SAI VOQ System

Israel Meilik,
June 2020

SAI VOQ System - Agenda

- Assumptions
- Add route exercise in a VOQ System
- VOQ System pipeline

Proposal assumptions

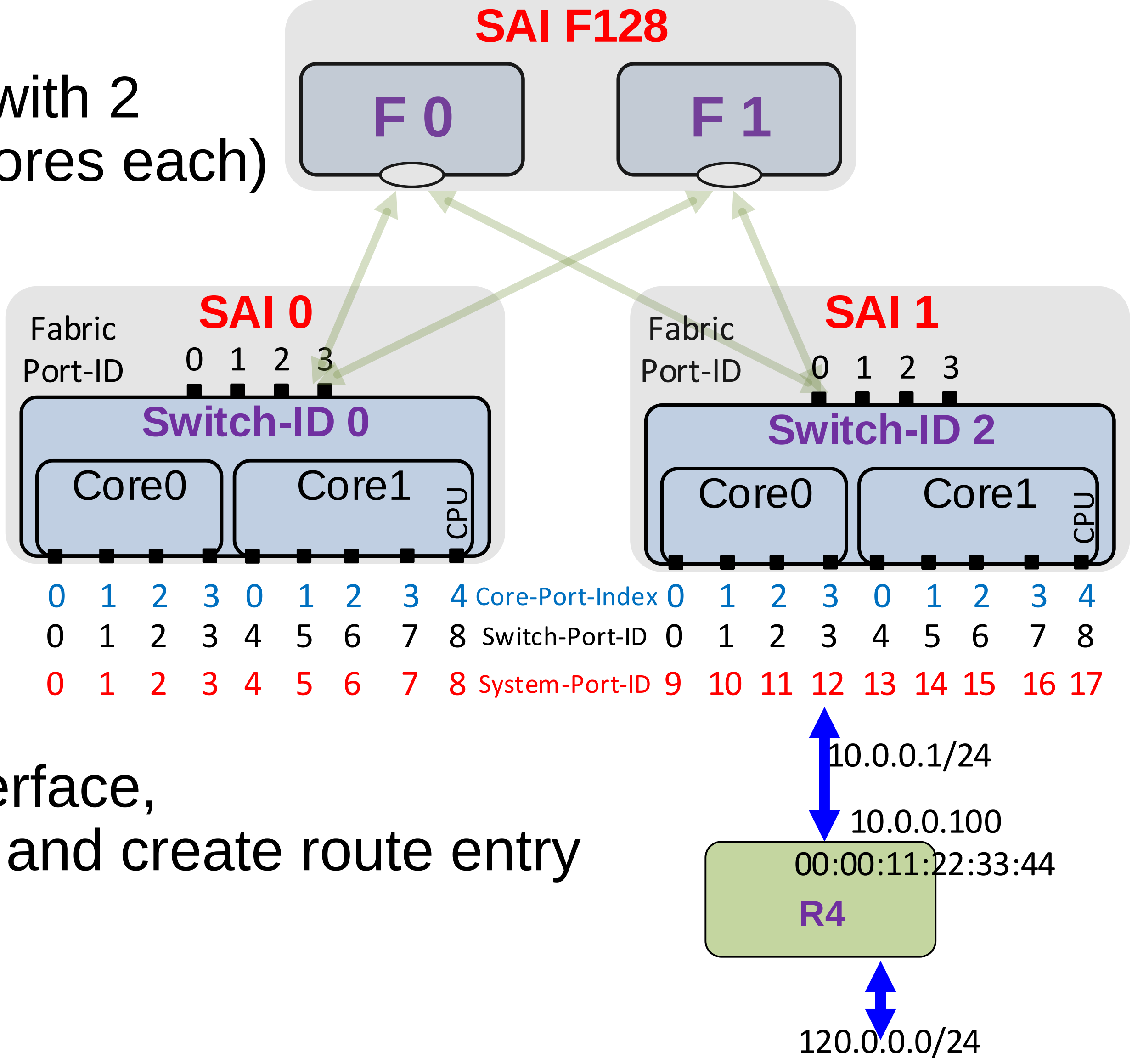
Assumptions

- LAG group has members only on a single VOQ Switch
  - No MLAG
- No IP Multicast
- All Routing Interfaces (RIF) are of type Port, we do not address (Port, VLAN) RIFs
- SAI has an association between System-Port and its local Port (part of SAI VOQ proposal)
- Static System, no ISSU handling/considerations at this phase

Add Route exercise

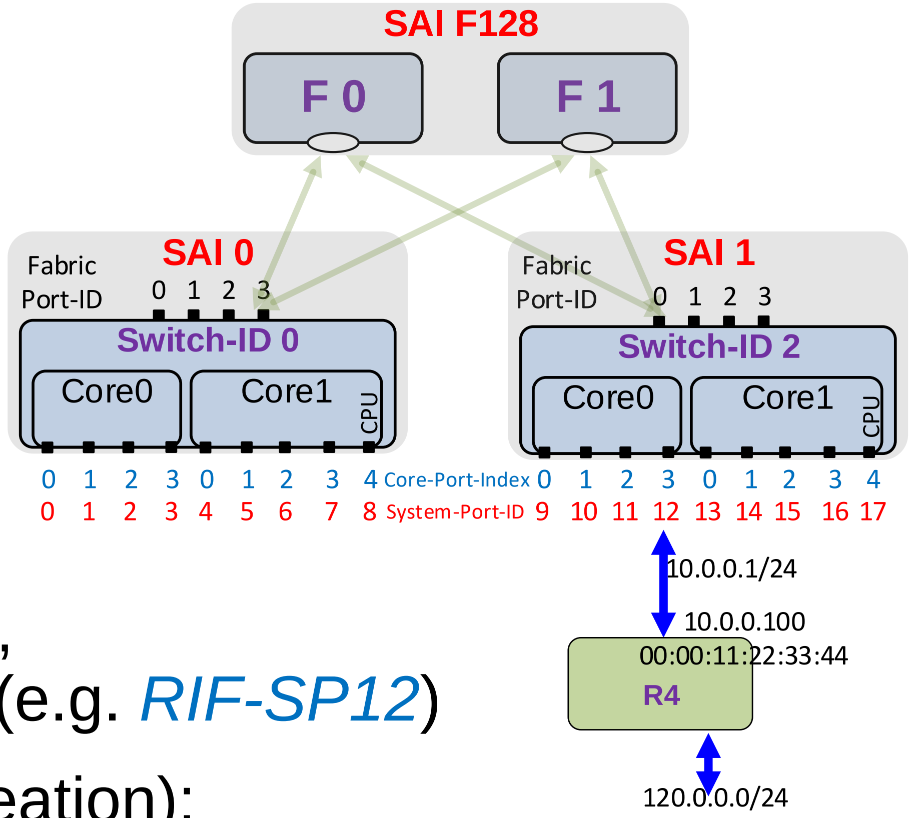
VOQ Switch System - Add Route exercise

- The exercise will refer to a VOQ system, with 2 Fabric devices, and 2 VOQ Switches (2 cores each)
- Fabric devices are passive (“transparent”) for this packet walk.
- Assumption: upper layer (e.g. SONiC) synchronize information/objects/DBs across all SAI images
- 4 SAI APIs are involved: create router interface, create neighbor entry, create next hop, and create route entry

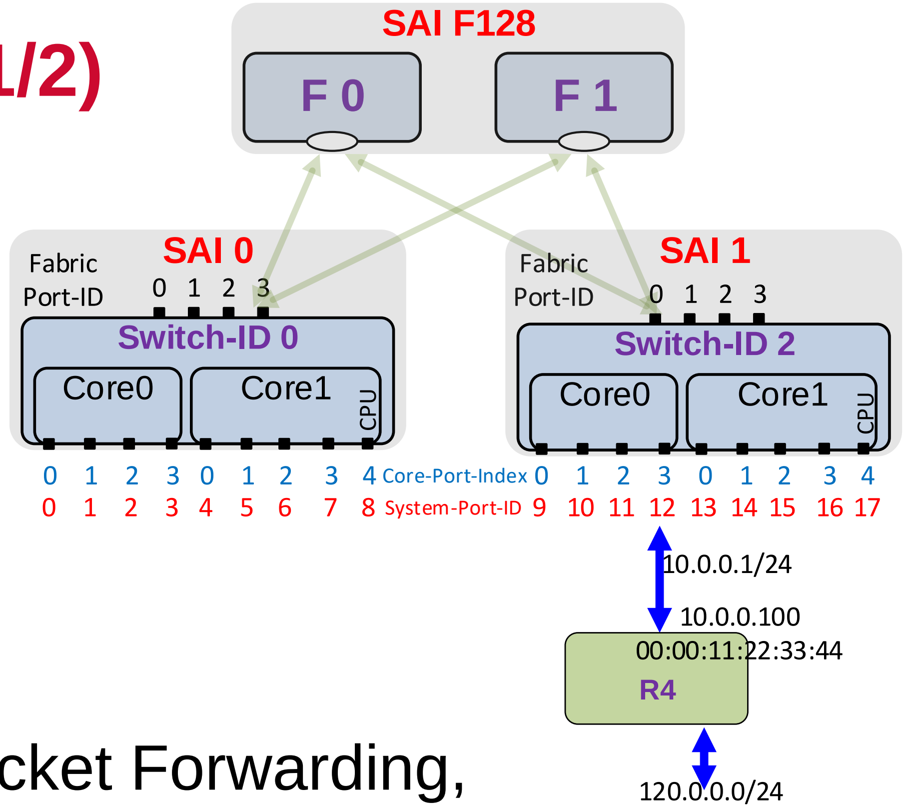


Add Route - RIF

- Call with System-Port instead of Port
- On SAI1:
  create router Interface (VRF=0, System-Port=12, MAC=00:00:12:34:56:78)
  ⇒ SAI object: *RIF-S1-SP12*
- VOQ-System-DB add a RIF object on system port 12, struct with all the RIF parameters, pointed by handle (e.g. *RIF-SP12*)
- On SAI0 (triggered by upper layer, after *RIF-SP12* creation):
  create RIF (VRF=0, System-Port=12, MAC=00:00:12:34:56:78)
  ⇒ SAI object: *RIF-S0-SP12*
  Note: function call parameters are provided by upper layer
- In steady state, all RIFs across all system ports exist on every SAI ASIC-DB in the system

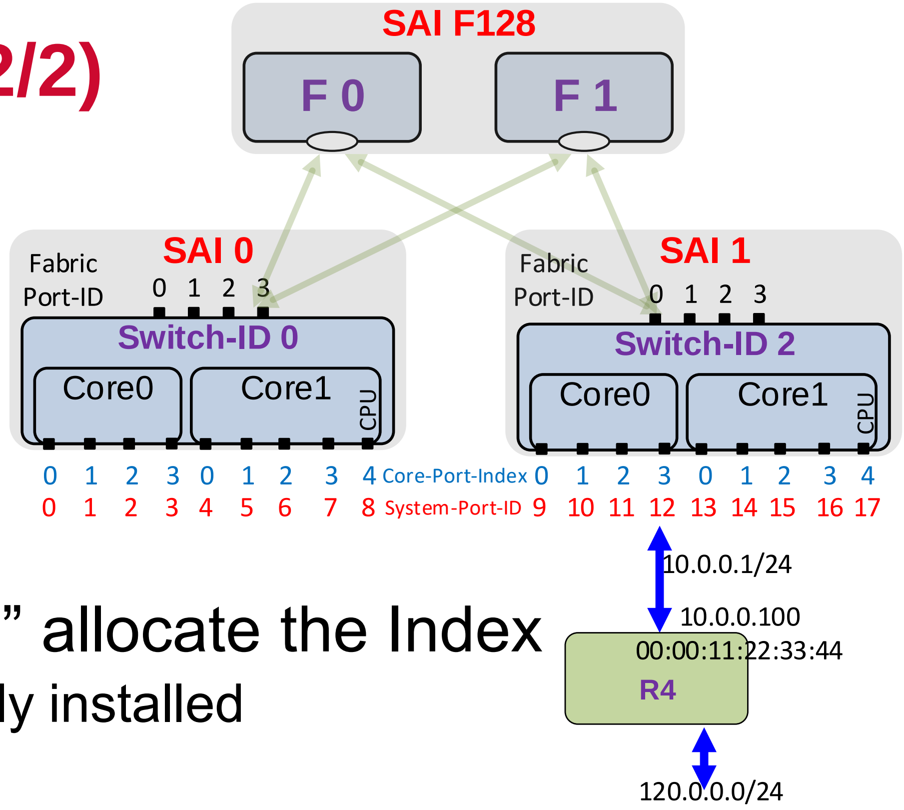


Add Route - Neighbor, functional partition (1/2)



- In a VOQ System, whereas the system behaves as a Single-Hop, in reality the packet is switched through two devices: Ingress Switch and Egress Switch
- This leads to partition of functionality, performed by single Switch in other system, between the Ingress and Egress switches in a VOQ system
- In many architecture the Ingress is responsible for packet Forwarding, i.e. determining the packet destination, and the Egress is responsible for editing the packet
- An alternative architecture of editing the packet on the Ingress device becomes extremely un-efficient in large distributed systems

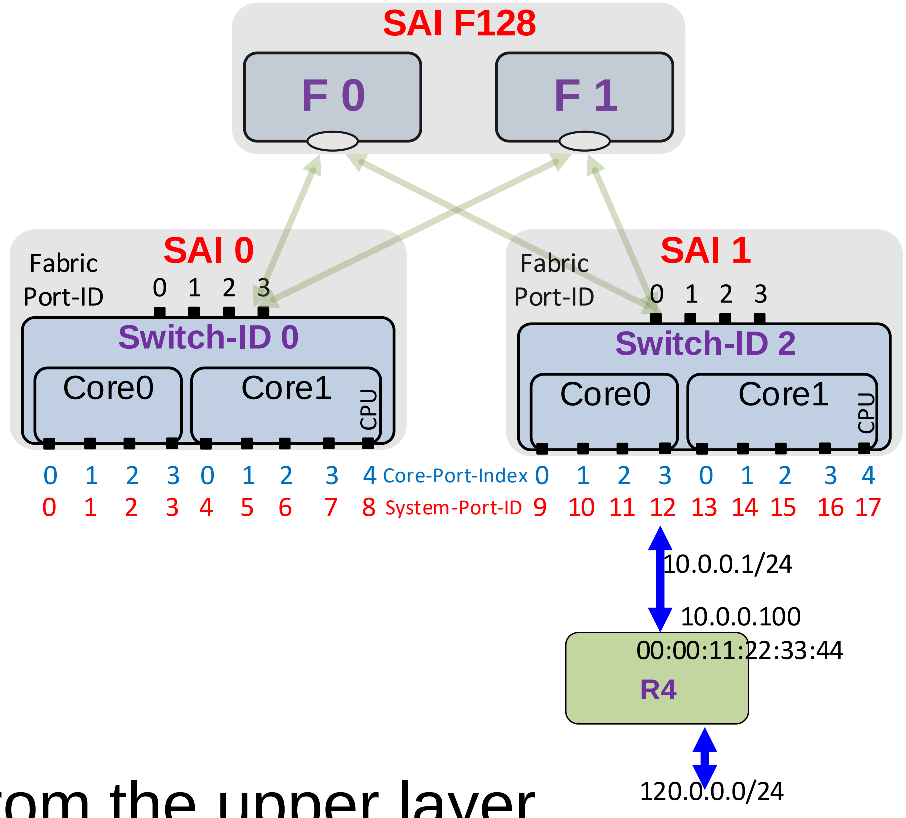
Add Route - Neighbor, functional partition (2/2)



- To facilitate this functional partition an Index (or few Indexes in other use-cases) is passed from Ingress device to Egress device
- There are few methods for allocation and management of these Indexes
- In some cases it is preferred to let the "owning device" allocate the Index
  - Owner device is the device where the "Egress entry" is actually installed
- In other cases it is preferred to impose the Index (for the "owning device") by an higher management layer (e.g. SONiC)

BROADCOM

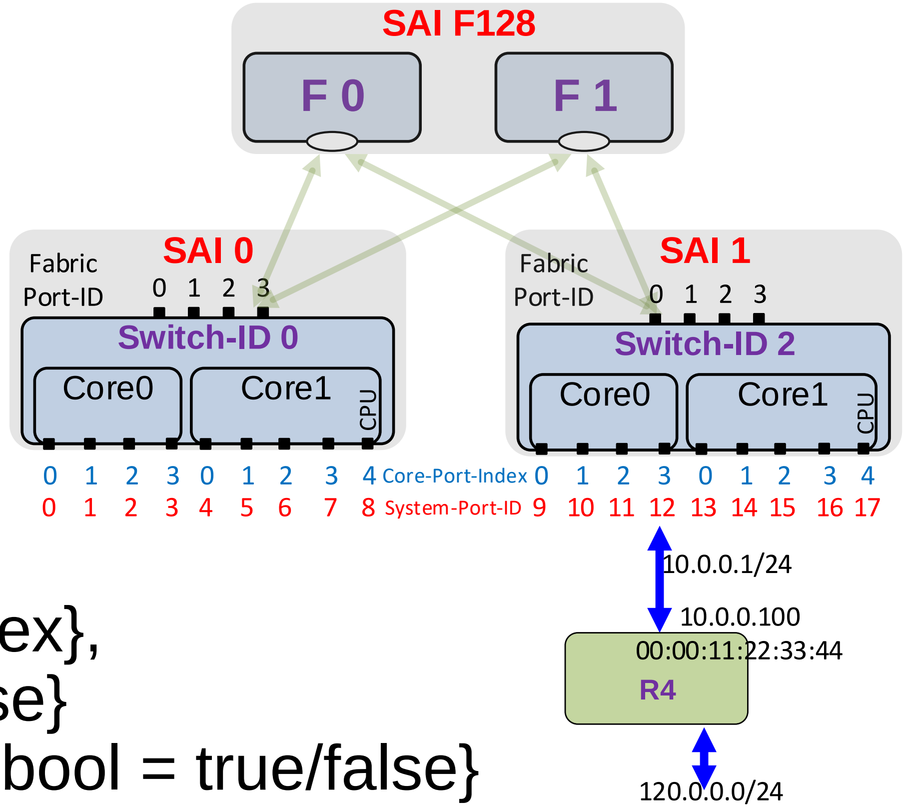
Add Route - Neighbor, new Attributes



- Proposal is to add three new Neighbor attributes:
  SAI-NEIGHBOR-ENCAP-IMPOSE-INDEX // flag
  SAI-NEIGHBOR-PRESENT // flag
  SAI-NEIGHBOR-ENCAP-INDEX // index
- SAI-NEIGHBOR-ENCAP-IMPOSE-INDEX
  is a flag, determining if the
  SAI-NEIGHBOR-ENCAP-INDEX will be self allocated
  (i.e. by the "owning device" SAI),
  or SAI-NEIGHBOR-ENCAP-INDEX will be imposed from the upper layer
- SAI-NEIGHBOR-PRESENT is a flag, determining if the Neighbor is "installed" on this device, it may be used by some functions such as counters
  - Note that the value of SAI-NEIGHBOR-PRESENT is updated in the event of Neighbor relocation to another Switch

BROADCOM

Add Route - Neighbor

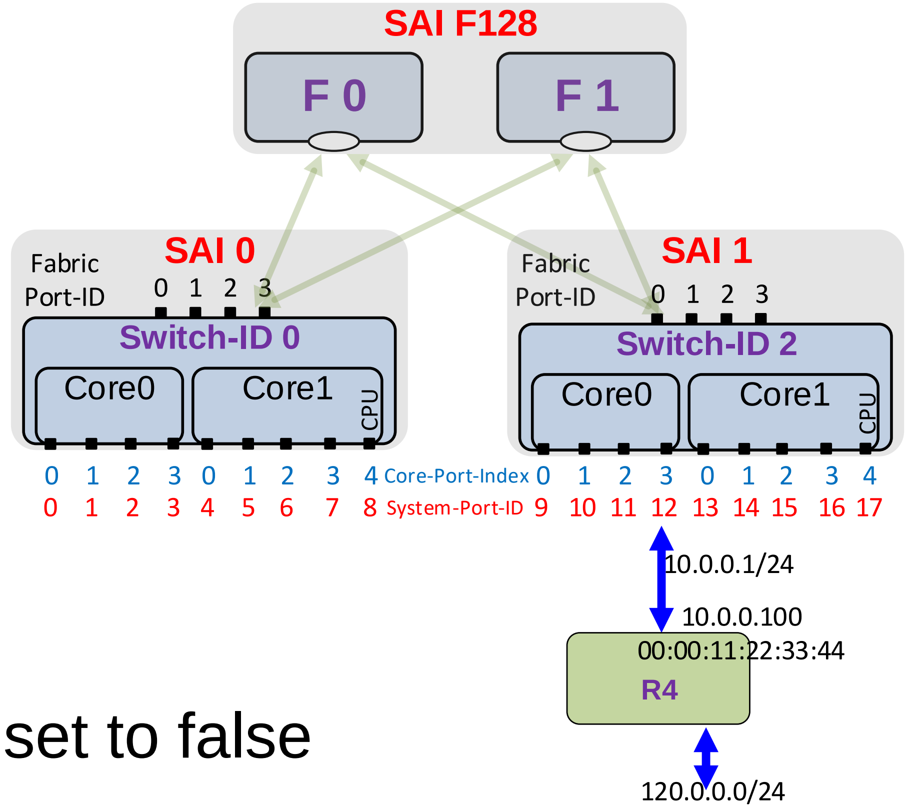


- API Calls:
  sai_attribute_t remote_attr_list[] = {
      …
      {SAI-NEIGHBOR-ENCAP-INDEX, .value.u32 = index},
      {SAI-NEIGHBOR-PRESENT, .value.bool = true/false}
      {SAI-NEIGHBOR-ENCAP-IMPOSE-INDEX, .value.bool = true/false}
      …
  };
  create neighbor entry (IP-Addr, MAC-Address, RIF,)
  get neighbor Index (IP-Addr, MAC-Address, RIF)
- Based on the flag values the SAI will either self allocate or impose the Out-LIF index

BROADCOM

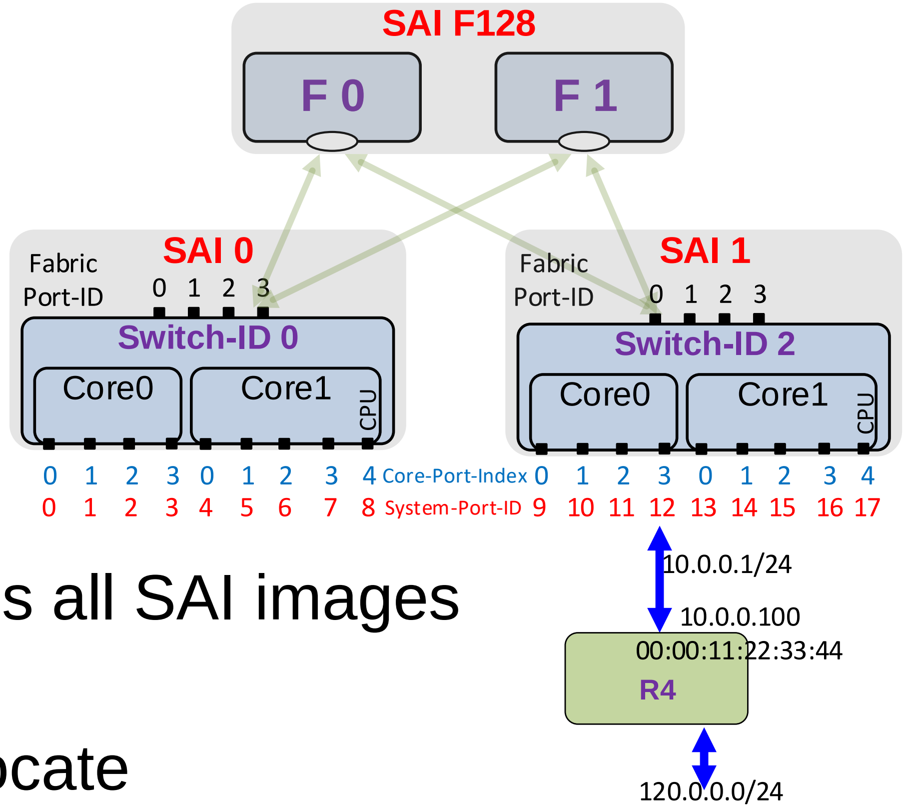
Add Route - Neighbor, usage method 1

- When the neighbor RIF is port (i.e. exist on a single device), it is recommended that SAI-NEIGHBOR-ENCAP-INDEX will be self indexed, i.e. assigned by the owning device SAI
- For example if the RIF on System-Port 12 is a port RIF, then it is expected that SAI of Switch-2 will allocated the SAI-NEIGHBOR-ENCAP-INDEX, i.e. SAI-NEIGHBOR-ENCAP-IMPOSE-INDEX will be set to false
- Neighbor creation will update the SAI-NEIGHBOR-ENCAP-INDEX, which will be synched to other SAIs by upper layer
- Later, when Switch-0 is creating the same neighbor, it is expected that it'll use both SAI-NEIGHBOR-ENCAP-IMPOSE-INDEX, and SAI-NEIGHBOR-ENCAP-INDEX, in the neighbor creation call



Add Route - Neighbor, usage method 2

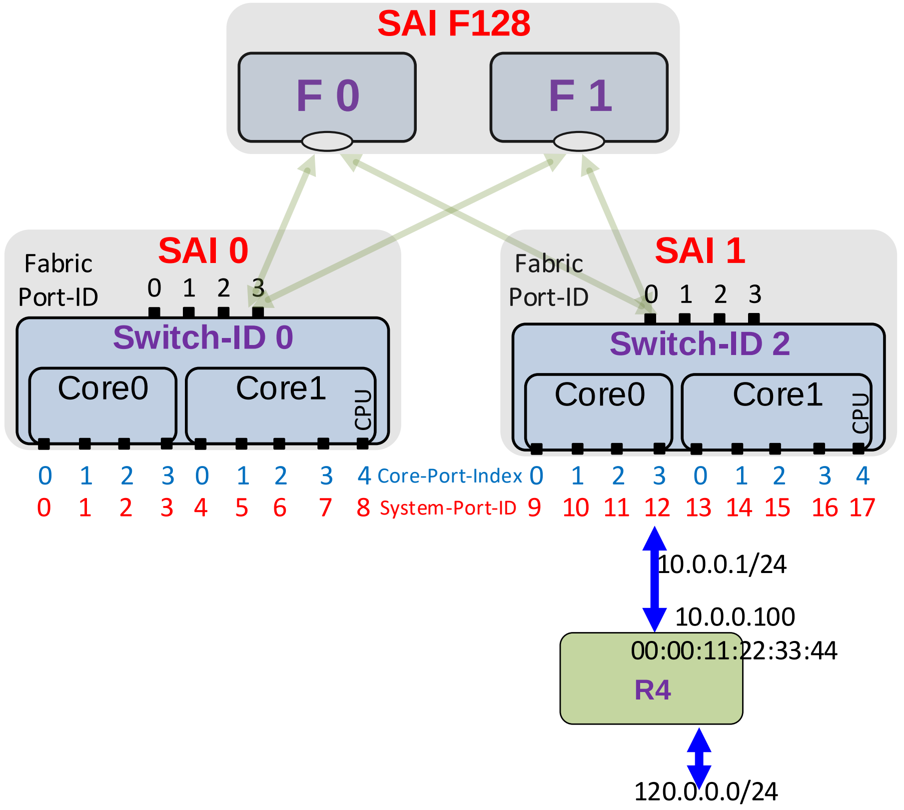
- When the neighbor RIF is VLAN port, or any other case where the neighbor can be relocated to other Switch or Core in the system, it is recommended that SAI-NEIGHBOR-ENCAP-INDEX will be imposed by upper layer
- The SAI-NEIGHBOR-ENCAP-INDEX is synced across all SAI images
- For example if the RIF on System-Port 12 is a VLAN RIF, then it is expected that upper layer will allocate SAI-NEIGHBOR-ENCAP-INDEX, and first impose it on Switch-2, and later on rest of the switches in the system (Switch-0 in this system)



BROADCOM

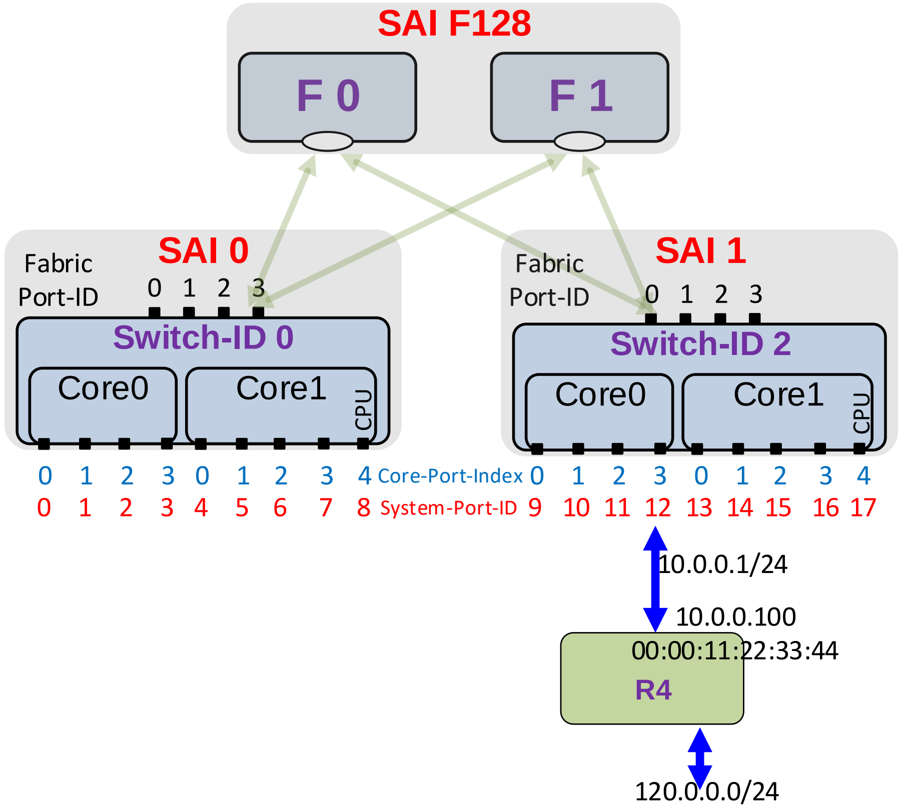
Add Route - Next Hop

- Next Hop creation is triggered by Neighbor creation
- On SAI1:
  create next hop (IP=10.0.0.100, RIF=RIF-S1-SP12)
  ⇒ SAI object: *NH-S1-R4*
- On SAI0:
  create next hop (IP=10.0.0.100, RIF=RIF-S0-SP12)
  ⇒ SAI object: *NH-S0-R4*



Add Route - Route entry

- Route updates between the different devices in the system is out of scope for this SAI sub group
- On SAI1:
  *create route entry (VRF=0, Prefix=120.0.0.0/24, *Next-Hop=*NH-S1-R4)*
  Add prefix to the Router table (LPM) pointing to the FEC associated with *NH-S1-R4*
- On SAI0:
  *create route entry (VRF=0, Prefix=120.0.0.0/24, *Next-Hop=*NH-S0-R4)*
  Add prefix to the Router table (LPM) pointing to the FEC associated with *NH-S0-R4*
- No change to create route call



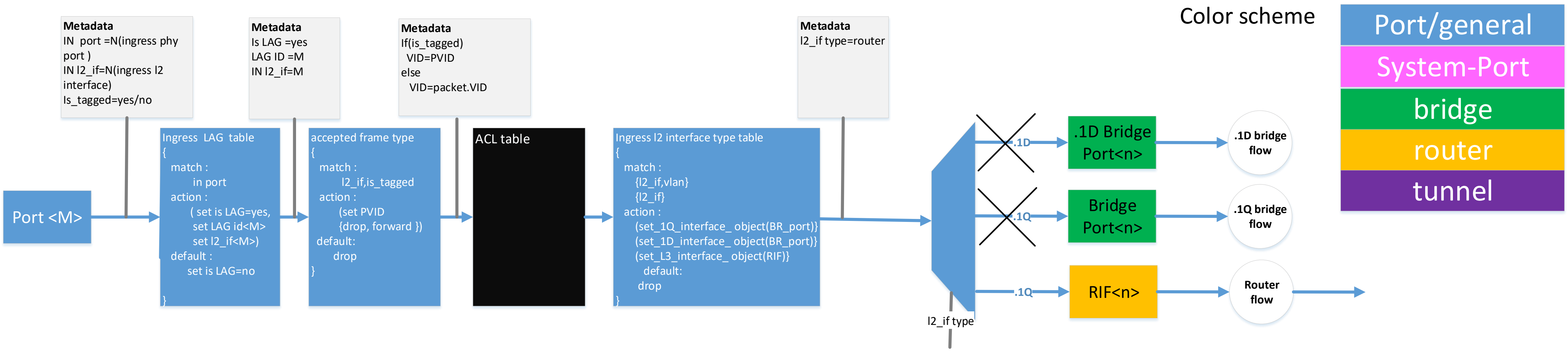
VOQ System pipeline

Pipeline - Ingress Port

- Split the pipeline into parts, to ease review
- Based on pipeline v7 - UC Routing
- Focus on RIF ports

- No major change

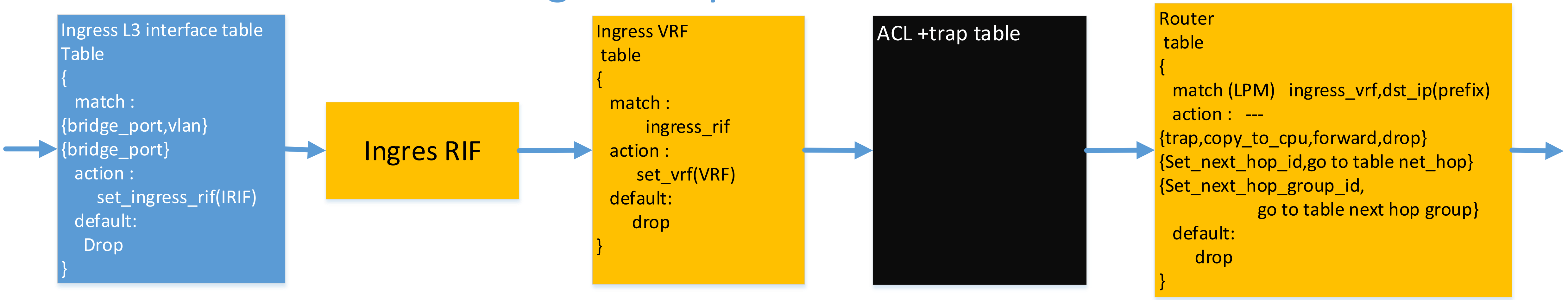
Ingress port flow



Pipeline – Route flow, part 1

- No major change

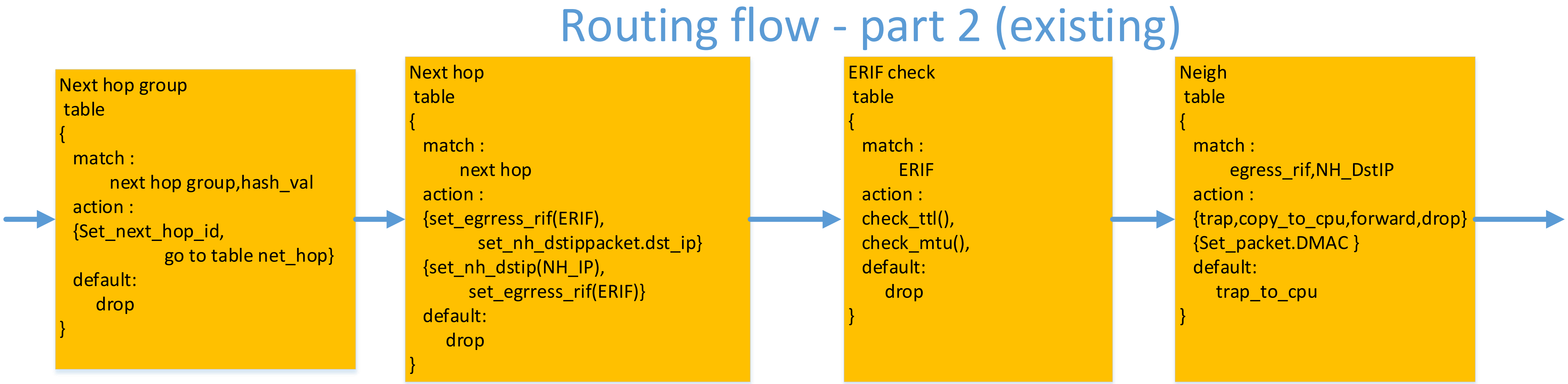
Routing flow - part 1



BROADCOM

Pipeline - Route flow, part 2 - Existing pipeline

- Note that Neighbor table set the Forwarding decision, but don’t set the destination port explicitly



Pipeline - Route flow, part 2 - Modified pipeline

- Split Neighbor table into two tables:
  - Neighbor Forwarding
    - Accessed with (egress_RIF, NH-DestIP) // same as existing
    - decides where to forward the packet: Trap/Snoop/Forward/Drop // same as existing
    - Provides the Index to the Neighbor Encapsulation table
  - Neighbor Encapsulation
    Provides the packet Dest-MAC // previously in the single Neighbor table

Routing flow - part 2

```
Next hop group
 table
{
  match :
        next hop group,hash_val
  action :
  {Set_next_hop_id,
                 go to table net_hop}
  default:
      drop
}
```

```
Next hop
 table
{
  match :
        next hop
  action :
  {set_egrress_rif(ERIF),
            set_nh_dstippacket.dst_ip}
  {set_nh_dstip(NH_IP),
           set_egrress_rif(ERIF)}
  default:
      drop
}
```

```
Neighbor-Forwarding
 table
{
  match :
        egress_rif,NH_DstIP
  action :
  {trap,
   copy_to_cpu,
   forward,
   drop}
  {Neighbor-Encap-Index}
  default:
      trap_to_cpu
}
```

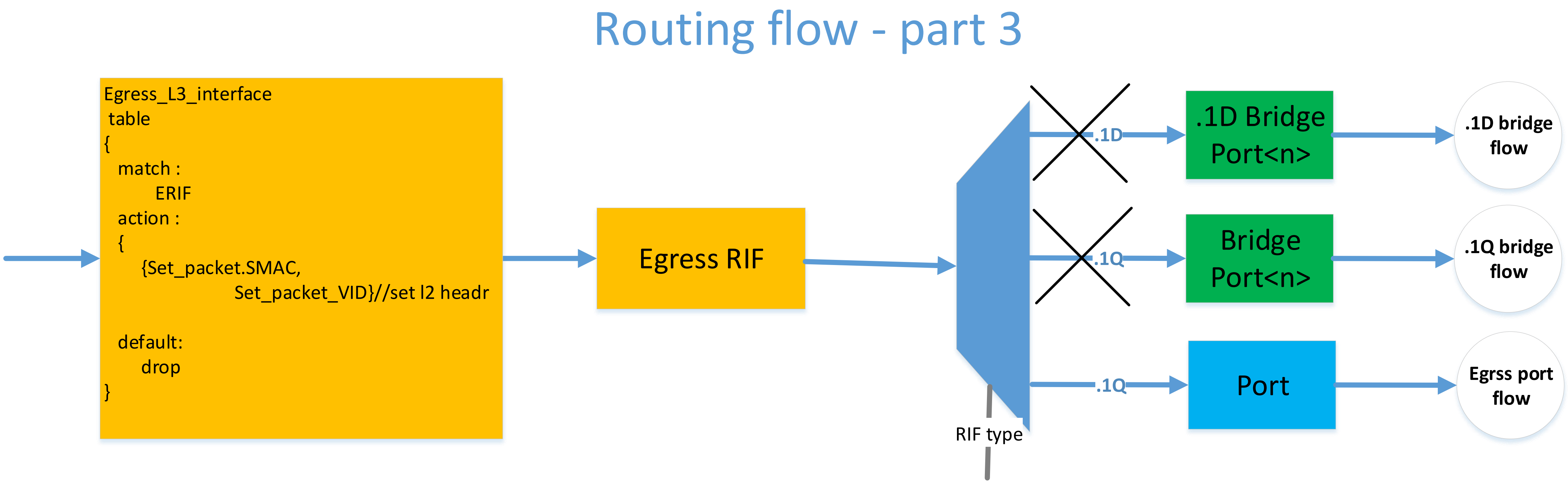
```
ERIF check
 table
{
  match :
        ERIF
  action :
  check_ttl(),
  check_mtu(),
  default:
      drop
}
```

```
Neighbor-Encapsulation
 table
{
  match :
        Neighbor-Encap-Index
  action :
  {Set_packet.DMAC}
  default:
      trap_to_cpu
}
```


BROADCOM

Pipeline – Route flow, part 3

- No major change



Thank You