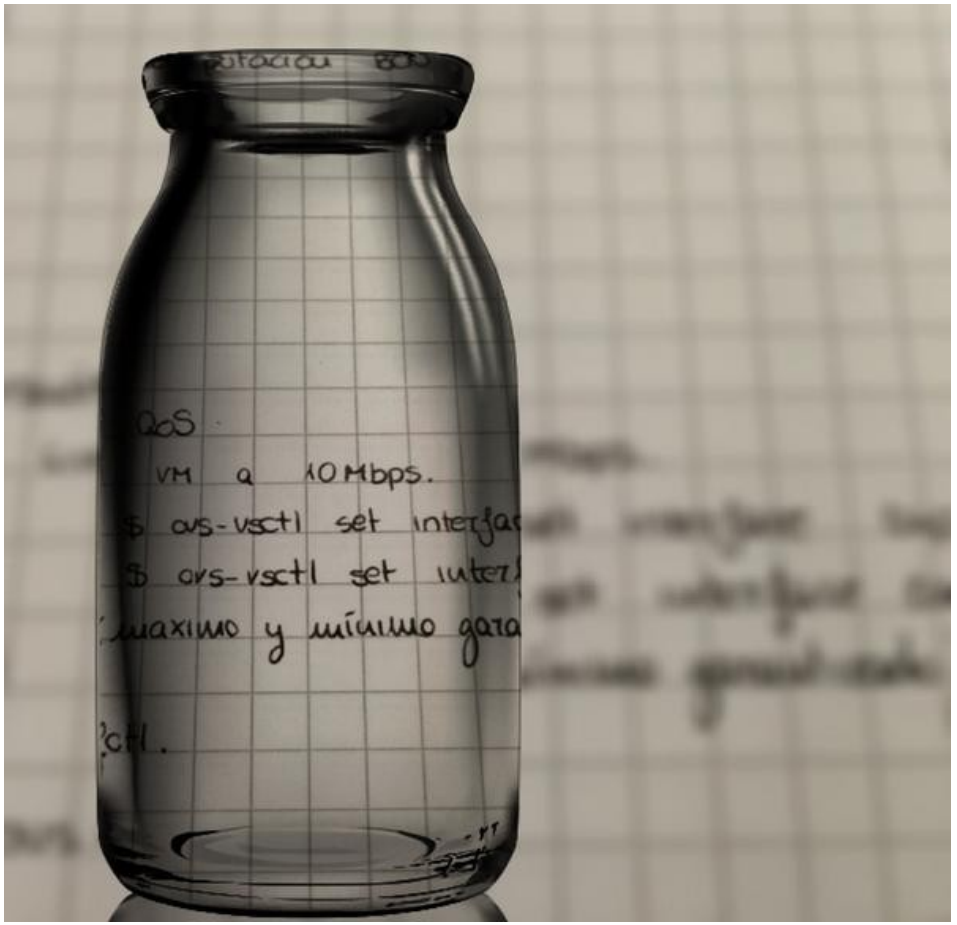
Open vSwitch



Ben Pfaff
Principal Engineer
VMware Networking & Security BU

Preview

- What is Open vSwitch?
- Programming Model
- Example
- What's Next?
- **Questions!**

What is Open vSwitch?

Semi-official description:

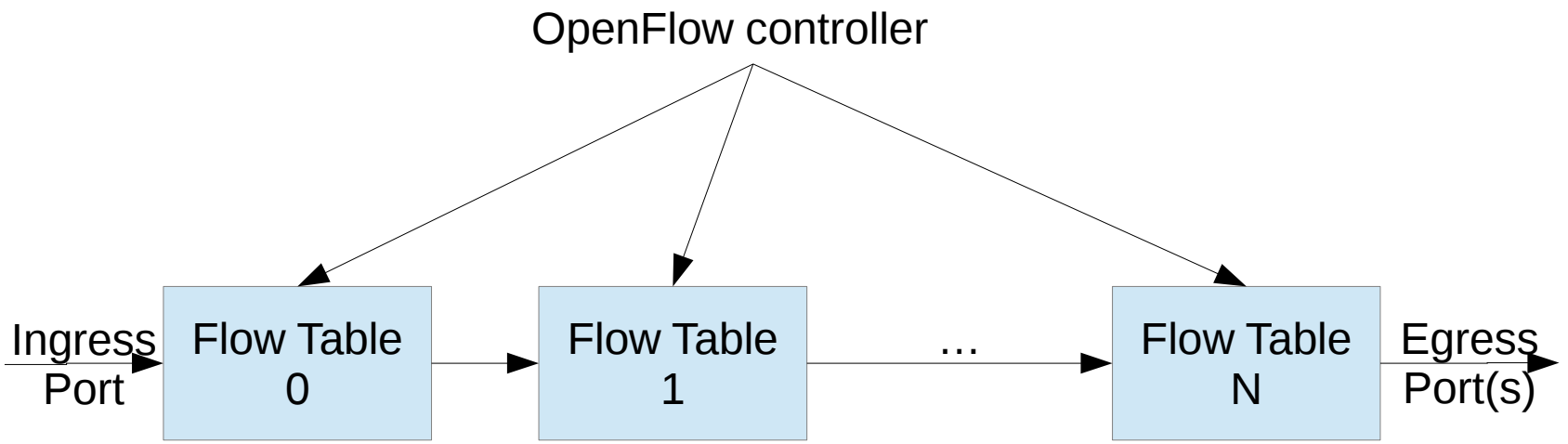
Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (e.g. NetFlow, sFlow, SPAN, RSPAN, CLI, LACP, 802.1ag).

The exciting parts:

- Write a program to control your network.
- Fast! (and getting faster)
- Everywhere! (and getting everyerwhere)

(Incomplete) List of Contributors

Programming Model



Flow Tables

IF a AND b AND ... THEN <actions1>

ELSE IF c AND d AND … THEN <actions2>

ELSE …

Conditions	Actions
Packet header matches	Output to port
Metadata matches:	Modify packet header or metadata
ingress port	Add/remove VLAN, MPLS, etc.
queuing information	Go to or recurse into flow table
tunnel information	Push/pop stack
Bitwise matches	Modify flow table
Range matches*	“Drop”
Set membership matches**	...
...	

priority=55, in_port=1,tcp,tcp_src=80, actions=output:2

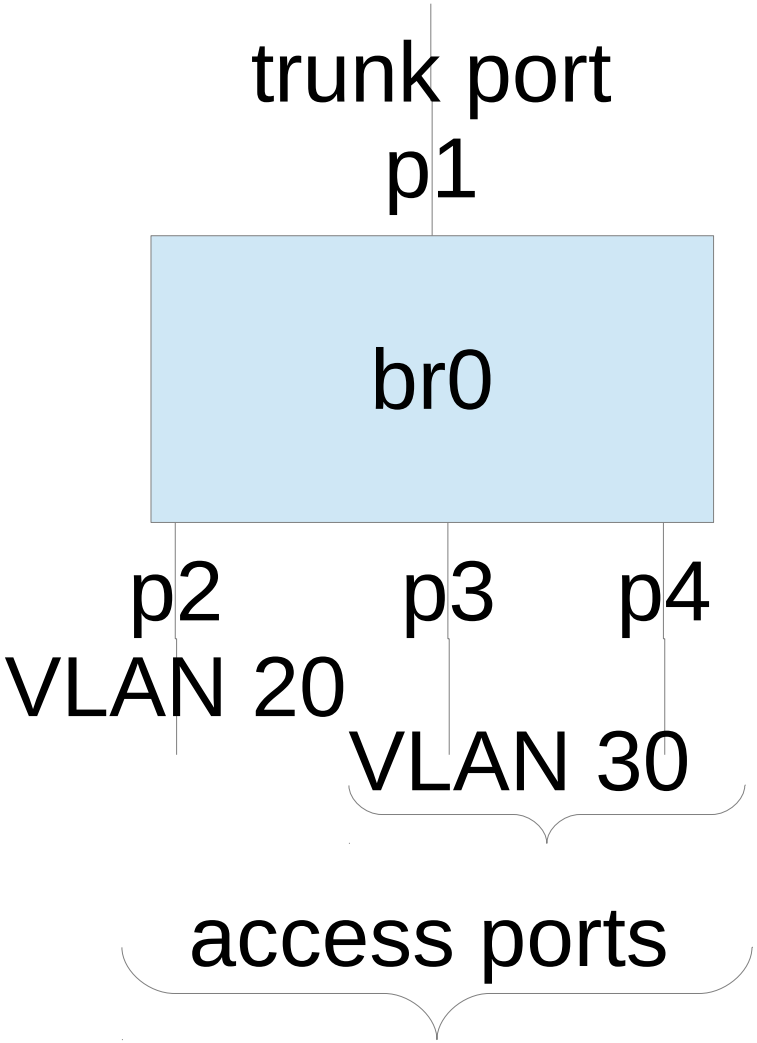
Programming Model Properties

- Somewhat good match for hardware ASICs
- Natural for network admins, unnatural for programmers
- Turing complete (with OVS extensions)
- Easy to implement, hard to optimize
  - (Wasn't obvious it needed optimization.)
- Embarrassingly parallel*

OVS Programming Example

- Notes and test cases in source tree
- The curse of “normal”.
- Normal pipeline:
    0. Admission control (reserved multicast, …).
    1. Ingress VLAN processing
    2. Learn source MAC+VLAN on ingress port.
    3. Look up egress port for dest MAC+VLAN.
    4. Egress VLAN processing.

Example: Test Case



Stage 0: Admission Control

priority=32768,
dl_src=01:00:00:00:00:00/01:00:00:00:00:00,
actions=drop

priority=32768,
dl_dst=01:80:c2:00:00:00/ff:ff:ff:ff:ff:f0, actions=drop

priority=0, actions=resubmit(,1)

Stage 1: Ingress VLAN Processing

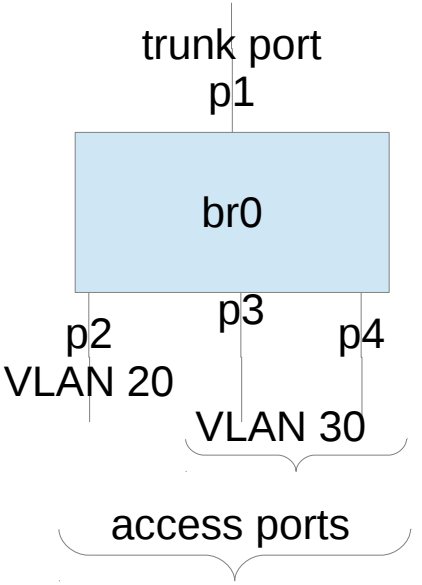
priority=0, actions=drop

priority=99, in_port=1, actions=resubmit(,2)

priority=99, in_port=2, vlan_tci=0,
actions=mod_vlan_vid:20, resubmit(,2)

priority=99, in_port=3, vlan_tci=0,
actions=mod_vlan_vid:30, resubmit(,2)

priority=99, in_port=4, vlan_tci=0,
actions=mod_vlan_vid:30, resubmit(,2)



Stage 2: VLAN+MAC learning

```
actions=
learn(
   table=10,
   NXM_OF_VLAN_TCI[0..11],
   NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[],
   load:NXM_OF_IN_PORT[] ->NXM_NX_REG0[0..15]),
resubmit(,3)
```

Stage 3: Look up egress port

priority=50, actions=resubmit(,10), resubmit(,4)

priority=99,
dl_dst=01:00:00:00:00:00/01:00:00:00:00:00,

actions=resubmit(,4)

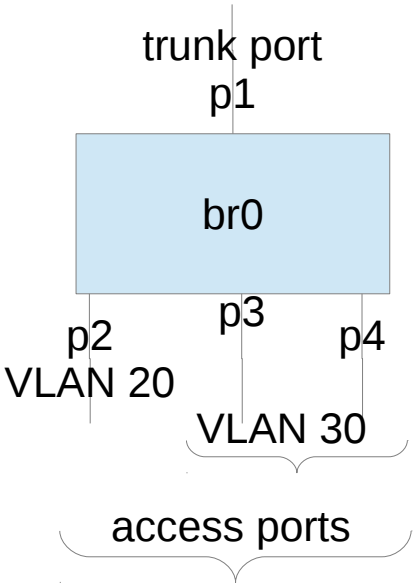
Stage 4: Egress VLAN Processing

reg0=1, actions=1

reg0=2, actions=strip_vlan,2
reg0=3, actions=strip_vlan,3
reg0=4, actions=strip_vlan,4

reg0=0, priority=99, dl_vlan=20, actions=1,strip_vlan,2
reg0=0, priority=99, dl_vlan=30, actions=1,strip_vlan,3,4
reg0=0, priority=50, actions=1

priority=0, actions=drop



What's Next?

- Open vSwitch 2014 Fall Conference: http://openvswitch.org/support/ovscon2014/
- L4: Connection tracking and stateful NAT
- DPDK
- Hardware acceleration
- Containers

Set Matches

IF ip_src ∈ { 10/8, 172.16/12, 192.168/16 }

AND ip_dst ∉ { 10/8, 172.16/12, 192.168/16 }

AND (tcp_src ∈ { 80, 443, 8080 }

OR tcp_dst ∈ { 80, 443, 8080 })

$$3 \times 3^* \times 3 \times 3 = 81$$

vs.

$$3 + 3^* + 3 + 3 = 12$$

Lessons Learned

- The best real packet classifiers are not found in papers.
- "I've rewritten the *!@# datapath N times, I'm not going to let iteration N+1 stop me."
- Standardization does not fix warts.
- Reactive programming is seductive, but doesn't scale.
- In-band control is hard.
- Guessing about performance is risky.

Questions?


WWW.OVSFEST.HU
ONLINEVIDEOSTARS
OVS industry
CHINO 29
VENDREDI 25 OCTOBRE
O.V.S
OVS
OVS industry
CARGO 39