gone, it propagated by Dynamic routing protocol to other router

Distributed

- deciding routing path is very distributed. No single point of failure.
- scale out nature.

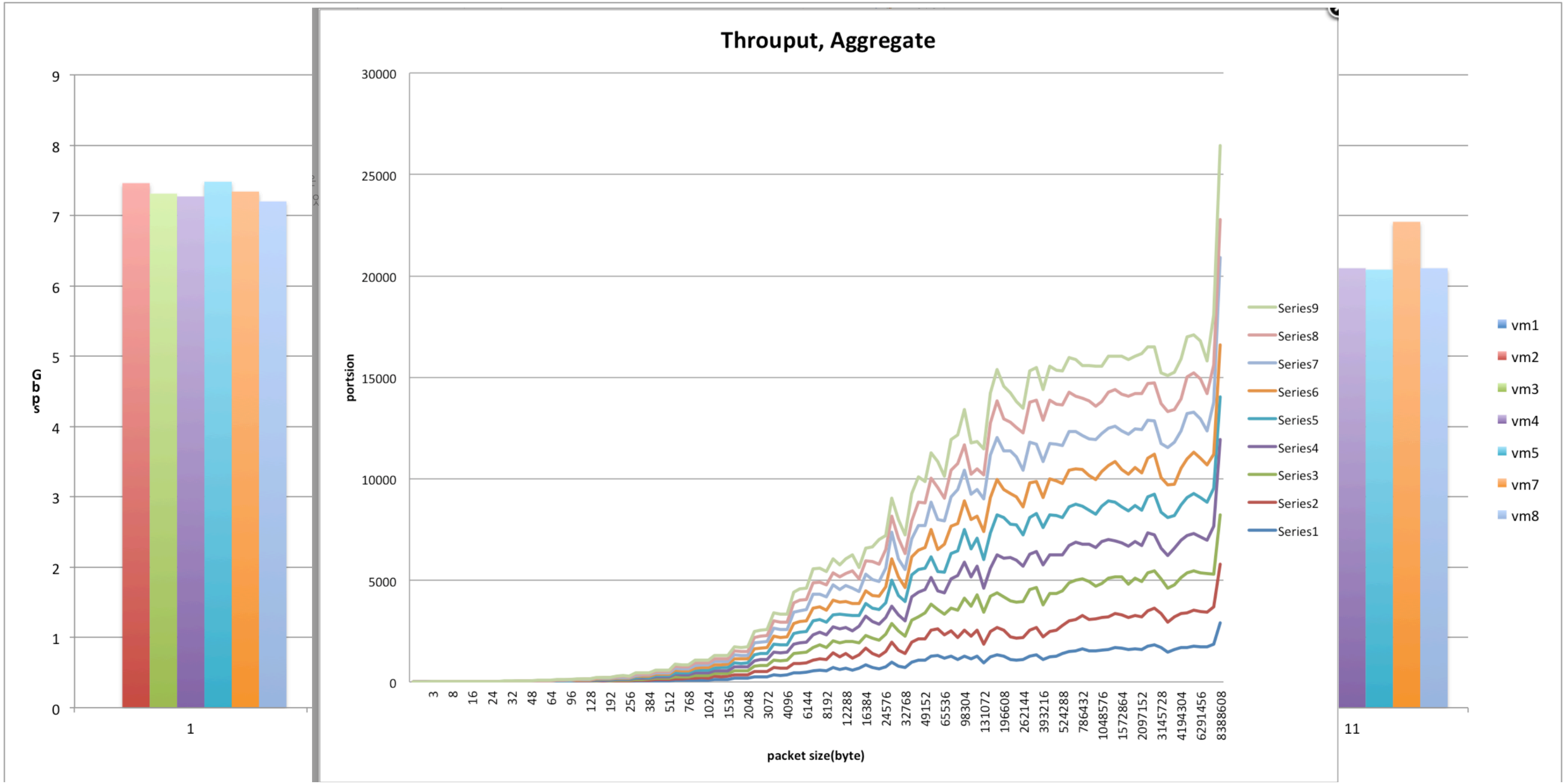
What we still have to solve?

Still many issue

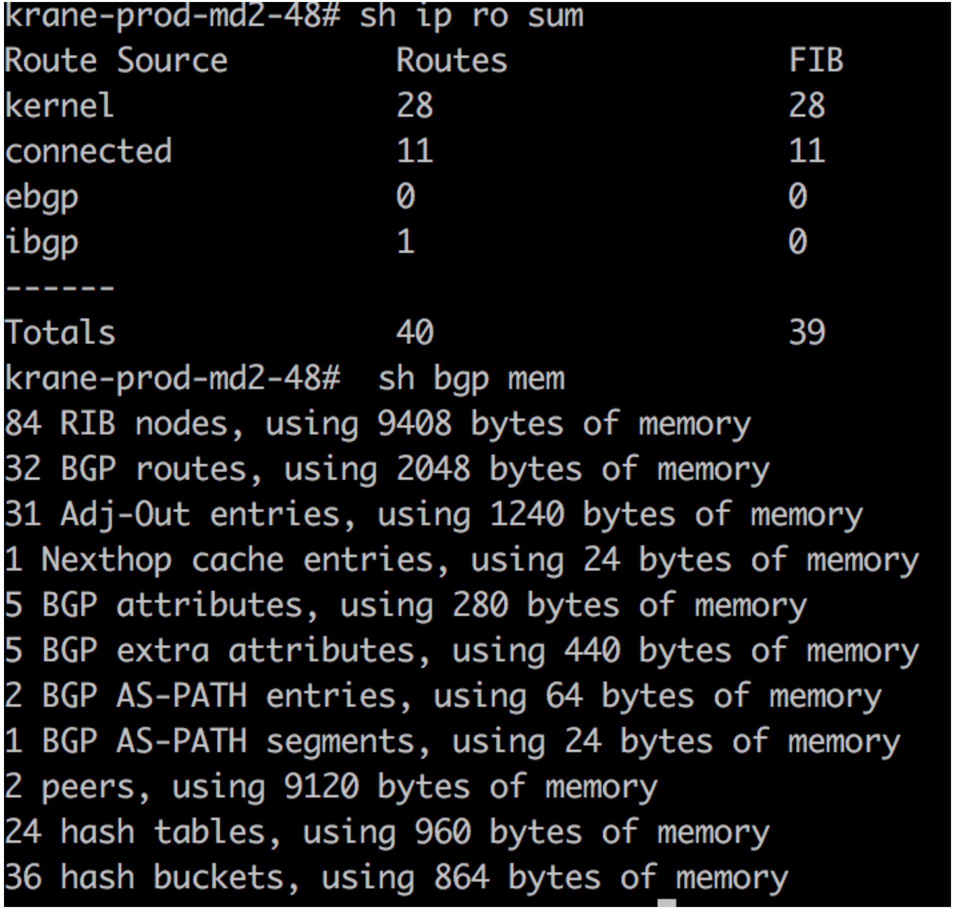
- Apply this to physical server
- Making Router setup by API (REST, RPC) using seed BGP(only advertising)
- ACL propagation using API (e.g. Flowspec)
- Shared storage base service

Performance Test VMs to VMs

kakao



Compute Node’s router status

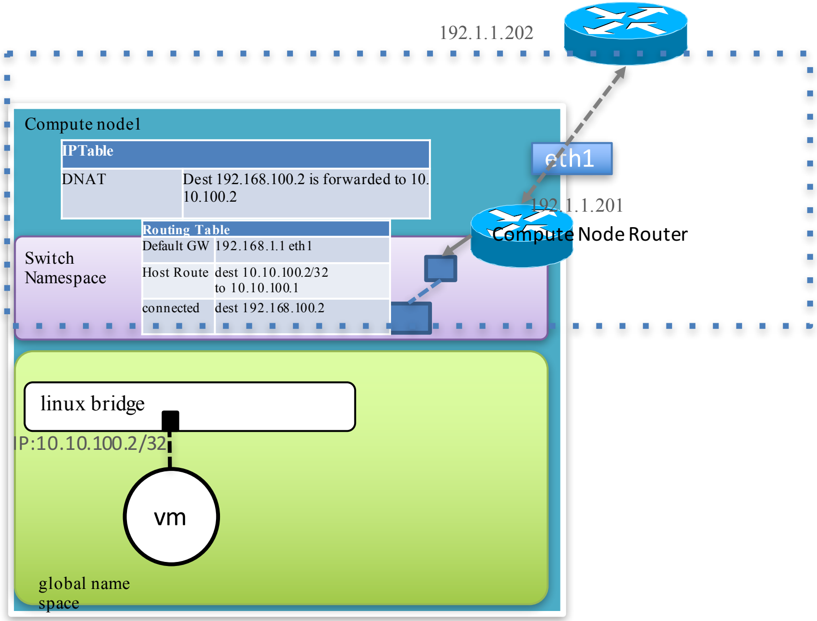


```
krane-prod-md2-48# sh ip ro sum
Route Source         Routes               FIB
kernel               28                   28
connected            11                   11
ebgp                 0                    0
ibgp                 1                    0
------
Totals               40                   39
krane-prod-md2-48#  sh bgp mem
84 RIB nodes, using 9408 bytes of memory
32 BGP routes, using 2048 bytes of memory
31 Adj-Out entries, using 1240 bytes of memory
1 Nexthop cache entries, using 24 bytes of memory
5 BGP attributes, using 280 bytes of memory
5 BGP extra attributes, using 440 bytes of memory
2 BGP AS-PATH entries, using 64 bytes of memory
1 BGP AS-PATH segments, using 24 bytes of memory
2 peers, using 9120 bytes of memory
24 hash tables, using 960 bytes of memory
36 hash buckets, using 864 bytes of memory
```

Application of /32bit network: /32bit route + DNAT → 1:1 NAT (A.K.A FloatingIP)

kakao

Routing Table	
1	10.10.100.2/32 via 192.1.1.201
2	10.10.100.3/32 via 192.168.1.202
3	192.168.100.2/32 via 192.168.1.201



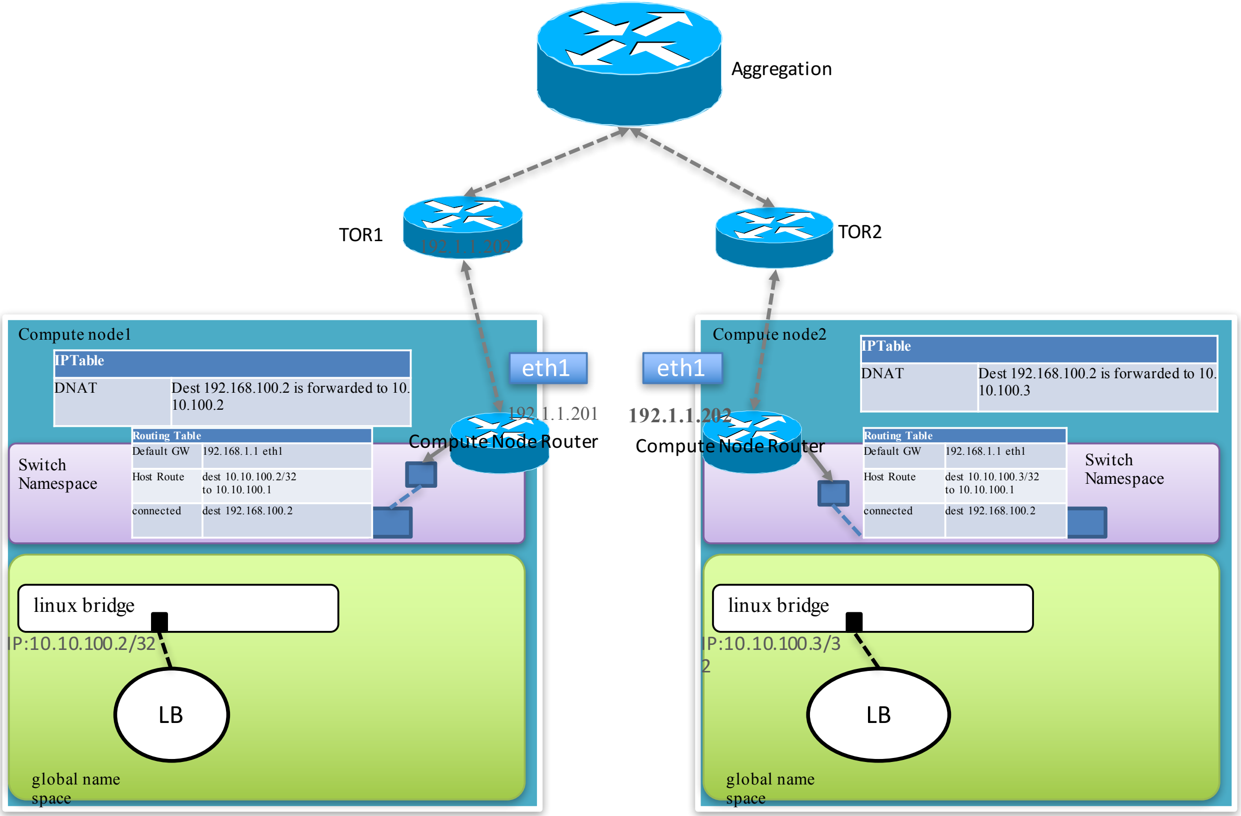
IPTable	
DNAT	Dest 192.168.100.2 is forwarded to 10.10.100.2

Routing Table	
Default GW	192.168.1.1 eth1
Host Route	dest 10.10.100.2/32 to 10.10.100.1
connected	dest 192.168.100.2

Application of /32bit network: ECMP + DNAT
→ Scalable Loadbalancer

kakao

VIP: 192.168.100.2 is ECMPed



Compute node1

IPTable	
DNAT	Dest 192.168.100.2 is forwarded to 10.10.100.2

Routing Table	
Default GW	192.168.1.1 eth1
Host Route	dest 10.10.100.2/32 to 10.10.100.1
connected	dest 192.168.100.2

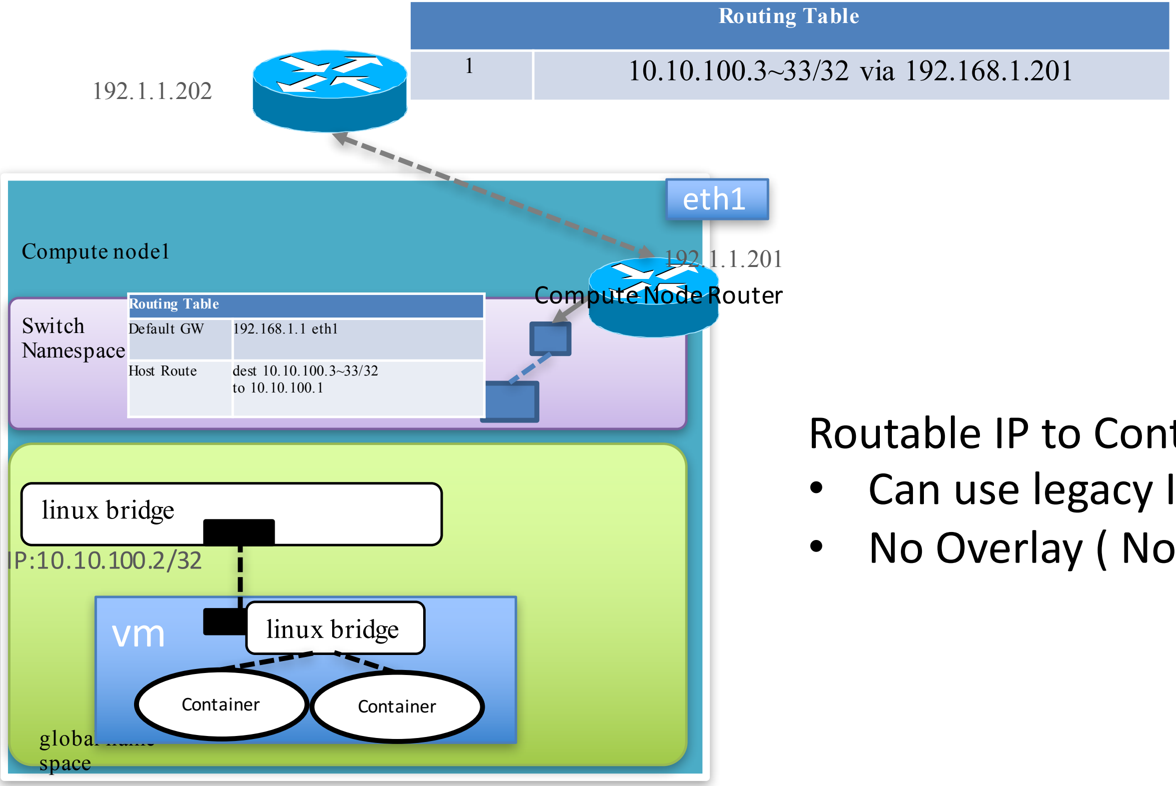
Compute node2

IPTable	
DNAT	Dest 192.168.100.2 is forwarded to 10.10.100.3

Routing Table	
Default GW	192.168.1.1 eth1
Host Route	dest 10.10.100.3/32 to 10.10.100.1
connected	dest 192.168.100.2

Application of /32bit network:

Multiple Routing Entry (AKA, Fixed IPs) + Container Bridge Network → Scalable Container Network



Routing Table	
1	10.10.100.3~33/32 via 192.168.1.201

Routing Table	
Default GW	192.168.1.1 eth1
Host Route	dest 10.10.100.3~33/32 to 10.10.100.1

Routable IP to Container:

- Can use legacy IP base Monitoring
- No Overlay (No complexity)

Q&A

Thanks