Not so simple switching

Brad Cowie

Waikato University

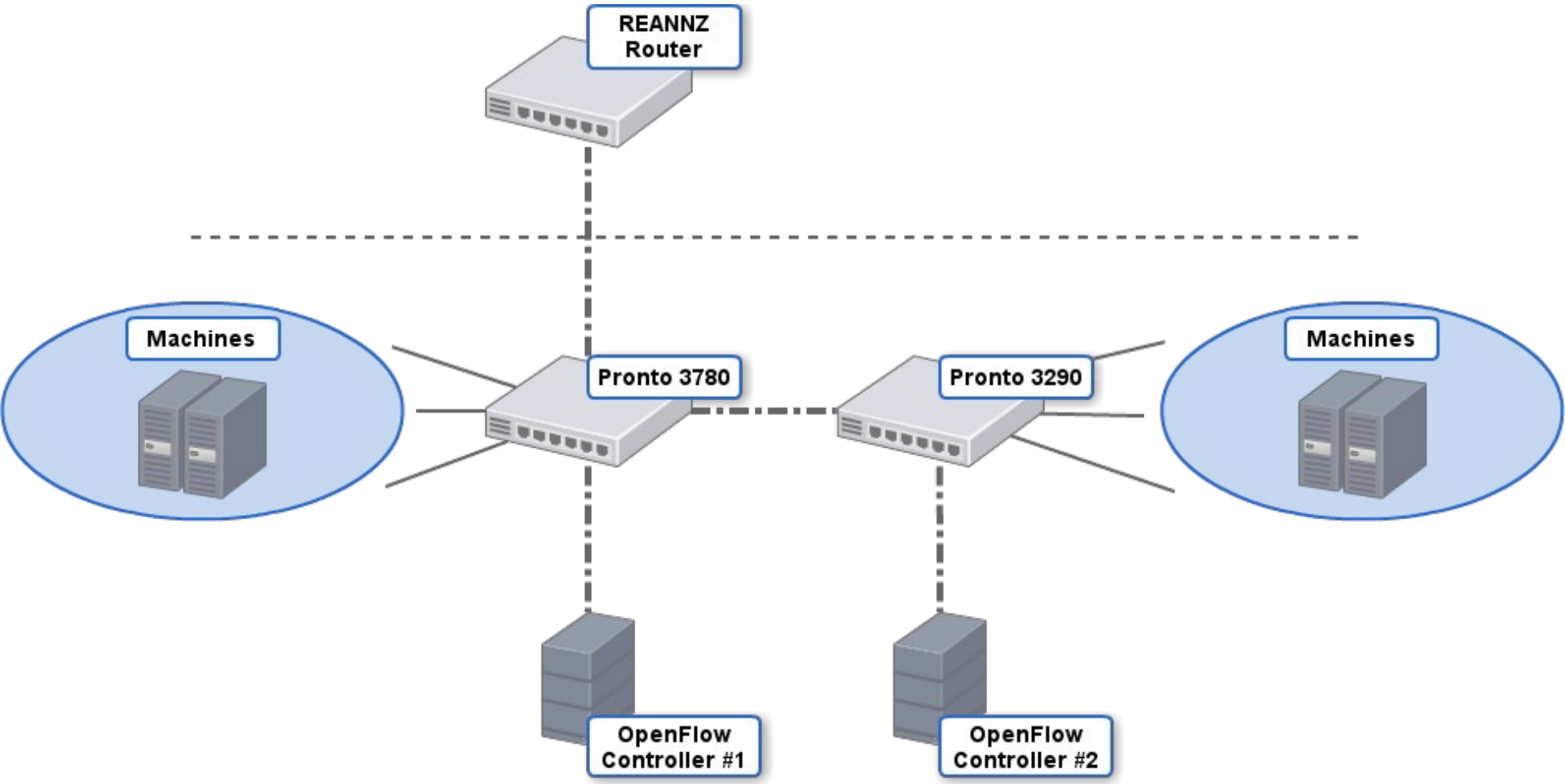
SDN at WAND

- Started with an Honours project in 2012 extending OpenvSwitch to support OpenFlow v1.1
- Our students helped with RouteFlow/Cardigan
- Further Honours/Masters/PhD projects

WAND SDN Testbed

- We like to be hands-on at WAND
- Test our projects and Open Source projects on real hardware at line rate
- Hardware from pica8
  - 1x Pronto 3290 (48x 1gig, 4x 10gig)
  - 1x Pronto 3780 (48x 10gig)
- Has REANNZ as an upstream
- Uses in-band control instead of a separate OpenFlow control network

Testbed Topology



Development Software

- OpenFlow v1.3
- Ryu (Python OpenFlow framework)
- OpenvSwitch (Linux software switch)
- Mininet (Simulation framework)
- Pronto PicOS 2.5 (Network “OS”)

Student Research Projects

- Chris Lorier - Building techniques for failure recovery in a Software Defined Network
  - Building protected paths
  - Fault detection (active vs passive)
- Adam Coxhead - Investigate replacement of STP with SDN
  - Dynamically discover network topology in controller
  - Install rules to keep broadcast traffic from being looped
  - Load balance traffic across redundant Layer 2 paths

Student Research Projects

- Craig Osborne - Investigate building OpenFlow-enabled BNG
- Karthik Sharma - Designing a distributed OpenFlow controller based on Ryu with a cassandra backend for topology and state sharing
- Chris Lorier – Development of a Distributed Router
- Joe Stringer – Contributing to OpenFlow 1.1 support in OpenvSwitch

Projects I work on

- RouteFlow
  - Pushes routes from Linux routing table onto OpenFlow-enabled switches as OpenFlow rules
  - Turns a set of switches into a distributed router
- Valve
  - What we'll be focusing on today

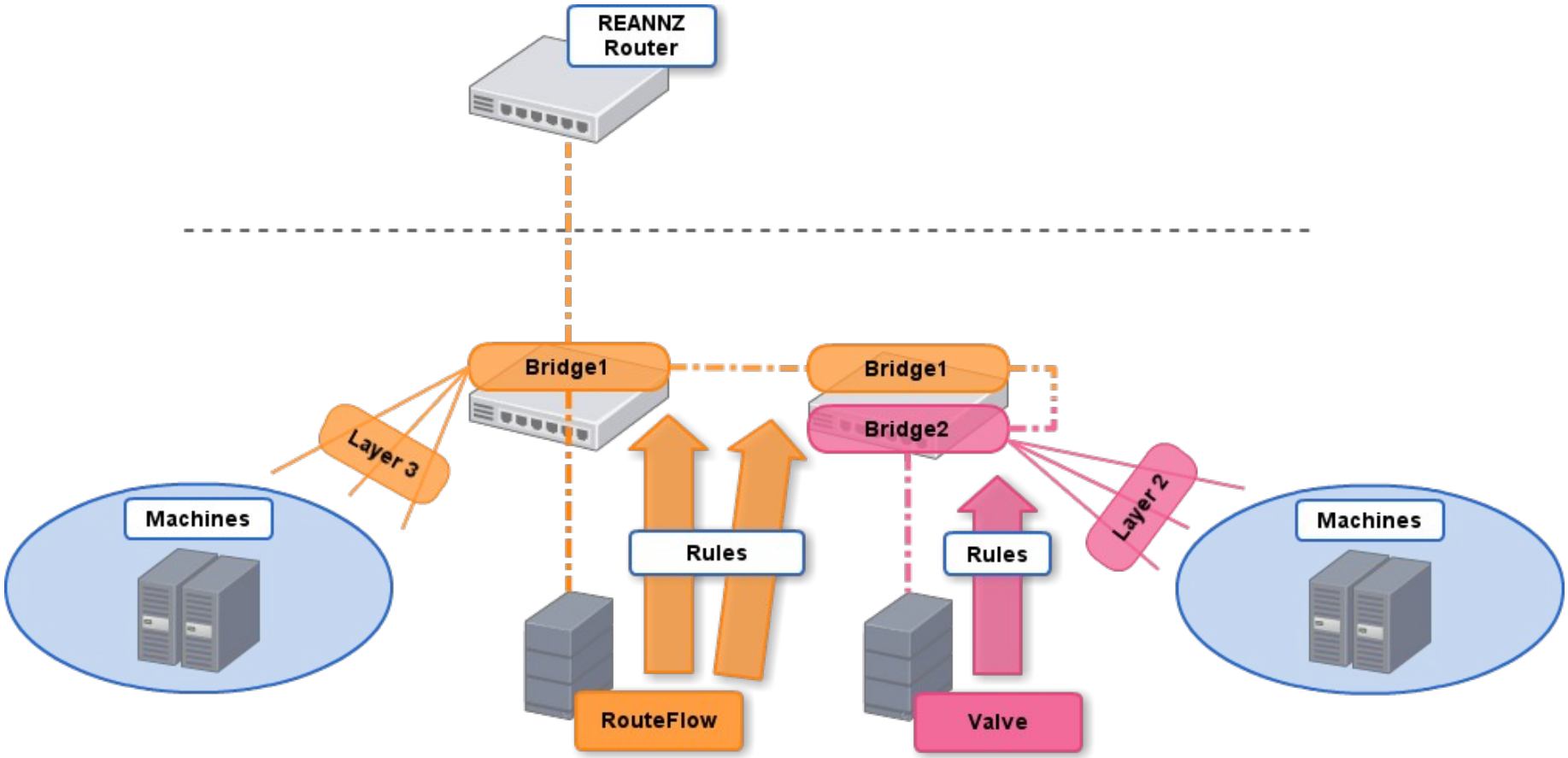
Valve

- “Hello world” application for SDN is a Layer 2 learning switch
- There are a lot of very simple ones
- We wrote another...
  - OpenFlow 1.3
  - VLANs
  - Configured with YAML
  - Access Control Lists
  - Control multiple datapaths
  - Port statistics
- https://github.com/openvapour/valve

Valve - Motivations

- Newly built SDN testbed
- First demonstration was running RouteFlow on testbed
- Wanted to bridge RouteFlow onto a traditional network
- Needed to add VLANs to SDN testbed
- Test the theory that SDN enables rapid development
    - Let's write this in an afternoon!

SDN Testbed Logical Architecture



Valve – Initial version

- 3 people
  - Myself
  - 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
- We made decision to standardise all our projects on OpenFlow v1.3
- 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 becomes **eth**_src/**eth**_dst
    - **dl_vlan** becomes **vlan_vid**
- VLAN actions have changed their names
  - OFPAction**Strip**Vlan() becomes OFPAction**Pop**Vlan()
  - OFPActionVlanVid()

    becomes

    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

- What became clear is that Valve needed to get better at configuration file parsing
- These new features pushed our previous parser (a large set of for loops) to the limits, it became difficult and costly time-wise to implement new features
- 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

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: 67}, action: drop}
    2:
```

Valve Configuration

```
default:
    type: untagged
    vlans: [10]
0000000000000001:
0000000000000002:
```

Valve Structure

```
~/valve$ ls
```

`api.py`	Valve APIs (REST etc)
`dp.py`	Datapath class
`acl.py`	ACL class
`port.py`	Port class
`vlan.py`	VLAN class
`valve.py`	Ryu App
`valve.yaml`	Configuration file

Running Valve

```
~/valve$ ryu-manager valve.py
```

Running Valve

```
~/valve$ ryu-manager valve.py
   loading app valve.py
   loading app ryu.controller.ofp_handler
   instantiating app None of DPSet
   creating context dpset
   creating context wsgi
   instantiating app valve.py of Valve
   instantiating app ryu.controller.ofp_handler of
   OFPHandler
   wsgi starting up on http://0.0.0.0:8080
```

Running Valve

```
~/valve$ ryu-manager valve.py
   INFO    dpid:1      Configuring datapath
   INFO    dpid:1      Configuring vid:10 ports:1
   INFO    dpid:1      Datapath configured
```

Reload configuration

- Edit configuration file
- Send process a SIGHUP:
  - `~$ pkill -SIGHUP -f “ryu-manager valve.py”`

Flow table usage

- On startup:
  - 1 match rule per VLAN
  - 1 match rule per untagged port
  - 1 match rule per ACL
- During runtime:
  - Broadcast and multicast match rule per MAC address
  - 2 unicast match rules per MAC address pair

Flow table usage

- VLAN match rule:
  - priority=9000
  - dl_vlan=**50**
  - actions=CONTROLLER,output:**3**,strip_vlan,output:**1**,output:**2**

- Untagged port match rule:
  - priority=9000
  - in_port=**1**
  - actions=CONTROLLER,output:**1**,output:**2**,
    push_vlan:0x8100,set_field:**50**->vlan_vid,output:**3**

Flow table usage

- Unicast match rules
  - priority=9001
  - in_port=**1**,dl_src=**11:11:11:11:11:11**,dl_dst=**22:22:22:22:22:22**
  - actions=output:**2**

  - priority=9001
  - in_port=**2**,dl_src=**22:22:22:22:22:22**,dl_dst=**11:11:11:11:11:11**
  - actions=output:**1**

Future work?

- REST API
- Statistics
  - Collectd
  - SNMP
- Use topology information
  - Smart proxy ARP
  - Get rid of STP (allow loops on the network)
- Fault detection / correction

SDN Community in NZ

- sdn-nz mailing list
- http://ecs.victoria.ac.nz/mailman/listinfo/sdn-nz

Questions?

- Fork us on github
- We accept pull requests!

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

- Contact me

  Brad Cowie

  brad@waikato.ac.nz