Faucet

– The Open Source Production Quality OpenFlow Switch

Brad Cowie

Applying SDN Principles

- Have now decoupled control plane on network devices from the forwarding plane
- What do we run on the control plane to configure the forwarding plane?
- Are no longer constrained by embedded CPU (ARM, PPC)
- Doesn’t need to run on proprietary OS (VxWorks, etc)

Faucet Introduction

- Open Source OpenFlow v1.3 Switch
- Normal switch features
  - VLANs
  - Inter-VLAN Routing
  - Port statistics (through gauge module)
  - Layer 3 features (BGP, static routing)
  - Flexible ACL rules
    - Filtering
    - Selective port mirroring (only mirror the traffic you want to see)
    - Policy based forwarding (lets us do 802.1x via external system)

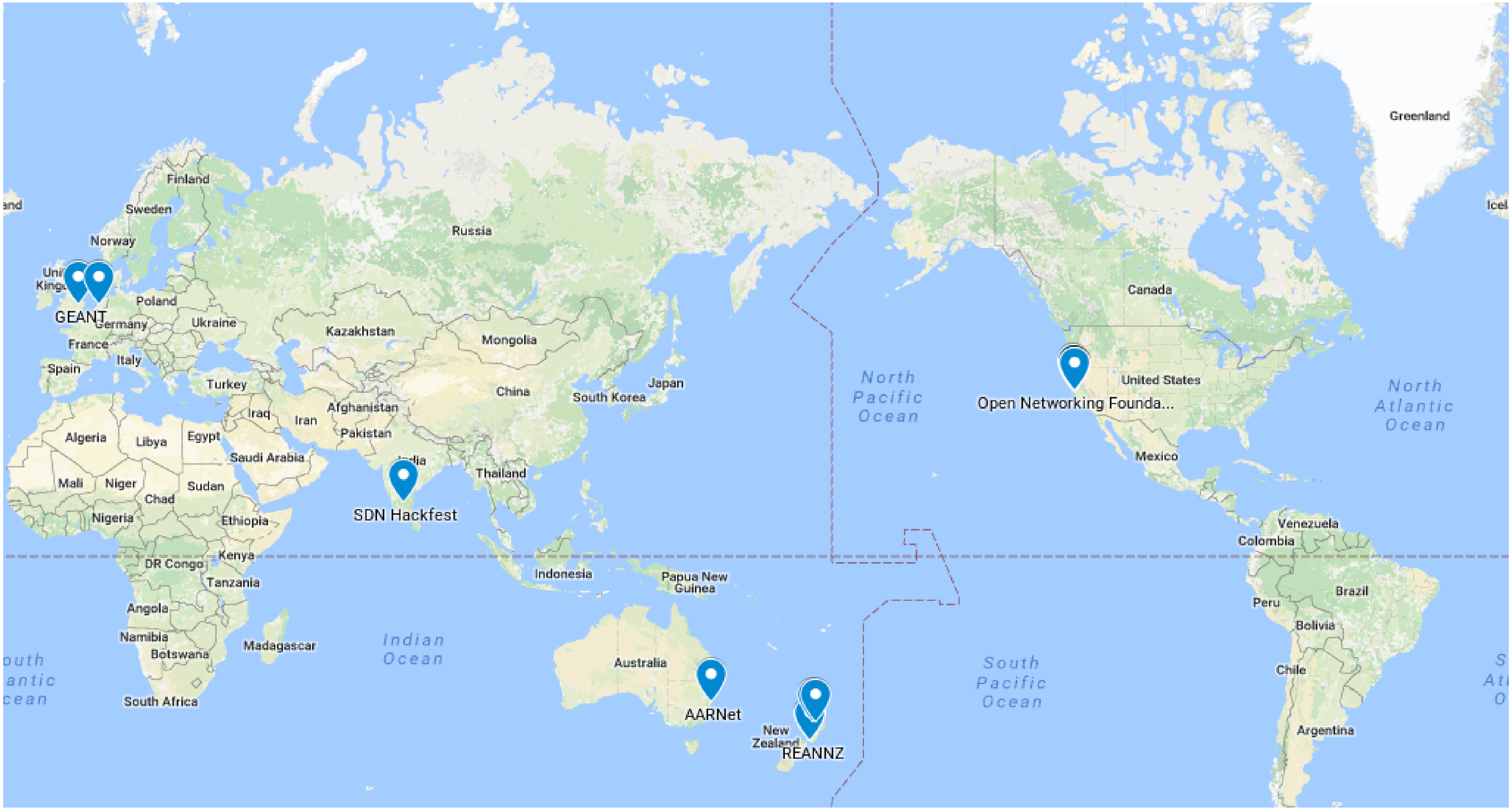
Faucet Introduction

- Follow normal software engineering principles
  - Comprehensive test suite
  - Travis for continuous integration testing
  - Written in Python (PEP8 style), uses Ryu framework
  - Open source on Github (we accept PRs!)

Motivation

- Rapid development lifecycle
  - Coded in Python
  - Parallel test suite runs virtualised in Docker against mininet or real hardware
- Benefits over regular hardware switch
  - Open source – can add your own features!
  - Easy to debug
  - Easy to administrate (YAML config file)
  - Devops can deploy a network like a regular application

Deployments



WAND. REANNZ. Victoria University. ESnet. GÉANT. Allied Telesis …

Faucet Components

- Ryu – OpenFlow controller
- Faucet – Ryu switching application
  - Valve (Datapath abstraction layer)
- Gauge – Ryu Monitoring and statistics application
  - InfluxDB (time-series DB)
  - PyODBC (RDBMS)
  - Grafana (Dashboard)
- External applications
  - Hostapd (802.1x support)
  - Peer with your favourite BGP daemon Quagga/FRRouting/Bird

Faucet Development

- Faucet is a Ryu application
- Ryu is an event driven OpenFlow framework and API
  - https://ryu.readthedocs.io/en/latest/
- Ryu features we use:
  - OpenFlow control channel
  - OpenFlow abstraction (crafting FlowMods, GroupMods, etc)
  - Packet parsing library
  - BGP library
- Code is on Github
  - https://github.com/REANNZ/faucet

Faucet Development

- faucet/
  - Configuration parsing: conf.py, config_parser.py, dp.py, port.py, vlan.py
  - Main ryu application: faucet.py
  - Datapath implementation: valve.py
  - Monitoring/statistics: gauge.py, watcher.py, watcher_conf.py
- tests/
  - faucet_mininet_test.py

Faucet Devices

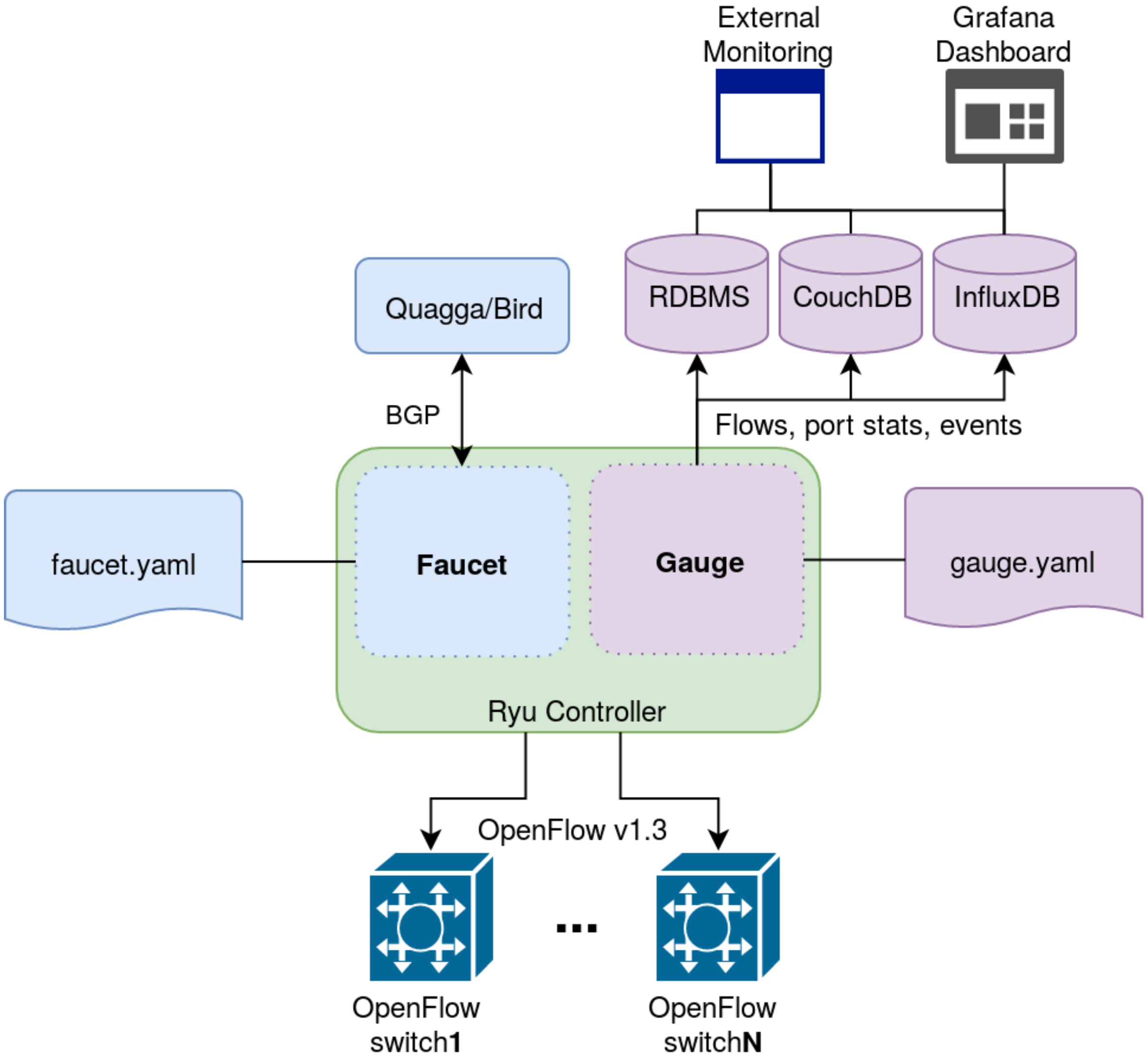
- Software switching
    - OpenvSwitch
    - Lagopus
- Hardware switching
    - Allied Telesis
    - NoviFlow
    - Netronome
    - HP Enterprise Aruba
    - Cisco
    - ZodiacFX Development Board

Running Faucet

- Installable with Python pip
- Or, Docker containers available on Docker hub
  - `docker pull faucet/faucet`
  - docker run:

    ```
    docker run -d \
        --name faucet \
        -v <path-to-config-dir>:/etc/ryu/faucet/ \
        -v <path-to-logging-dir>:/var/log/ryu/faucet/ \
        -p 6633:6633 \
        faucet/faucet
    ```
- Or, Prebuilt VM appliance
  - https://susestudio.com/a/ENQFFD/ryu-faucet

Faucet Architecture



Faucet Flooding

- Configurable flooding modes
- Default flooding behaviour
  - Flood all unknown unicast packets to VLAN
- Secure flooding
  - Can disable unicast flooding on a port, so that it doesn’t receive unknown unicast traffic
  - Broadcast/multicast is still flooded so ND and ARP will continue to work

Faucet Access Control Lists

- We use Ryu’s OpenFlow parser to handle ACLs
- This means you can define very fine-grained security policy on a port
- Rules are applied in order so you have control over how they apply to traffic
- We support Port ACLs and VLAN ACLs currently
  - Egress ACLs should be supported soon
- Supported actions:
  - Allow or Drop (filtering)
  - Output to port (port mirroring, NFV offload, etc)

Faucet Learning

- Configurable learning modes
- Default learning behaviour
  - Send traffic for unknown MACs to controller to learn SRC_MAC and DST_MAC
  - Use hard_timeout for ETH_SRC table and idle_timeout for ETH_DST table to expire learned MAC addresses
  - Relearn when MAC moves
- Permanent learn
  - Never timeout ETH_SRC or ETH_DST MAC rules
  - Hosts can't move ports once learned
- Max hosts
  - Limit how many MAC addresses may be learned on a port

Faucet Virtual IP addresses

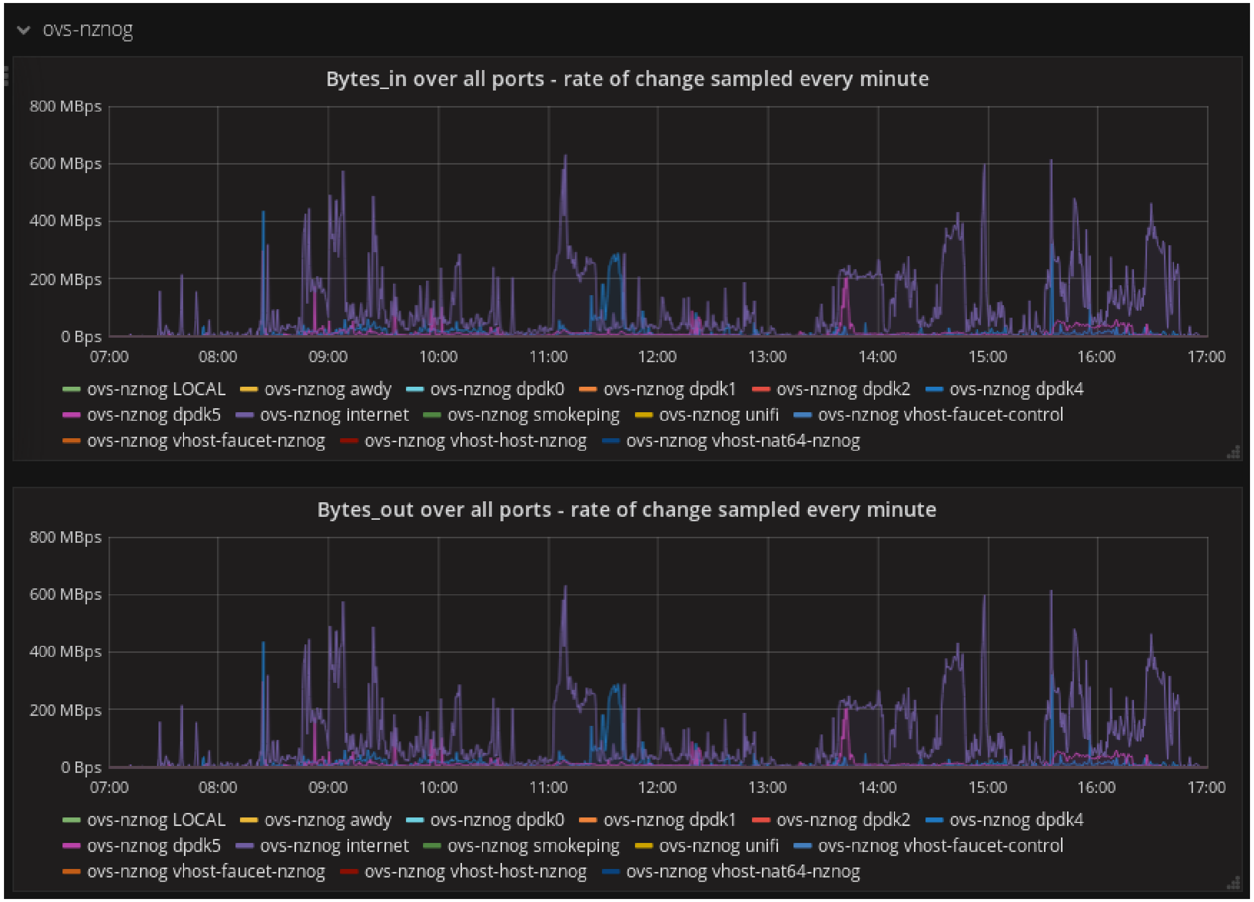
- a.k.a Faucet VIPs
- Allows Faucet controller to be present on the network
- Hand out Faucet VIP as gateway address to clients
- Install OpenFlow rules to catch ARP & ND packets destined for Faucet VIP and send these to the controller
- Reply with Faucet’s magic MAC (0e:00:00:00:00:01)
- Use this MAC address to identify packets for routing
- All routing happens on physical hardware in silicon

Faucet Monitoring

- Need a method of gaining visibility of our datapath
  - Faults
  - Capacity planning
- Gauge is a Ryu application
- Polls OpenFlow switches for port statistics
- Registers itself to receive datapath events (link up/down)
- Stores statistics in a time-series database InfluxDB
- Stores OpenFlow rules in JSON file or RDBMS

Faucet Monitoring

- Statistics are viewable via the Grafana dashboard



Faucet Configuration

```
---
version: 2

dps:
  …

vlans:
  …

routers:
  …

acls:
  …
```

Faucet Configuration – Datapaths

```
dps:
    0x000000000001:
        name: "test-switch-1"
        hardware: "Allied-Telesis"
        interfaces:
            1:
                native_vlan: 100
                acl_in: 1
            2:
                description: "trunk port"
                tagged_vlans: [100,200]
    0x000000000002:
        name: "test-switch-2"
        hardware: "Open vSwitch"
        interfaces:
            1:
                native_vlan: 100
            2:
                description: "trunk port"
                tagged_vlans: [100,200]
```

Faucet Configuration – VLANs

```
vlans:
    100:
        name: "customer vlan"
    200:
        name: "server vlan"
```

Faucet Configuration – Routing

```
Vlans:
    300:
        name: "customer"
        faucet_vips: ["192.168.0.1/24"]
        routes:
            - route:
                ip_dst: '172.16.0.0/24'
                ip_gw:  '192.168.0.2'
    400:
        name: "wan"
        faucet_vips: ["10.0.0.1/24"]
        bgp_port: 9179
        bgp_as: 64500
        bgp_routerid: "192.0.2.1"
        bgp_neighbor_addresses: ["127.0.0.1"]
        bgp_neighbor_as: 64501
```

Faucet Configuration – InterVLAN Routing

```
routers:
    router-1:
        vlans: [300, 400]
```

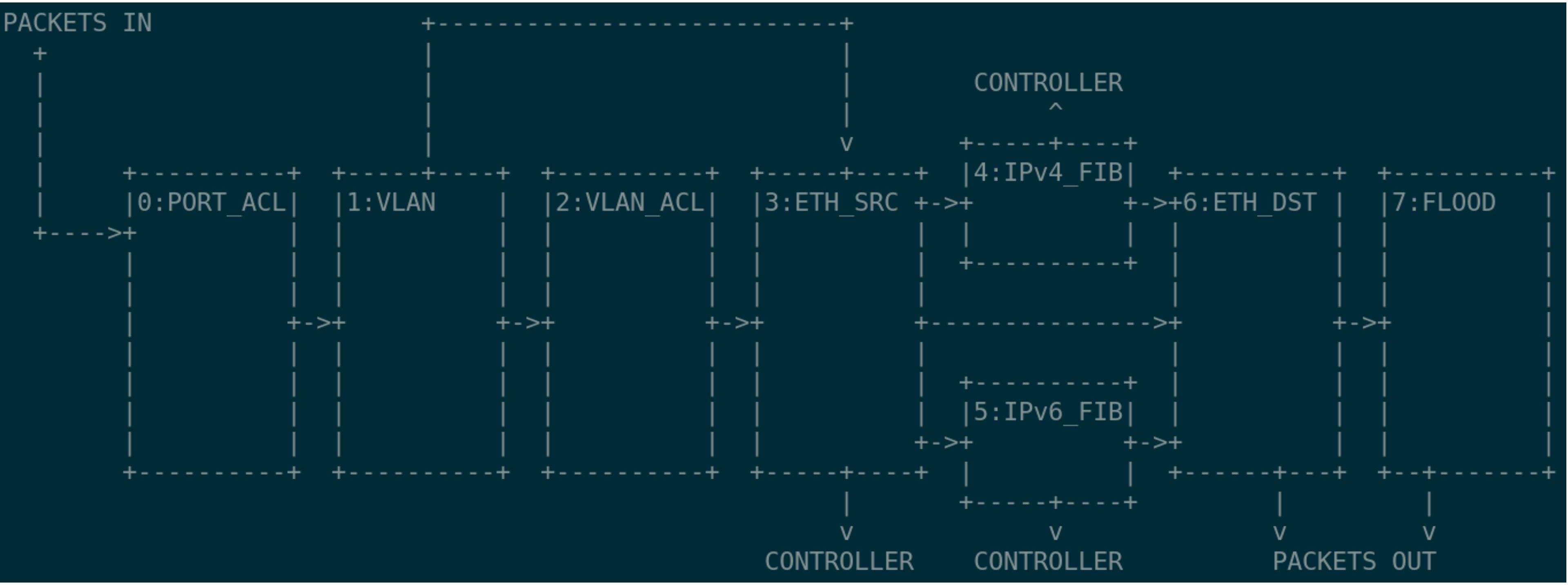
Faucet Configuration - ACLs

```
acls:
    1:
        - rule:
            dl_dst: "ff:ff:ff:ff:ff:ff"
            dl_type: 0x800
            nw_proto: 17
            nw_src: "0.0.0.0"
            nw_dst: "255.255.255.255"
            tp_src: 68
            tp_dst: 67
            actions:
                output:
                    port: 1
```

Gauge Configuration

```
---
faucet_configs:
    - 'config/faucet.yaml'
dbs:
    ft_file:
        type: 'text'
        file: 'flow_table.JSON'
    influx:
        type: 'influx'
        influx_db: 'faucet'
        influx_host: 'localhost'
        influx_port: 8086
watchers:
    flow_table_poller:
        type: 'flow_table'
        dps: ['switch1']
        interval: 40
        db: 'ft_file'
    port_state_logger:
        type: 'port_state'
        dps: ['switch1']
        db: 'influx'
    port_stats_poller:
        type: 'port_stats'
        dps: ['switch1']
        interval: 40
        db: 'influx'
```

Faucet Pipeline



```
PACKETS IN                  +---------------------------+
  +                         |                           |
  |                         |                           |        CONTROLLER
  |                         |                           |             ^
  |                         |                           v       +-----+----+
  |     +----------+  +-----+----+  +----------+  +-----+----+  |4:IPv4_FIB|  +----------+  +----------+
  |     |0:PORT_ACL|  |1:VLAN    |  |2:VLAN_ACL|  |3:ETH_SRC +->+          +->+6:ETH_DST |  |7:FLOOD   |
  +---->+          |  |          |  |          |  |          |  |          |  |          |  |          |
        |          |  |          |  |          |  |          |  +----------+  |          |  |          |
        |          |  |          |  |          |  |          |                |          |  |          |
        |          +->+          +->+          +->+          +---------------->+          +->+          |
        |          |  |          |  |          |  |          |                |          |  |          |
        |          |  |          |  |          |  |          |  +----------+  |          |  |          |
        |          |  |          |  |          |  |          |  |5:IPv6_FIB|  |          |  |          |
        |          |  |          |  |          |  |          +->+          +->+          |  |          |
        +----------+  +----------+  +----------+  +-----+----+  |          |  +------+---+  +--+-------+
                                                        |       +-----+----+        |         |
                                                        v             v              v         v
                                                   CONTROLLER    CONTROLLER          PACKETS OUT
```

Faucet Flows – Table 0: Port ACL

- Apply user supplied ACLs to a port and send to next table

Faucet Flows – Table 1: VLAN

- Match fields: *in_port, vlan_vid, eth_src, eth_dst, eth_type*
- Operations
  - Drop STP BPDUs
  - Drop LLDP
  - Drop broadcast sourced traffic
  - Drop traffic from sources spoofing Faucet’s magic MAC address
  - For tagged ports
    - Match VLAN_VID and send to next table
  - For untagged ports
    - Push VLAN frame onto packet with VLAN_VID representing ports native VLAN and send to next table
  - Unknown traffic is dropped

Faucet Flows – Table 2: VLAN ACL

- Apply user supplied ACLs to a VLAN and send to next table

Faucet Flows – Table 3: ETH_SRC

- Match fields: *in_port, vlan_vid, eth_src, eth_dst, eth_type, ip_proto, icmpv6_type, ipv6_nd_target, arp_tpa, ipv4_src*
- Operations
  - Handle layer 3 traffic by sending to IPv4 or IPv6 FIB table
  - Send traffic destined for Faucet via packet in message
  - For source MAC addresses we have learned send to ETH_DST
  - Unknown traffic is
    - Sent to controller via packet in (for learning)
    - Sent to ETH_DST table

Faucet Flows – Table 4: IPv4 FIB

- Match fields: *vlan_vid, eth_type, ip_proto, ipv4_src, ipv4_dst*
- Operations
  - Route IP traffic to a next-hop for each route we have learned
  - Set eth_src to Faucet's magic MAC address
  - Set eth_dst to the resolved MAC address for the next-hop
  - Decrement TTL
  - Send to ETH_DST table
  - Unknown traffic is dropped

Faucet Flows – Table 5: IPv6 FIB

- Match fields: *vlan_vid, eth_type, ip_proto, icmpv6_type, ipv6_dst*
- Operations
  - Route IP traffic to a next-hop for each route we have learned
  - Set eth_src to Faucet's magic MAC address
  - Set eth_dst to the resolved MAC address for the next-hop
  - Decrement TTL
  - Send to ETH_DST table
  - Unknown traffic is dropped

Faucet Flows – Table 6: ETH_DST

- Match fields: *vlan_vid, eth_dst*
- Operations
  - For destination MAC addresses we have learned output packet towards that host (popping VLAN frame if we are outputting on an untagged port)
  - Unknown traffic is sent to FLOOD table

Faucet Flows – Table 7: FLOOD

- Match fields: *vlan_vid, eth_dst*
- Operations
  - Flood broadcast within VLAN
  - Flood multicast within VLAN
  - Unknown traffic is flooded within VLAN

Faucet Unit Testing

- Uses Python unittest
- Runs Faucet code against virtual network topologies
- Virtual network provided by Mininet
- Also runs pylint on code
- Tests can optionally be run against real hardware

Faucet Unit Testing

- Running unit tests
  - docker build -t reannz/faucet-tests -f Dockerfile.tests .
  - apparmor_parser -R /etc/apparmor.d/usr.sbin.tcpdump
  - modprobe openvswitch
  - sudo docker run --privileged -ti reannz/faucet-tests

Faucet Scaling

- Deployed Faucet at NZNOG17 conference in Tauranga
- Need to generate a lot of traffic to find bottlenecks
  - Spin up 500 dockers on a laptop to simulate clients
- Lessons learned:
  - Reduce traffic on control channel, less CPU time spent on parsing control packets
  - Reduce number of PACKET_OUT messages by being smart about resolving end hosts (exponential backoff, random time variation between packets, etc)

Further Information

- Github
  - https://github.com/reannz/faucet
- Faucet blog
  - https://faucet-sdn.blogspot.co.nz
- Faucet troubleshooting guide
  - https://faucet-sdn.blogspot.co.nz/2016/06/faucet-troubleshootingfaq.html
- Faucet mailing lists
  - https://list.waikato.ac.nz/mailman/listinfo/faucet-dev
  - https://lists.geant.org/sympa/info/faucet-users

Questions