Flow-based tunneling for SR-IOV using switchdev API

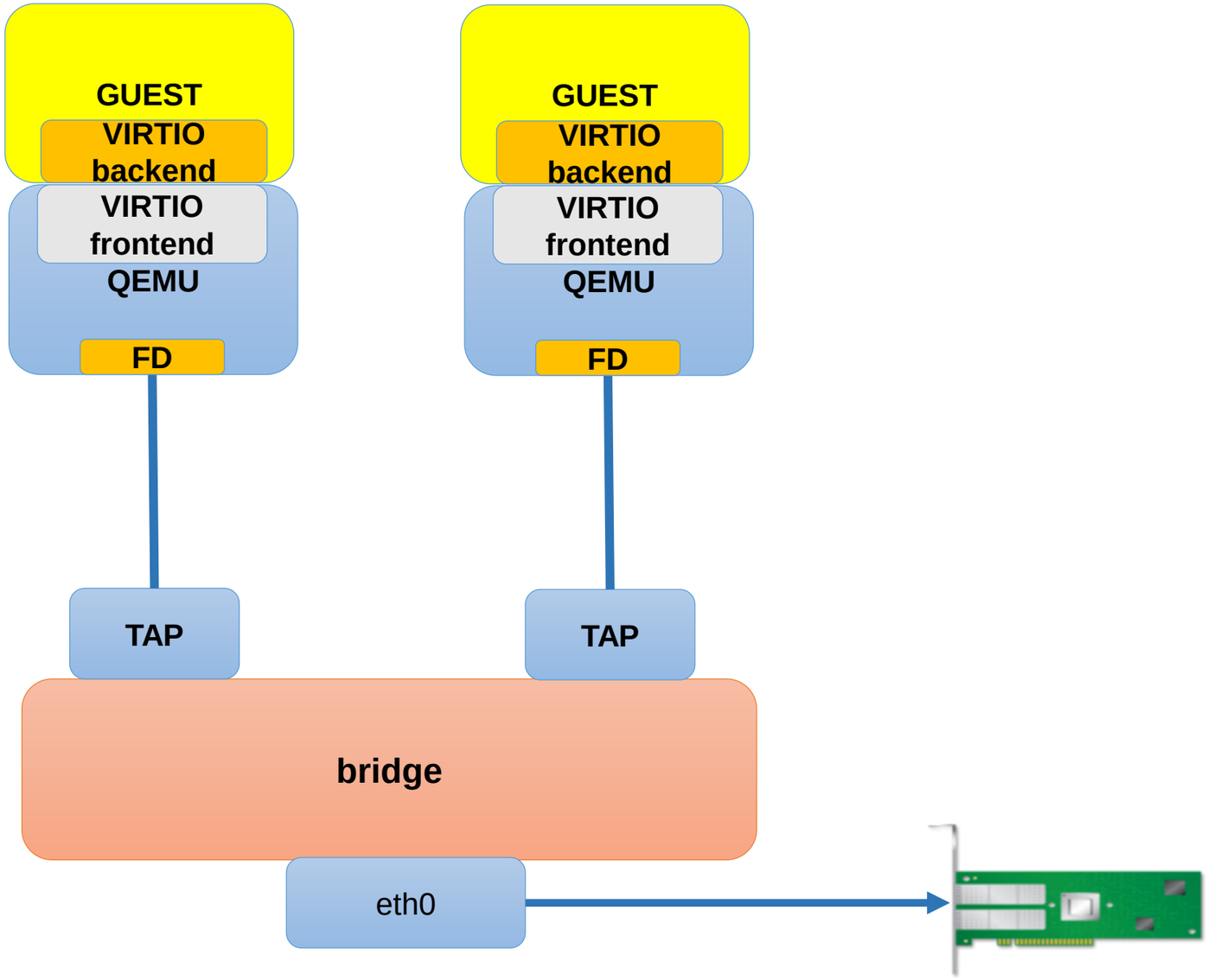
Ilya Lesokhin, Haggai Eran, Or Gerlitz

February 2016

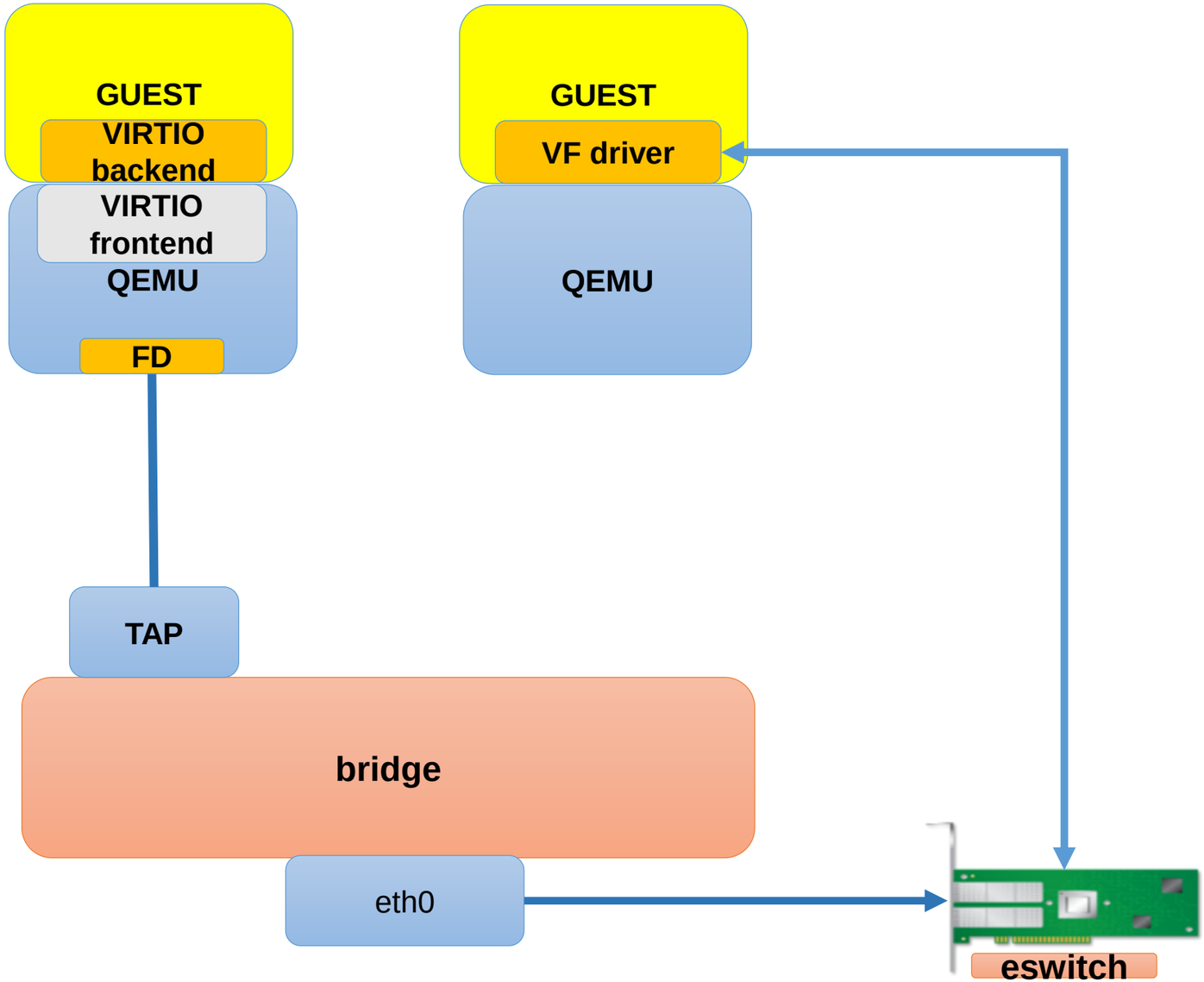
Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™

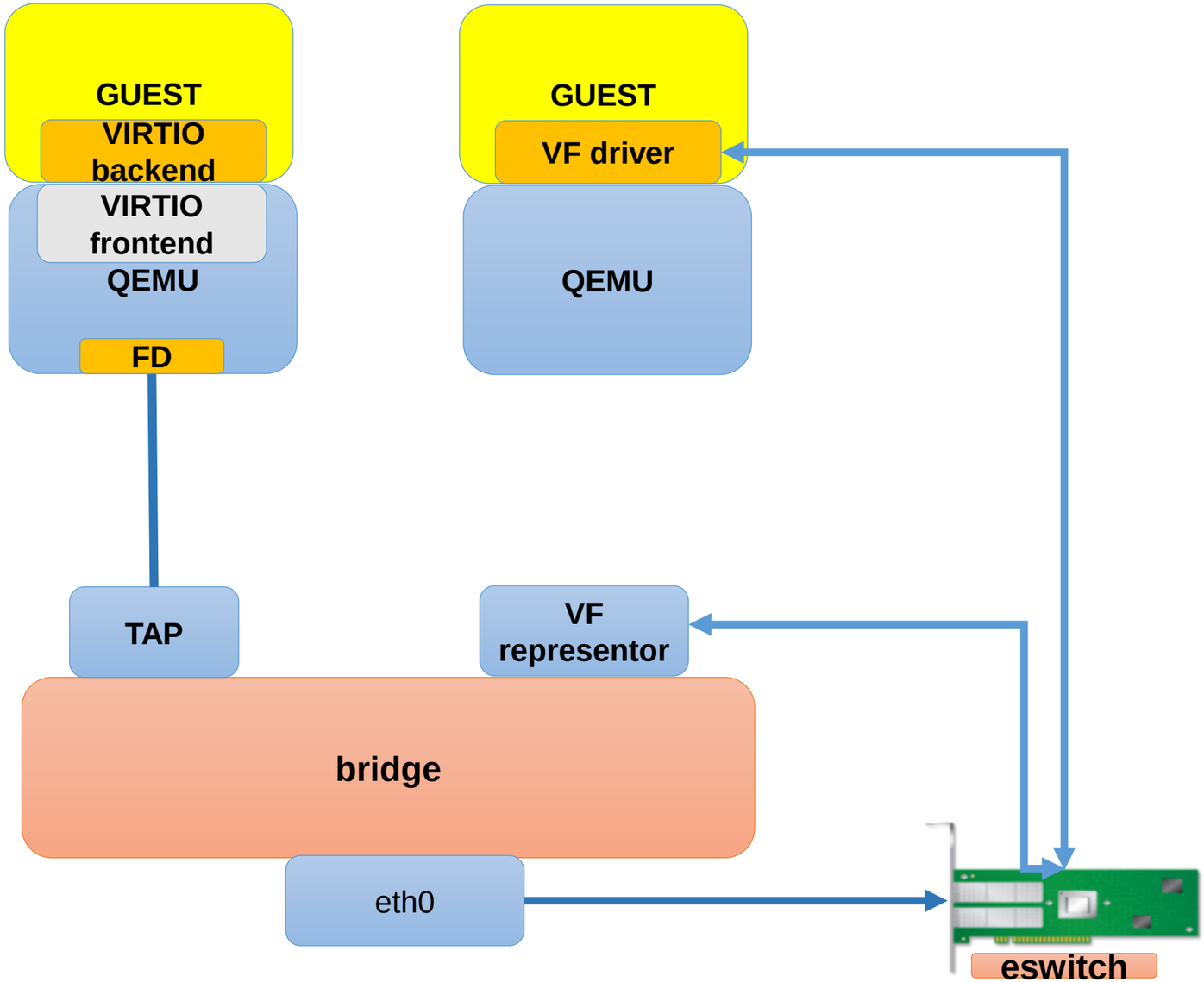
Para-virt networking model



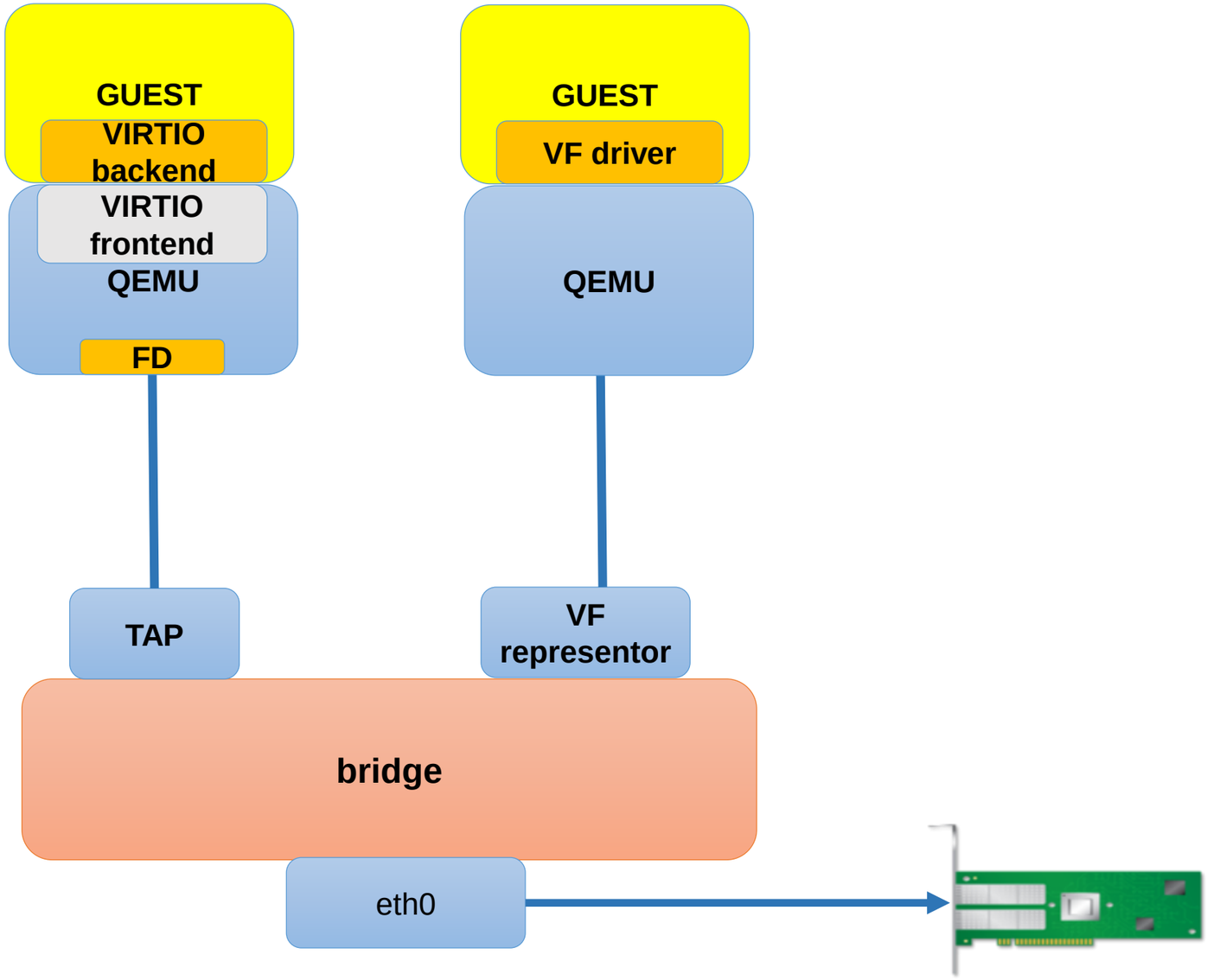
SRIOV has its own management



SRIOV has its own management

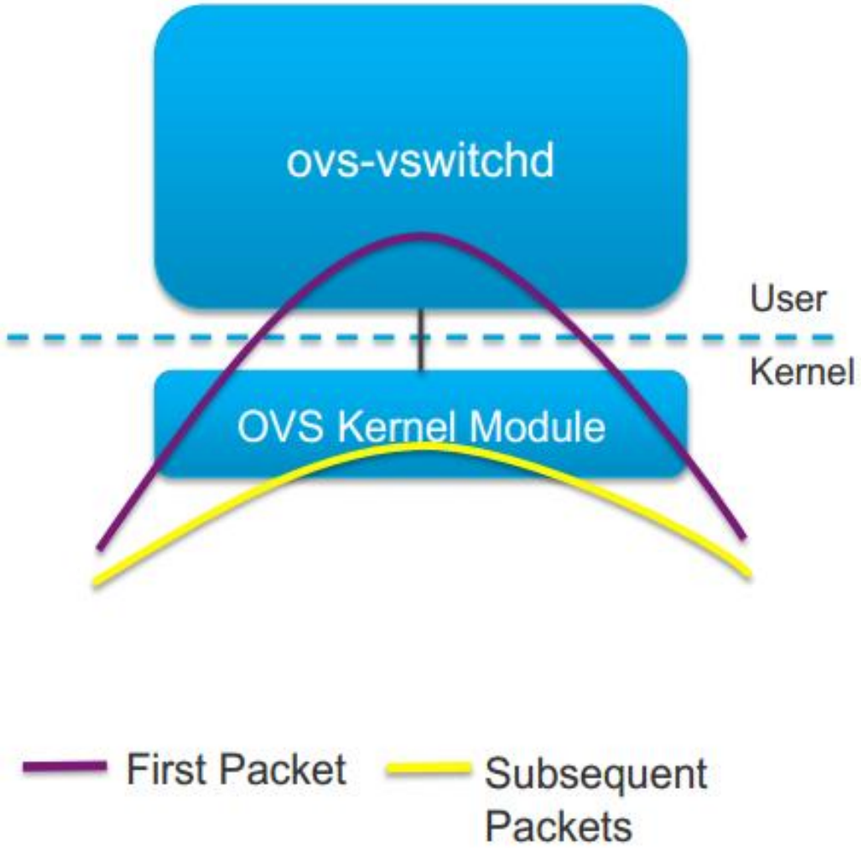


SRIOV has its own management



Open vSwitch

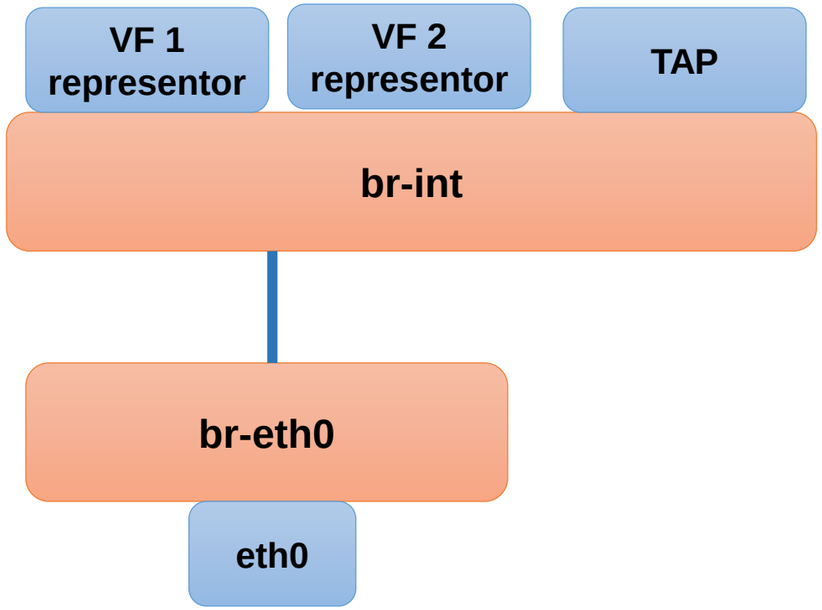
- Forwarding
  - Flow-based forwarding
  - Decisions are made in user space
  - First packet of a new flow is directed to ovs-vswitchd, following packets hit cached entry in kernel



- OVS Overview
  - http://openvswitch.org/slides/OpenStack-131107.pdf

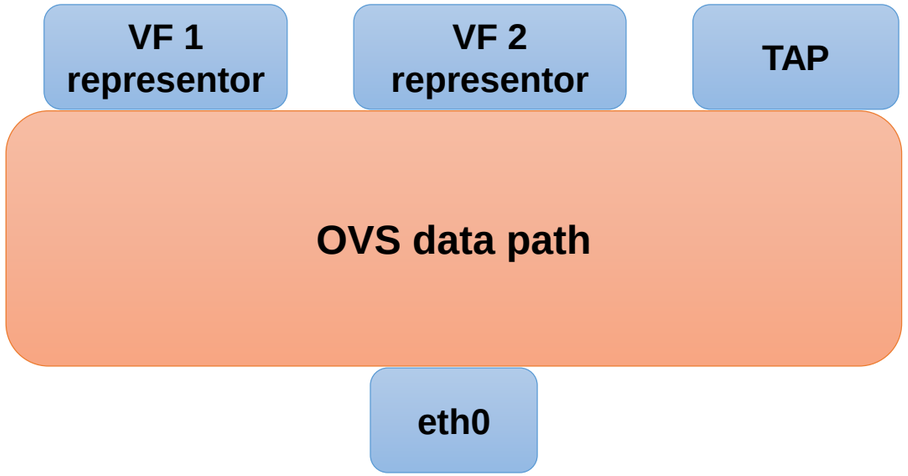
OVS data path

Per bridge complex open flow rules with priority



OVS-vswitchd

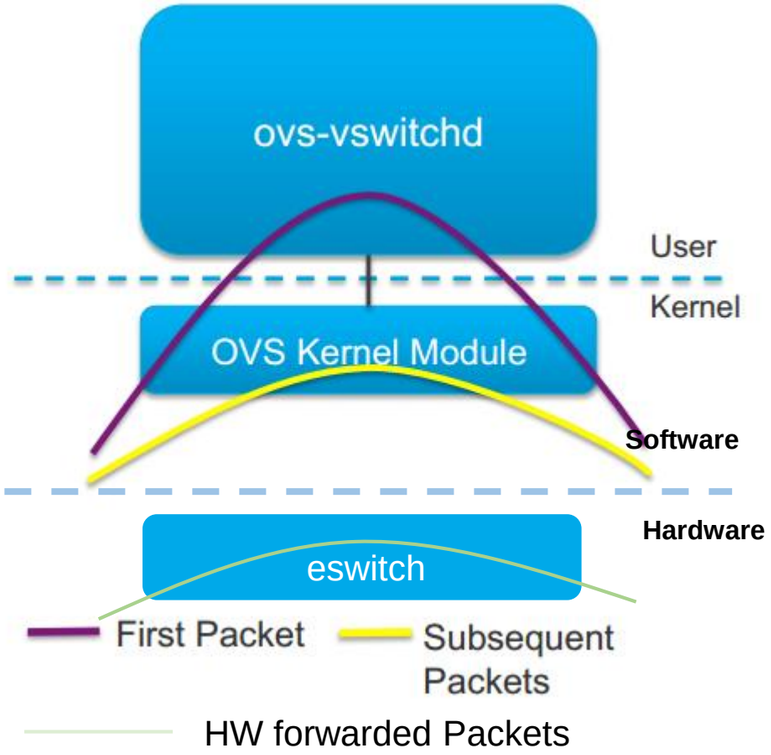
One bridge with simple mutually exclusive rules



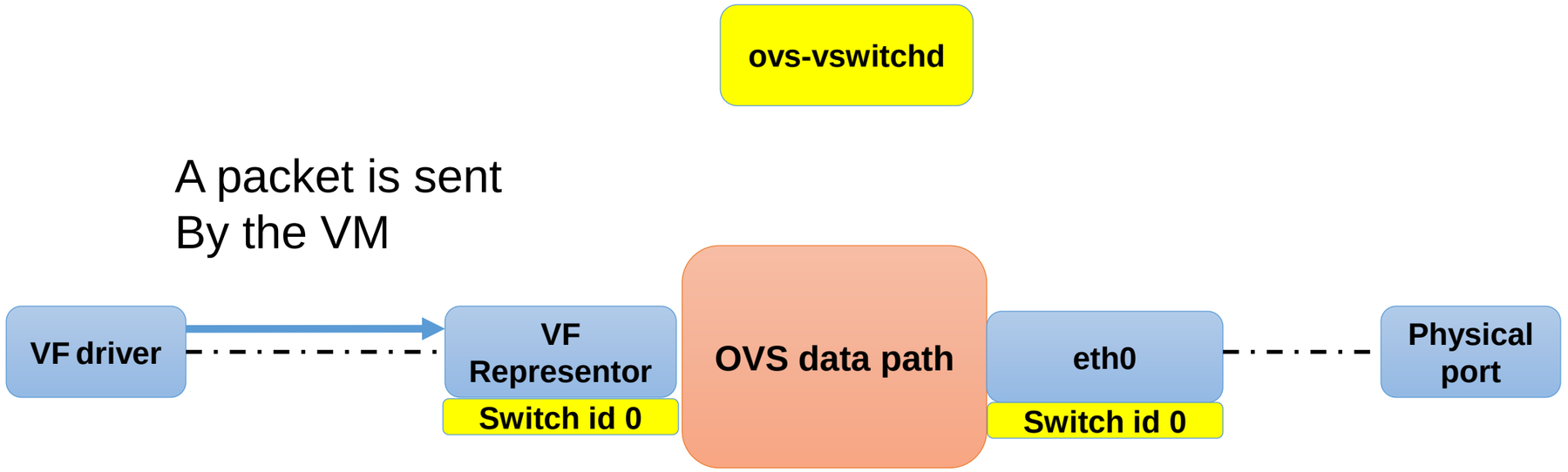
OVS offload – solution: adding the Hardware layer to the forwarding plane

- The NIC Embedded Switch is layered below the kernel module
- A miss in the eswitch causes the packet to be forwarded to the kernel data path and possibly to user space
- When a cached entry is inserted to the data path we try to offload it to the hardware.

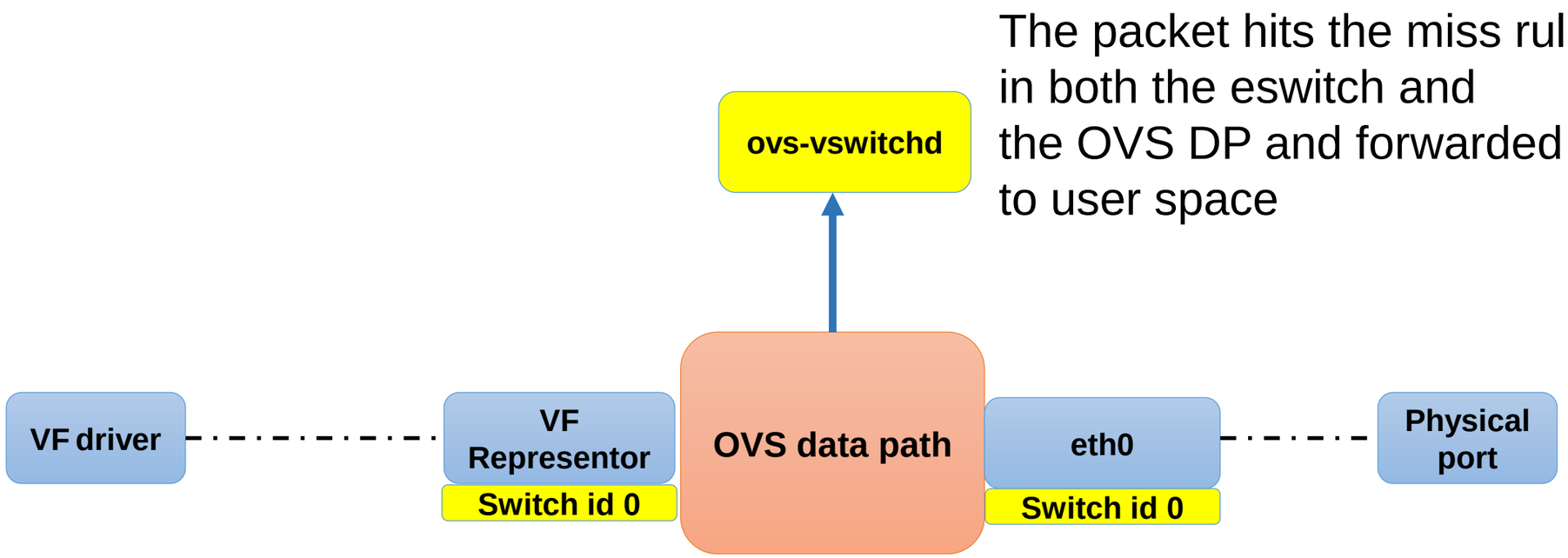
Retain the “first packet” concept and enable the “fast-est” path – via the HW switch



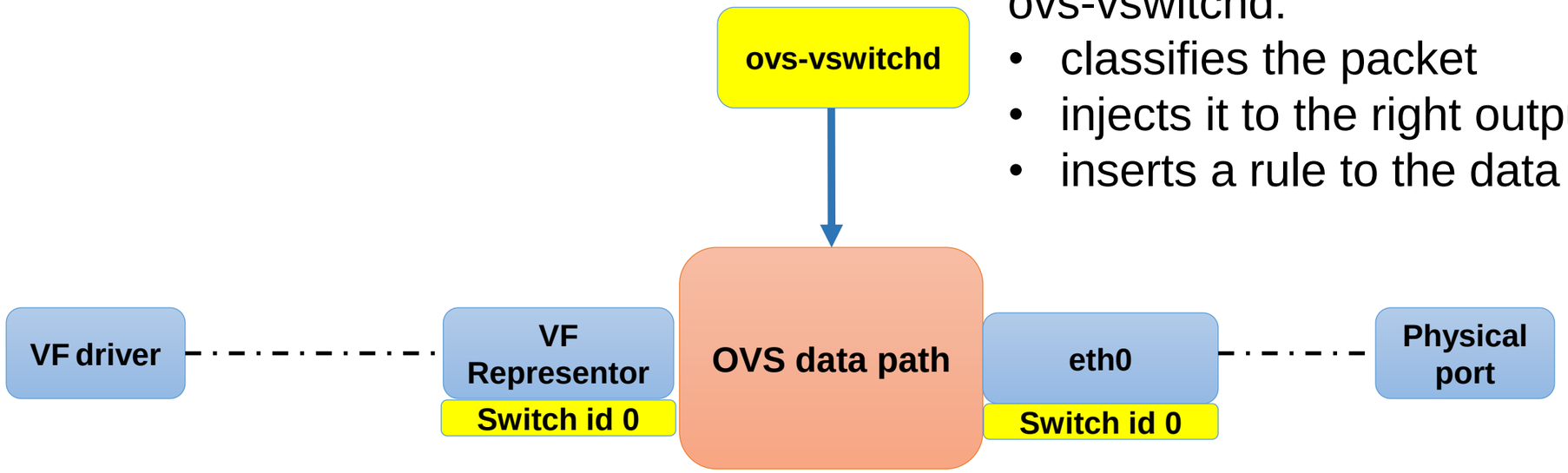
Offloading rules to the hardware



Offloading rules to the hardware



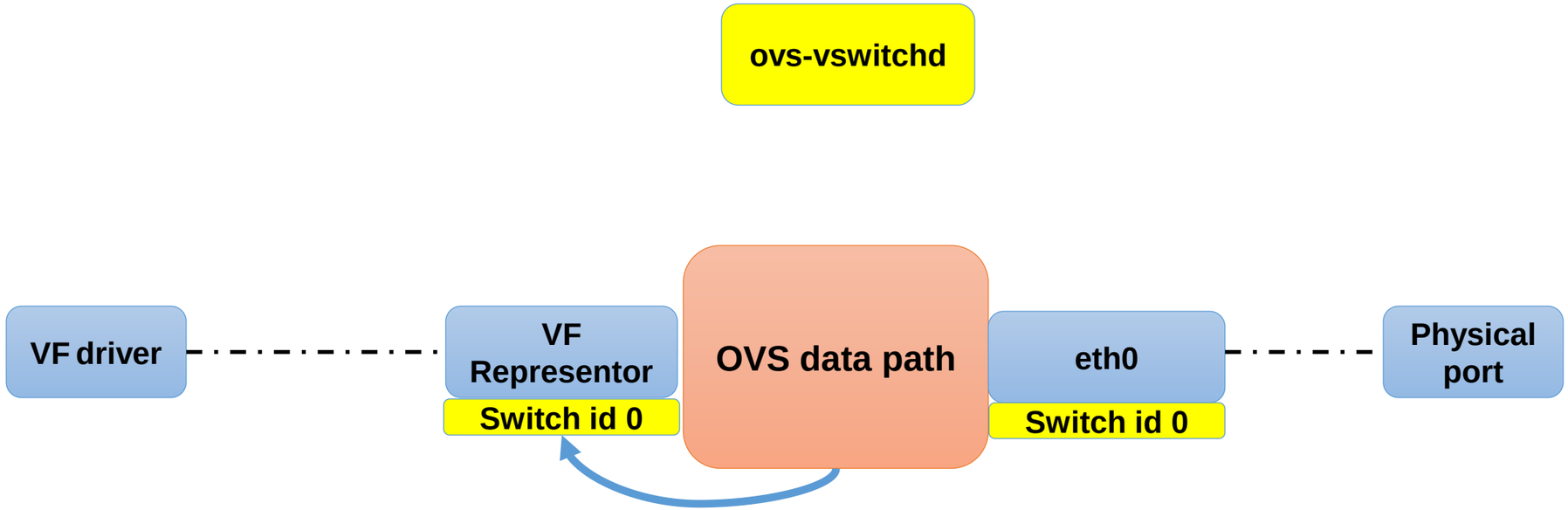
Offloading rules to the hardware



ovs-vswitchd:

- classifies the packet
- injects it to the right output port
- inserts a rule to the data path

Offloading rules to the hardware



Our hook in the OVS DP uses the ingress vport of the flow to offload that flow.
The offloading is done through switchdev.

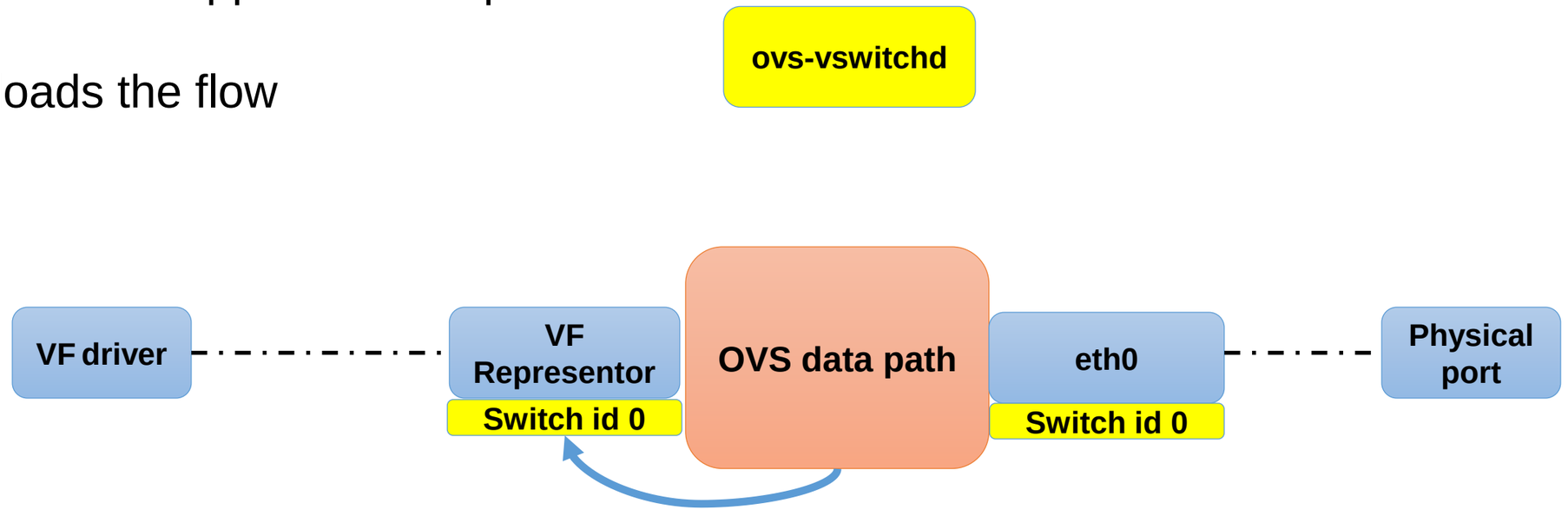
This is well defined as the OVS DP mandates a full match on the ingress port.

Offloading rules to the hardware

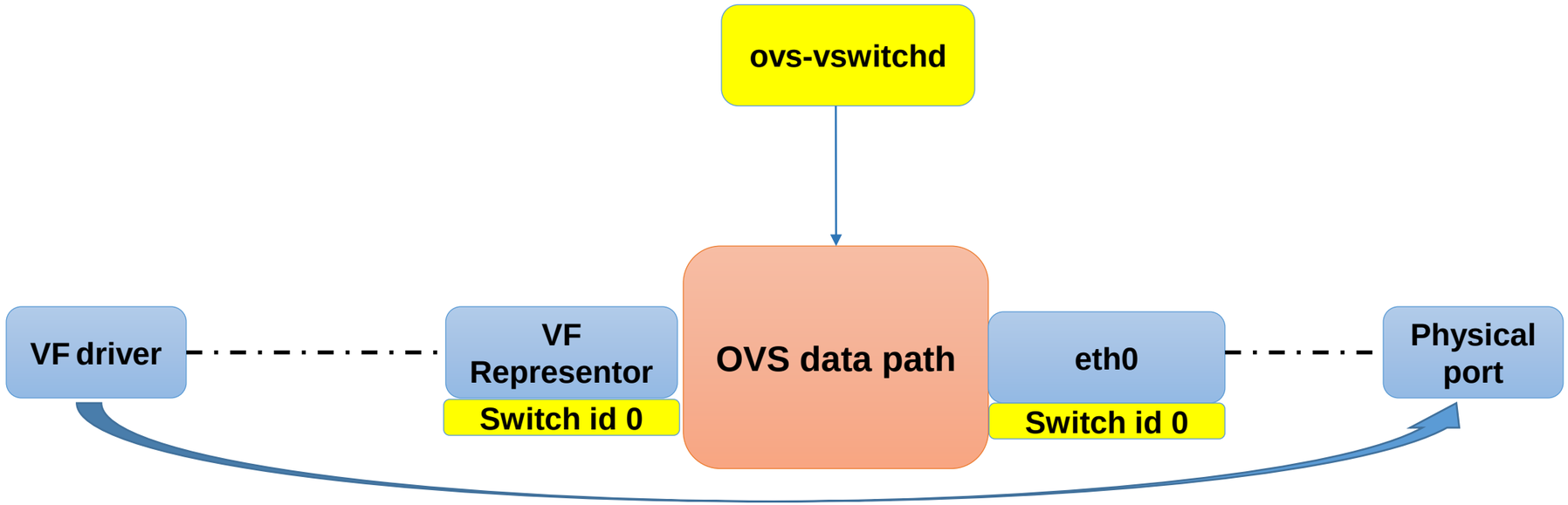
The eswitch driver verifies that:

1. The source and destination have the same switch ID
2. The device supports the required matching
3. The device supports the required actions

And offloads the flow



Offloading rules to the hardware



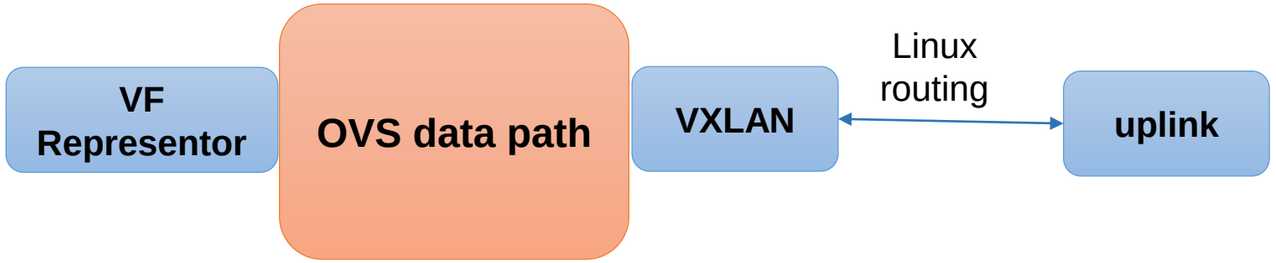
The following packets in the same flow go directly to the physical port without any hypervisor involvement.

VLAN push/pop vs. tunneling in OVS

In_port=VF representor,(Match criteria)
action=**push_vlan(id=…),output:uplink**

In_port=uplink,(Match)
action=**pop_vlan,output:VF representor**

In_port=VF representor, (Match criteria)
action=**set_attr(vni=…, dst_ip=…),output:vxlan**

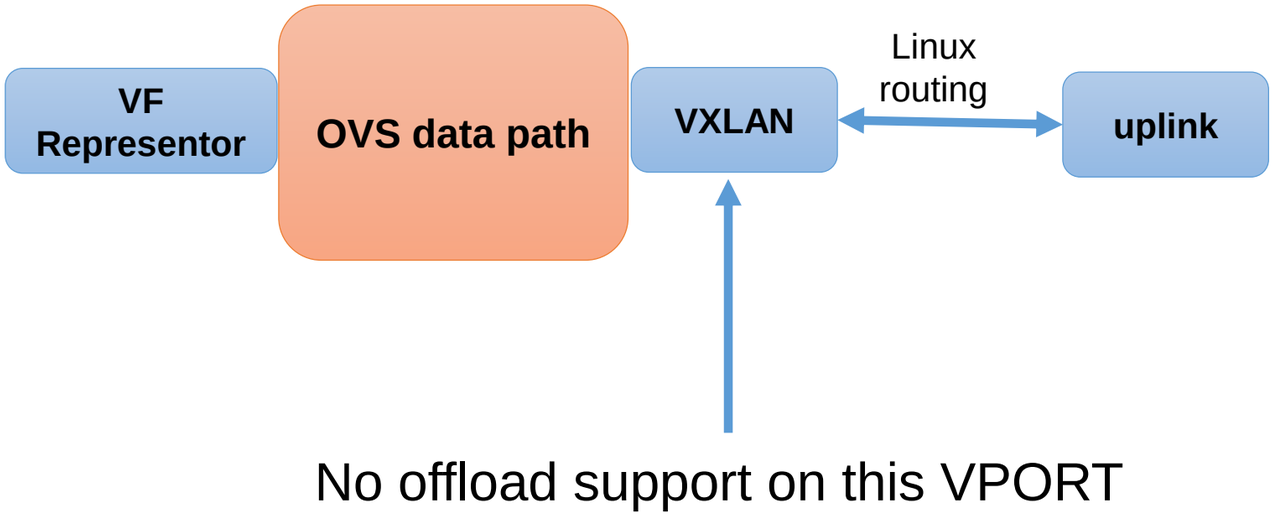


In_port=vxlan,(Match)
action=**output:VF representor**

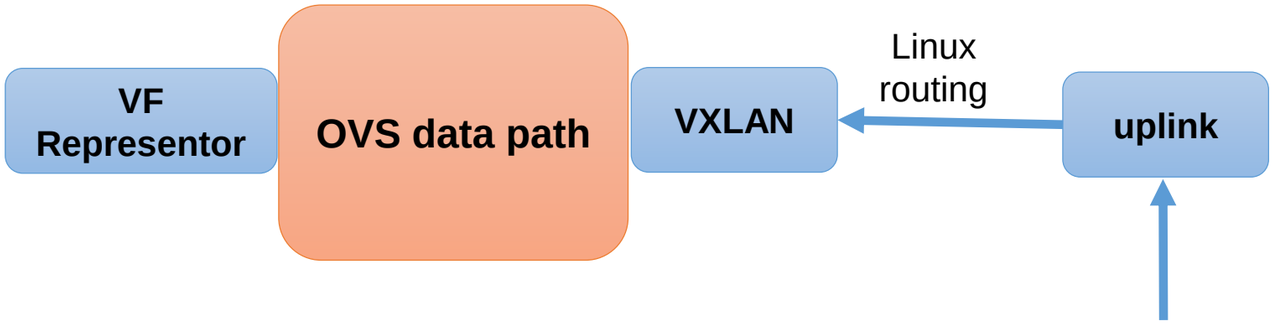
The OVS Data path rule lacks:

- Routing information
- Layer two information

Tunneled flows can’t be offloaded through ingress port



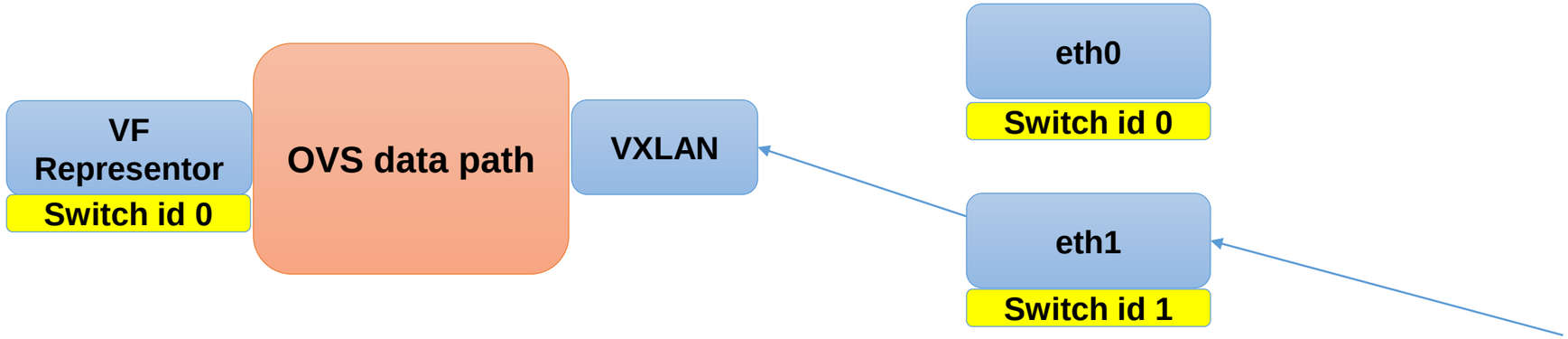
Find “real” ingress port for offloading



Use fib_lookup(…) to get the “real” ingress port of the flow

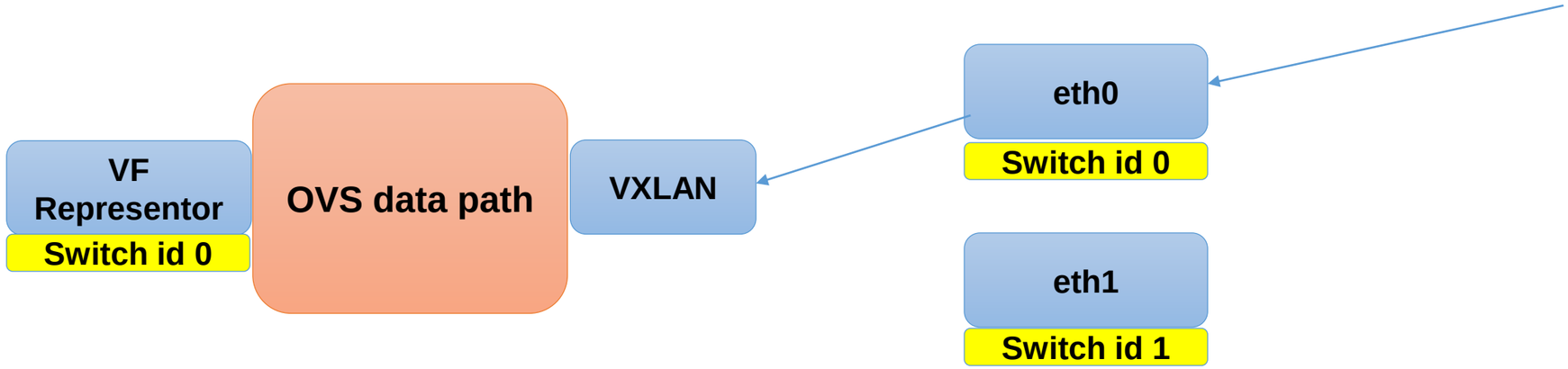
+ Driver need to listen for routing changes

Decapsulation ingress port problem



A vxlan packet comes through eth1.
A rule is inserted to the data path to forward the packet to the VF.
The flow is not offloaded because the VF and eth1
Belong to different switch.

Decapsulation ingress port problem



Due to routing changes, the flow we inserted earlier now comes through eth0.
Since the flow is already in the data path, we don’t try to offload it again and
The flow is not offloaded.

Solution

- Add a new functions
  - int switchdev_tunneled_flow_add(struct net *net, struct sw_flow *flow)
  - int switchdev_tunneled_flow_del(struct net *net, struct sw_flow *flow)
- switchdev will maintain a sorted data structure mapping

Outer dst IP => (flow, offloading device [if offloaded])

- Upon fib update: [switchdev_fib_ipv4_add (192.168.0.3/24)]
  - Foreach relevant flow:
    - Do routing to find target device (to account for policy routing)
    - Remove flow from previous offloading device.
    - Ask the new device to offload the flow.

MTU

- Encapsulating a packet might cause it to exceed the MTU
- The VF representor should reflect the VF MTU. The driver will refuse to offload flows if the representor MTU + encapsulation header > PORT MTU

Conclusions

- Flow based offloads provides SR-IOV performance with Para-virt like flexibility.
- Our work shows the feasibility for openvswitch offloading with and without tunneling.
- We hope our work can be the basis for upstream flow based offloading support.

Open issues

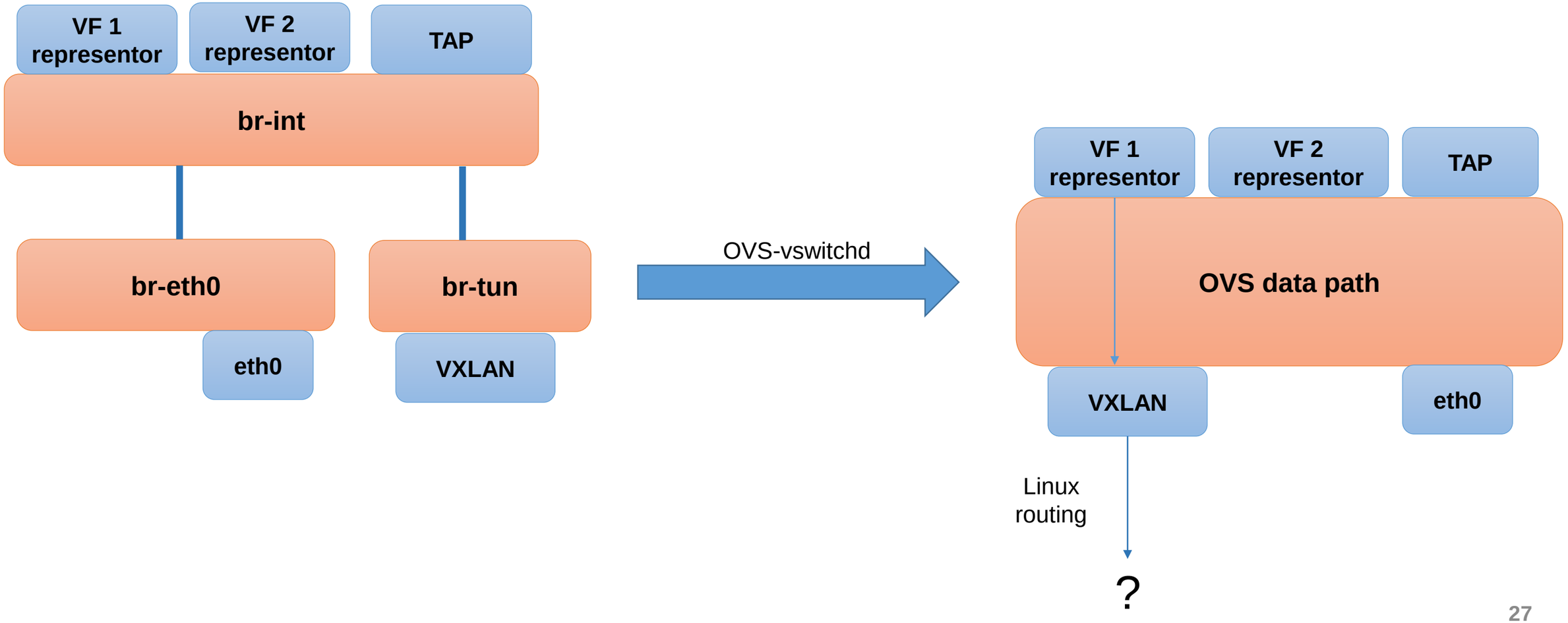
- Who should ask the driver to offload the flows?
  - The OVS data path?
  - ovs-vswitchd? through what interface? TC? Netlink?
- Who should maintain the list of decapsulated flows?
  - switchdev?
  - A new offload management module?
- How should we represent flows? sw_flow? cls_flower?

Questions?

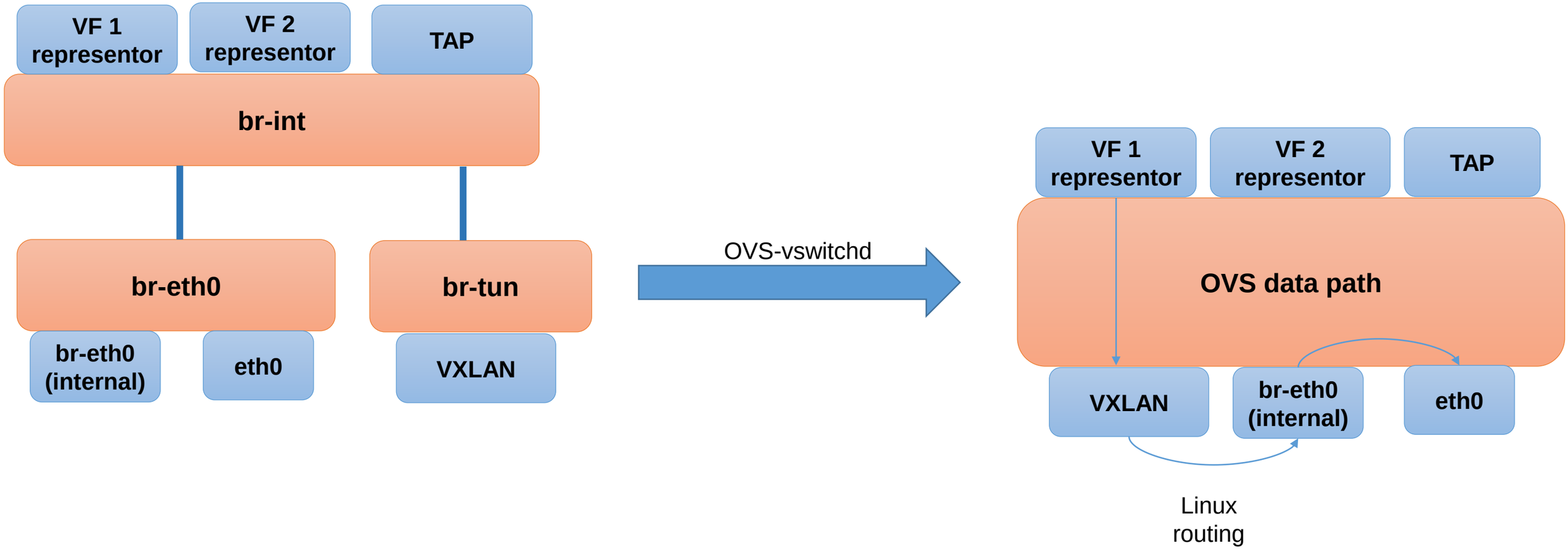
Thank you

Backup

How flat and vxlan networks work side by side?

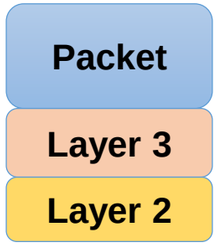


Two passes through the data path



Tunnel actions don’t have layer 2 information

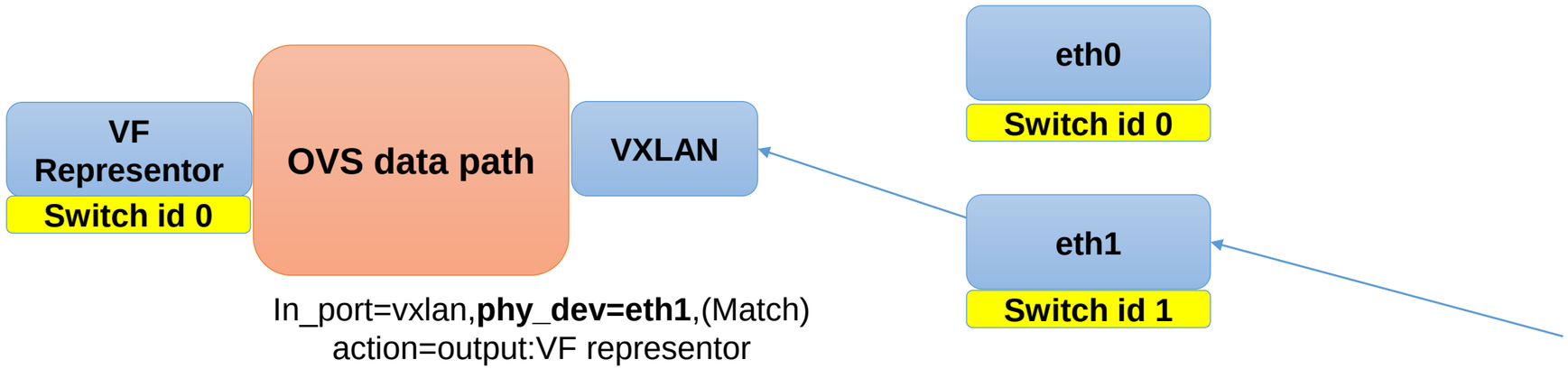
The software does encapsulation in two stages. First it adds the later 3 information, and then it does neighbor look up to add layer 2 information.



For hardware this is not necessarily the case. Do we want to hide in in the Driver or have an intermediate layer that will ask the driver two offload

The rule only when all the information is available?

Stronger matching in OVS DP



Stronger matching in OVS DP