P4 Applied to a vSwitch Data Plane

Dan Daly

Intel Corp

June 4, 2015

How is a vSwitch Defined?

- literal: *Virtual Switch*, like a nexus 1000v
- de facto: Open vSwitch
- marketing: uses OpenFlow, enables SDN/NFV
- Kitchen sink: switches, routes, filters, tracks
  connections, ACLs, VPN,
  load balancer, snort, etc.

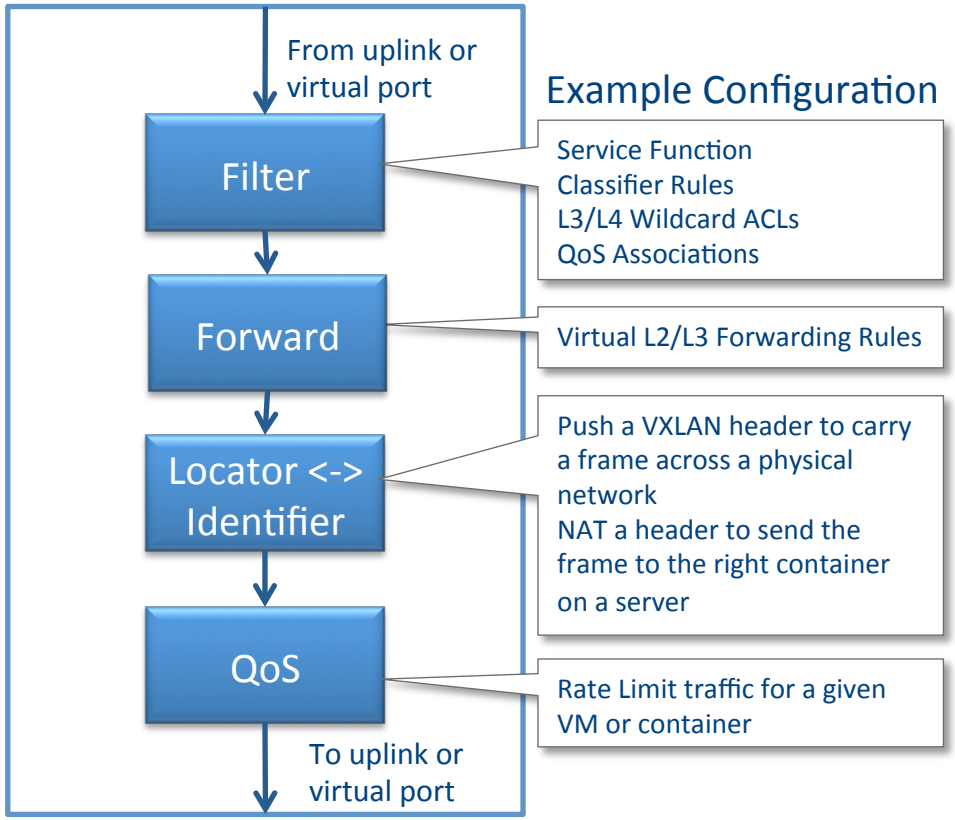
Why the question?

- Define the functionality of the vSwitch
- Optimize the vSwitch Data Plane on the server

- Problem Statement:
  - Virtualize the entire network
  - Virtual L2 and L3 domains
  - Virtual network services
  - High throughput, low latency & jitter

Abstract Virtual Switch Data Plane

- Requirements:
  - Virtual Forwarding
  - Locator/Identifier Separation
  - Filtering
  - QoS



Virtual Forwarding

- Locator: Marked on the packet to say where it should go
- Domain: Marked on the packet or in metadata as additional context
- Destination: a port, into a tunnel, a NAT, or into another pipeline (trap case)

{Locator, Domain} -> Destination

Locator/Identifier Separation

- Locator (Virtual) separate from Identifier (Physical)
- LISP, VXLAN, NVGRE, do this
- Address Translation can also do this

{Locator, Domain} -> Add(Identifier)

{Locator, Domain} -> Remove(Identifier)

- Irrespective of approach, the vSwitch operates on Locators, physical network can rely on Identifiers

Filtering

```
table filtering {
reads {
        standard_metadata.ingress_port :
                wildcard;
        ethernet.dst_mac : wildcard;
        ethernet.src_mac : wildcard;
        …
        ip.src_ip : wildcard;
        ip.dst_ip : wildcard;
        l4.src_port : wildcard;
        l4.dst_port : wildcard;
        tcp.flags : wildcard;
        vxlan.vni : wildcard;
        nsh.nsp : wildcard;
        nsh.nsi : wildcard;
}
```

Enable simple state tracking

Match on metadata tags for service chaining / forward graphs

```
actions {
        count;
        drop;
        set_egress_port;
        set_tunnel_id;
        decap;
        route_via_ecmp(group);
        set.vlan1;
        push_vlan(tpid, tci);
        pop_vlan;
        mirror,
        sample(probability, actions);
    }
}
```

QoS

- Simplest Example: Each virtual port is given a virtual speed, so that each VM or container cannot exceed its virtual allotment

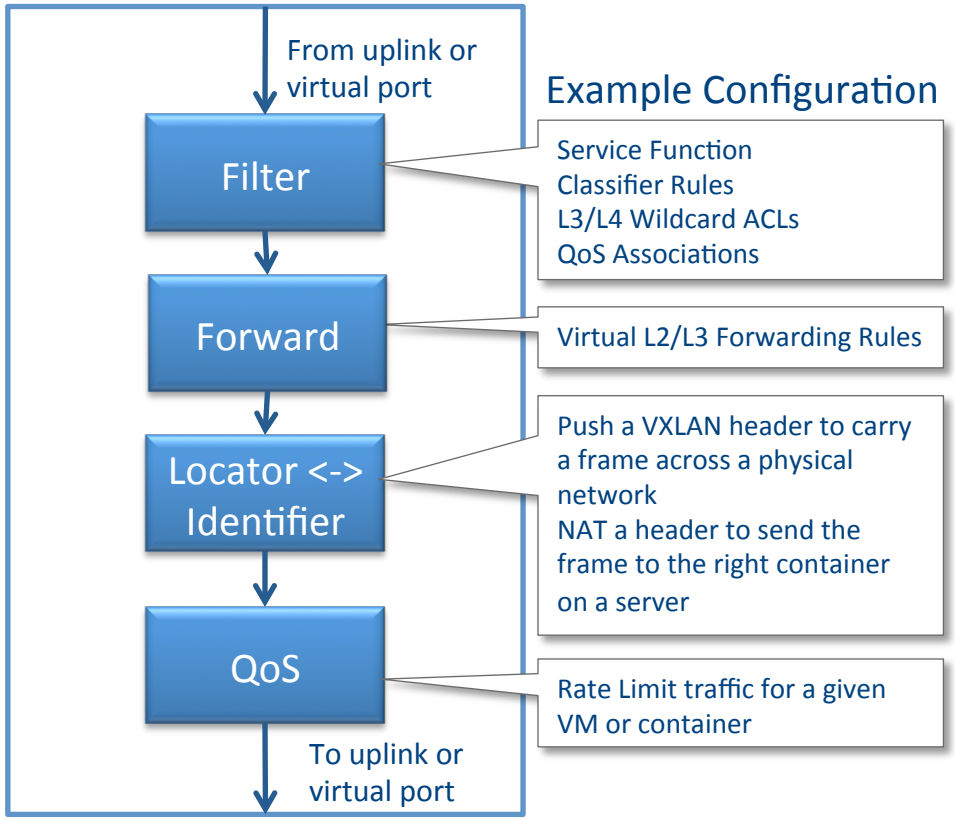
```
attribute {
   type { configuration, per-port }
   name { port.limit.bandwidth}
   description { "Set a bandwidth limit for the
given port.  Note that the platform may restrict
or even change the set of possible
configurations" }
   range { 1e6..100e9}
}
```

P4 Definition of a vSwitch Data Plane

- Concisely Defines:
  - Match Conditions
  - Actions Supported
  - Tables
  - Configuration
  - Headers
- Specifies application intent

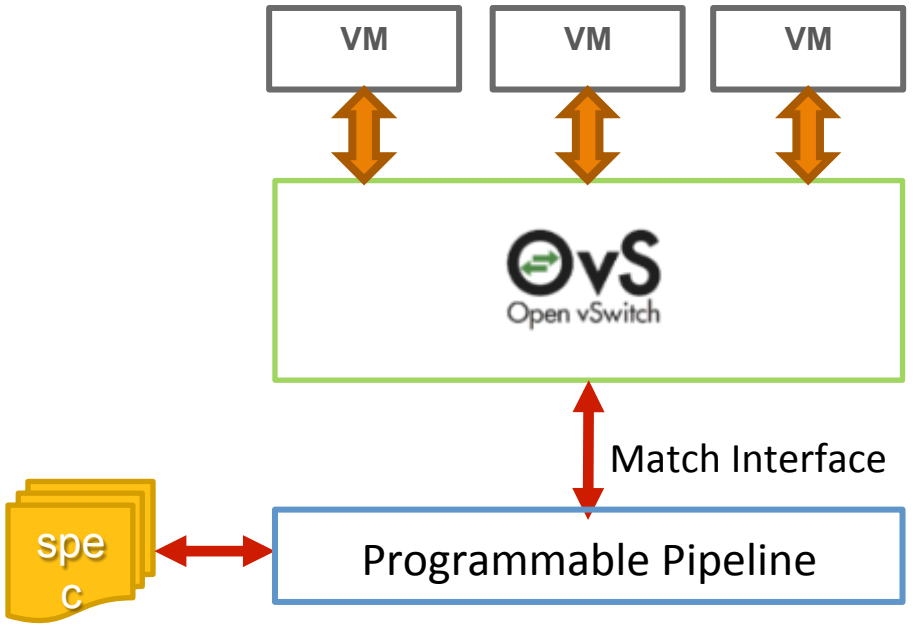
  "I want to filter on X"

  "My tunnel looks like Y"



OvS Integration

- Adding a programmable data plane under OvS
- Similar to how the kernel data plane is exposed
- Converts OvS rules into the underlying data plane



https://github.com/match-interface

Summary

- P4 can be used to concisely define the requirements for a vSwitch Data Plane
  - What can the vSwitch do? Read the P4
- P4 could also be used to define future requirements
  - Example: Hooks for connection tracking
- Open source project to integrate a programmable pipeline underneath OvS: https://github.com/match-interface