SDN Tutorial

Dean Pemberton – NSRC

UNIVERSITY OF OREGON

REANNZ

PICA8 OPEN NETWORKING

IEEE

catalyst

VICTORIA UNIVERSITY OF WELLINGTON

NSRC
Network Startup Resource Center

Who am I

- Dean Pemberton
  - **NSRC**
    - Trainer/Network Engineer
  - **Victoria University of Wellington**
    - SDN Research Associate
  - **InternetNZ**
    - Technical Policy Advisor

You probably have questions

- What is SDN?
- What's wrong with the network I have now?
- What can an SDN do?

Software Defined Networking is…

- The stupidest name ever invented.

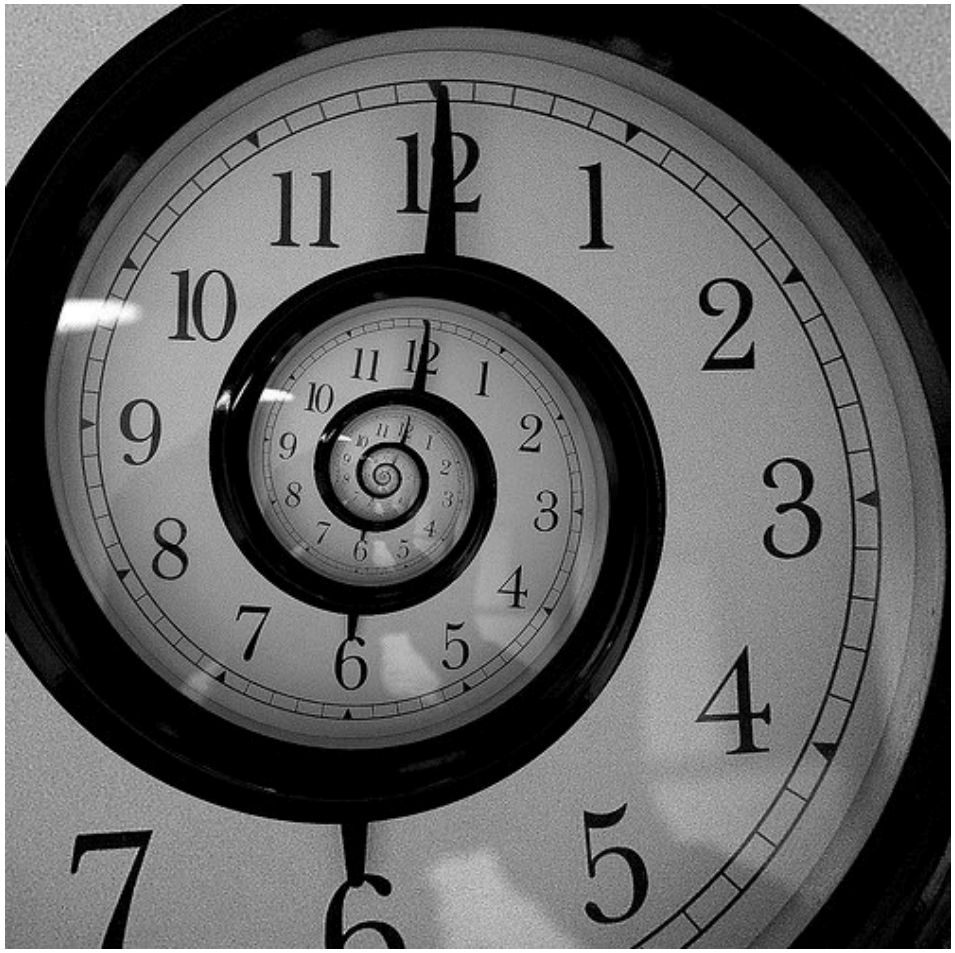
Software Defined Networking is…

- SDN allows network administrators to manage network services through abstraction of lower level functionality.
- This is done by decoupling the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the selected destination (the data plane).

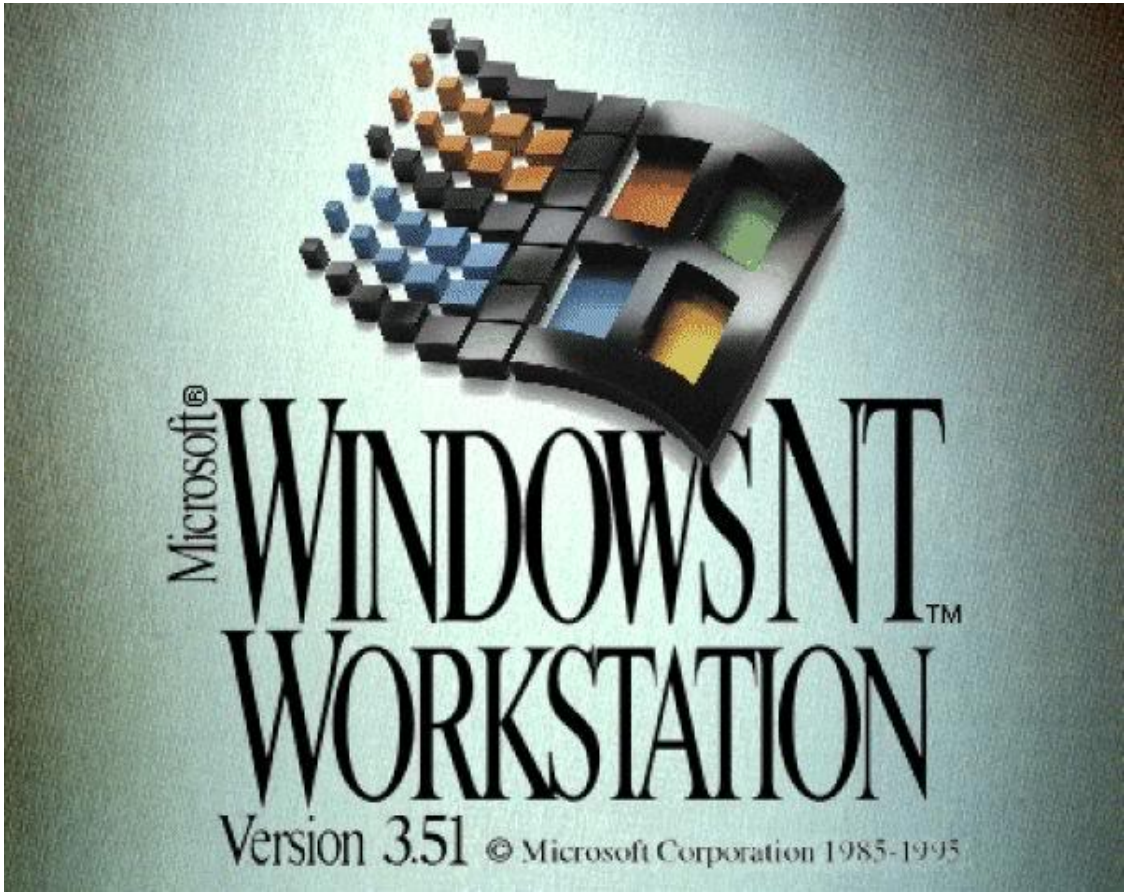
Software Defined Networking

- You've probably had Software Defined Networking for years?
- Anyone own a Juniper M-Series?
- It was just that you were never allowed to define or control the software.

Lets go back in time



Remember this…



Remember when…

- If the features you wanted were supplied by the operating system you were in luck.
- =)
- If the features you wanted were not supplied by the operating system, there were limited opportunities to expand it to include those features.
- =(

Enter choice

End User Innovation

- With Open Source Operating System Software control over the development and deployment of OS features is placed in the hands of the users.
- If you need a feature, even if you are the only one on the planet who wants it, you have a way to develop and deploy it.

A world without…

- Facebook
  - http://www.developer.com/open/article.php/3894566/Inside-Facebooks-Open-Source-Infrastructure.htm
- Google
  - https://developers.google.com/open-source/
- Android
- etc.

Now think about current network equipment…

- Do we currently live in a world more like the closed source OS past?
- Or the current OS world where end users can innovate.

Current Network Feature Roadmap

- You have a good idea
- You go to your network vendor and pitch the idea
- Your network vendor asks how many units you're going to buy
- That number is not enough
- Nothing happens regarding your good idea

Current Example

- “Hi Mr Load Balancing Vendor, I’m a ccTLD in a small country, we face a set of unique challenges with regard to managing bandwidth and protecting against DDoS attacks. We own 2 of your units and were wondering if you might be able to develop some features to assist us in these unique challenges”
- *CLICK* brrrrrrrrrrrrrrr

Another Example

- “We are pleased to announce that after months of development the new version of our networking software will support <feature X which you don’t need>. The price for the next software upgrade with be double to re-coup this development cost”

What if we lived in a world where…

- You could start an open source project where people could develop the features you actually needed your platform to support.
- You didn't need to pay for features that you were never going to use.
- You didn't need to worry about bugs in code you were never going to use.

This works today for OSs

- If you need a new extension to Apache/ BIND/MySQL/etc. then you can have someone develop them for you.
- What if you could do the same thing for all the features in your:
  - Switches
  - Routers
  - Load Balancers
  - Firewalls

Software Defined Networking

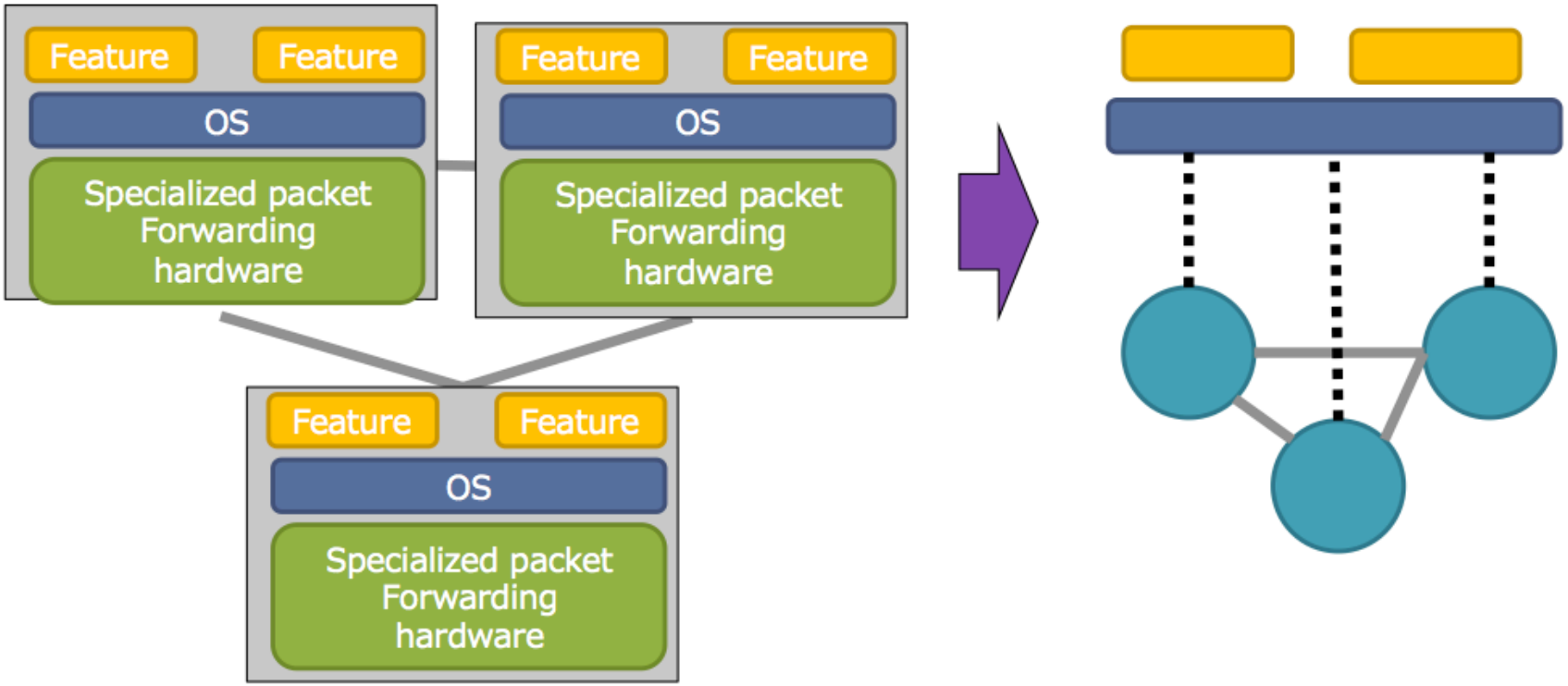
- Allows you to do just that.
- It allows you to take back control of the software that controls your network
- It allows you to drive the speed and direction of the innovation of features within that software.

How?

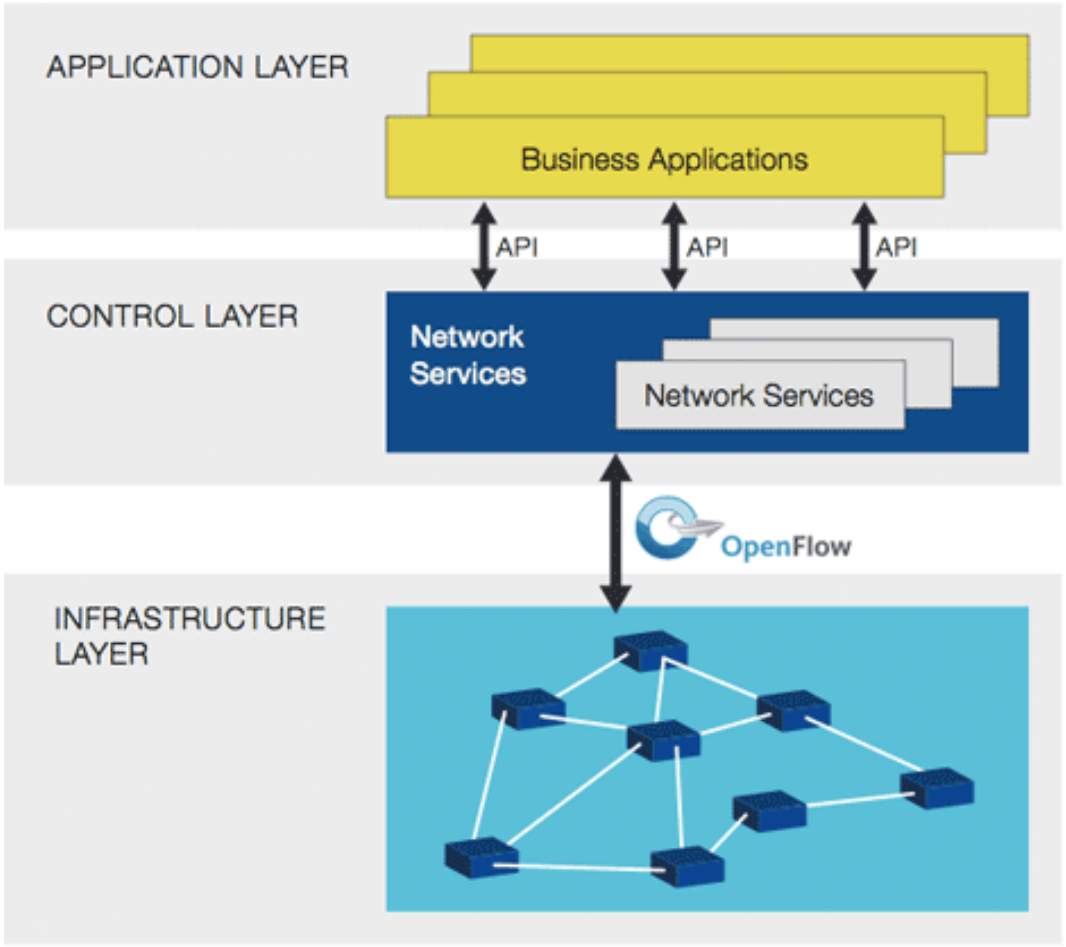
Software defined networking (SDN)

- Separates control and data plane:
  - Open interface between control and data plane (OpenFlow)
  - Network control and management features in software

…SDN


Feature
Feature
OS
Specialized packet Forwarding hardware
Feature
Feature
OS
Specialized packet Forwarding hardware
Feature
Feature
OS
Specialized packet Forwarding hardware

Linton 3 Layer Model

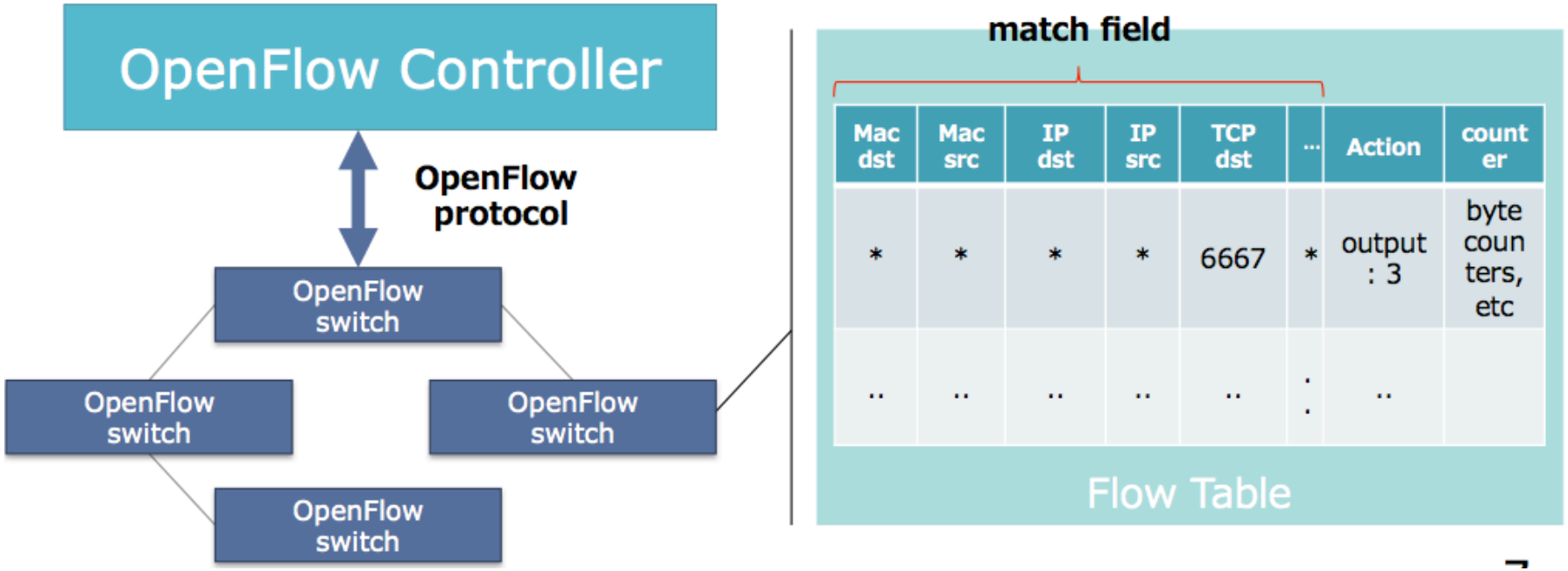


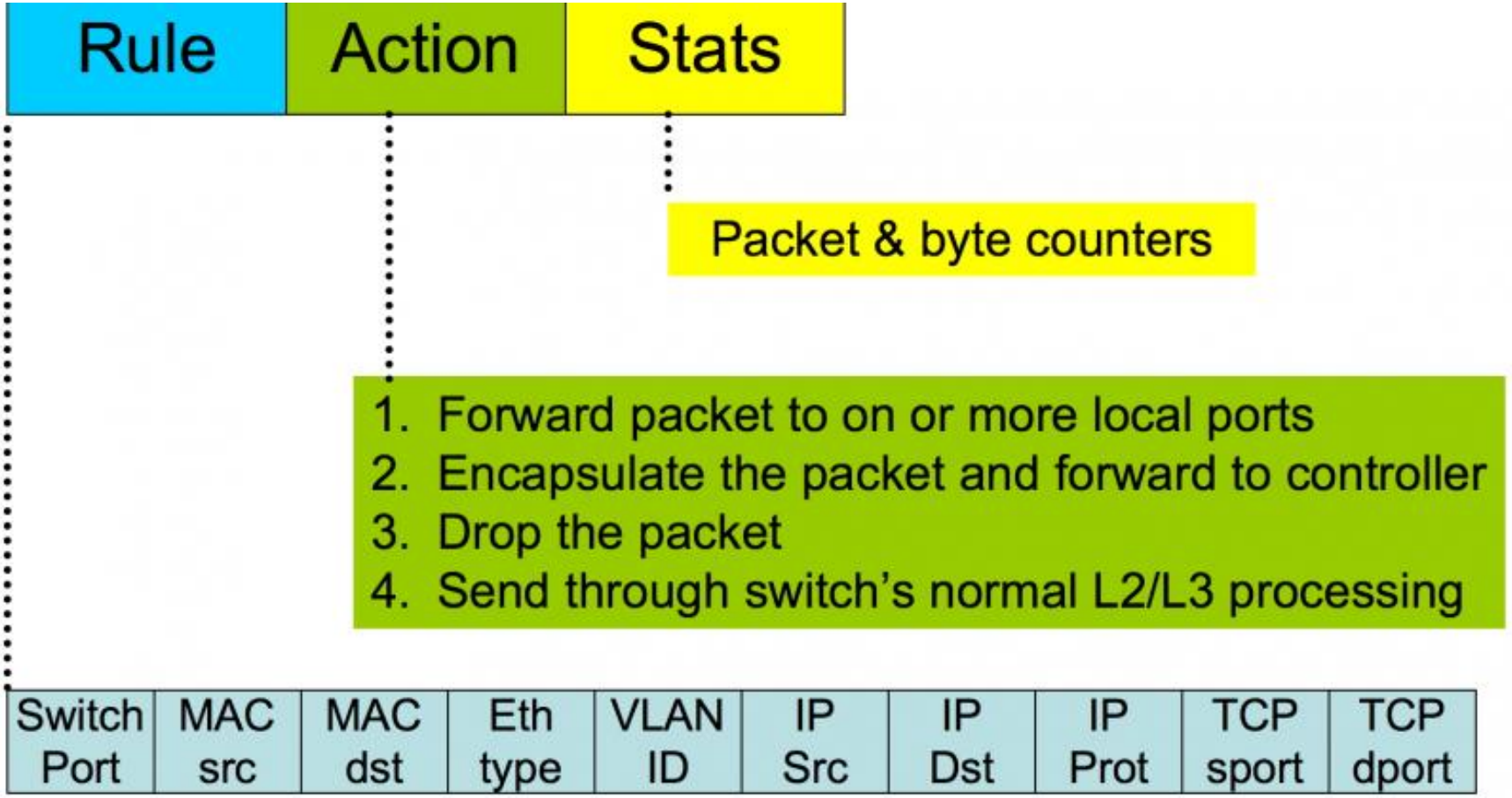
Lessons from history ☺

- "If you know what you're doing, 3 layers is enough; if you don't, 17 layers won't help you.”
- [B]eware of the panacea peddlers: just because you wind up naked doesn't make you an emperor.
  - Michael A Padlipsky

Openflow overview

- One of the key technologies to realize SDN
- Open interface between control and data plane




Rule
Action
Stats
Packet & byte counters
1. Forward packet to on or more local ports
2. Encapsulate the packet and forward to controller
3. Drop the packet
4. Send through switch's normal L2/L3 processing
Switch Port
MAC src
MAC dst
Eth type
VLAN ID
IP Src
IP Dst
IP Prot
TCP sport
TCP dport

Flow Rule Examples

	Input Port	Source MAC	Dest MAC	Ether Type	VLAN ID	Source IP@	Dest IP@	IP Proto	IP SrcPort	IP DstPort
MASKS										
Ethernet Switching	*	*	12:2E	*	*	*	*	*	*	*
IP Routing	*	*	*	*	*	*	1.2.3.4	*	*	*
App Firewall	*	*	*	*	*	*	*	*	*	443
Flow Switching	Port6	12:2E	17:FF	0800	VLAN7	1.2.3.4	4.3.2.1	06	11317	80
VLAN + App	*	*	*	*	VLAN7	*	*	*	*	80
Port + Ethernet + IP	Port6	12:2E	*	0800	*	*	4.3.2.1	06	*	*

UNIVERSITY OF OREGON

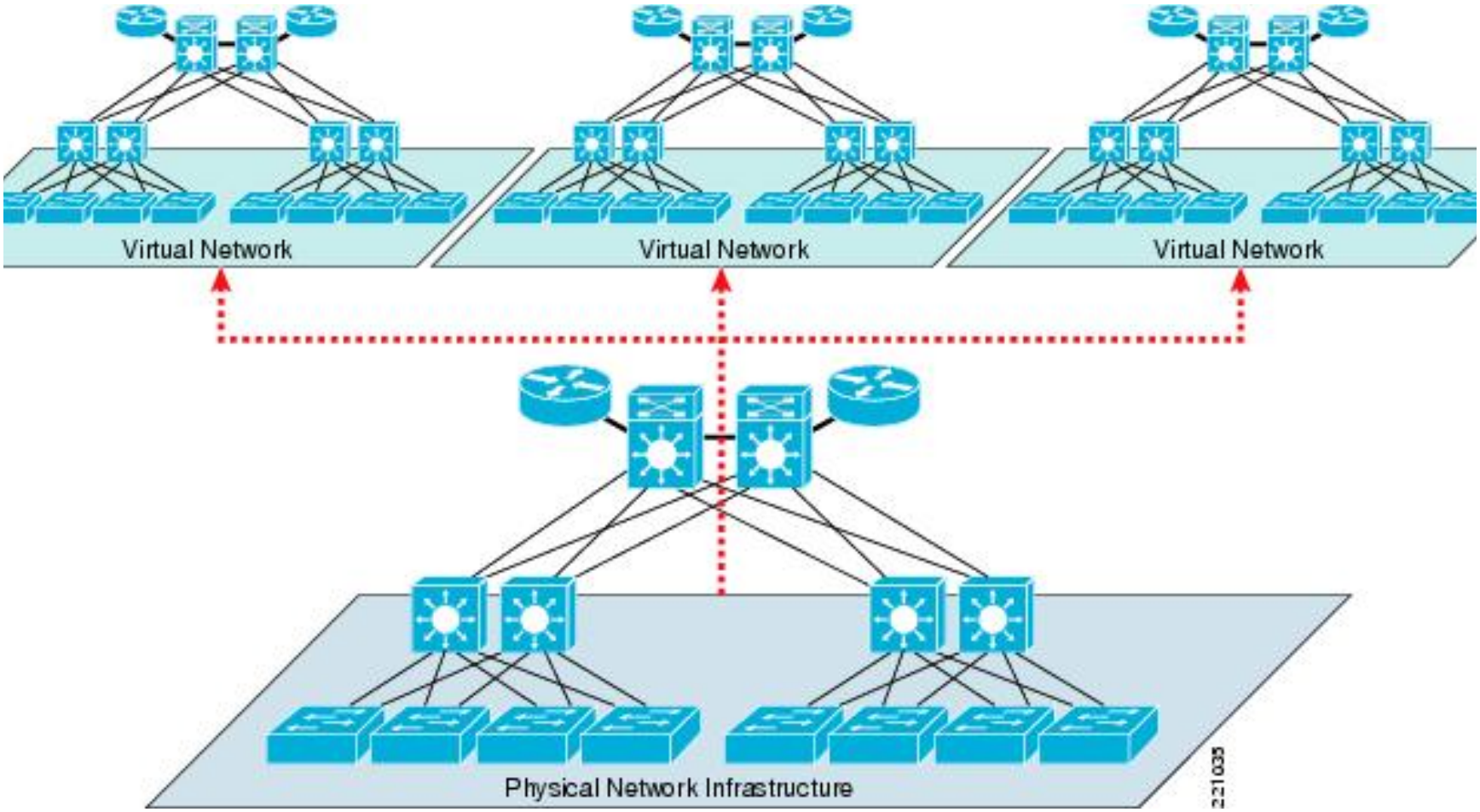
NSRC
Network Startup Resource Center

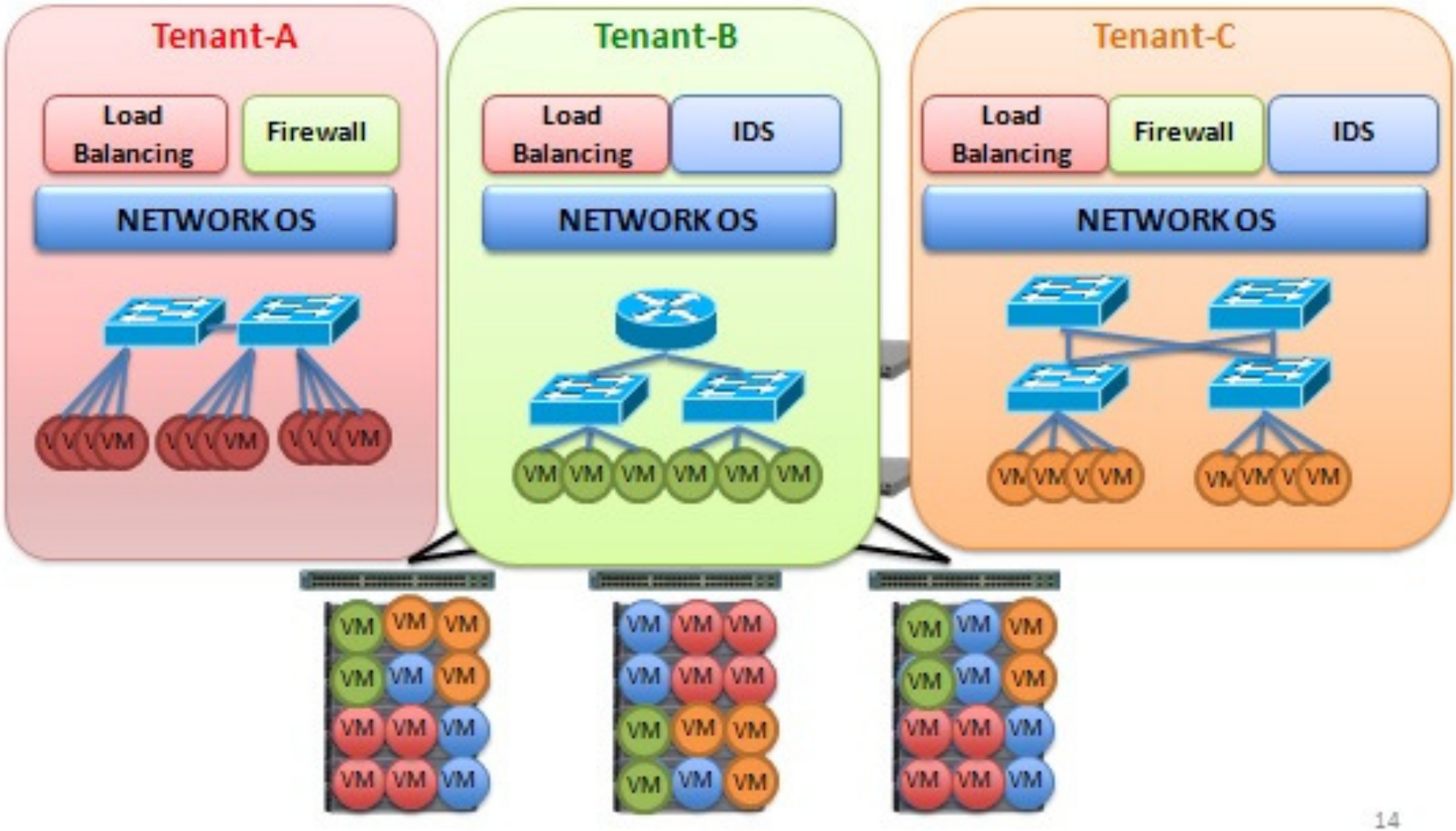
Examples

Layer 2 – Switches

- Network Virtualisation
- Data Centre
- Multi Tennant
- FlowVisor

- Each customer not only gets their own ‘network’ they can control it with their own controller.

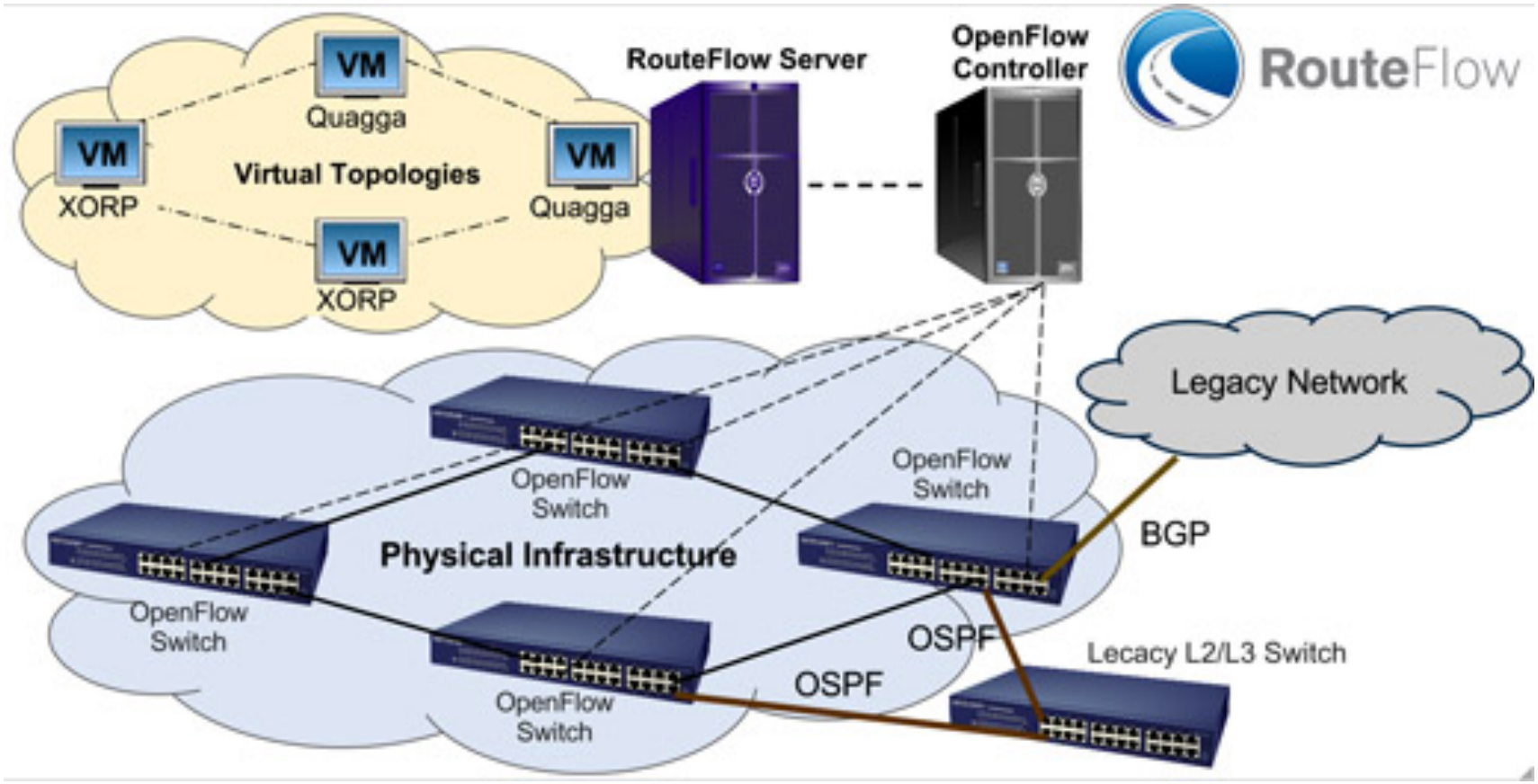

Virtual Network
Virtual Network
Virtual Network
Physical Network Infrastructure
221035


Tenant-A
Load Balancing
Firewall
NETWORK OS
Tenant-B
Load Balancing
IDS
NETWORK OS
Tenant-C
Load Balancing
Firewall
IDS
NETWORK OS
VM

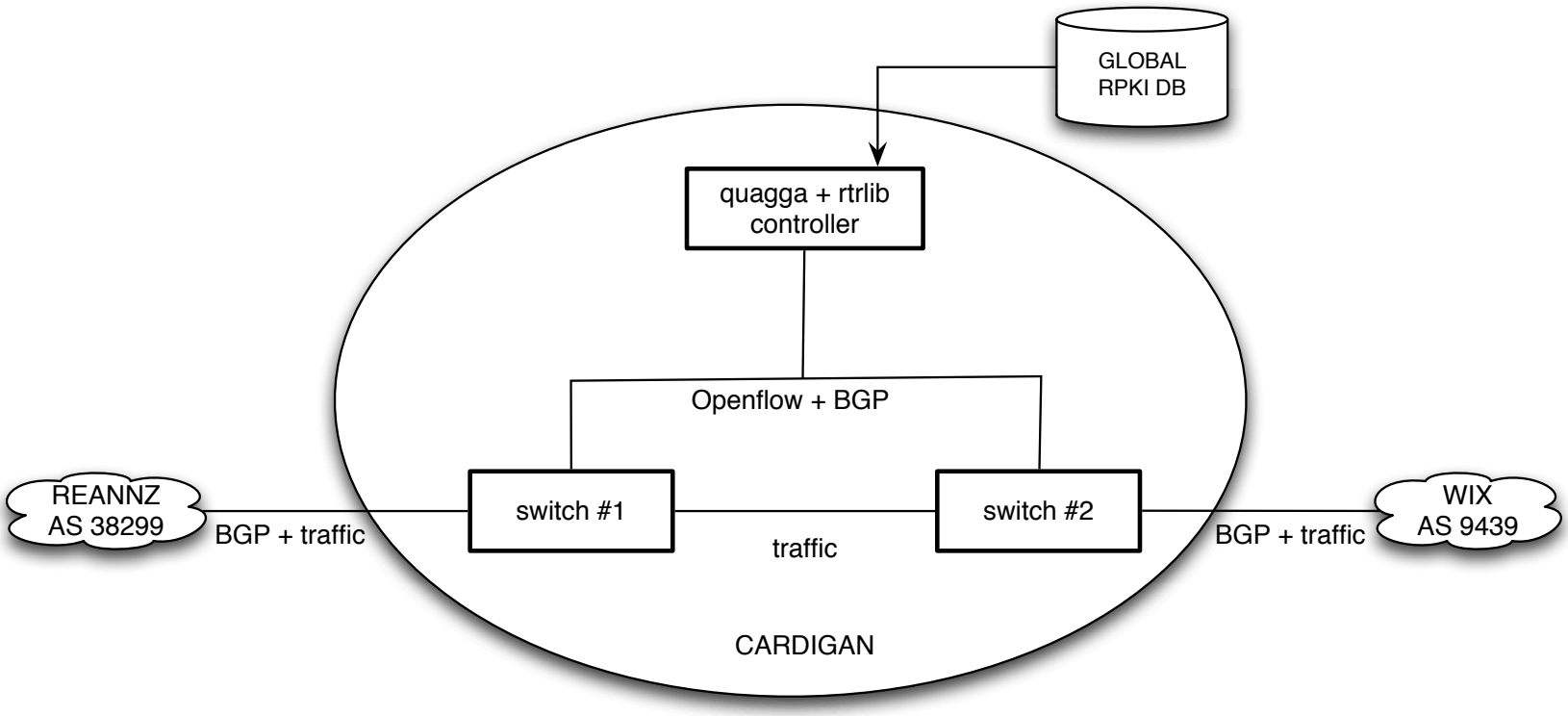
Layer 3 – Routers

- RouteFlow
- What if you were able to take any number of ports throughout you network and draw them together into a router?

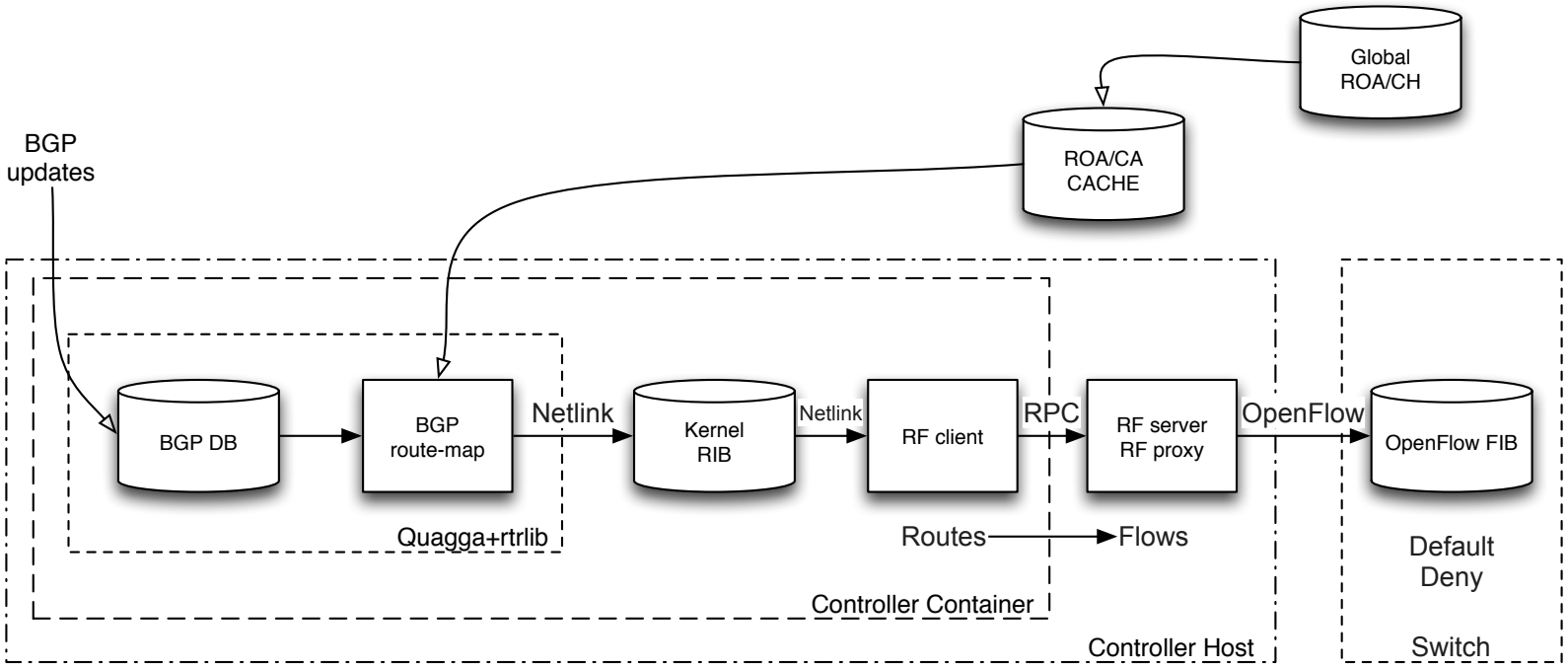
RouteFlow


VM
Quagga
VM
XORP
Virtual Topologies
VM
Quagga
VM
XORP
RouteFlow Server
OpenFlow Controller
RouteFlow
Legacy Network
OpenFlow Switch
OpenFlow Switch
Physical Infrastructure
BGP
OpenFlow Switch
OSPF
Lecacy L2/L3 Switch
OpenFlow Switch
OSPF

Cardigan overview



Cardigan details



Layer 3 – Routers

- Being able to add new features without waiting for vendor support
- RPKI

Layer 4 – Load Balancers

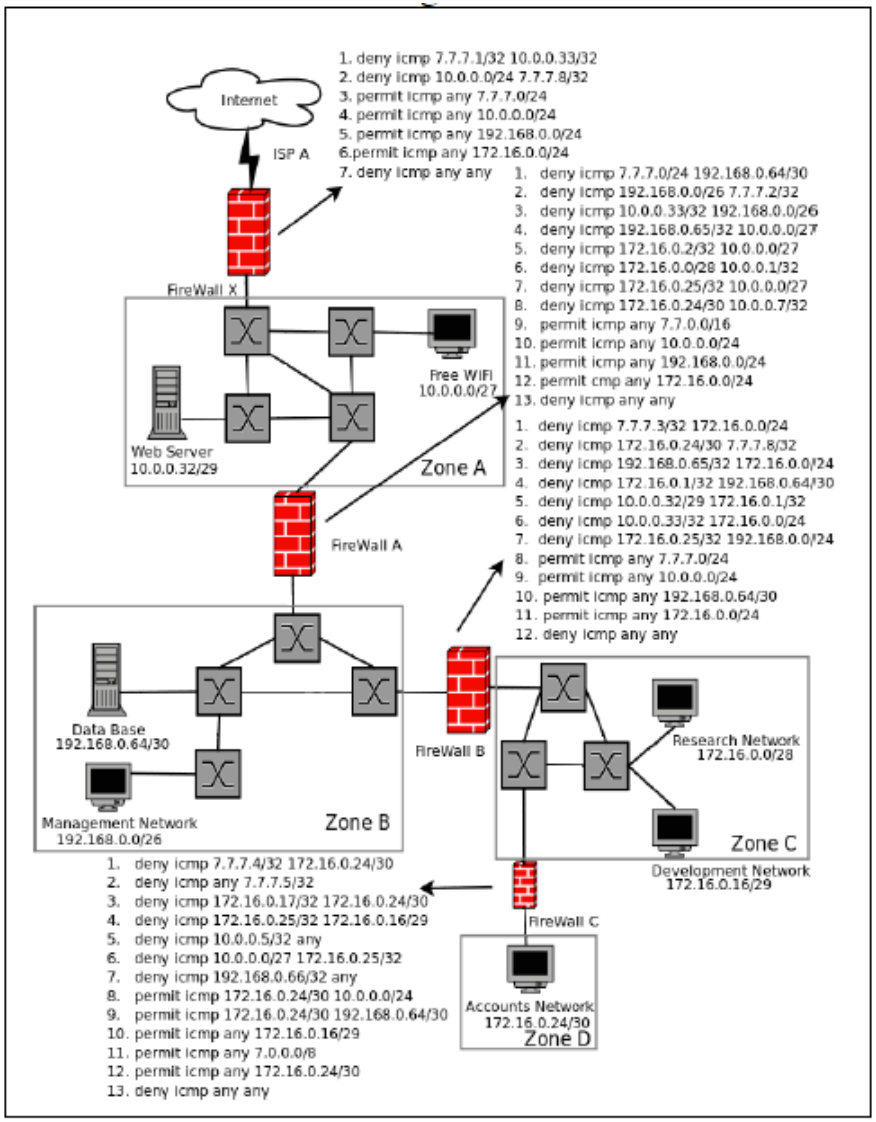
- Load Balancers need to take into account not only complex information about network latency, congestion and performance, but also the load on each of the servers that they are balancing traffic across.
- They also need to know how the balanced application deals with certain situations
- The best person to know that is YOU

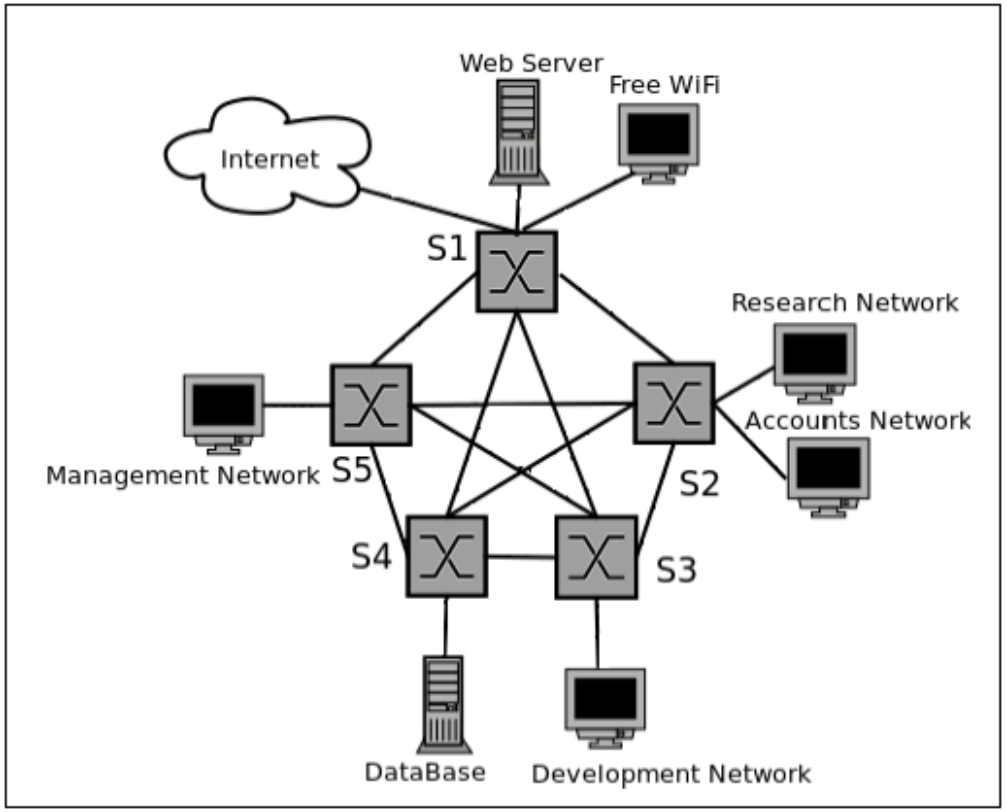
Layer 4 – Load Balancers

- Wang, Richard, Dana Butnariu, and Jennifer Rexford. "OpenFlow-based server load balancing gone wild." Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services. USENIX Association, 2011.
- Handigol, Nikhil, et al. "Plug-n-Serve: Load-balancing web traffic using OpenFlow." ACM SIGCOMM Demo (2009).
- Koerner, Marc, and Odej Kao. "Multiple service load-balancing with OpenFlow." High Performance Switching and Routing (HPSR), 2012 IEEE 13th International Conference on. IEEE, 2012.

Layer 4+ - Firewalls

- We install firewalls everywhere
- They are expensive
- What if we could somehow virtualise them and deploy them only where needed.


1. deny icmp 7.7.7.1/32 10.0.0.33/32
2. deny icmp 10.0.0.0/24 7.7.7.8/32
3. permit icmp any 7.7.7.0/24
4. permit icmp any 10.0.0.0/24
5. permit icmp any 192.168.0.0/24
6.permit icmp any 172.16.0.0/24
7. deny icmp any any
Internet
ISP A
FireWall X
Free WiFi
10.0.0.0/27
Web Server
10.0.0.32/29
Zone A
1. deny icmp 7.7.7.0/24 192.168.0.64/30
2. deny icmp 192.168.0.0/26 7.7.7.2/32
3. deny icmp 10.0.0.33/32 192.168.0.0/26
4. deny icmp 192.168.0.65/32 10.0.0.0/27
5. deny icmp 172.16.0.2/32 10.0.0.0/27
6. deny icmp 172.16.0.0/28 10.0.0.1/32
7. deny icmp 172.16.0.25/32 10.0.0.0/27
8. deny icmp 172.16.0.24/30 10.0.0.7/32
9. permit icmp any 7.7.0.0/16
10. permit icmp any 10.0.0.0/24
11. permit icmp any 192.168.0.0/24
12. permit cmp any 172.16.0.0/24
13. deny icmp any any
FireWall A
1. deny icmp 7.7.7.3/32 172.16.0.0/24
2. deny icmp 172.16.0.24/30 7.7.7.8/32
3. deny icmp 192.168.0.65/32 172.16.0.0/24
4. deny icmp 172.16.0.1/32 192.168.0.64/30
5. deny icmp 10.0.0.32/29 172.16.0.1/32
6. deny icmp 10.0.0.33/32 172.16.0.0/24
7. deny icmp 172.16.0.25/32 192.168.0.0/24
8. permit icmp any 7.7.7.0/24
9. permit icmp any 10.0.0.0/24
10. permit icmp any 192.168.0.64/30
11. permit icmp any 172.16.0.0/24
12. deny icmp any any
Data Base
192.168.0.64/30
Management Network
192.168.0.0/26
Zone B
FireWall B
Research Network
172.16.0.0/28
Zone C
Development Network
172.16.0.16/29
1. deny icmp 7.7.7.4/32 172.16.0.24/30
2. deny icmp any 7.7.7.5/32
3. deny icmp 172.16.0.17/32 172.16.0.24/30
4. deny icmp 172.16.0.25/32 172.16.0.16/29
5. deny icmp 10.0.0.5/32 any
6. deny icmp 10.0.0.0/27 172.16.0.25/32
7. deny icmp 192.168.0.66/32 any
8. permit icmp 172.16.0.24/30 10.0.0.0/24
9. permit icmp 172.16.0.24/30 192.168.0.64/30
10. permit icmp any 172.16.0.16/29
11. permit icmp any 7.0.0.0/8
12. permit icmp any 172.16.0.24/30
13. deny icmp any any
FireWall C
Accounts Network
172.16.0.24/30
Zone D


Web Server
Free WiFi
Internet
S1
Research Network
Accounts Network
Management Network
S5
S2
S4
S3
DataBase
Development Network

Layer 4+ - Firewalls

- Porras, Philip, et al. "A security enforcement kernel for OpenFlow networks." Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012.
- Stabler, Greg, et al. "Elastic IP and security groups implementation using OpenFlow." Proceedings of the 6th international workshop on Virtualization Technologies in Distributed Computing Date. ACM, 2012.
- Gamayunov, Dennis, Ivan Platonov, and Ruslan Smeliansky. "Toward Network Access Control With Software-Defined Networking."

Next generation of engineers

- SDN being taught to undergrads in Q3/2014 at VUW

NZNOG 2014 SDN Install Tutorial

- SDN Intro
- Ryu – OpenFlow Controler
- Open vSwitch
- RouteFlow

- Building a L2 Switch
- Building a L3 Router

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

NZNOG SDN Install Tutorial

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

Takeaways

- SDN separates the control of the network from the elements involved in actually forwarding the packets
- This allows us to have a holistic view of the network not available before
- SDN allows you to control the direction and speed on innovation.
- Active area of development
- Watch this space

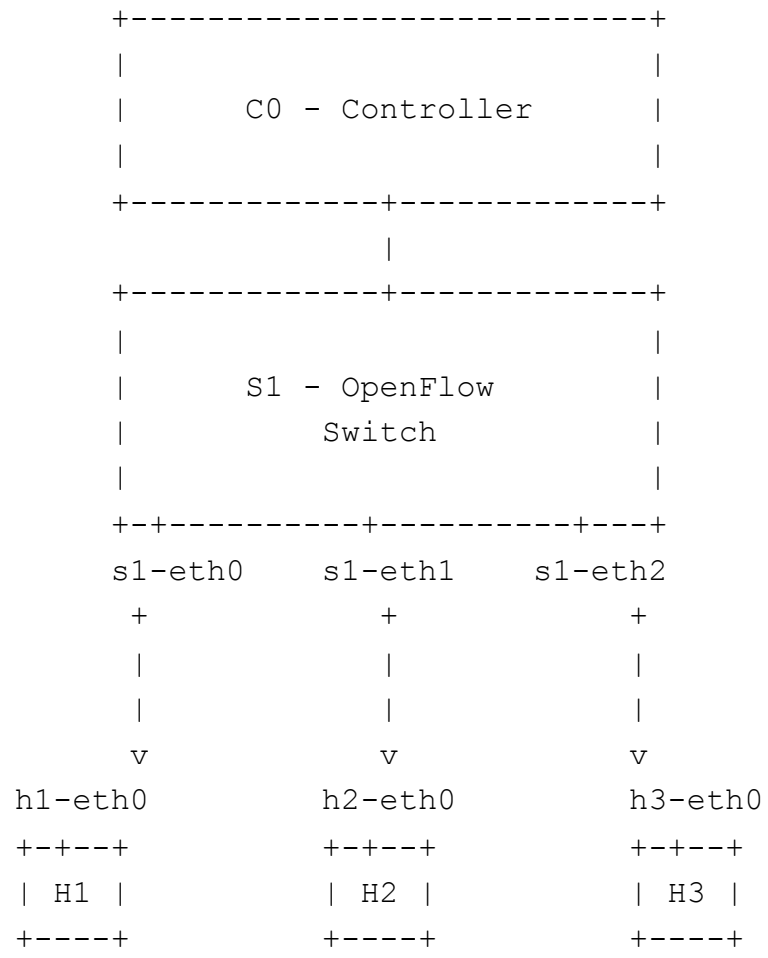
Questions

Do you have any questions?

Hands ON!

- We have VMs for you all

Topology


C0 - Controller
S1 - OpenFlow Switch
s1-eth0
s1-eth1
s1-eth2
h1-eth0
h2-eth0
h3-eth0
H1
H2
H3

Connecting

- Open a terminal window on your machine. If you don't know how to do this ask an instructor for help.
- At the prompt type:
- `ssh mininet@10.10.0.<NUMBER>`

- This IP might be different but you can view it on the VM console

Starting the RYU Openflow controller

- Start the ryu controller with the Simple Switch application

- `# ryu-manager --verbose ./simple_switch_13.py`

Housekeeping

- Make sure that things are in a clean state before we start

- `root@mininet-vm:~# killall controller`
- `root@mininet-vm:~# mn -c`

Become root

- All of the actions in this exercise are done as the root user, so if you are not root already type the following in both windows:

```
mininet@mininet-vm:~$ sudo bash
root@mininet-vm:~#
```

Open Two SSH windows

- We will use two windows for this demo. One for the Control Plane (ryu) and the other for the Data Plane (mininet)

Simple Switch

Create a table called mac_to_port ;

If {packet_in to switch}

{ Parse packet to reveal src and dst MAC addr;

Store in the dictionary the mapping between src_mac and the in_port;

Lookup dst_mac in mac_to_port dict of switch s1 to find next hop;

If { next hop is found}

{ create flow_mod ;

send;

}

else

flood all ports ≠ in_port;

Starting Mininet

- Start mininet with 3 hosts connected to 1 switch

```
# mn --topo=tree,1,3 --mac \
--controller=remote \
--switch ovsk,protocols=OpenFlow13
```

Passing Packets

- mininet> h1 ping h2
- mininet> dpctl dump-flows -O OpenFlow13

Increase Network Size

```
#mn --topo=tree,1,10 --mac \
--controller=remote \
--switch ovsk,protocols=OpenFlow13
```

Running a high bandwidth flow

```
mininet> iperf
```

Questions

Do you have any questions?

Valve

- Layer 2 learning switch with extras
  - OpenFlow 1.3
  - VLANs
  - Configured with YAML
  - Access Control Lists
  - Control multiple datapaths
  - Port statistics

- https://github.com/openvapour/valve

Valve - Motivations

- Just built SDN testbed at Waikato University
- Wanted to bridge it onto a traditional network
- Needed to add VLANs to SDN testbed
- Thought we would test the theory that SDN enables rapid development/prototyping of new features

Valve – Initial version

- 3 people
  - Brad Cowie
  - Joe Stringer
  - Chris Lorier
- 6 hours over two days
- Why so slow?
  - Vendor bugs and work arounds

Valve – Initial version

- OpenFlow 1.0
- Implemented in Ryu
- ~150 lines of code
- Supported tagged/untagged/trunk ports
- Accomplished our goal of bridging networks

Valve – Second version

- We like to run latest firmware on our switches, regular upgrade cycle
- OpenFlow v1.0 quickly became a bad idea
- Needed to add OpenFlow v1.3 support to Valve

Valve – Second version

- Changes between OF1.0 and OF1.3 relevant to valve:
- FlowMod messages now include “instructions”
- Some fields have changed their names
  - **dl_src/dl_dst** ==> **eth_src/eth_dst**
  - **dl_vlan** ==> **vlan_vid**
- VLAN actions have changed their names
  - **OFPActionStripVlan**() ==> **OFPActionPopVlan**()
  - **OFPActionVlanVid**() ==>
  - **OFPActionPushVlan**() + **OFPActionSetField**(**vlan_vid**)

Valve – Second version

- Matching on VLAN ID has changed
- OFPVID_PRESENT bit...

- From OpenFlow 1.3 spec:

OXM field	oxm_value	oxm_mask	Matching packets
absent	-	-	Packets *with* and *without* a VLAN tag
present	OFPVID_NONE	absent	Only packets *without* a VLAN tag
present	OFPVID_PRESENT	OFPVID_PRESENT	Only packets *with* a VLAN tag regardless of its value
present	*value* | OFPVID_PRESENT	absent	Only packets with VLAN tag and VID equal *value*

Table 12: Match combinations for VLAN tags.

Valve – Further improvements

- Access Control Lists
- Default configuration elements
- Multiple Datapaths
- Live reconfiguration

Valve – Third version

- Valve needed to get better at configuration file parsing
- New version includes an Object-orientated design with configuration file handling spread out across the classes
- ~700 lines of code

Valve Configuration

- Try to make every element optional
- Reduces barrier to entry for simple architectures
  - git clone
  - add 2 or 3 lines of config
  - run it

Valve Configuration

- 0000000000000001:
  - 1:
    - type: untagged
    - vlans: [10]
  - 2:
    - type: tagged
    - vlans: [10]

Valve Configuration

- 0000000000000001:
  - default:
    - type: untagged
    - vlans: [10]
  - 1:
    - type: untagged
    - vlans: [10]
    - acls:
      - {match: {eth_type: 0x0800, ip_proto: 7, udp_src: 68}, action: drop}
  - 2: