Introduction

Requirements of NTT network

- □ NTT groups have provided various services with reliability and scalability **by dedicated high-end routers.**

High-end core routers

◆ Example of NTT Regional Communications Business

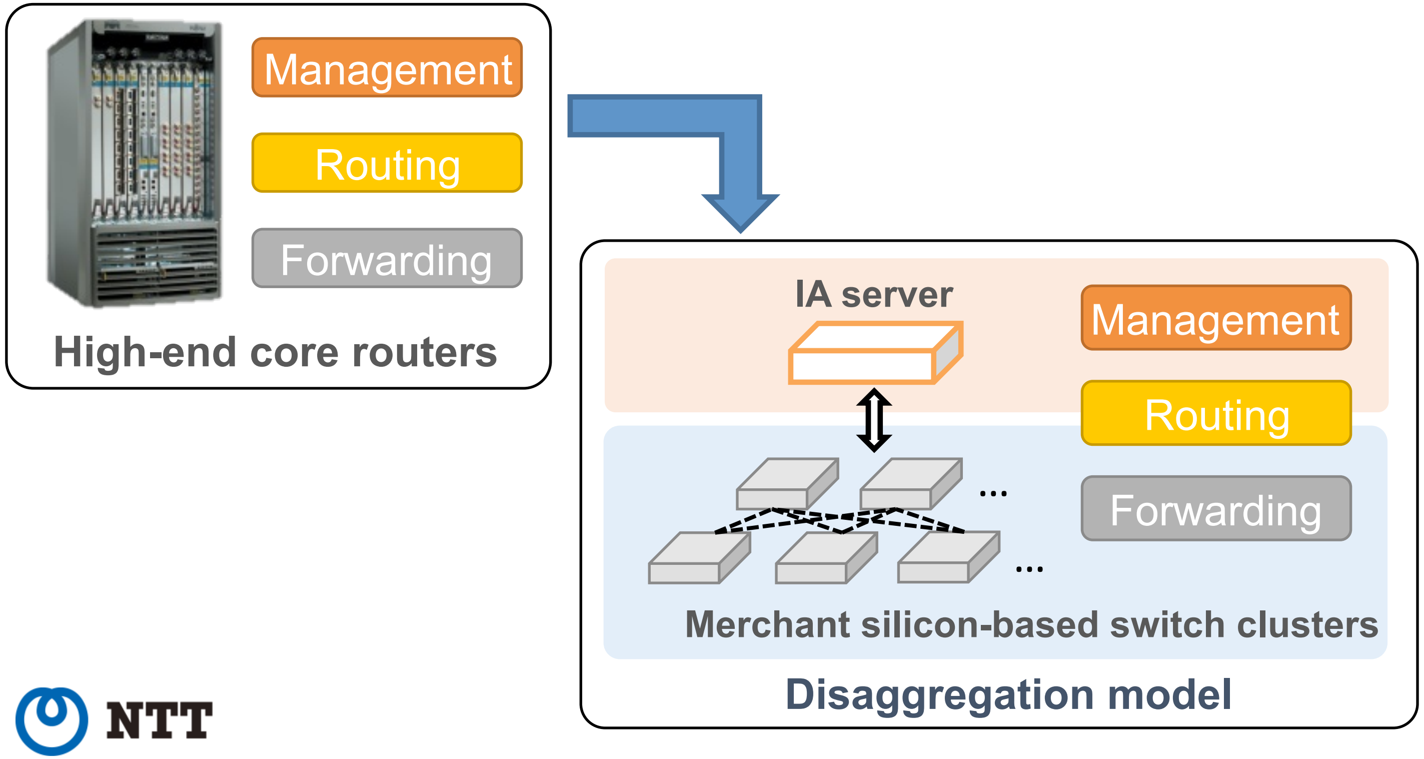
Service		
	Network services	Internet(IPv4, IPv6), Telephone, Telecast
	Additional functions	PPPoE, IPv6 native etc.
Scalability		
	Route	Over a few hundreds of thousands routes
	Traffic	Over tens of Tbps
Quality		
	Reliability	Redundancy of each function Rapid failover time - In-device failure < a few seconds - Inter-device failure < a few tens of seconds
	Recovery operation	Internet: within 2 hours VoIP: within less time than internet

Introduction

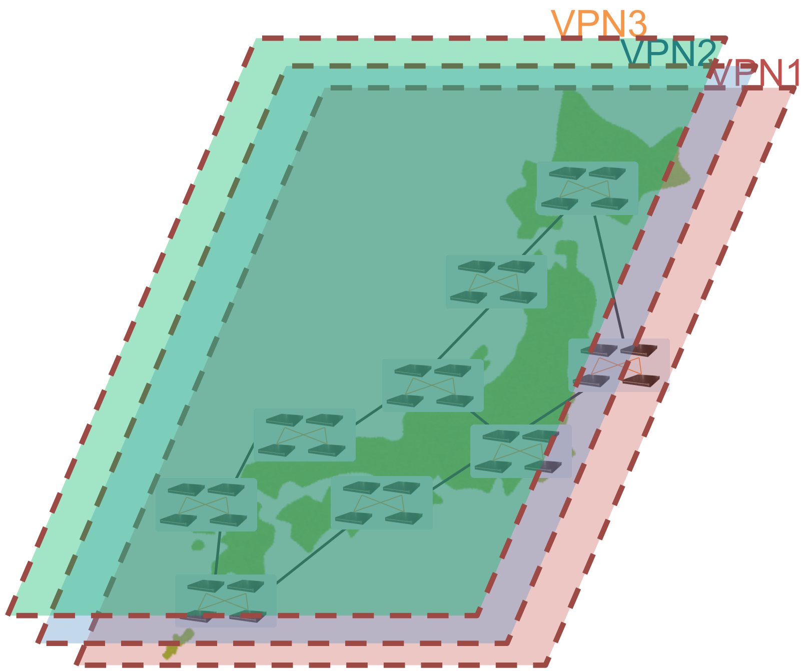
Expectation for future carrier network

- □ Disaggregation of dedicated high-end core routers
  - ✓ Especially in OTT, a merchant silicon-based switch has great demands.
  - ✓ CAPEX/OPEX savings and flexibility can be expected with commodity products.
- □ Providing E2E VPN service throughout carrier network
  - ✓ Wide-area underlay and VPN network function is required to meet the various network requirements.

◆ Disaggregation of dedicated high-end core routers



◆ VPN service throughout carrier network



Introduction

Activity

Our main activity

✓ Developing network architecture using commodity products

Multi Service Fabric

Beluganos

- https://github.com/multi-service-fabric/msf
- https://github.com/beluganos/beluganos

✓ Discussion with an open community about carrier requirements

ONF

TELECOM INFRA PROJECT

- https://www.opennetworking.org/
- https://telecominfraproject.com/

Our working about ONF

- ONOS/CORD verification
- Trying to close the gap between the verification results and our requirements

Today, we want to Introduce new carrier-grade IP fabric architecture,
and further expectation for programmable ASIC technique about P4/Stratum

Base network architecture

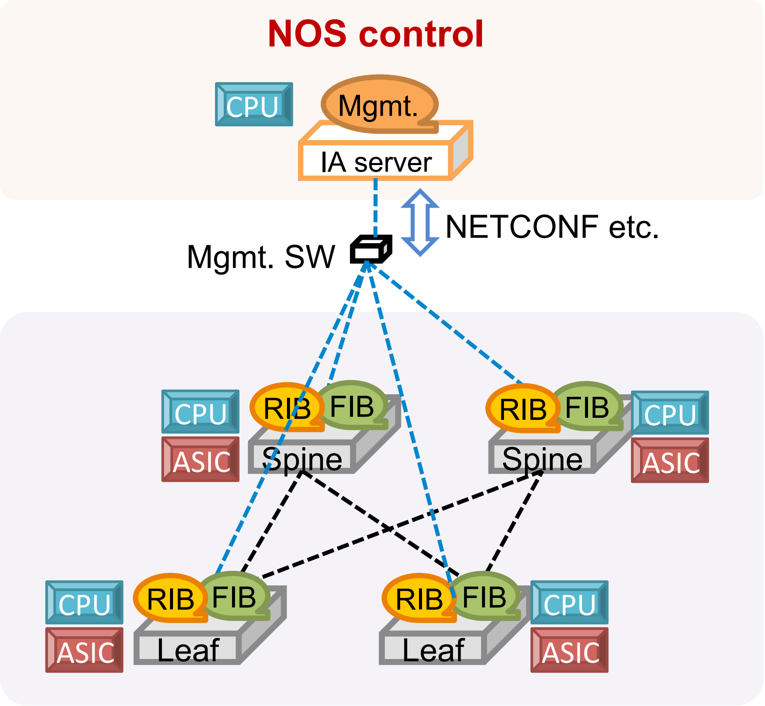
□ There are two base architectures regarding the deployment of the routing function.

- ✓ Distributed control architecture – Deploy routing functions on all switches
- ✓ Centralized control architecture – Deploy routing functions on central controller

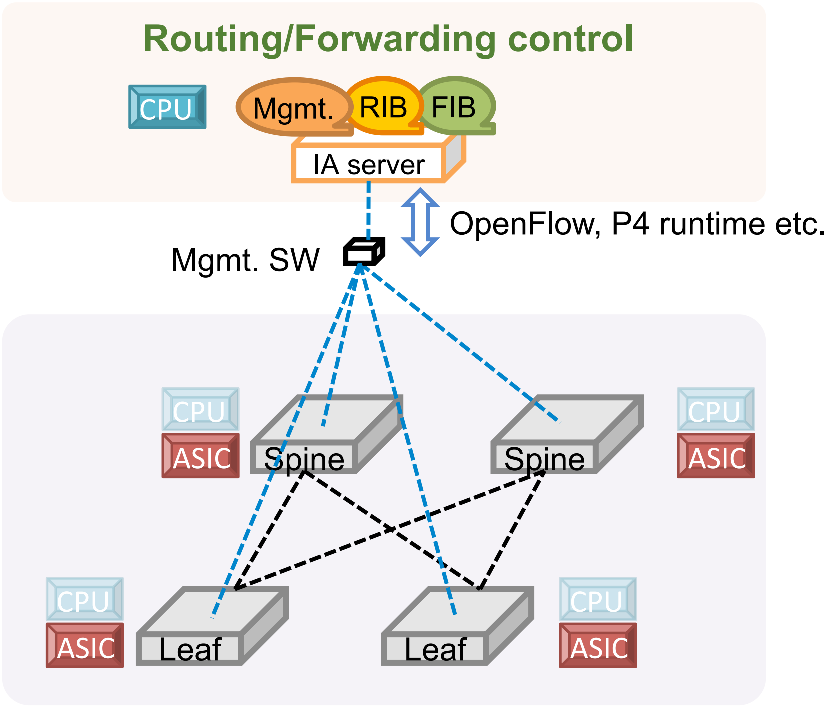
➢ Legend

- Mgmt. — SW device and fabric management function
- RIB — RIB construction function such as “quagga”
- FIB — FIB construction function such as “ONOS” and “Ryu”
- ▭ — Forwarding function with forwarding database
- CPU — Central processing unit for software processing
- ASIC — ASIC

◆ Distributed control architecture



◆ Centralized control architecture

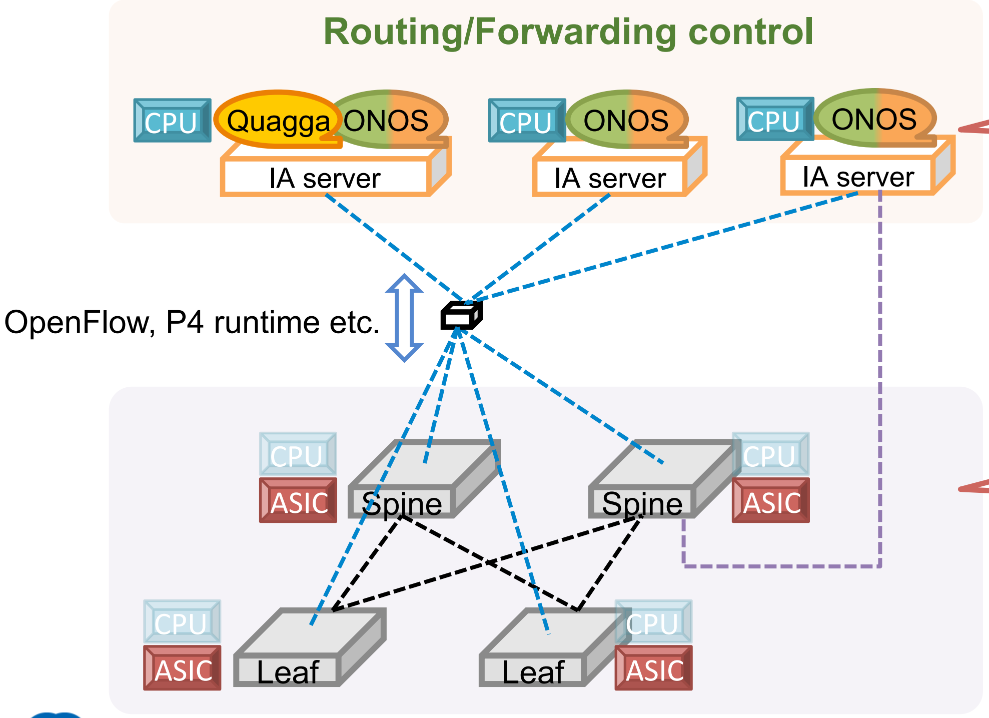


Disaggregate network architecture

ONOS solution for centralized control

- □ Traditional centralized control architecture could control multiple disaggregated devices as a single logical node, but there are a few disadvantage points.
  - ✓ The controller process load on IA server comes larger with the increase of the number of switches.
  - ✓ Management switch will be a single point of failure in total network PoD.
- □ ONOS has solved these problem with some original techniques.

◆ ONOS architecture



✓ The controller process load distribution

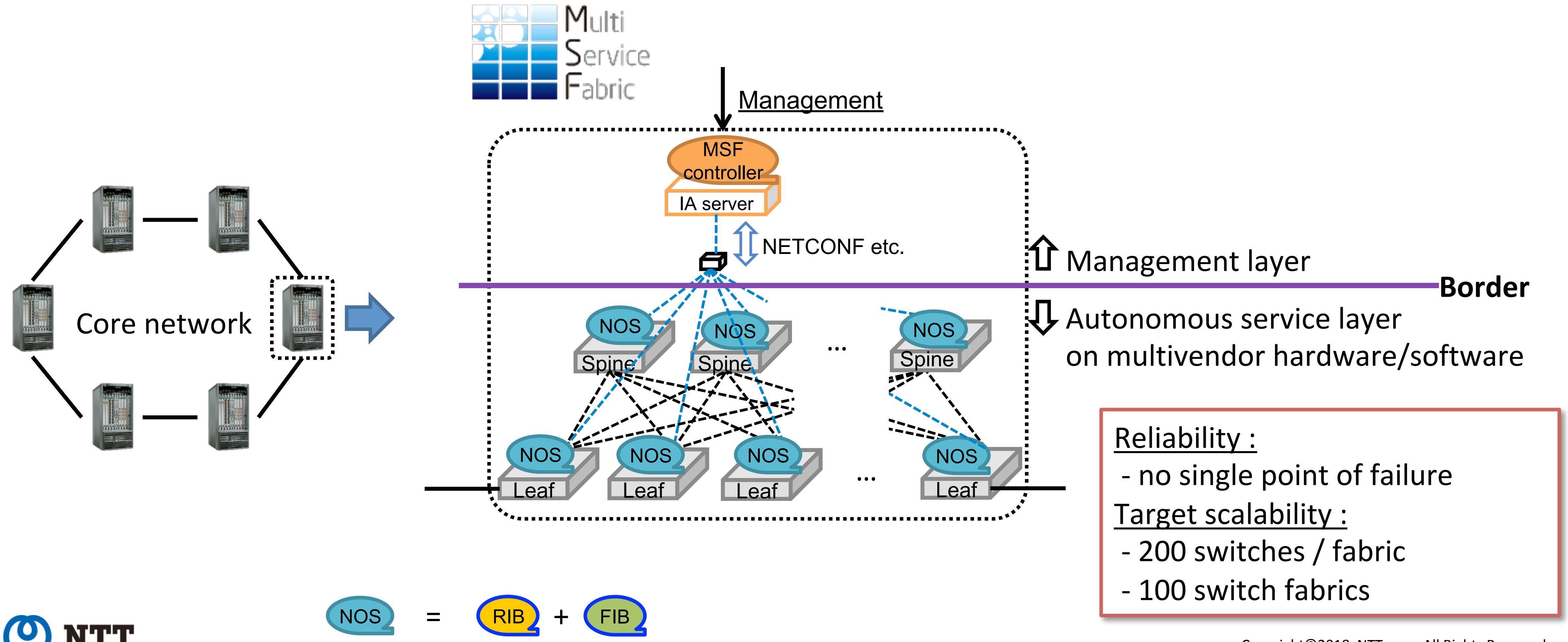
- ➢ ONOS forming cluster capability enable the distribution of the calculation load into multiple IA servers.
  => High reliability and scalability

✓ SPOF avoidance on the management switch

- ➢ ONOS enable backup connection via data port on the switch by additional development of OF-DPA/Indigo
  => High reliability

Current disaggregated network architecture in NTT

- NTT have developed disaggregated network architecture “Multi-Service Fabric” with distributed and autonomous control technique for keeping today’s stable and reliable network architecture.



Reliability :

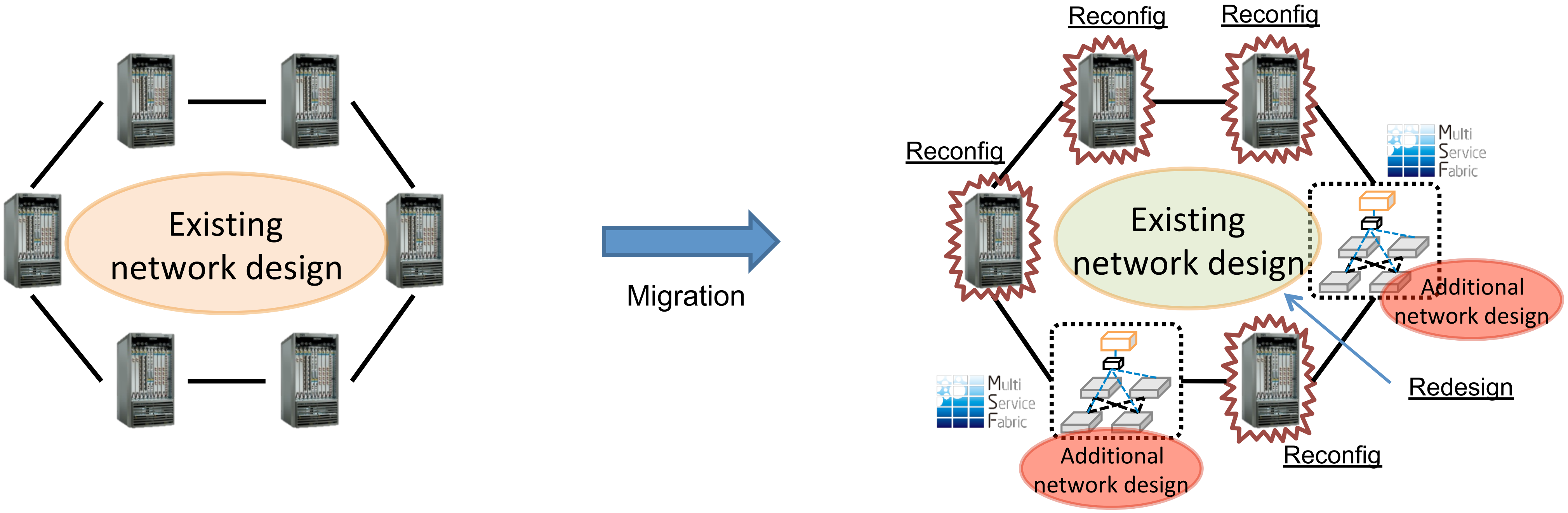
- no single point of failure

Target scalability :

- 200 switches / fabric
- 100 switch fabrics

Further improvement of disaggregated architecture

- □ Considering compatibility for existing carrier network, however, the distributed control architecture would give impact to the existing design of the whole network.
- □ So, for further improvement of disaggregated network architecture, we need both advantages of centralized and distributed control technique.



Disaggregate network architecture

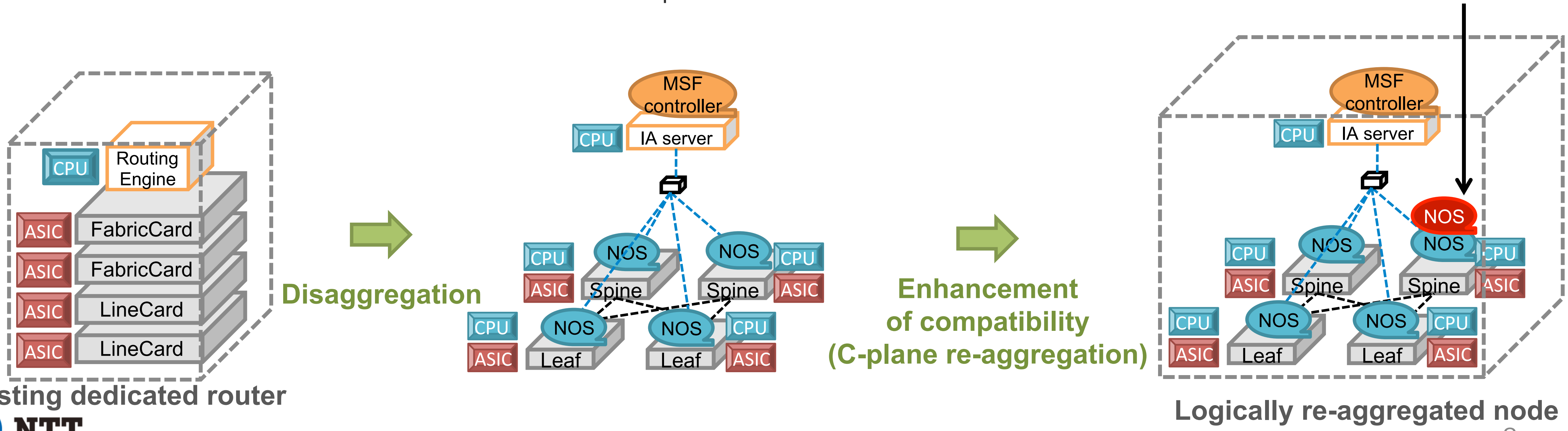
Further improvement of disaggregated architecture

□ Goal

Controlling disaggregated IP fabric as a single logical node with reliability, scalability and compatibility like existing carrier dedicated high-end router.

- Requirement
  - Carrier’s high reliability and scalability by autonomous stable service layer
  - Compatibility of existing network design
- Technique
  - Deployment of additional centralized network control function on switch
  - Flexible flow construction on ASIC from multiple network control functions

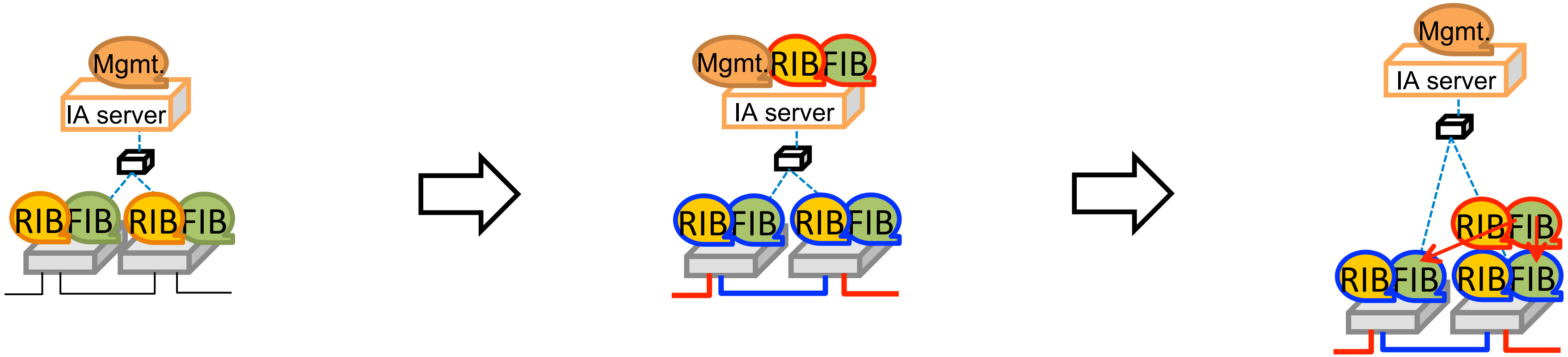
Deployment of re-aggregate centralized control function



Proposal overview

◆ Proposal 1: Deployment of the partially centralize control function

➢ We divide existing routing functions into two types, internal and external,
and external functions perform inbound-based centralized control of switch fabric.



➢ Distributed architecture

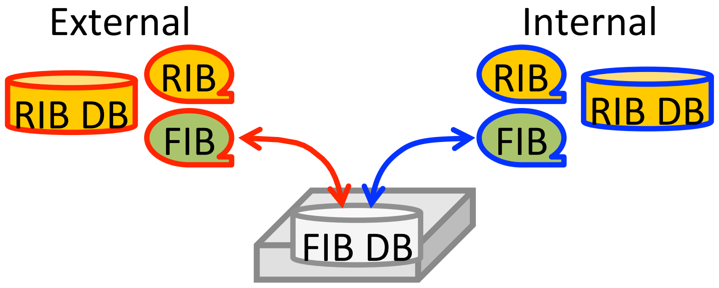
➢ Deploying additional centralized control function

➢ Inbound-based centralized control

◆ Proposal 2: Combining two routing information on ASIC table

✓ The internal and external routing information should be stored separately in the switch fabric.

✓ The forwarding information should be constructed
by internal and external FIB construction functions individually.

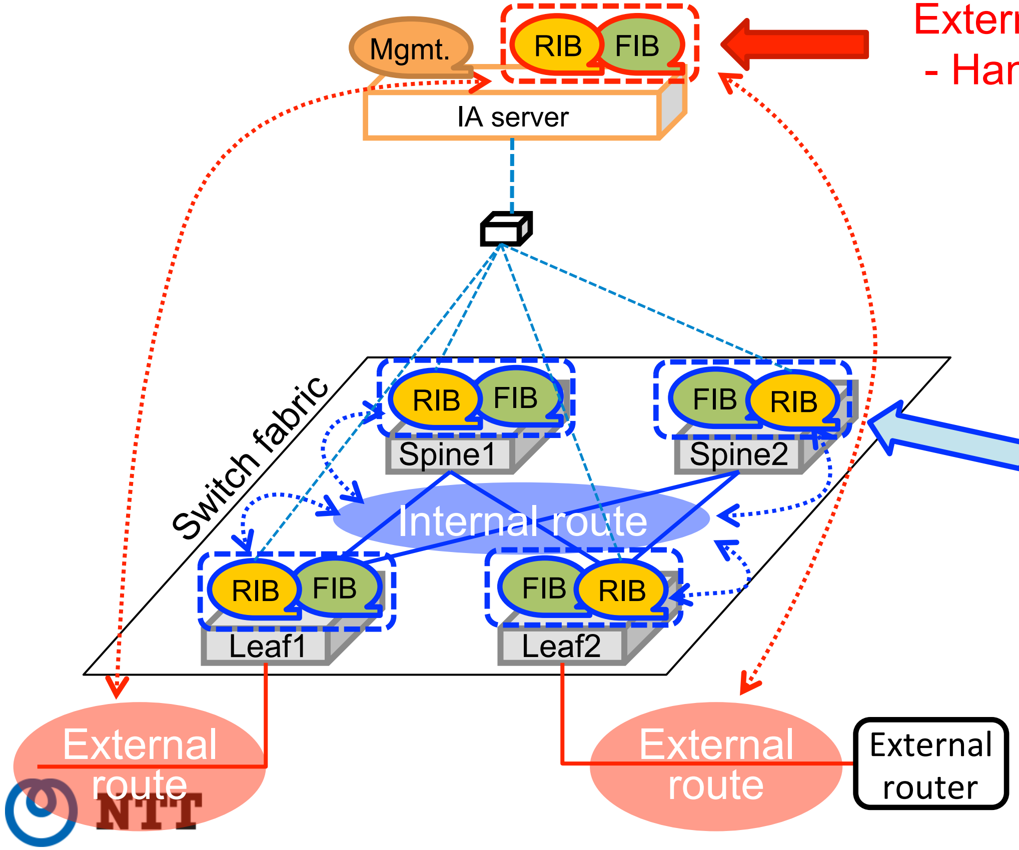


Combining on ASIC table

Proposed architecture

Proposal 1-(1): Deployment of additional centralized function

- We divided existing network functions (RIB and FIB construction functions) into two types of functions, for the internal route information and for the external route information of the fabric.
  - ✓ The external route should be centrally controlled to keep compatibility as dedicated high-end router.
  - ✓ The internal route should be distributedly controlled to keep high reliability of existing network.

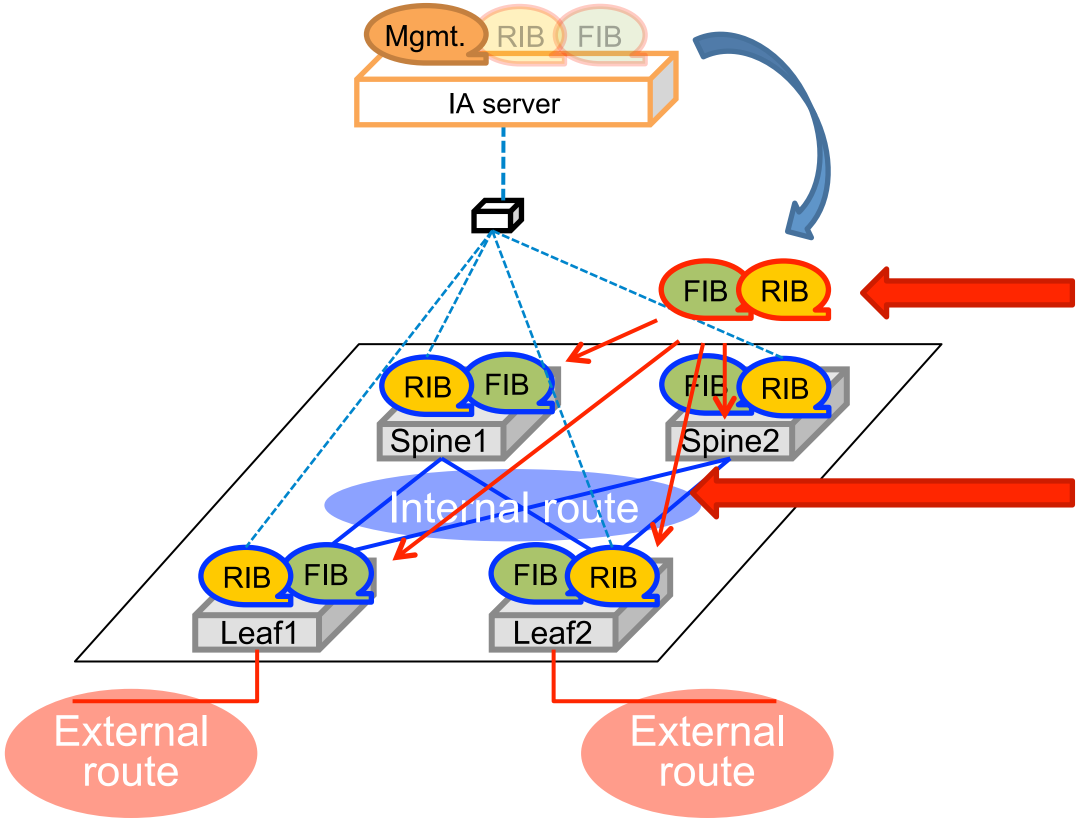


External function (Centralized control)
- Handling route information on the outside of the switch fabric

Internal functions (Distributed control)
- Handling route information on the inside of the switch fabric

Proposal 1: Inbound-based centralized control

- In order to keep fabric reliability, centralized control functions are deployed into the switch and centralized control connection is connected via data port.

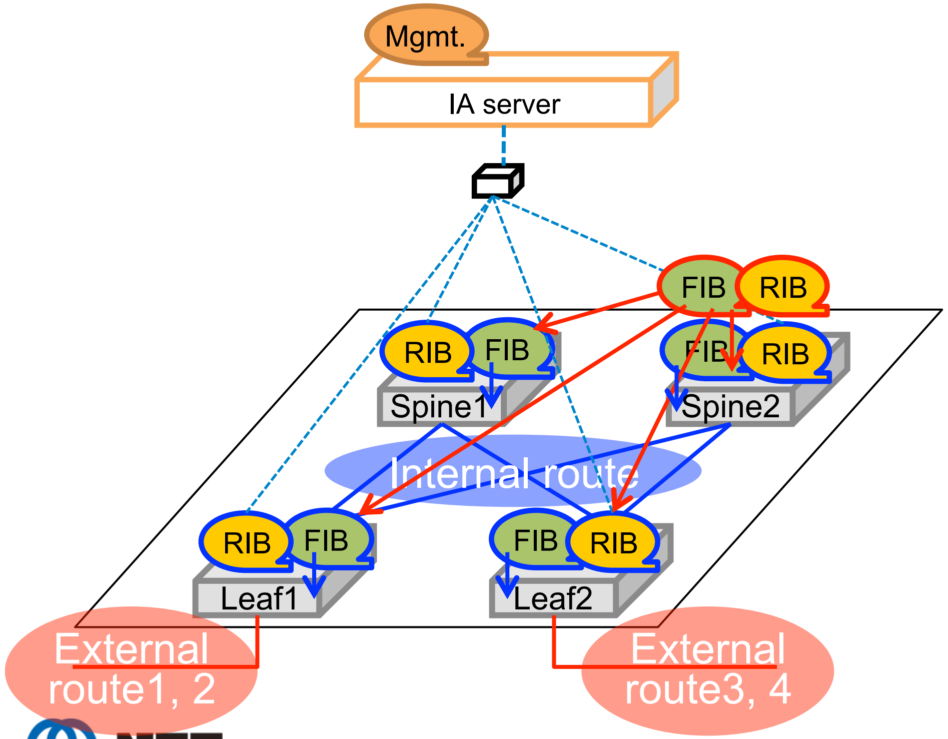


Deploying centralized control functions on the switch

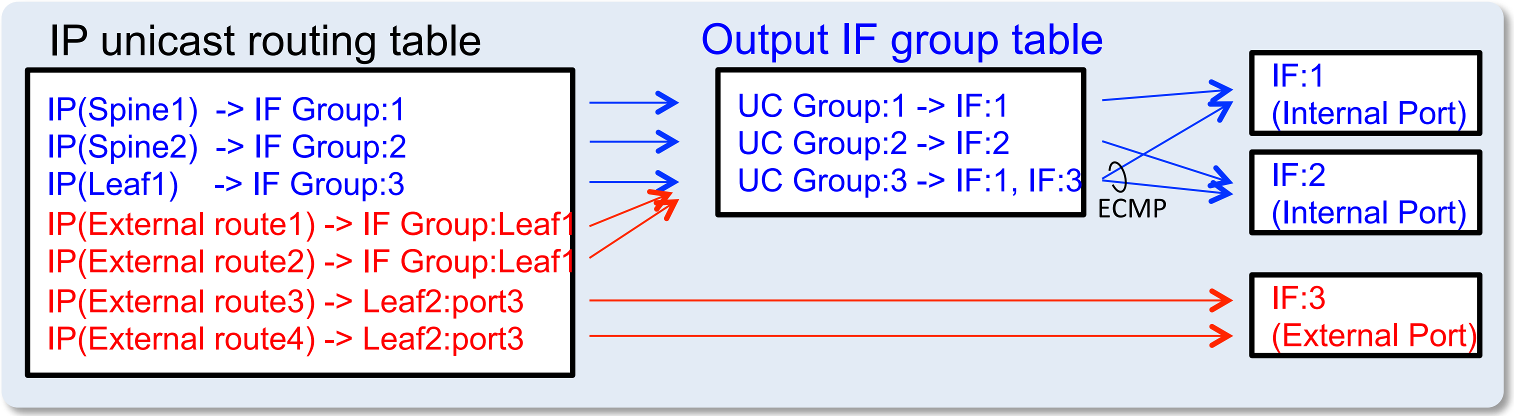
In-bound centralized control via data port on switch
(connection route has already solved by internal functions)

Proposal 2: Combining two information on ASIC table

- To forward an injected packet from external router properly, both of internal/external route information should be constructed on ASIC. So we applied the recursively looking up method on the ASIC by utilizing ASIC TTP.
  - ✓ Multiple functions could independently construct flow rule to ASIC.
  - ✓ Even when node or link failure occurs, re-calculation load is independent of each other’s route information.



Example of ASIC flow table on Leaf2



IP unicast routing table

IP(Spine1) -> IF Group:1
IP(Spine2) -> IF Group:2
IP(Leaf1) -> IF Group:3
IP(External route1) -> IF Group:Leaf1
IP(External route2) -> IF Group:Leaf1
IP(External route3) -> Leaf2:port3
IP(External route4) -> Leaf2:port3

Output IF group table

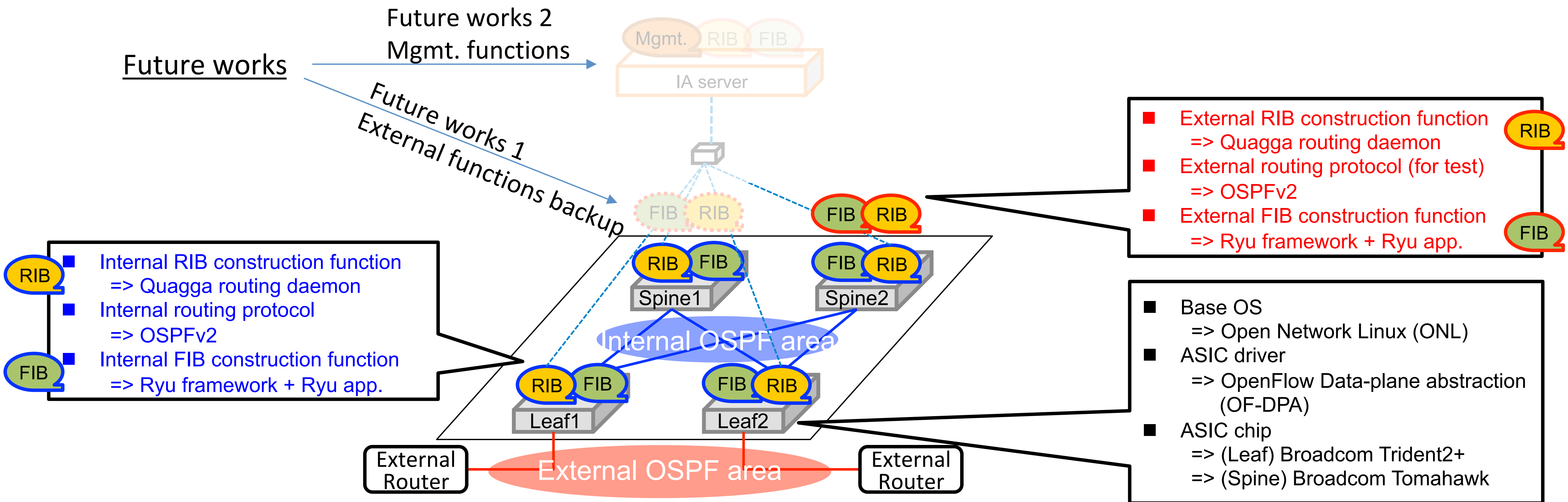
UC Group:1 -> IF:1
UC Group:2 -> IF:2
UC Group:3 -> IF:1, IF:3

IF:1 (Internal Port)
IF:2 (Internal Port)
IF:3 (External Port)

Test Implementation and Evaluation

Test implementation (OpenFlow)

- □ We implemented the proposed architecture by using open source software.
  We adopted Ryu framework* and Quagga routing suite** this time because we already have knowledge to deploy these functions directly on the switch base OS (ONL).
- □ We tested three viewpoints, logical node control, amount of calculation load and switching time when internal link failure. (compared with distributed control architecture)



- ■ Internal RIB construction function
  => Quagga routing daemon
- ■ Internal routing protocol
  => OSPFv2
- ■ Internal FIB construction function
  => Ryu framework + Ryu app.

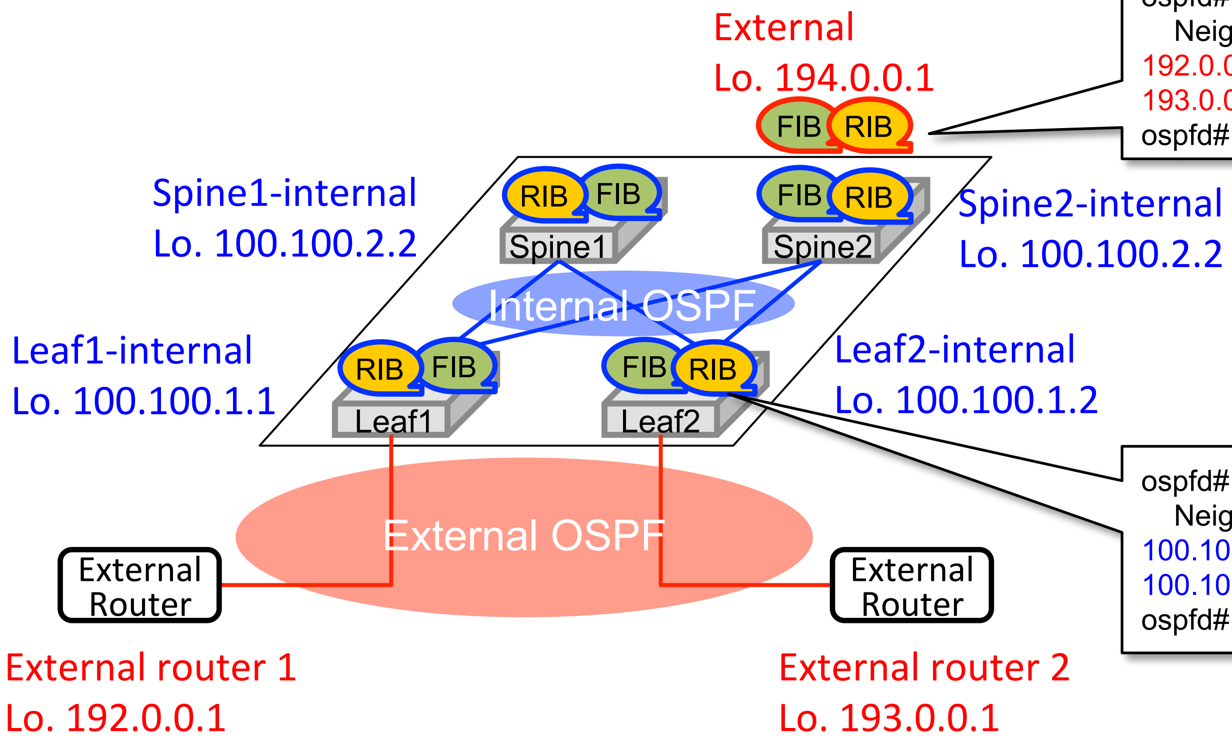
- ■ External RIB construction function
  => Quagga routing daemon
- ■ External routing protocol (for test)
  => OSPFv2
- ■ External FIB construction function
  => Ryu framework + Ryu app.

- ■ Base OS
  => Open Network Linux (ONL)
- ■ ASIC driver
  => OpenFlow Data-plane abstraction (OF-DPA)
- ■ ASIC chip
  => (Leaf) Broadcom Trident2+
  => (Spine) Broadcom Tomahawk

* Ryu framework - https://osrg.github.io/ryu/ ** Quagga routing suite - https://www.quagga.net/

Result 1. Controlling switches as a single logical node

- Firstly, we confirmed the status of routing functions in the fabric.
  - ✓ External routers have connected only external routing functions in switch fabric
  - ✓ Internal functions have connected each other and not connected to external routers



```
ospfd# show ip ospf neighbor
   Neighbor ID Pri State           Dead Time Address         Interface           RXmtL RqstL DBsmL
192.0.0.1         0 Full/DROther     30.391s 172.16.3.2      910102:172.16.3.1       0     0     0
193.0.0.1         0 Full/DROther     31.300s 172.16.4.2      910202:172.16.4.1       0     0     0
ospfd#
```

```
ospfd# show ip ospf neighbor
   Neighbor ID Pri State           Dead Time Address         Interface           RXmtL RqstL DBsmL
100.100.2.2       1 Full/DR          30.292s 100.100.0.18    2253:100.100.0.17       0     0     0
100.100.2.1       1 Full/DR          30.284s 100.100.0.14    2254:100.100.0.13       0     0     0
ospfd#
```

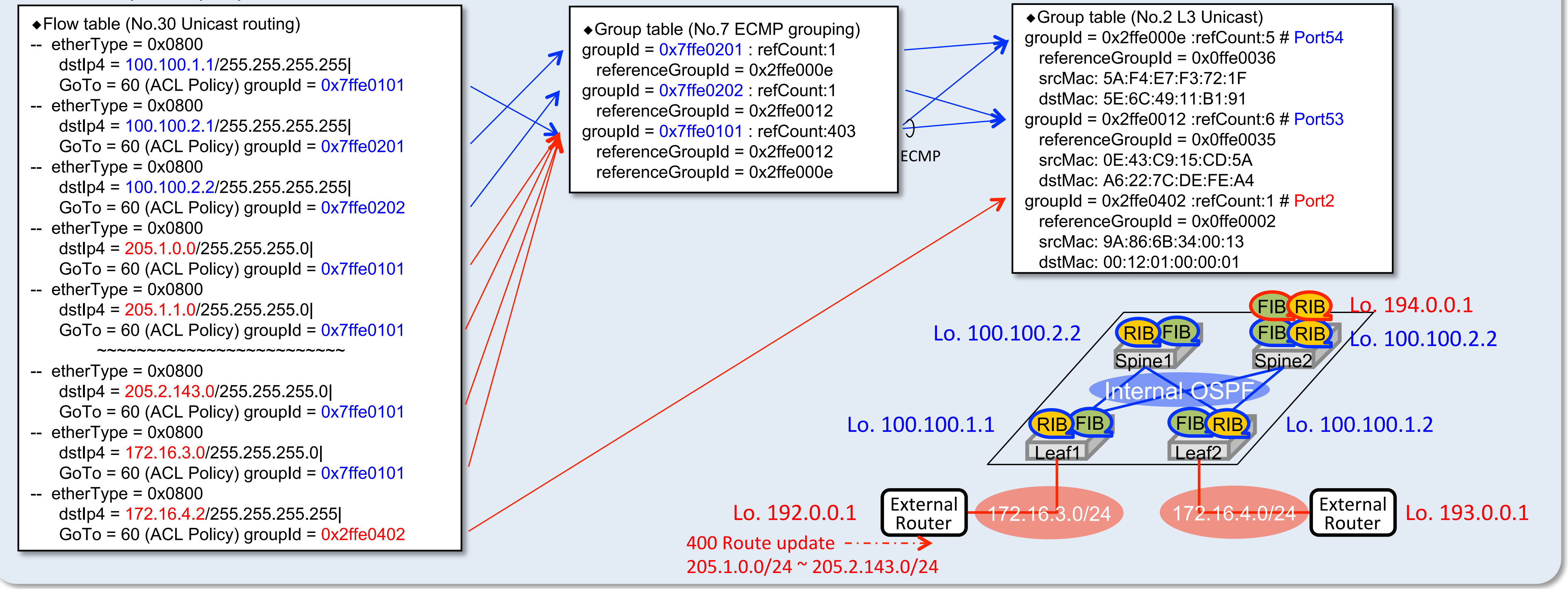
Proposed architecture could control multiple switches as a single logical node with distributed processing capability in the switch fabric

Test Implementation and Evaluation

Result 2. Combining routing information on ASIC

- □ Next, we experimentally demonstrated flow combining technique on “Broadcom TTP”.
  - ✓ We adopted “ECMP group table” to aggregate the information about output port in internal fabric
- □ Internal/external function can individually construct flows and packets were forwarded properly.

➢ Example of proposed architecture flow rule on ASIC of Leaf2



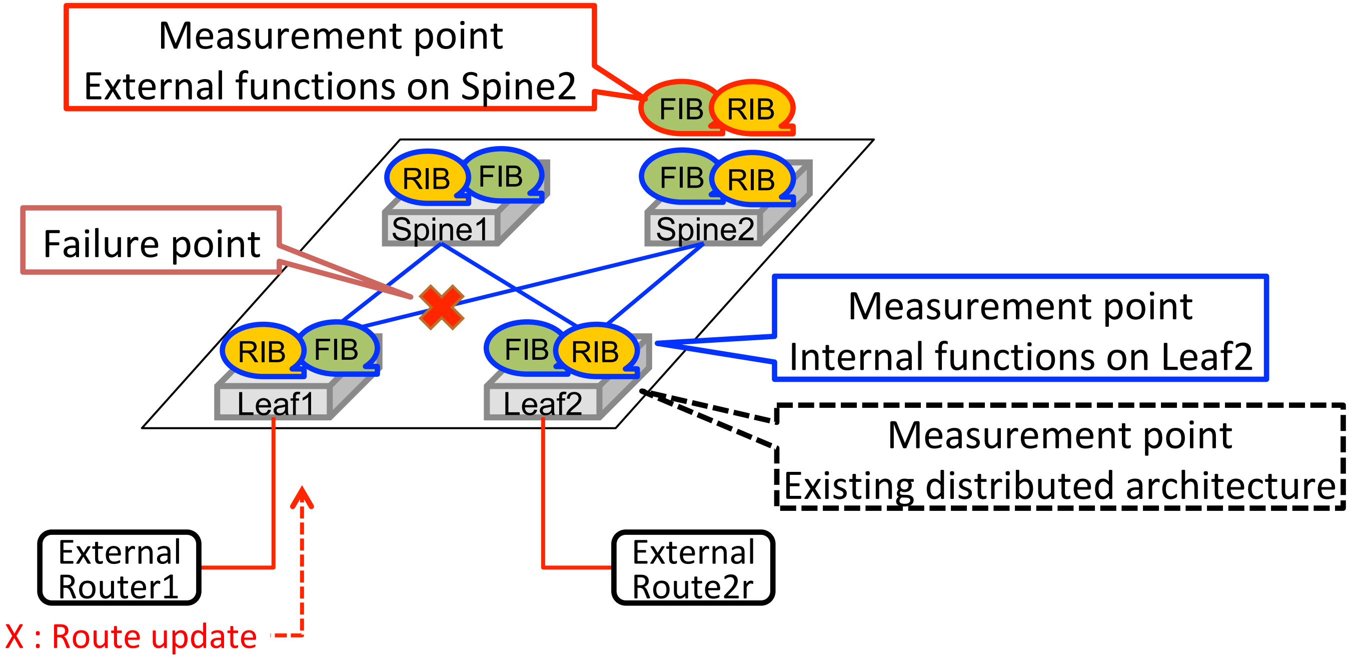
```
◆Flow table (No.30 Unicast routing)
-- etherType = 0x0800
    dstIp4 = 100.100.1.1/255.255.255.255|
    GoTo = 60 (ACL Policy) groupId = 0x7ffe0101
-- etherType = 0x0800
    dstIp4 = 100.100.2.1/255.255.255.255|
    GoTo = 60 (ACL Policy) groupId = 0x7ffe0201
-- etherType = 0x0800
    dstIp4 = 100.100.2.2/255.255.255.255|
    GoTo = 60 (ACL Policy) groupId = 0x7ffe0202
-- etherType = 0x0800
    dstIp4 = 205.1.0.0/255.255.255.0|
    GoTo = 60 (ACL Policy) groupId = 0x7ffe0101
-- etherType = 0x0800
    dstIp4 = 205.1.1.0/255.255.255.0|
    GoTo = 60 (ACL Policy) groupId = 0x7ffe0101
        ~~~~~~~~~~~~~~~~~~~~~~~~~~
-- etherType = 0x0800
    dstIp4 = 205.2.143.0/255.255.255.0|
    GoTo = 60 (ACL Policy) groupId = 0x7ffe0101
-- etherType = 0x0800
    dstIp4 = 172.16.3.0/255.255.255.0|
    GoTo = 60 (ACL Policy) groupId = 0x7ffe0101
-- etherType = 0x0800
    dstIp4 = 172.16.4.2/255.255.255.255|
    GoTo = 60 (ACL Policy) groupId = 0x2ffe0402
```

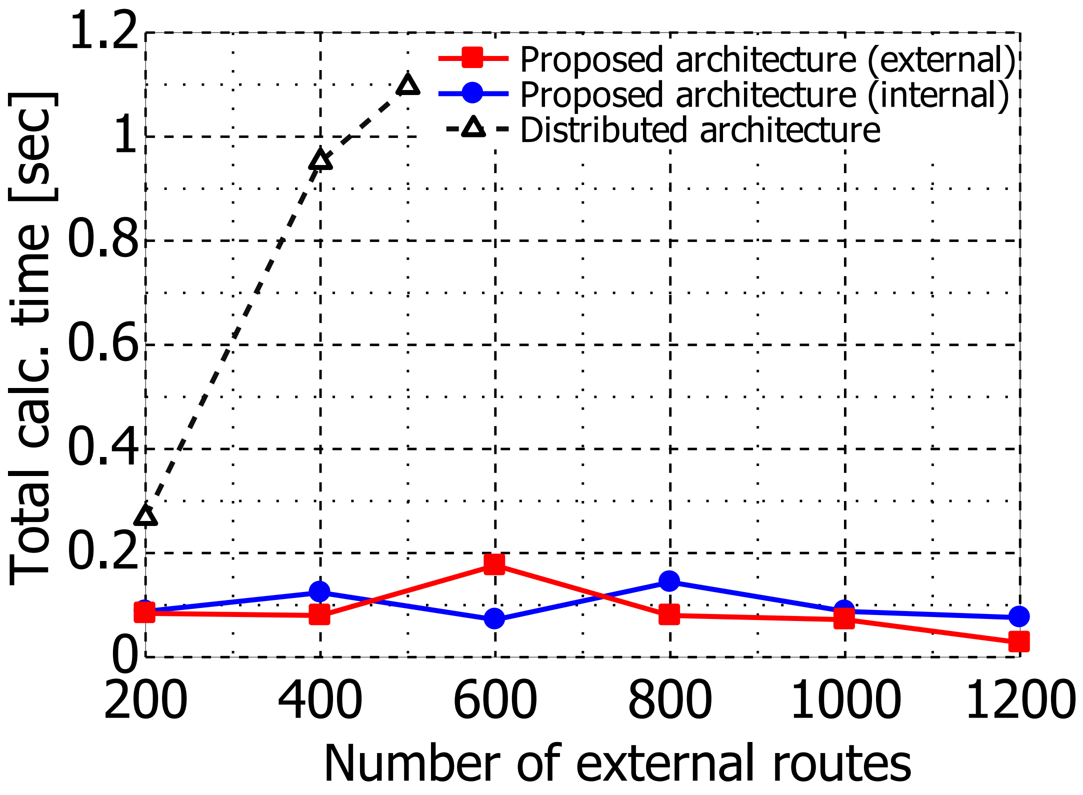
```
◆Group table (No.7 ECMP grouping)
groupId = 0x7ffe0201 : refCount:1
  referenceGroupId = 0x2ffe000e
groupId = 0x7ffe0202 : refCount:1
  referenceGroupId = 0x2ffe0012
groupId = 0x7ffe0101 : refCount:403
  referenceGroupId = 0x2ffe0012
  referenceGroupId = 0x2ffe000e
```

```
◆Group table (No.2 L3 Unicast)
groupId = 0x2ffe000e :refCount:5 # Port54
  referenceGroupId = 0x0ffe0036
  srcMac: 5A:F4:E7:F3:72:1F
  dstMac: 5E:6C:49:11:B1:91
groupId = 0x2ffe0012 :refCount:6 # Port53
  referenceGroupId = 0x0ffe0035
  srcMac: 0E:43:C9:15:CD:5A
  dstMac: A6:22:7C:DE:FE:A4
groupId = 0x2ffe0402 :refCount:1 # Port2
  referenceGroupId = 0x0ffe0002
  srcMac: 9A:86:6B:34:00:13
  dstMac: 00:12:01:00:00:01
```

Result 3. Calculation load of internal/external functions

- □ We measured calculation load of each component when internal link failure occurred in proposed architecture and distributed architecture.
- □ We could confirm that proposed architecture reduced the calculation load by dividing route information into internal and external.

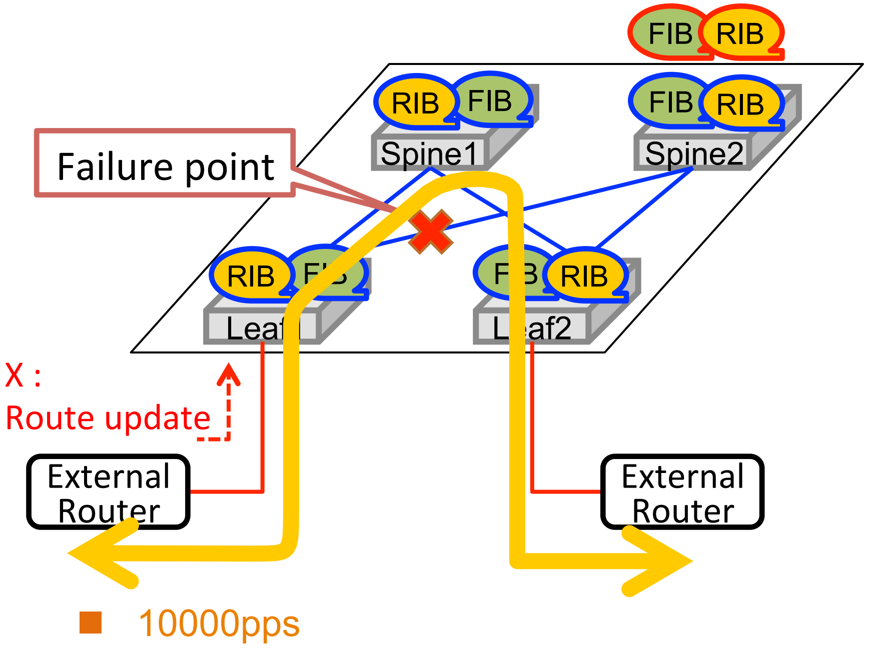


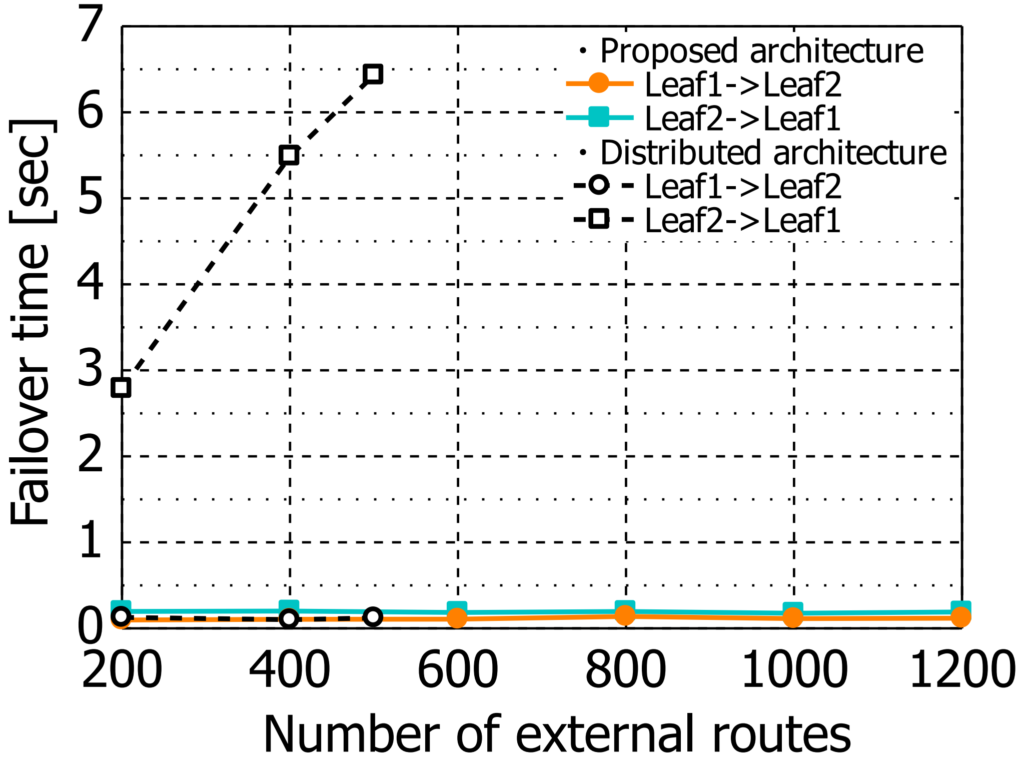


- ■ Measurement contents
  CPU total calculation time
- ■ Calculation component
  Quagga (zebra, ospfd)
  Ryu+Ryu app.

Result 4. Combining routing information on FIB

- Finally, we measured failover time when internal link failure occurred as a function of the number of external routes.
- In proposed architecture, we could confirm that failover time of internal link failure is independent of the number of external routes by dividing internal/external forwarding rules.



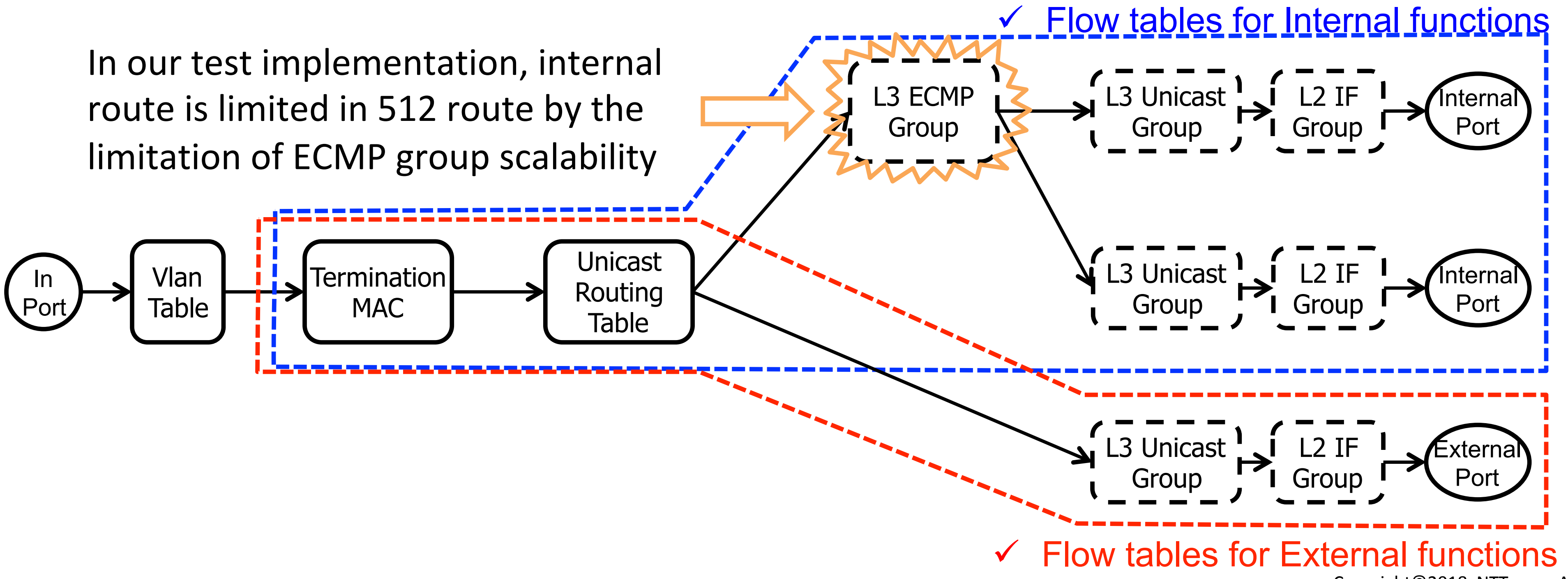


Expectation for P4/P4 runtime

1. Flexible and common TTP

- □ Test implementation highly depends on Broadcom chip original TTP.
  It has a possibility to lead some restrictions about scalability or something (internal route is limited in 512 route), and it also leads to dependence on specific chip implementation.
- □ For further flexible deployment of network functions and expansion of target chip, we are expecting for programmable chip techniques to define an appropriate table for our proposal.

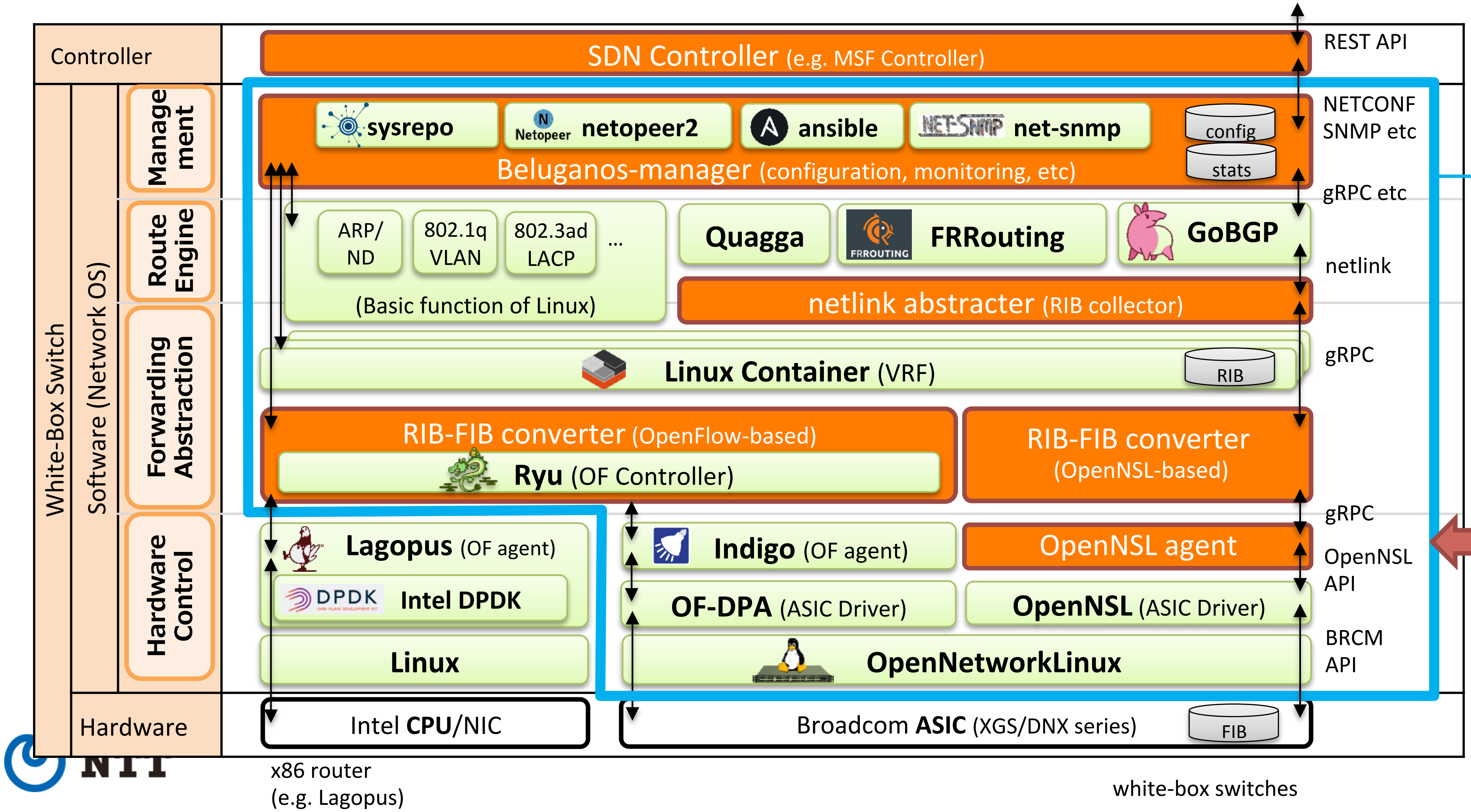
◆ Part of Broadcom chip TTP



Expectation for P4/P4 runtime

2. ASIC driver performance

- □ We are also developing open-source-based carrier-grade network OS, “Beluganos”, and we could confirm that switch performance highly depends on ASIC driver.
(About failover time, OpenNSL was approximately 20-times faster than OF-DPA)
- □ We expect P4/P4 runtime for more flexible definition of flow construction protocol.



Controller

SDN Controller (e.g. MSF Controller)

REST API

White-Box Switch

Software (Network OS)

Management

sysrepo | netopeer2 | ansible | net-snmp | config | stats

Beluganos-manager (configuration, monitoring, etc)

NETCONF
SNMP etc

gRPC etc

Route Engine

ARP/ND | 802.1q VLAN | 802.3ad LACP | …

(Basic function of Linux)

Quagga | FRRouting | GoBGP

netlink

netlink abstracter (RIB collector)

gRPC

Linux Container (VRF) | RIB

Forwarding Abstraction

RIB-FIB converter (OpenFlow-based)

Ryu (OF Controller)

RIB-FIB converter (OpenNSL-based)

gRPC

Hardware Control

Lagopus (OF agent)

Intel DPDK

Indigo (OF agent)

OpenNSL agent

OpenNSL API

OF-DPA (ASIC Driver)

OpenNSL (ASIC Driver)

Linux

OpenNetworkLinux

BRCM API

Hardware

Intel CPU/NIC

Broadcom ASIC (XGS/DNX series) | FIB

x86 router
(e.g. Lagopus)

white-box switches

Beluganos

OF: OpenFlow
RIB: Routing Information Base
FIB: Forwarding Information Base
GPB: Google Protocol Buffer

[Legend]

- Software

Open source (or published binary)

NTT’s effort (Open source)

data

- Hardware

Hardware

Hardware driver performance could not be clearly now. We are expecting that P4/P4 runtime clarifies definition of ASIC interface.

Conclusion

- □ Proposal of new IP fabric control architecture, which combines distributed control techniques with centralized control techniques.
  - ✓ Deploying two types of routing functions, proposed architecture enables the high-compatibility for existing network design with today's carrier-network autonomous stability.
  - ✓ By the inbound-based centralized control, a single point of failure in PoD could be avoided.

- □ Confirmation of the improvement from existing distributed architecture by test implementation of OpenFlow
  - ✓ We experimentally demonstrated proposed architecture by Quagga and Ryu-based OpenFlow control.
  - ✓ Test implementation enabled controlling four switches fabric as a single logical node.
  - ✓ Combining multiple network function information on ASIC saves CPU resources and leads fast failover time without advertising internal route to the outside of the logical node.
    (Compared with conventional distributed control architecture)

- □ Expectation for P4 or programmable ASIC technique to lead to
  - ✓ Flexible network function deployment
  - ✓ Fully utilizing hardware performance
  - ✓ Vendor agnostic chip control