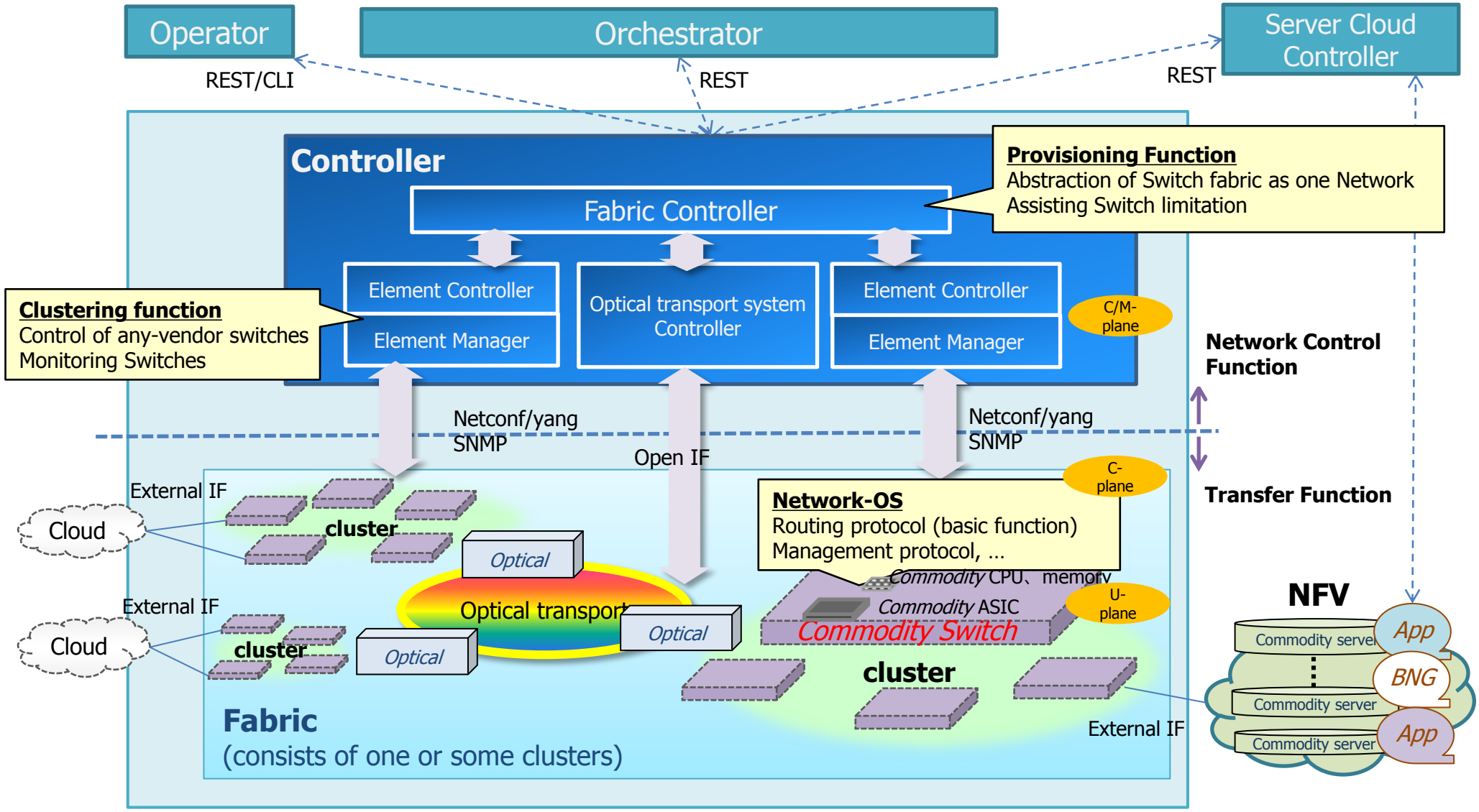
Technical Details

Jan. 2018

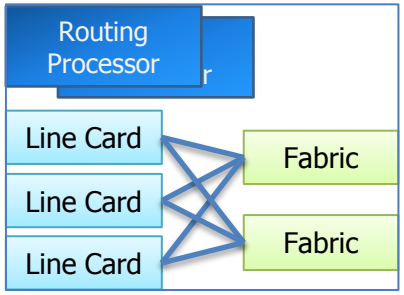
Architecture outline



Disaggregation concept

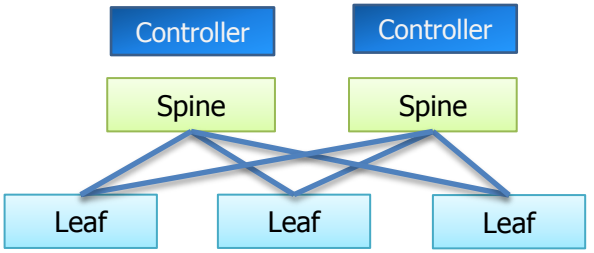
- Disaggregation for big device (e.g., high-end core routers) can be come due to technical progression for merchant silicon.
- Multi Service Fabric is a research for disaggregating router with distributed control plane which consists of standardized routing protocols and standard physical interface.
- Each device has autonomously control plane basically.
- SDN controller is centralized management system for numerous network nodes.
- Controller uses the same cluster service model to manage multi vendor switches.

Big Device(e.g., high-end core routers)



Disaggregation (Open and Any-Vendor)

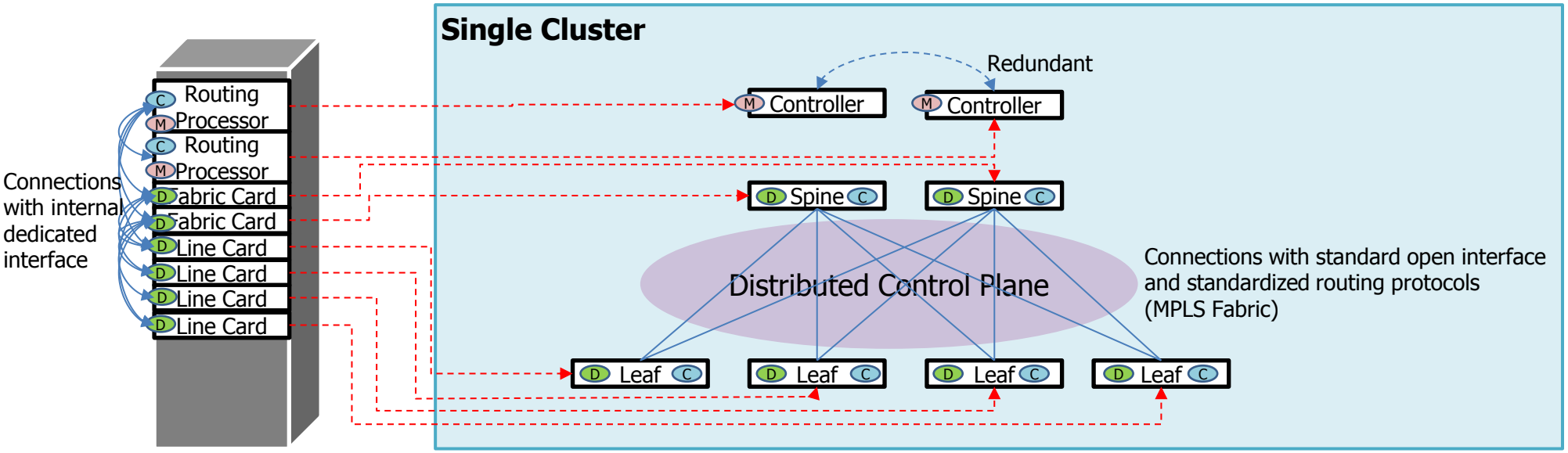
Switch fabric



Function	Big Device		Component	
Management Plane	Routing Processor	Processing Module	Controller	IA Server (VM)
Control Plane			Spine / Leaf	Datacenter SW
Data Plane (Total Switching)	Fabric	Switch Fabric Module	Spine	Datacenter SW
Data Plane (Service Scalability)	Line Card	Line Card Module with Distributed ASIC	Leaf	Datacenter SW

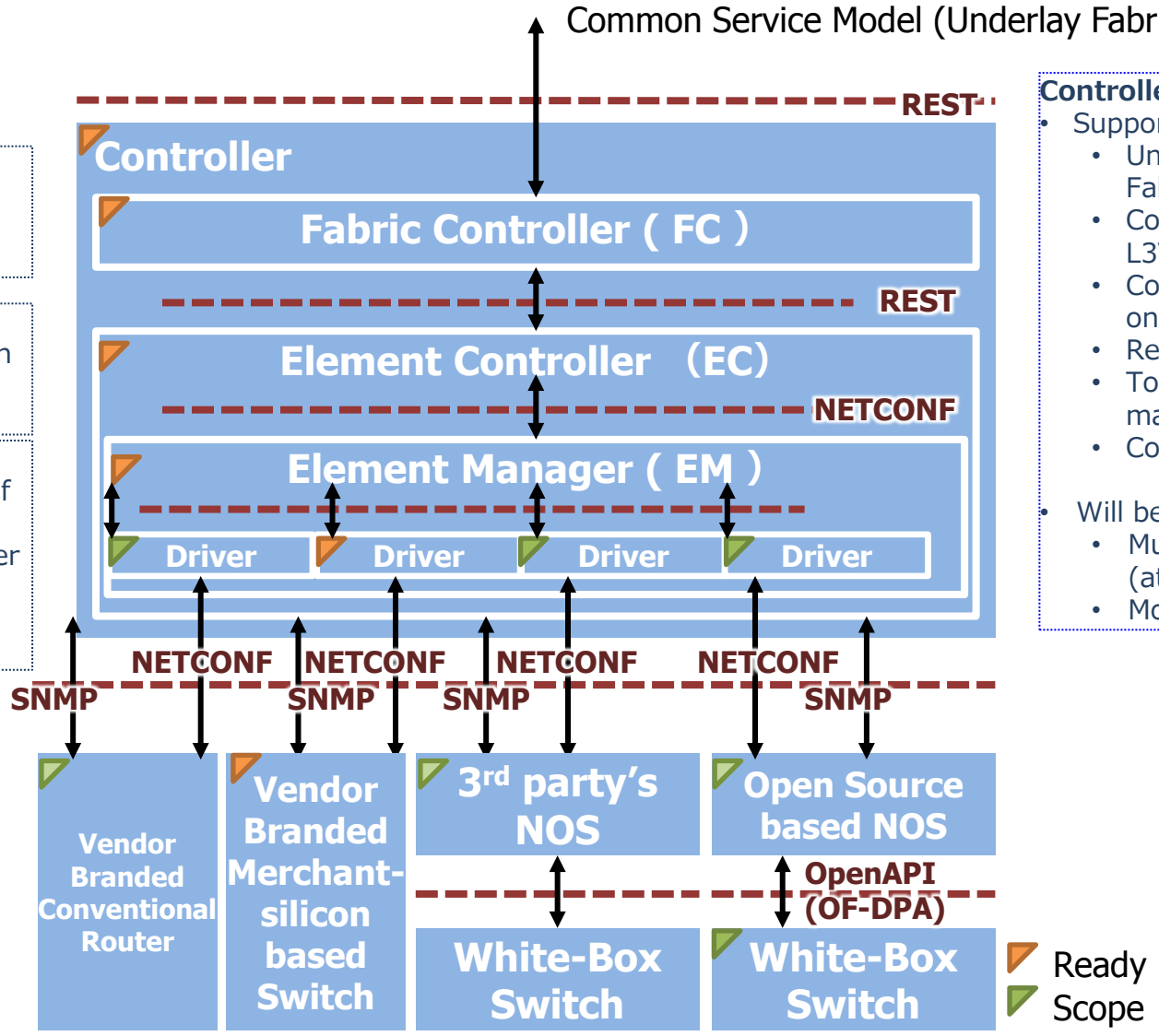
Fabric architecture

- Controller manages nodes' configuration and status.
- OSPF and LDP are used in underlay network configuration. VXLAN (L2VPN) and MP-BGP (L3VPN) are used in overlay service configuration.
- Cluster Scalability depends on mainly Switch hardware.



- Underlay network : OSPF, LDP
- VPN (overlay) route exchange: iBGP
- Supporting LAG between Spine and Leaf, Leaf and CE
- Supporting Redundancy for CE (Dual-home Device)
  - L2 : VXLAN (EVPN multi-home will be deployed in Dec.2017)
  - L3 : VRRP, eBGP, OSPF

Controller architecture



FC
- Managing switch fabric model and network model

EC
- Conversion between logical and physical information

EM
- Absorb difference of vender-specific
- Provide switch driver to support device models for each device type

Controller
- Supported:
  - Underlay control of Data Center Fabric (at a single location)
  - Control of configuration for L3VPN(RFC4364)
  - Control of configuration for VXLAN on HWVTEP
  - Redundancy mechanism
  - Topology and Traffic statistics management and visualization
  - Combination of OpenStack neutron
- Will be supported:
  - Multiple fabrics network models (at many locations)
  - More detailed policy (QoS, Filter)

Controller design

Fabric Controller (FC)

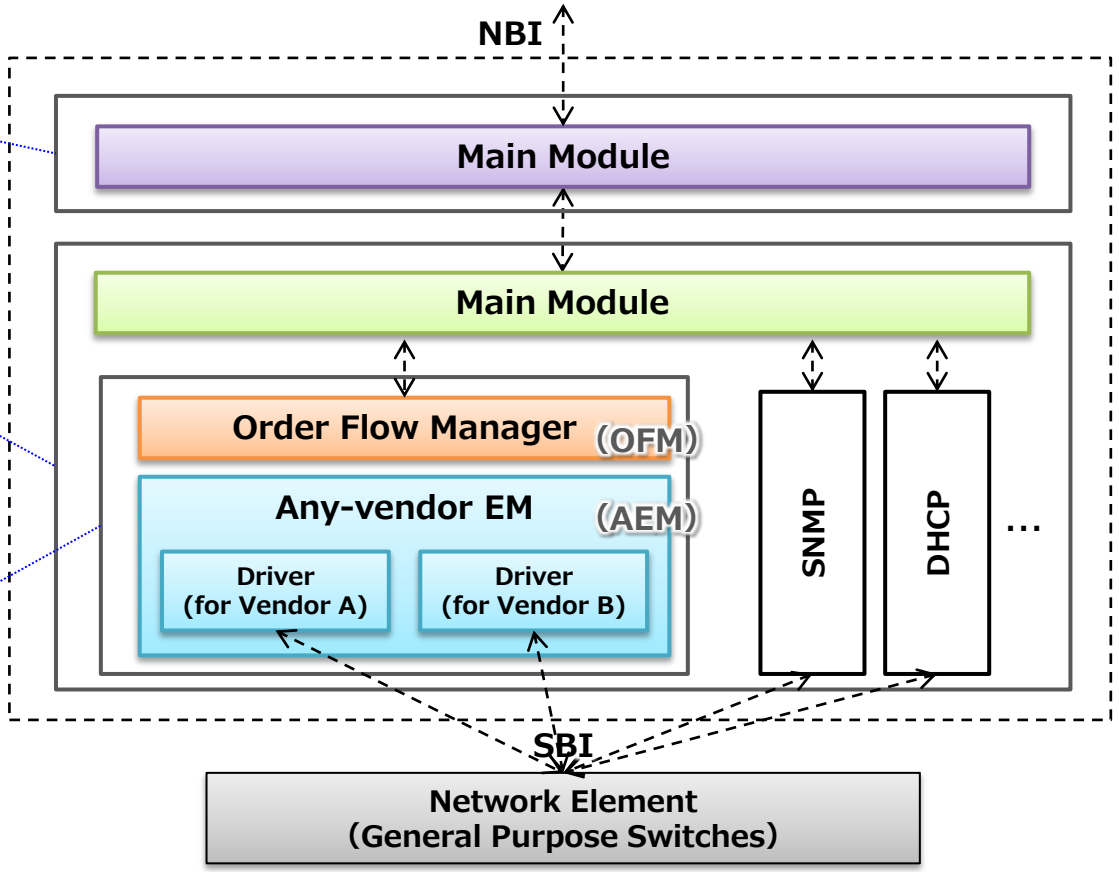
Abstraction (show as a one logical switch)

Element Controller(EC)

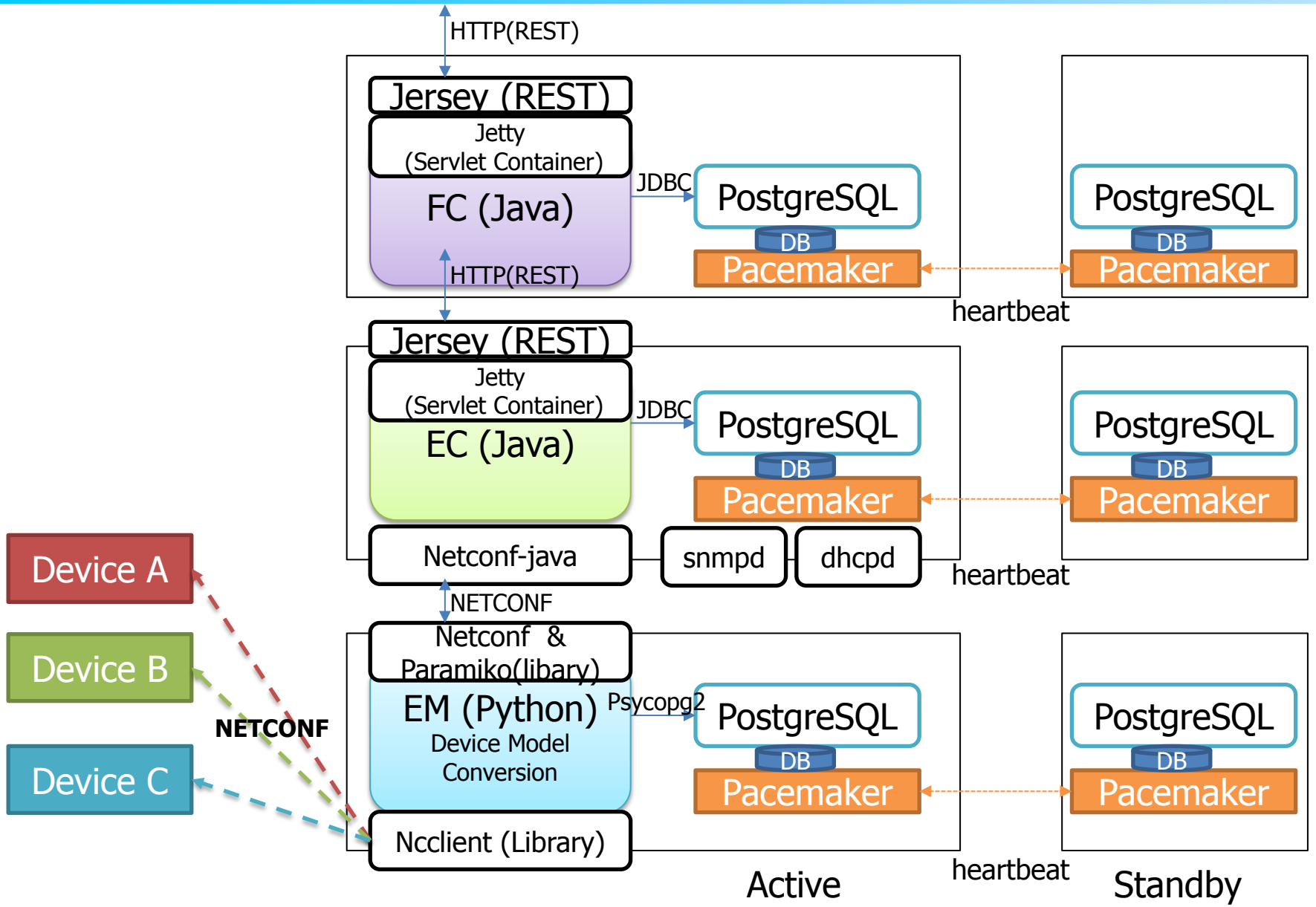
Physical – Logical mapping

Element Manager(EM)

Conceals vendor-specific

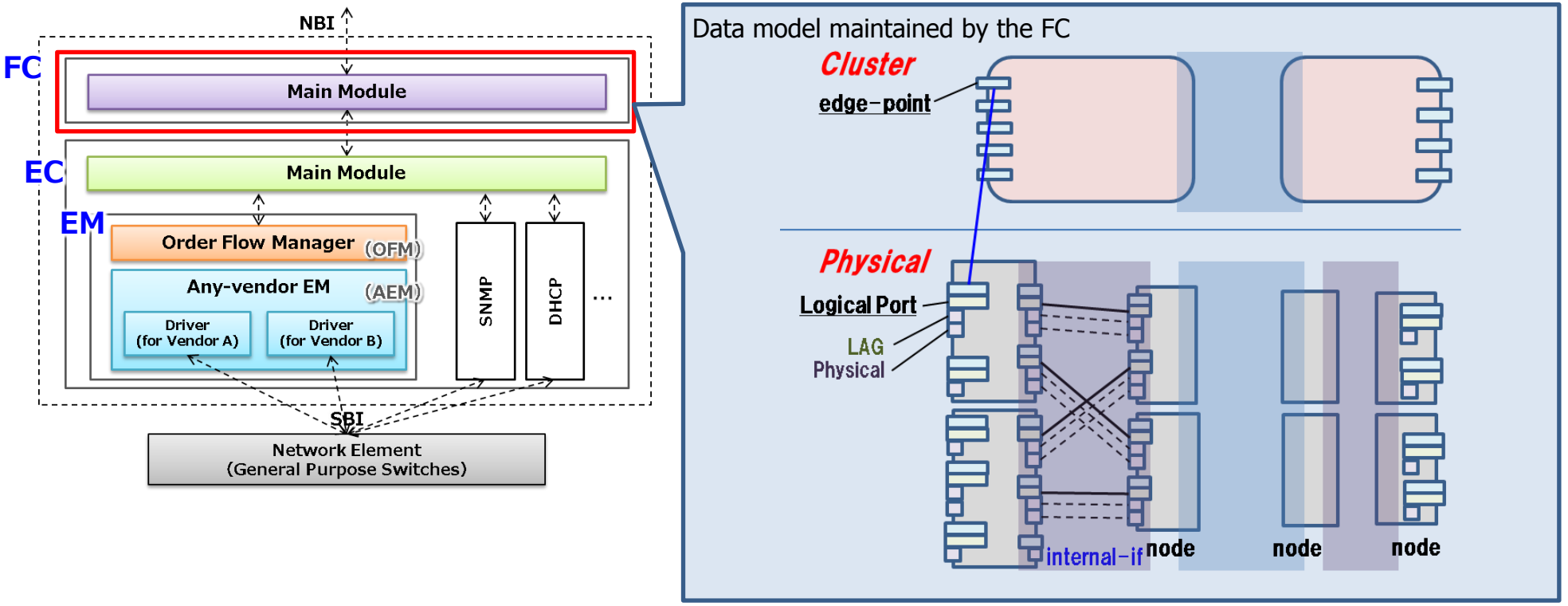


Software component



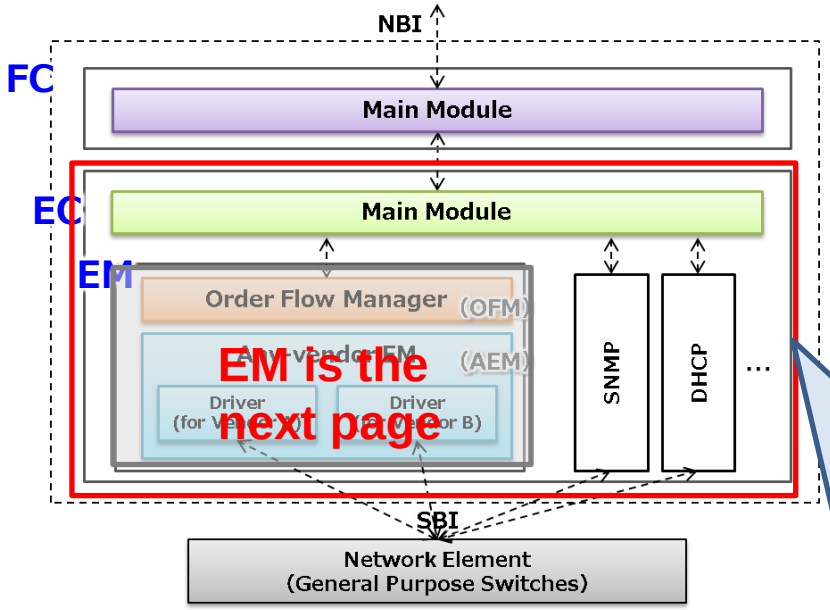
Fabric Controller (FC)

- **FC provides Network abstraction, management and the interface for northbound systems.**
  - **Network Abstraction : Show multiple switches as one logical switch**. “Edge-point” is defined in order to indicate the port (unique ID among a single cluster). This enables to hide the physical information to northbound.
  - Network Management : Maintain the network topology with logical information (edge-points).
    - Interface types and vendor-specific information is hidden by the EC.
    - EC does the mapping of physical port to logical port, FC does the mapping of logical port to edge-point.



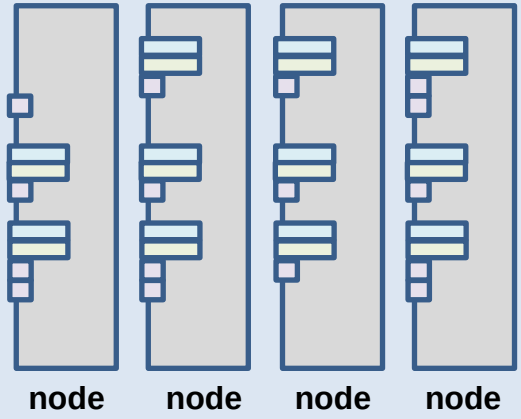
Element Controller (EC)

- **EC provides physical-logical mapping, concealment of vendor-specific information**. It also provides control interface to switch devices.
  - physical-logical mapping : Maintain mapping of physical ports to logical ports.
  - Concealment of vendor-specific information : Consolidate the difference between vendors MIB. This is injected via REST IF.



SBI protocols: SNMP Trap/MIB, Netconf, BGP

Data model maintained in the EC



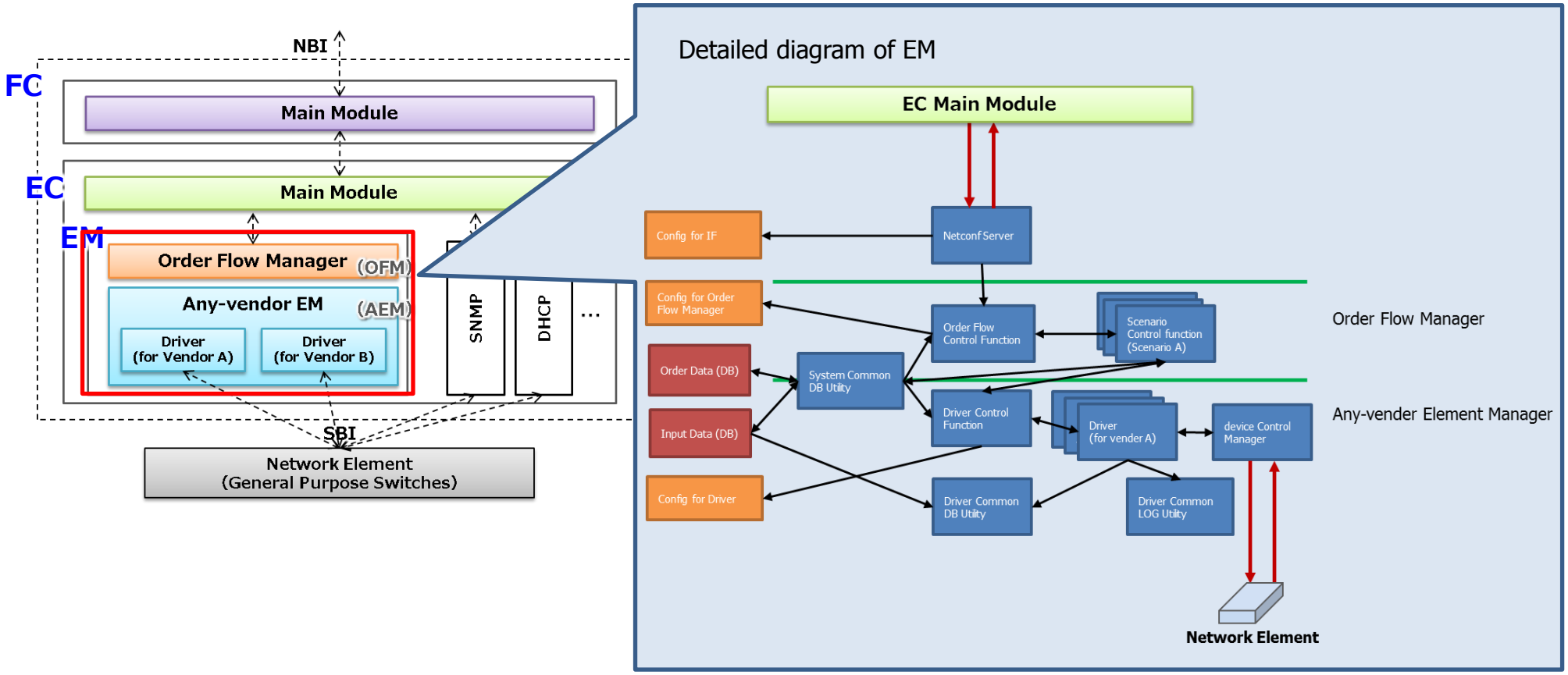
Does not maintain topology info

Physical-logical mapping table

Device	Logical Port	IF name
A	Port1	xe-0/0/1
	Port2	xe-0/0/2
B	port1	TenGigE0/0/0/1
	port2	TenGigE0/0/0/2
	⋮	

Element Manager (EM)

- **Concealment of vendor-specific configurations** and order flow management. It also provides control interface to switch devices.
  - **Concealment of vendor-specific configurations : Enabled by drivers implemented for each vendor products.**
  - Order Flow Management : Manage the configuration to multiple devices with one single transaction. Executes roll-back in case of error.



Supported functions and products

model		QFX5100 -48S	QFX5100 -24Q	QFX5200 -32C	MX240	NCS5001	NCS5011	NCS5501 -SE	OcNOS +AS5812	OcNOS +AS7712
OS version		Junos14.1X53 -D35	Junos14.1X53 -D35	Junos15.1X53 -D30Flex-image	Junos15.1 R5.5	IOS-XR 6.1.2	IOS-XR 6.1.2	IOS-XR 6.1.31/6.3.1	IPBASE-1.3.0V-alpha2	IPBASE-1.3.1
type	Spine	Yes	Yes	Yes	×	×	Yes	×	×	Yes
	L3-Leaf(※1)	Yes	×	Yes	×	Yes	×	Yes	×	×
	L2-Leaf(※2)	Yes	×	×	×	×	×	×	Yes	×
	core-router(※3)	×	×	×	Yes	×	×	×	×	×
L3CP type	direct(v4)	Yes	—	Yes	Yes	×	—	Yes	—	—
	direct(v6)	Yes	—	×	Yes	×	—	Yes	—	—
	VRRP(v4)	Yes	—	Yes	×	×	—	Yes	—	—
	VRRP(v6)	Yes	—	×	×	×	—	Yes	—	—
	BGP(v4)	Yes	—	Yes	Yes	Yes	—	Yes	—	—
	BGP(v6)	Yes	—	×	Yes	×	—	Yes	—	—
	static(v4)	Yes	—	Yes	×	Yes	—	Yes	—	—
	static(v6)	Yes	—	×	×	×	—	Yes	—	—
L2CP type	EVPN (multi-home)	Yes	—	—	—	—	—	—	×	—
	EVPN (single)	Yes	—	—	—	—	—	—	Yes	—

Supported functions and products

model		QFX5100 -48S	QFX5100 -24Q	QFX5200 -32C	MX240	NCS5001	NCS5011	NCS5501 -SE	OcNOS＋ AS5812	OcNOS＋ AS7712
OS version		Junos14.1X53 -D35	Junos14.1X53 -D35	Junos15.1X53 -D30Flex-image	Junos15.1 R5.5	IOS-XR 6.1.2	IOS-XR 6.1.2	IOS-XR 6.1.31/6.3.1	IPBASE-1.3.0V-alpha2	IPBASE-1.3.1
Interface	1G-LX	Yes	×	×	Yes	Yes	×	Yes	Yes	×
	10G-LR (Non Breakout)	Yes	×	×	Yes	Yes	×	Yes	Yes	×
	40G-SR4	Yes	Yes	Yes	×	Yes	Yes	Yes	Yes	Yes
	100G-SR4	×	×	Yes	×	Yes	Yes	Yes	×	×
	10G-SR*4 (Breakout-IF)	×	Yes	Yes	×	×	Yes	×	×	×
Control	Netconf	Yes	Yes	Yes	Yes	Yes	Yes	Yes	×	—
	CLI	×	×	×	×	×	×	×	Yes	—
MIB	L3 physical	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	L3VLAN	×	—	Yes	Yes	×	×	×	—	—
	L2 physical	Yes	—	—	—	—	—	—	Yes	Yes
	L2VLAN	×	—	—	—	—	—	—	×	×
how to configure	ZTP	Yes	Yes	Yes	×	Yes	Yes	Yes	×	×
	manual	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Update Jan.2018

Controller functions

Category	Function	Description
Model information management	Register model information	**Register the switch model to be used in controller** You can put the model name, OS version, ID of each physical interface, and capability ..etc.
Underlay management	Leaf/Spine control	**Add and Delete Leaf-node or Spine-node** You can automatically add the nodes into the network by registering the link connection information in the controller, and turning on the node.
	Interface control	**Physical<->Link-Aggregation or Breakout-IF** You can convert physical-IF to LAG-IF or Breakout-IF, or check the information on each interface
	edge-point control	**Specify any interfaces as edge-point** The end point to which end-user connects to slice, called connection-point; CP, can be created on the edge-point.
Overlay management	Slice control	**Create/delete L2/L3 slice, also called VPN**
	CP control	**Create/delete end point to which end-user connects to slice**

Update Jan.2018

Controller functions

Category	Function	Description
Network Operation	Fault detection	**Detection of link or switch failure** You can detect failures at physical, link-aggregation, or breakout-Ifs.
	Traffic measurement	**Measurement on traffic at any interface** You can specify the traffic on each interface, and if you set the traffic threshold in the interface, it is aslo possible to notify
Controller Operation	Controller state	**The state of the controller** You can acquire CPU and memory utilization.
	Controller log	**Get the controller log** You can get the controller processing log.

Update Jan.2018

Controller API

For details, refer to multi-service-fabric/fabric-controller/API

Class	Group	Interface description	Method
Common	Processing request	Getting list of operational state	GET
		Getting information of detailed operation state	GET
	Controller status confirmation	Getting controller state	GET
	Controller log	Getting controller log	GET
Underlay management	Equipment-type information management	Registering equipment information	POST
		Getting equipment list in switch cluster	GET
		Getting equipment information	GET
		Deleting equipment information	DELETE
	Node information	Getting list of nodes	GET
	Leaf management	Adding Leaf-node	POST
		Getting list of Leaf-nodes	GET
		Getting information of Leaf-node	GET
		Deleting Leaf-node	DELETE
		Updating Leaf-node	PUT
		Adding Spine-node	POST
		Getting list of Spine-nodes	GET
		Getting information of Spine-node	GET
		Deleting Spine-node	DELETE

Update Jan.2018

Controller API

For details, refer to multi-service-fabric/fabric-controller/API

Class	Group	Interface description	Method
Underlay management	RR (BGP Route Reflector) management	Getting list of RR-node	GET
		Getting information of RR-node	GET
	Interface information	Getting list of interfaces	GET
	Interface management (Physical interface)	Getting list of physical interfaces	GET
		Getting information of physical interface	GET
		Updating physical interface	PUT
	Interface management (Breakout interface)	Creating or deleting breakout interface	PATCH
		Getting list of breakout interfaces	GET
		Getting information of breakout interface	GET
	Interface management (Internal-link interface)	Getting list of internal-link interfaces	GET
		Getting information of internal-link interface	GET
	Interface management (Link aggregation interface)	Creating Link-aggregation interface	POST
		Getting list of Link-aggregation interfaces	GET
		Getting information of Link-aggregation interface	GET
		Deleting information of Link-aggregation interface	DELETE
	Edge point management	Creating edge-point	POST
		Getting list of edge-points	GET
		Getting information of edge-point	GET
		Deleting edge-point	DELETE

Update Jan.2018

Controller API

For details, refer to multi-service-fabric/fabric-controller/API

Class	Group	Interface description	Method
Overlay management	Slice	Creating Slice	POST
		Deleting Slice	DELETE
		Getting information of Slice	GET
		Getting list of Slices	GET
	CP	Creating or deleting CP	PATCH
		Creating CP	POST
		Deleting CP	DELETE
		Getting information of CP	GET
		Getting lists of CP	GET
		Creating or deleting static route	PATCH
Traffic information	Traffic information	Getting list of IF traffic	GET
		Getting IF traffic	GET
		Getting list of CP traffic	GET
		Getting CP traffic	GET
Fault detection	Failure detection	Getting list of failures	GET

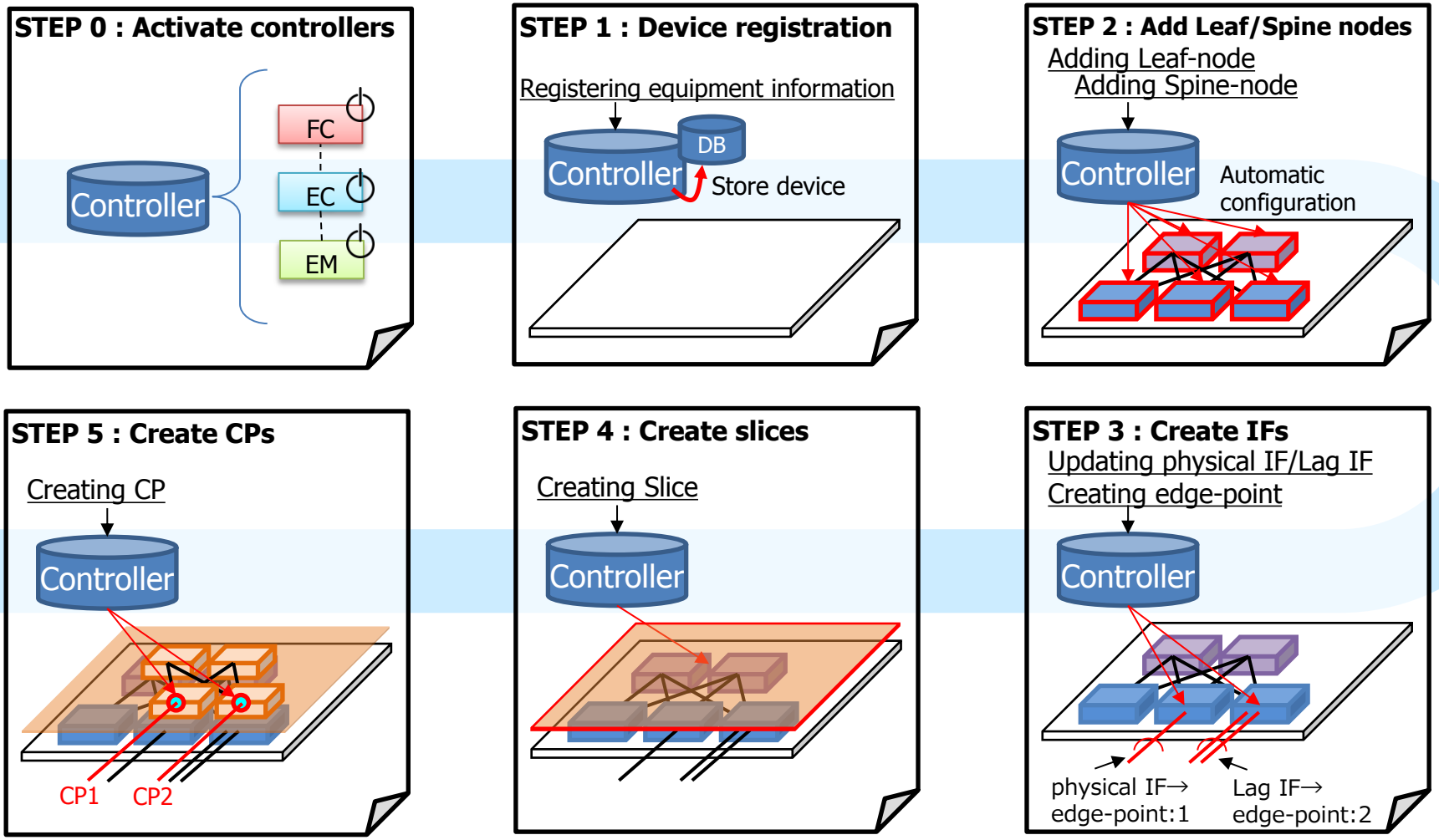
Update Jan.2018

Controller API(Notification)

For details, refer to multi-service-fabric/fabric-controller/API

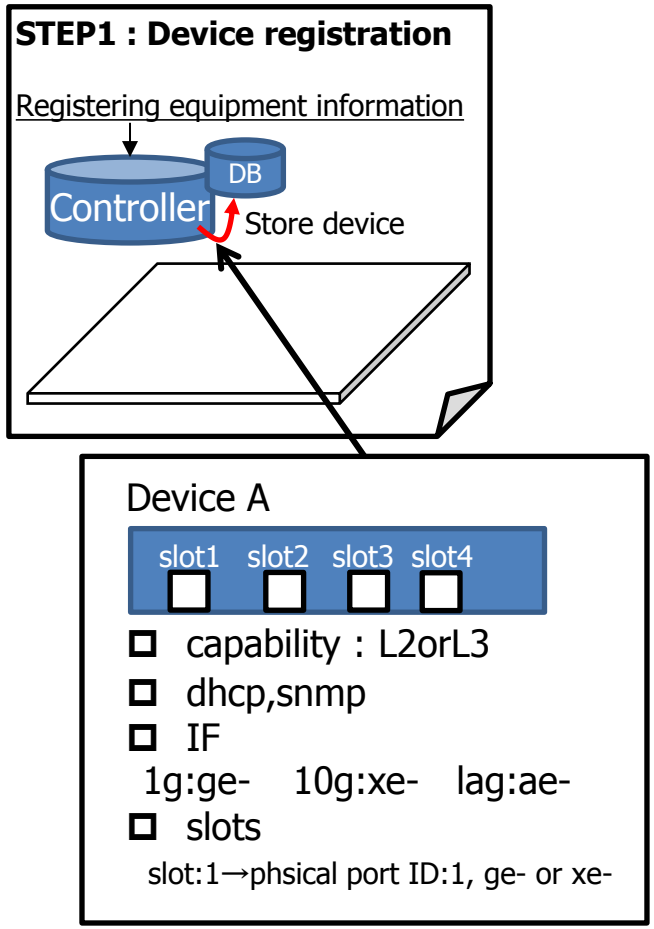
Group	Interface description	Method
Common	Processing result	PUT
	controller status	PUT
Traffic information	Traffic information	PUT
Failure detection	Failure information	PUT

Basic operation (activate controllers ~ create CPs)



Basic operation (Step1 : Device registration)

- **Interface name : Registering equipment information**
- **URI : /v1/equipment-types**
- **Register device information to be used in the cluster.**

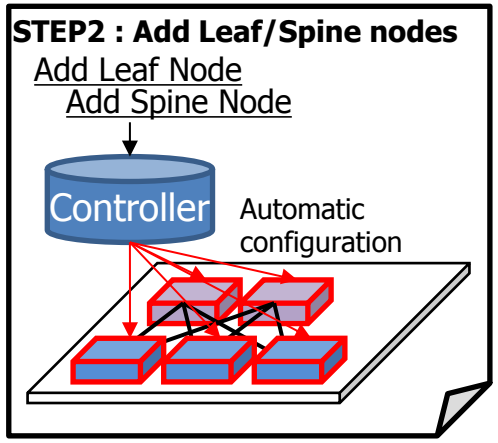


Typical parameters

body	overview	remarks
platform	platform	
firmware	firmware version	
capability	l2/l3 VPN compatibility	
dhcp/snmp	DHCP, SNMP	
if_definitions	IF information (port, speed, prefix, ..)	
slots	slot information	Mapping of physical port ID and slot.

Basic operation (Step2 : Add Leaf/Spine nodes)

- **Interface name : Adding Leaf-node, Adding Spine-node**
- **URI : /v1/clusters/{cluster_id}/nodes/leafs, /v1/clusters/{cluster_id}/nodes/spines**



- When Leaf is added, the controller also sets the appropriate configuration for the connected Spine.
- If you add the device that has been already configured (you don't use ZTP), you set the "provisioning" body is "false", and set the same conditions for other parameters.
- Please refer to Page.28 for your understanding of automatic configuration using ZTP.
- You need to add Leaf/Spine node one by one.

Typical parameters (Adding Leaf-node)

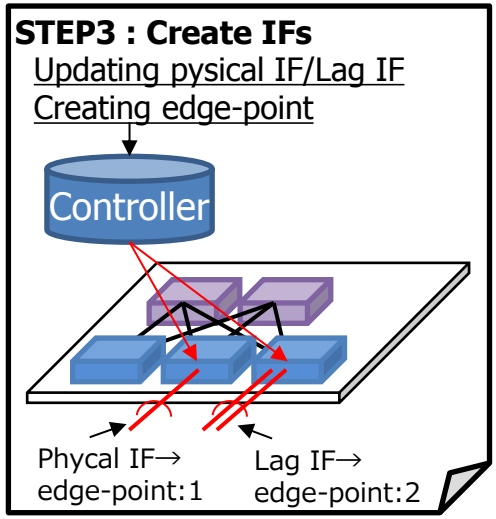
body	overview	remarks
node_id	Serial number for device	Created by FC
equipment_type_id	model ID	
provisioning	Device setting necessity flag	True: Built-in device not set False: Embed setting device
VPN_type	l2/l3 VPN type	One of "l2" and "l3"
plane	Belonging side	Set "1".
internal_links	Internal link information	

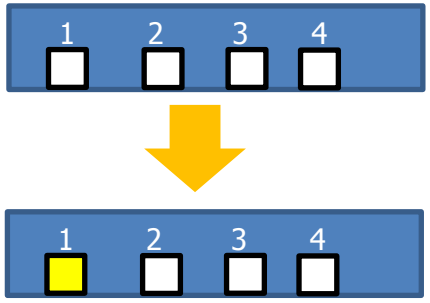
Typical parameters (Adding Spine-node)

body	overview	remarks
node_id	Serial number for device	Created by FC
equipment_type_id	model ID	
provisioning	Device setting necessity flag	True: Built-in device not set False: Embed setting device
internal_links	Internal link information	

Basic operation (Step3 : Create IFs) - physical interface -

- Interface name : Updating information of physical interface
- URI : /v1/clusters/{cluster_id}/nodes/{fabric_type}/{node_id}/interfaces/physical-ifs/{if_id}
- Determine the speed of the physical interface. (not confirmed at device registration)
- The selectable speed is the value defined at device registration.





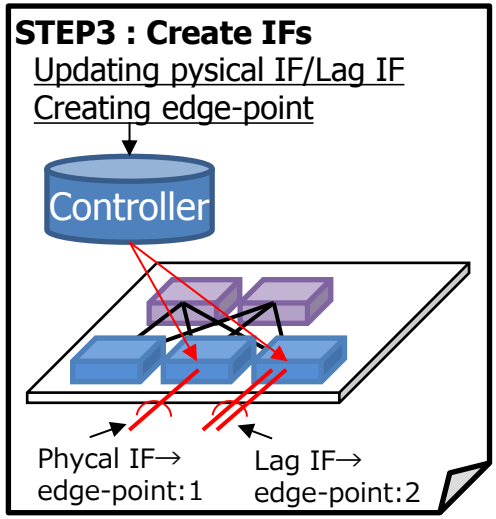
set speed

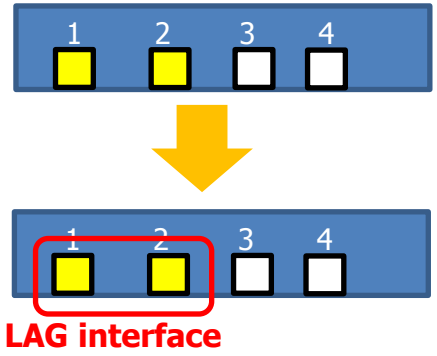
Parameters

body	overview	remarks
cluster_id	Switch cluster ID	Identify the target physical IF
fabric_type	Device type	
node_id	Device ID	
if_id	Physical IF ID	
action	Control type	
speed	IF speed	

Basic operation (Step3 : Create IFs) - LAG interface -

- **Interface name : Creating Link-aggregation interface**
- **URI : /v1/clusters/{cluster_id}/nodes/{fabric_type}/{node_id}/interfaces/lag-ifs**
- **Create the LAG-IF from several phsical interfaces set speed.**





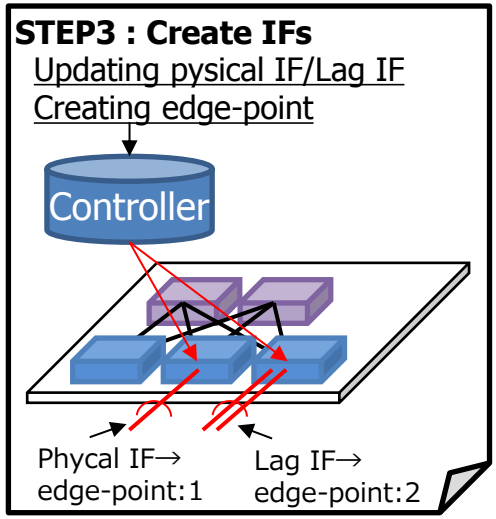
Parameters

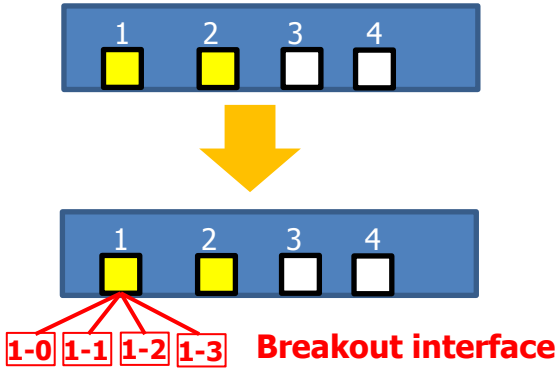
body	overview	remarks
cluster_id	Switch cluster ID	Identify the target physical IF
fabric_type	Device type	
node_id	Device ID	
physical_if_ids	List of Physical IF ID	

Update Jan.2018

Basic operation (Step3 : Create IFs) - Breakout interface -

- **Interface name : Creating or deleting breakout interface**
- **URI : /v1/clusters/{cluster_id}/nodes/{fabric_type}/{node_id}/interfaces/breakout-ifs**
- **Create the Breakout-IF**



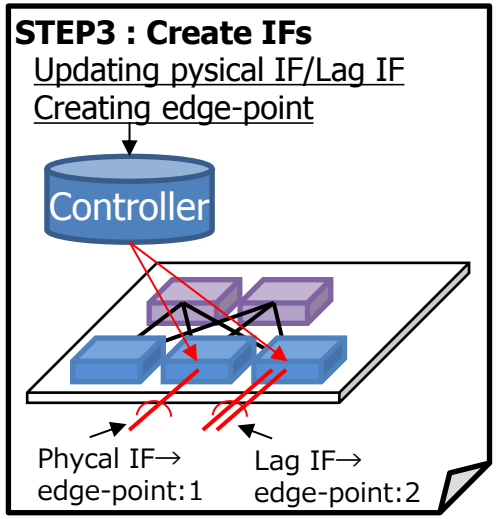


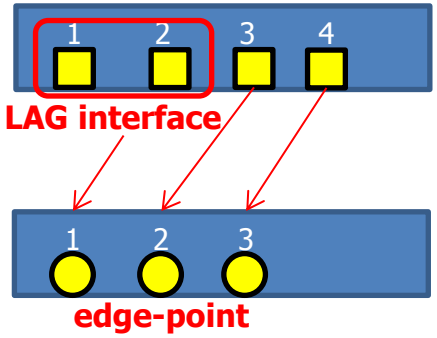
Parameters

body	overview	remarks
op	control type	create: "add" delete: "remove"
path	Breakout IF ID	"/" + "breakout IF ID"
physical_if_id	target physical IF ID	
division_number	number of divisions of physical IF	
breakout_if_speed	Speed of each breakout IF	

Basic operation (Step3 : Create IFs) - edge-point -

- **Interface name : Creating edge-point**
- **URI : /v1/clusters/{cluster_id}/points/edge-points**
- **Create edge-point so that the upper systems do not identify the interface type. The CP is registered on the edge-point.**
- **You can not register another edge-point in the IF where the edge-point is already registered.**



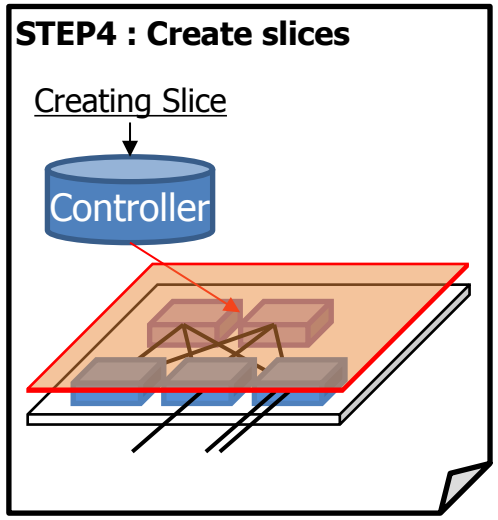


Parameters

body	overview	remarks
cluster_id	Switch cluster ID	
leaf_node_id	Leaf device ID	
laag_if_id	LAG IF ID	Specify either LAGIF ID or Physical IF ID
physical_if_ids	Physical IF ID	

Basic operation (Step4 : Create slices)

- **Interface name : Creating Slice**
- **URI : /v1/slices/{slice_type}**

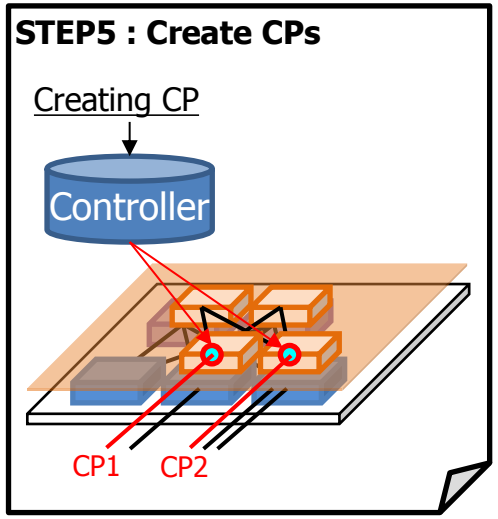


Parameters

body	overview	remarks
slice_type	Slice type	"l2vpn" : L2 slice "l3vpn" : L3 slice
slice_id	Slice ID	If it is not specified, FC creates ID.

Basic operation (Step5 : Create CPs)

- **Interface name : Creating CP**
- **URI : /v1/slices/{slice_type}/{slice_id}/cps**
- **CP is set above the edge-point.**
- **L3CP needs to specify the protocol to be used. (BGP, OSPF, static, VRRP)**



Parameters (slice type -> L2 slice)

body	overview	remarks
slice_type	Slice ID	"l2vpn"
slice_id	Slice ID	
cluster_id	Switch cluster ID	
edge_point_id	Edge-point ID to be created for CP	
vlan_id	VLAN ID	VLAN ID of CP
cp_id	Create CP ID	
port_mode	Port mode of VLAN	"access" or "trunk"

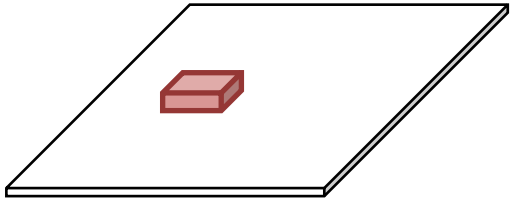
Typical parameters (slice type -> L3 slice)

body	overview	remarks
slice_type	Slice ID	"l2vpn"
slice_id	Slice ID	
ipv4_addr	Housing equipment IF address	
bgp	Information for BGP	specified when setting BGP
static_routes	Static Route information list	specified when setting static
vrrp	information for VRRP	specified when setting VRRP

Basic operation (internal link setting)

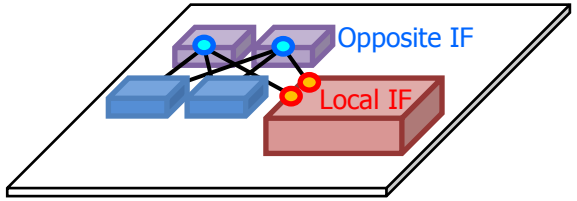
- **When Leaf/Spine node is added, internal link setting is done automatically.**
- **In order to set the internal link, you describe the following parameters on the body by adding Leaf/Spine-node, but when adding the first device, set it to null.**
  - **Adding Leaf node**
    - ✓**Information of Local IF (Leaf) and opposite IF (Spine)**
  - **Adding Spine node**
    - ✓**Information of opposite IF (Leaf) and Local IF (Leaf)**

<First unit>



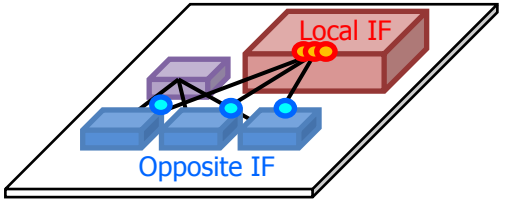
internal_links: null

<Adding Leaf node>



internal_links
->physical_links information list
-> Physical IF ID (local)
-> Port speed type (local)
-> Opposite Spine device ID
-> Opposite Spine Physical IF ID

<Adding Spine node>

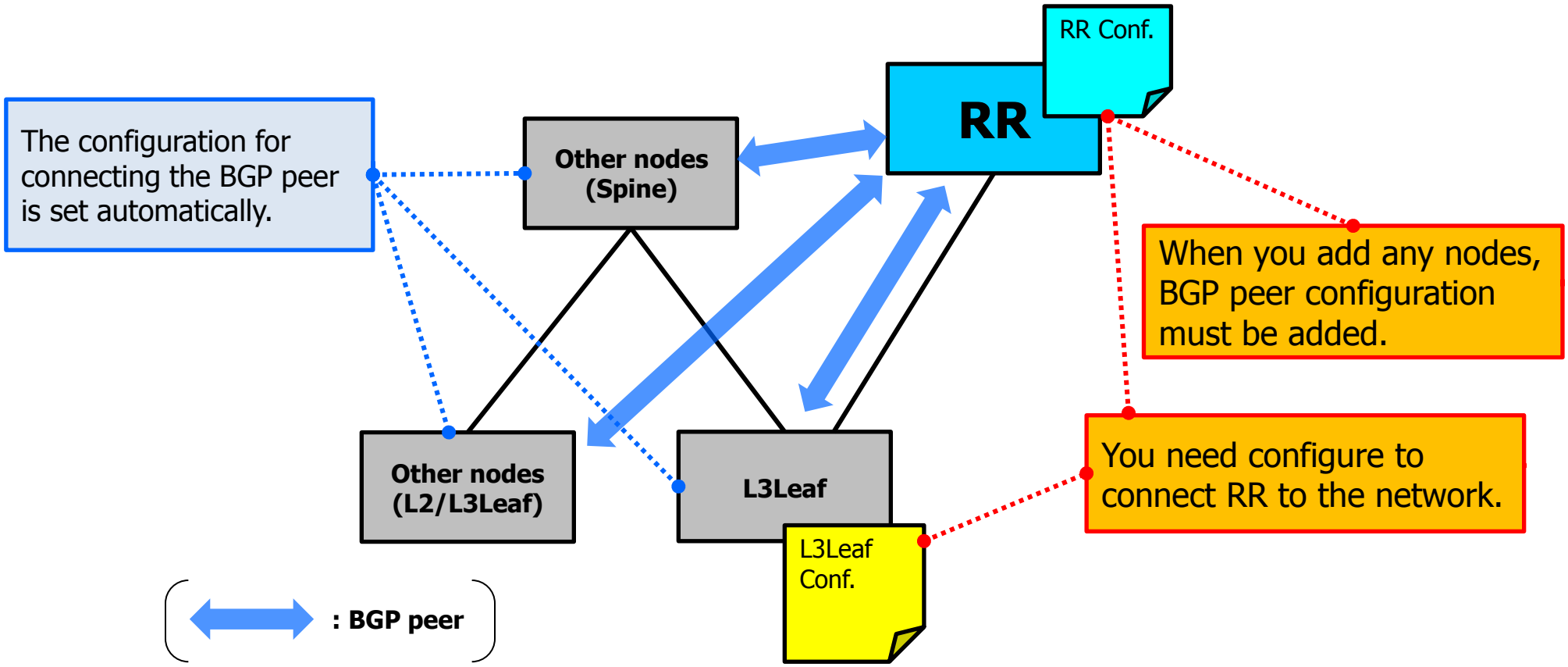


internal_links
- physical_links information list
-> Physical IF ID (local)
-> Port speed type (local)
-> Opposite Spine device ID
-> Opposite Spine Physical IF ID

: added node : Leaf : Spine

Basic operation (RR (BGP Route Reflector) setting)

- **You need to set the RR configuration and network setting yourself, because the current controller does not implement the function of automating the setting of RR.**
- **The node configuration for peering the neighbor with RR is set automatically when Leaf/Spine is added, by registering the ID and loopback address of RR in the initial configuration of FC. But you need to configure the added node as a neighbor in RR conf. when you add any nodes.**
- **When both L2Leaf and L3Leaf exist, RR must be connected to L3Leaf.**



Basic operation (sample configuration of RR)

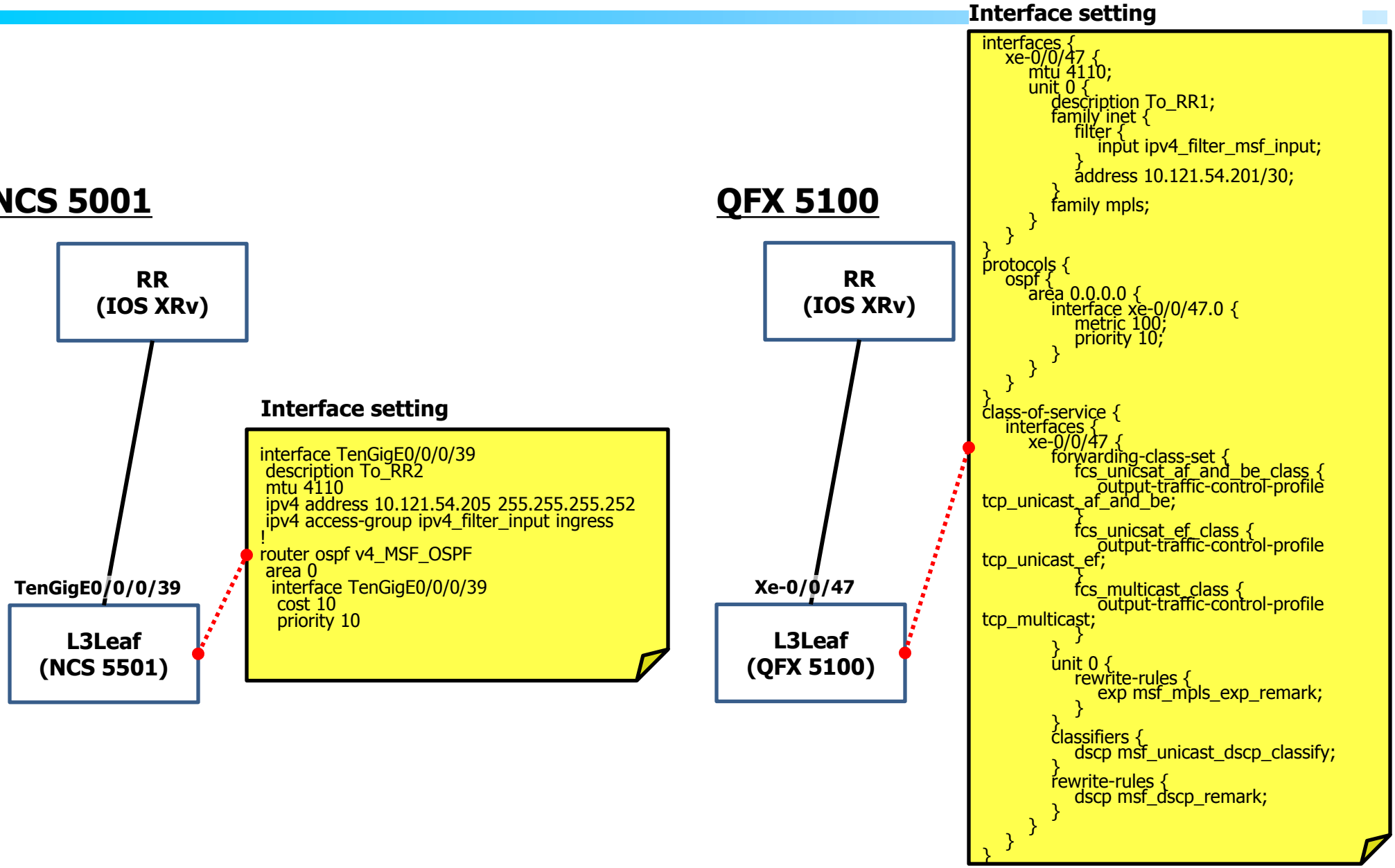
```
hostname sosetsu-RR1
clock timezone JST 9
logging trap alerts
logging buffered 12500000
logging buffered debugging
logging facility local5
logging source-interface Loopback0
service timestamps log datetime msec
service timestamps debug datetime msec
telnet vrf default ipv4 server max-servers 100
domain lookup disable
ntp
 server 192.168.134.14
 source MgmtEth0/RSP0/CPU0/0
 update-calendar
!
interface Loopback0
 ipv4 address 10.0.100.1 255.255.255.255
!
interface MgmtEth0/0/CPU0/0
 ipv4 address 192.168.2.36 255.255.0.0
!
interface GigabitEthernet0/0/0/0
 description To_Leaf4
 mtu 4110
 ipv4 address 10.121.54.202 255.255.255.252
!
route-policy PASS_ALL
  pass
end-policy
!
router ospf v4_MSF_OSPF
 router-id 10.0.100.1
 mpls ldp auto-config
 dead-interval 40
 hello-interval 10
 timers throttle spf 200 200 2000
 area 0
  interface Loopback0
   cost 10
   passive enable
  !
  interface GigabitEthernet0/0/0/0
   cost 100
   priority 10
  !
 !
!
router bgp 64050
 timers bgp 30 90
 bgp router-id 10.0.100.1
 address-family vpnv4 unicast
 !
```

```
neighbor 10.0.1.1
 remote-as 64050
 update-source Loopback0
 address-family vpnv4 unicast
  route-policy PASS_ALL in
  route-reflector-client
  route-policy PASS_ALL out
 !
!
neighbor 10.0.1.2
 remote-as 64050
 update-source Loopback0
 address-family vpnv4 unicast
  route-policy PASS_ALL in
  route-reflector-client
  route-policy PASS_ALL out
 !
!
neighbor 10.0.1.3
 remote-as 64050
 update-source Loopback0
 address-family vpnv4 unicast
  route-policy PASS_ALL in
  route-reflector-client
  route-policy PASS_ALL out
 !
!
neighbor 10.0.1.4
 remote-as 64050
 update-source Loopback0
 address-family vpnv4 unicast
  route-policy PASS_ALL in
  route-reflector-client
  route-policy PASS_ALL out
 !
!
neighbor 10.0.1.5
 remote-as 64050
 update-source Loopback0
 address-family vpnv4 unicast
  route-policy PASS_ALL in
  route-reflector-client
  route-policy PASS_ALL out
 !
!
neighbor 10.0.1.6
 remote-as 64050
 update-source Loopback0
 address-family vpnv4 unicast
  route-policy PASS_ALL in
  route-reflector-client
  route-policy PASS_ALL out
 !
!
!
```

```
mpls ldp
 router-id 10.0.100.1
 interface GigabitEthernet0/0/0/0
  discovery hello holdtime 15
  discovery hello interval 5
 !
!
ssh server vrf default
end
```

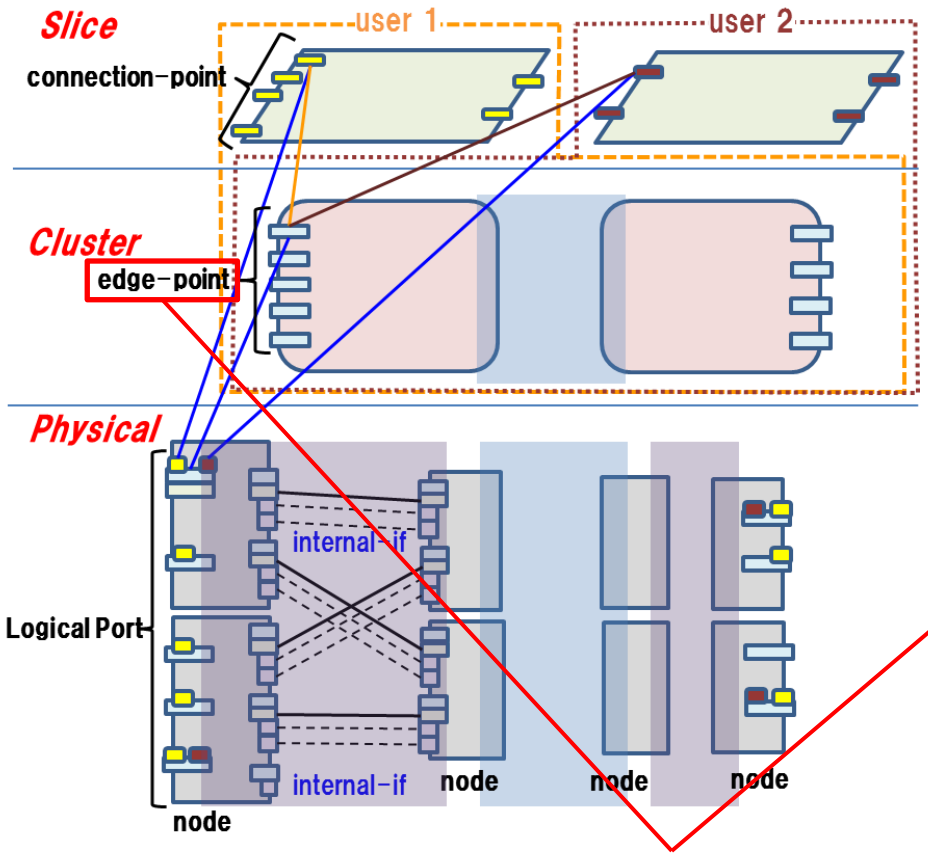
Configuration of the BGP neighbor
When you add any nodes, you need to add it.

Basic operation (sample configuration to connect RR)

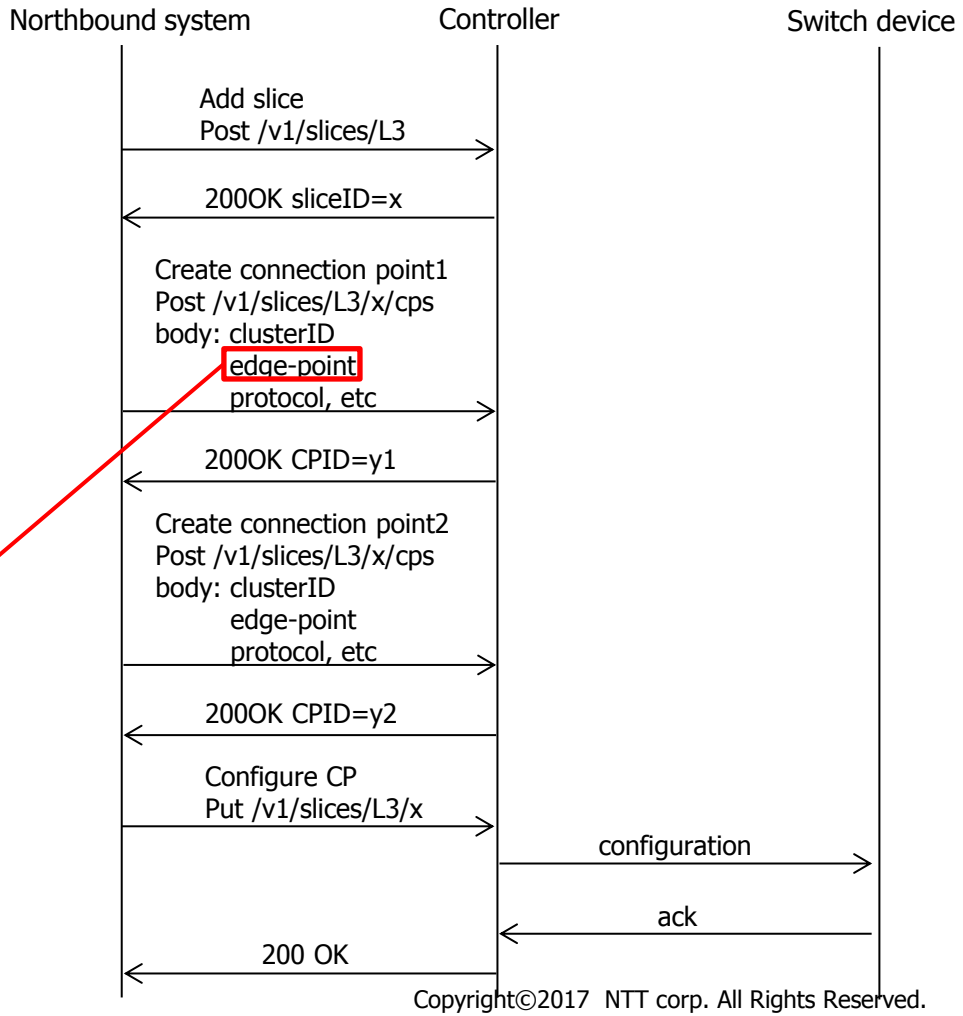


Network slicing (add/remove Layer2/3 VPN)

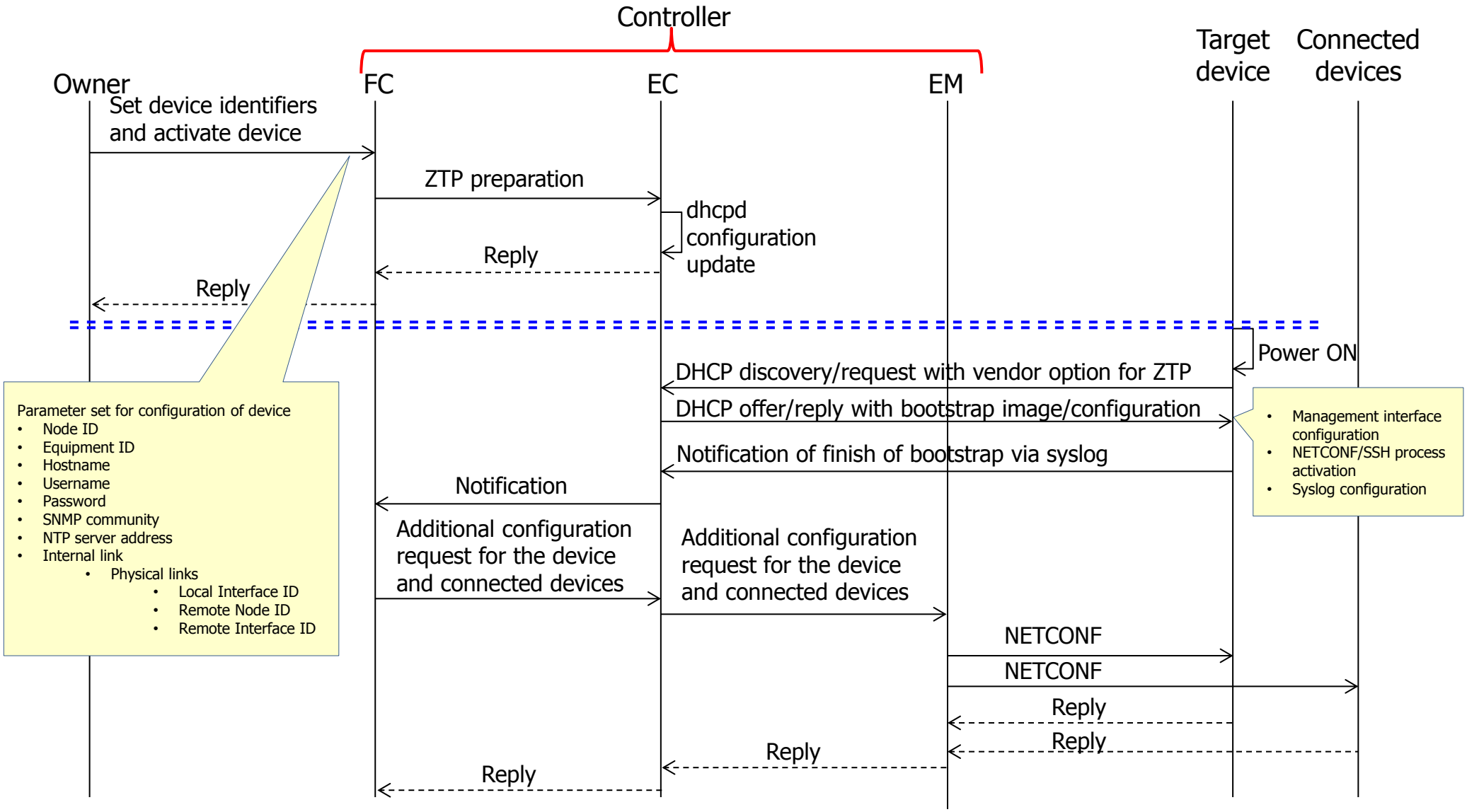
- ◆ Creates network slices (VPN) by selecting the edge-points.
- ◆ Does not need to be aware of the physical structure, nor the configurations of the devices.



Select the edge-points, and the slice to let the points be attached to.

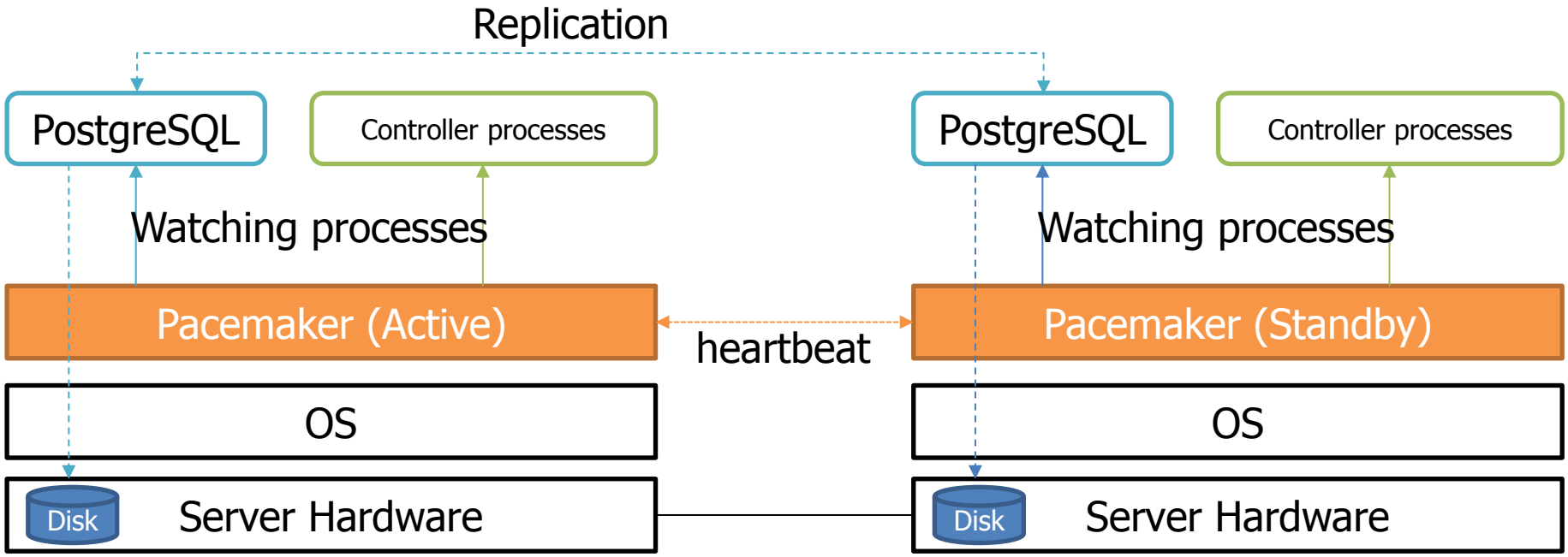


Automatically configuration using ZTP and NETCONF



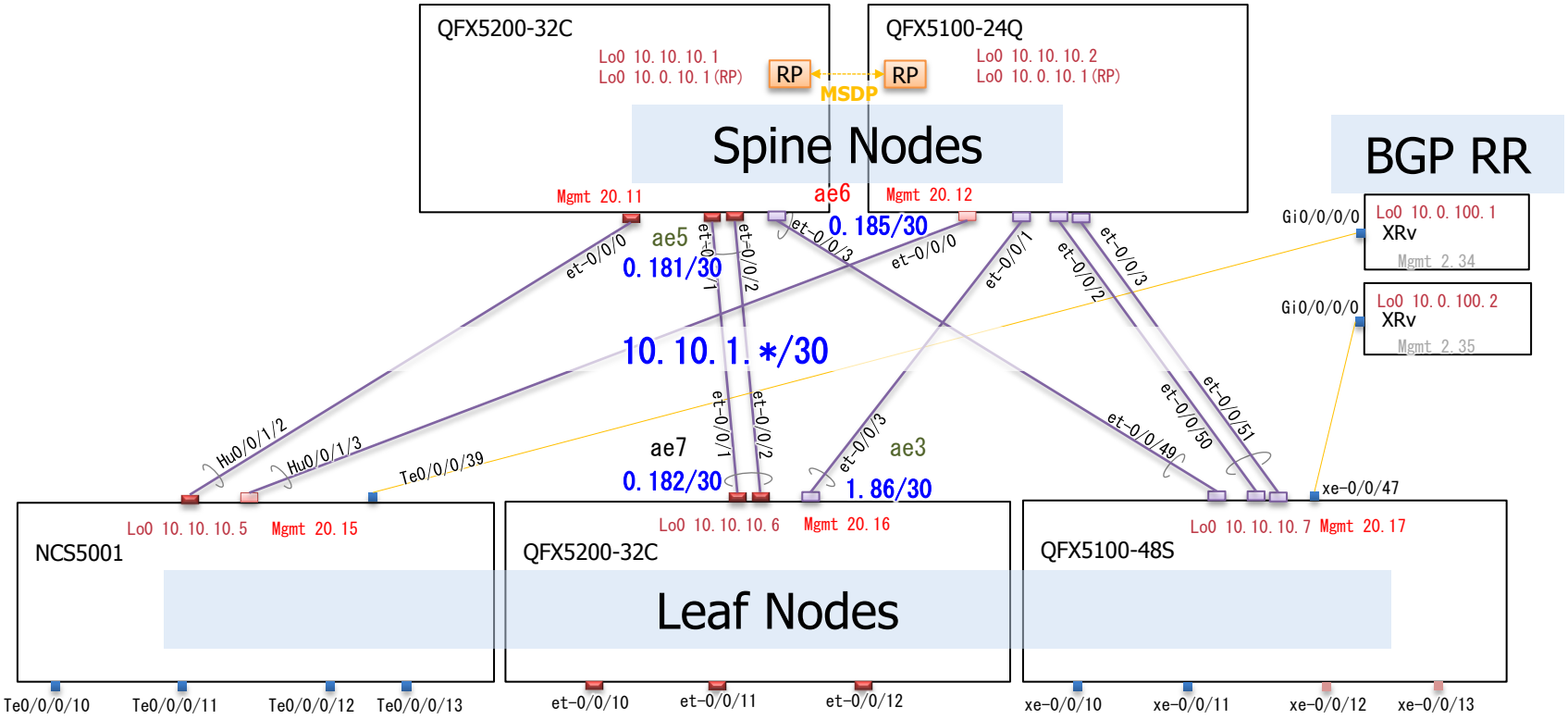
Redundancy/Failover

- We choose proven redundancy mechanism for controller.
  - Pacemaker (http://clusterlabs.org/) for heartbeat
  - PostgreSQL clustering replication
- Pacemaker provides
  - Failover mechanism with heartbeat
  - Watching processes and control process status
  - Providing Virtual IP mechanism.



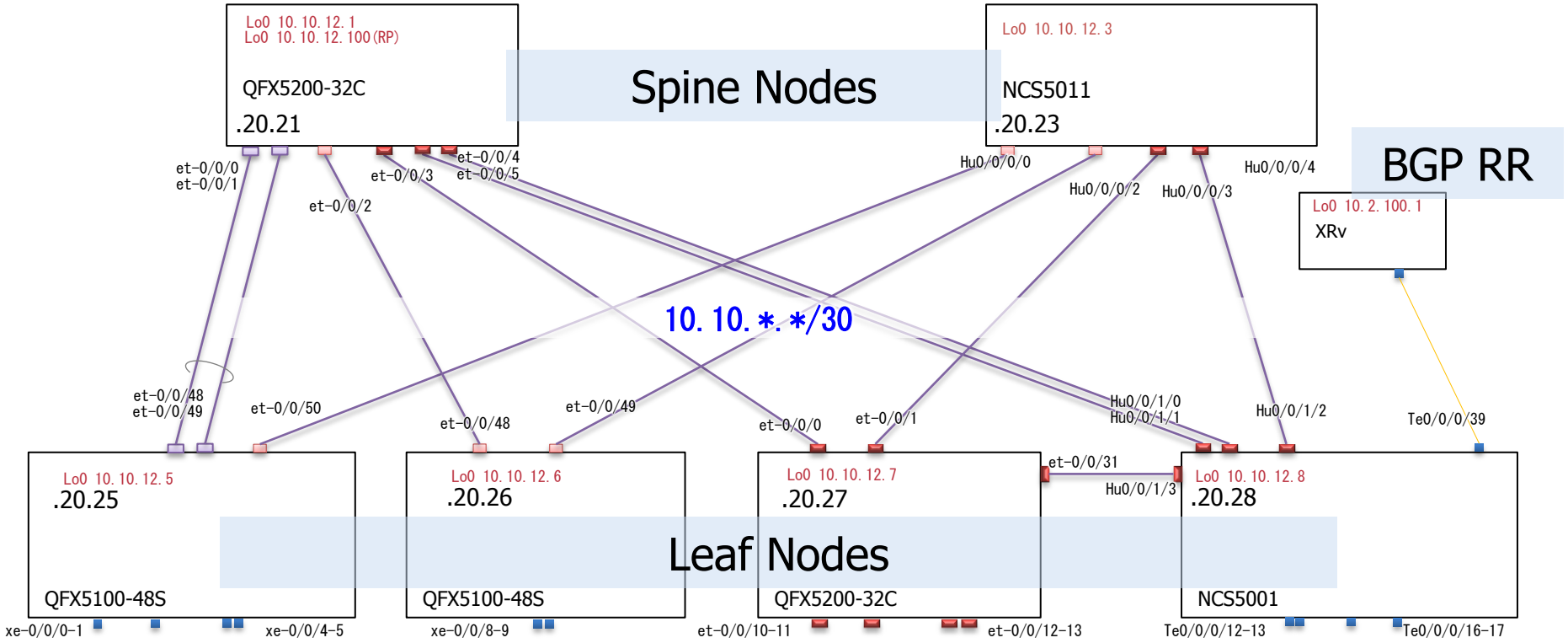
Test environment

The following environment has already been tested.



Test environment

The following environment has already been tested.



Notice

- All company names and product names mentioned in this document are registered trademarks or trademarks of their respective companies.

- This document is not sponsored by, endorsed by or affiliated with Cisco Systems, Inc. Cisco, the Cisco logo, Cisco Systems and Cisco IOS are registered trademarks or trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

