

---

# Faucet Documentation

**Faucet Developers**

**Mar 25, 2019**



---

# Contents

---

<b>1</b>	<b>User Documentation</b>	<b>1</b>
1.1	Introduction to Faucet . . . . .	1
1.2	Tutorials . . . . .	5
1.3	Installation . . . . .	45
1.4	Configuration . . . . .	52
1.5	Monitoring . . . . .	70
1.6	Configuration Recipe Book . . . . .	71
1.7	Vendor-specific Documentation . . . . .	72
1.8	External Resources . . . . .	103
<b>2</b>	<b>Developer Documentation</b>	<b>105</b>
2.1	Developer Guide . . . . .	105
2.2	Architecture . . . . .	110
2.3	Testing . . . . .	113
2.4	Fuzzing . . . . .	118
2.5	Source Code . . . . .	119
<b>3</b>	<b>Quick References</b>	<b>175</b>
3.1	Frequently Asked Questions . . . . .	175
<b>4</b>	<b>Indices and tables</b>	<b>179</b>
	<b>Python Module Index</b>	<b>181</b>



## 1.1 Introduction to Faucet

### 1.1.1 What is Faucet?

Faucet is a compact open source OpenFlow controller, which enables network operators to run their networks the same way they do server clusters. Faucet moves network control functions (like routing protocols, neighbor discovery, and switching algorithms) to vendor independent server-based software, versus traditional router or switch embedded firmware, where those functions are easy to manage, test, and extend with modern systems management best practices and tools. Faucet controls OpenFlow 1.3 hardware which delivers high forwarding performance.

You can read more about our approach to networking by reading our ACM Queue article [Faucet: Deploying SDN in the Enterprise](#).

### 1.1.2 What is Gauge?

Faucet has two main OpenFlow controller components, Faucet itself, and Gauge. Faucet controls all forwarding and switch state, and exposes its internal state, e.g. learned hosts, via Prometheus (so that an open source NMS such as Grafana graph it).

Gauge also has an OpenFlow connection to the switch and monitors port and flow state (exporting it to Prometheus or InfluxDB, or even flat text log files). Gauge, however, does not ever modify the switch's state, so that switch monitoring functions can be upgraded, restarted, without impacting forwarding.

### 1.1.3 Why Faucet?

#### Design

Faucet is designed to be very small, simple (1000s of lines of code, versus millions in other systems), and keep relatively little state. Faucet does not have any implementation-specific or vendor driver code, which considerably reduces complexity. Faucet does not need connectivity to external databases for forwarding decisions. Faucet provides

“hot/hot” high availability and scales through the provisioning of multiple Faucets with the same configuration - Faucet controllers are not inter-dependent.

### Performance and scaling

As well as being compact, Faucet offloads all forwarding to the OpenFlow switch, including flooding if emulating a traditional switch. Faucet programs the switch pre-emptively, though will receive packet headers from the switch if, for example, a host moves ports so that the switch’s OpenFlow FIB can be updated (again, if traditional switching is being emulated). In production, Faucet controllers have been observed to go many seconds without needing to process a packet from a switch. In cold start scenarios, Faucet has been observed to completely program a switch and learn connected hosts within a few seconds.

Faucet uses a multi-table packet processing pipeline as shown in *Faucet Openflow Switch Pipeline*. Using multiple flow tables over a single table allows Faucet to implement more complicated flow-based logic while maintaining a smaller number of total flows. Using dedicated flow tables with a narrow number of match fields, or limiting a table to exact match only, such as the IPv4 or IPv6 FIB tables allows us to achieve greater scalability over the number of flow entries we can install on a datapath.

A large network with many devices would run many Faucets, which can be spread over as many (or as few) machines as required. This approach scales well because each Faucet uses relatively few server resources and Faucet controllers do not have to be centralized - they can deploy as discrete switching or routing functional units, incrementally replacing (for example) non-SDN switches or routers.

An operator might have a controller for an entire rack, or just a few switches, which also reduces control plane complexity and latency by keeping control functions simple and local.

### Testing

Faucet follows open source software engineering best practices, including unit and systems testing (python unittest based), as well static analysis (pytype, pylint, and codecov) and fuzzing (python-af). Faucet’s systems tests test all Faucet features, from switching algorithms to routing, on virtual topologies. However, Faucet’s systems tests can also be configured to run the same feature tests on real OpenFlow hardware. Faucet developers also host regular PlugFest events specifically to keep switch implementations broadly synchronized in capabilities and compatibility.

## 1.1.4 Release Notes

### 1.7.0 Release Notes

We are making a few potentially breaking features in faucet 1.7.0. This document covers how to navigate the changes and safely upgrade from earlier versions to 1.7.0.

#### 1. Configuration and log directory changed

Starting in 1.7.0 and onwards faucet has changed which directories it uses for configuration and log files. The new paths are:

Old path	New path
/etc/ryu/faucet	/etc/faucet
/var/log/ryu/faucet	/var/log/faucet

Faucet 1.7.0 when being installed by pip will automatically attempt to migrate your old configuration files to /etc/faucet assuming it has permissions to do so. Failing this faucet when started will fallback to loading configuration from /etc/ryu/faucet. The search paths for configuration files are documented on the *Environment variables* page.

---

**Note:** Consider the `/etc/ryu/faucet` directory deprecated, we will in a future release stop reading config files stored in this directory.

---

If you currently set your own configuration or log directory by setting the appropriate environment variables you will be unaffected. In most other cases the migration code or the fallback configuration search path will allow the upgrade to 1.7.0 to be seamless. We have however identified two cases where manual intervention is required:

### Dockers

Dockers will need to be started with new mount directories, the commands to start a 1.7.0 docker version of faucet or gauge are detailed in the *Installation with Docker* section.

### Virtualenvs

We are unable to migrate configuration files automatically when faucet is run inside of a virtualenv, please copy the configuration directory over manually.

## 2. Changing default flood mode

Currently faucet defaults to using `combinatorial_port_flood` when it comes to provisioning flooding flows on a datapath, faucet implicitly configures a datapath like this today:

```
dps:
  mydp:
    combinatorial_port_flood: True
```

The default is `True`, in 1.7.0 and previously. The default will change to `False` in 1.7.1.

When `True`, flood rules are explicitly generated for each input port, to accommodate early switch implementations which (differing from the OpenFlow standard - see below) did not discard packets output to the packet input port. `False` generates rules per faucet VLAN which results in fewer rules and better scalability.

See [OpenFlow 1.3.5 specification](#), section B.6.3:

```
The behavior of sending out the incoming port was not clearly defined
in earlier versions of the specification. It is now forbidden unless
the output port is explicitly set to OFPP_IN_PORT virtual port
(0xffff8) is set.
```

## 1.9.0 Release Notes

There are some changes in version 1.9.0 of faucet that may affect how you use it. Below are the changes and how they might affect you.

### 1. Removing support for older python versions

Starting from faucet 1.9.0 and onwards, faucet now requires a version of python 3.5 or newer to function.

Most currently supported distributions of linux should have a version of python that is compatible, with the notable exception of Debian Jessie which is no longer supported by faucet.

### 2. Change BGP configuration syntax

Previously, BGP configuration for faucet was attached to a VLAN, for example:

Listing 1: Older style bgp configuration

```
vlan:
  internet:
    description: 'internet peering'
    vid: 200
    bgp_routerid: '127.0.0.2'
    bgp_as: 14031
    bgp_neighbor_as: 14031
    bgp_neighbor_addresses: ['127.0.0.1', '::1']
    bgp_server_addresses: ['127.0.0.2', '::1']
    bgp_port: 9179
    bgp_connect_mode: 'passive'
```

As BGP peering in faucet now has the ability to resolve next hops in all VLANs, we have elected to move where BGP is configured.

We have now implemented a new `bgp` router type that can be configured in faucet, similar to how inter-VLAN routing works today, for example this is an example of the new syntax showing how we would convert the configuration shown above:

Listing 2: Newer style bgp configuration

```
vlan:
  internet:
    description: 'internet peering'
    vid: 200

router:
  internet-router:
    bgp:
      vlan: internet
      routerid: '127.0.0.2'
      as: 14031
      neighbor_as: 14031
      neighbor_addresses: ['127.0.0.1', '::1']
      server_addresses: ['127.0.0.2', '::1']
      port: 9179
      connect_mode: 'passive'
```

It is also possible to combine inter-VLAN routing and bgp routing in a single routing instance:

Listing 3: Newer style bgp configuration (with IVR)

```
vlan:
  office:
    description: 'internet peering'
    vid: 100
  internet:
    description: 'internet peering'
    vid: 200

router:
  office-internet-router:
    vlans: [office, internet]
    bgp:
      vlan: internet
```

(continues on next page)



(continued from previous page)

```

routerid: '127.0.0.2'
as: 14031
neighbor_as: 14031
neighbor_addresses: ['127.0.0.1', '::1']
server_addresses: ['127.0.0.2', '::1']
port: 9179
connect_mode: 'passive'

```

For more information on what each option does, please see the *BGP* documentation section.

## 1.1.5 Getting Help

We use maintain a number of mailing lists for communicating with users and developers:

- [faucet-announce](#)
- [faucet-dev](#)
- [faucet-users](#)

We also have the [#faucetsdn](#) IRC channel on [freenode](#).

A few tutorial videos are available on our [YouTube channel](#).

The [faucetsdn blog](#) and [faucetsdn twitter](#) are good places to keep up with the latest news about faucet.

If you find bugs, or if have feature requests, please create an issue on our [bug tracker](#).

## 1.2 Tutorials

### 1.2.1 Installing faucet for the first time

This tutorial will run you through the steps of installing a complete faucet system for the first time.

We will be installing and configuring the following components:

Component	Purpose
faucet	Network controller
gauge	Monitoring controller
prometheus	Monitoring system & time series database
grafana	Monitoring dashboard

This tutorial was written for Ubuntu 16.04, however the steps should work fine on any newer supported version of Ubuntu or Debian.

### Package installation

1. Add the faucet official repo to our system:

```

sudo apt-get install curl gnupg apt-transport-https lsb-release
echo "deb https://packagecloud.io/faucetsdn/faucet/$(lsb_release -si | awk '
↪{print tolower($0)}')/ $(lsb_release -sc) main" | sudo tee /etc/apt/sources.
↪list.d/faucet.list

```

(continues on next page)

(continued from previous page)

```
curl -L https://packagecloud.io/faucetsdn/faucet/gpgkey | sudo apt-key add -
sudo apt-get update
```

2. Install the required packages, we can use the `faucet-all-in-one` metapackage which will install all the correct dependencies.

```
sudo apt-get install faucet-all-in-one
```

## Configure prometheus

We need to configure prometheus to tell it how to scrape metrics from both the faucet and gauge controllers. To help make life easier faucet ships a sample configuration file for prometheus which sets it up to scrape a single faucet and gauge controller running on the same machine as prometheus. The configuration file we ship looks like:

Listing 4: prometheus.yml

```
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is
  ↪every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1
  ↪minute.
  # scrape_timeout is set to the global default (10s).

# Load rules once and periodically evaluate them according to the global 'evaluation_
  ↪interval'.
rule_files:
  - "faucet.rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from
  ↪this config.
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'faucet'
    static_configs:
      - targets: ['localhost:9302']
  - job_name: 'gauge'
    static_configs:
      - targets: ['localhost:9303']
```

To learn more about what this configuration file does you can look at the [Prometheus Configuration Documentation](#). The simple explanation is that it includes an additional `faucet.rules.yml` file that performs some automatic queries in prometheus for generating some additional metrics as well as setting up scrape jobs every 15 seconds for faucet listening on `localhost:9302` and gauge listening on `localhost:9303`.

Steps to make prometheus use the configuration file shipped with faucet:

1. Change the configuration file prometheus loads by editing the file `/etc/default/prometheus` to look like:

Listing 5: /etc/default/prometheus

```
# Set the command-line arguments to pass to the server.
ARGS="--config.file=/etc/faucet/prometheus/prometheus.yml"
```

- Restart prometheus to apply the changes:

```
sudo systemctl restart prometheus
```

## Configure grafana

Grafana running in it's default configuration will work just fine for our needs. We will however need to make it start on boot, configure prometheus as a data source and add our first dashboard:

- Make grafana start on boot and then start it manually for the first time:

```
sudo systemctl daemon-reload
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

- To finish setup we will configure grafana via the web interface.

First load `http://localhost:3000` in your web browser (by default both the username and password are admin).

- The web interface will first prompt us to add a data source. Use the following settings then click Save & Test:

```
Name: Prometheus
Type: Prometheus
URL: http://localhost:9090
```

- Next we want to add some dashboards so that we can later view the metrics from faucet.

Hover over the + button on the left sidebar in the web interface and click Import.

We will import the following dashboards, just download the following links and upload them through the grafana dashboard import screen:

- [Instrumentation](#)
- [Inventory](#)
- [Port Statistics](#)

## Configure faucet

For this tutorial we will configure a very simple network topology consisting of a single switch with two ports.

- Configure faucet

We need to tell faucet about our topology and VLAN information, we can do this by editing the faucet configuration `/etc/faucet/faucet.yaml` to look like:

Listing 6: /etc/faucet/faucet.yaml

```
vlan:
  office:
    vid: 100
    description: "office network"

dps:
  sw1:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        name: "host1"
        description: "host1 network namespace"
        native_vlan: office
      2:
        name: "host2"
        description: "host2 network namespace"
        native_vlan: office
```

**Note:** Tabs are forbidden in the YAML language, please use only spaces for indentation.

This will create a single VLAN and a single datapath with two ports.

## 2. Verify configuration

The `check_faucet_config` command can be used to verify faucet has correctly interpreted your configuration before loading it. This can avoid shooting yourself in the foot by applying configuration with typos. We recommend either running this command by hand or with automation each time before loading configuration.

```
check_faucet_config /etc/faucet/faucet.yaml
```

This script will either return an error, or in the case of successfully parsing the configuration it will return a JSON object containing the entire faucet configuration that would be loaded (including any default settings), for example:

```
[{'advertise_interval': 30,
  'arp_neighbor_timeout': 30,
  'cache_update_guard_time': 150,
  'combinatorial_port_flood': False,
  'cookie': 1524372928,
  'description': 'sw1',
  'dot1x': None,
  'dp_acls': None,
  'dp_id': 1,
  'drop_broadcast_source_address': True,
  'drop_spoofed_faucet_mac': True,
  'egress_pipeline': False,
  'fast_advertise_interval': 5,
  'faucet_dp_mac': '0e:00:00:00:00:01',
  'global_vlan': 0,
  'group_table': False,
  'hardware': 'Open vSwitch',
  'high_priority': 9001,
  'highest_priority': 9099,
  'idle_dst': True,
```

(continues on next page)

(continued from previous page)

```
'ignore_learn_ins': 10,
'interface_ranges': OrderedDict(),
'interfaces': {'host1': {'acl_in': None,
                        'acls_in': None,
                        'description': 'host1 network namespace',
                        'dot1x': False,
                        'enabled': True,
                        'hairpin': False,
                        'hairpin_unicast': False,
                        'lacp': 0,
                        'lacp_active': False,
                        'lldp_beacon': OrderedDict(),
                        'loop_protect': False,
                        'loop_protect_external': False,
                        'max_hosts': 255,
                        'max_lldp_lost': 3,
                        'mirror': None,
                        'native_vlan': 'office',
                        'number': 1,
                        'opstatus_reconf': True,
                        'output_only': False,
                        'override_output_port': None,
                        'permanent_learn': False,
                        'receive_lldp': False,
                        'stack': OrderedDict(),
                        'tagged_vlans': [],
                        'unicast_flood': True},
               'host2': {'acl_in': None,
                        'acls_in': None,
                        'description': 'host2 network namespace',
                        'dot1x': False,
                        'enabled': True,
                        'hairpin': False,
                        'hairpin_unicast': False,
                        'lacp': 0,
                        'lacp_active': False,
                        'lldp_beacon': OrderedDict(),
                        'loop_protect': False,
                        'loop_protect_external': False,
                        'max_hosts': 255,
                        'max_lldp_lost': 3,
                        'mirror': None,
                        'native_vlan': 'office',
                        'number': 2,
                        'opstatus_reconf': True,
                        'output_only': False,
                        'override_output_port': None,
                        'permanent_learn': False,
                        'receive_lldp': False,
                        'stack': OrderedDict(),
                        'tagged_vlans': [],
                        'unicast_flood': True}},

'lacp_timeout': 30,
'learn_ban_timeout': 51,
'learn_jitter': 51,
'lldp_beacon': OrderedDict(),
'low_priority': 9000,
```

(continues on next page)

(continued from previous page)

```
'lowest_priority': 0,
'max_host_fib_retry_count': 10,
'max_hosts_per_resolve_cycle': 5,
'max_resolve_backoff_time': 64,
'max_wildcard_table_size': 1280,
'metrics_rate_limit_sec': 0,
'min_wildcard_table_size': 32,
'multi_out': True,
'nd_neighbor_timeout': 30,
'ofchannel_log': None,
'packetin_pps': None,
'priority_offset': 0,
'proactive_learn_v4': True,
'proactive_learn_v6': True,
'stack': None,
'strict_packet_in_cookie': True,
'table_sizes': OrderedDict(),
'timeout': 300,
'use_classification': False,
'use_idle_timeout': False}]
```

### 3. Reload faucet

To apply this configuration we can reload faucet which will cause it to compute the difference between the old and new configuration and apply the minimal set of changes to the network in a hitless fashion (where possible).

```
sudo systemctl reload faucet
```

### 4. Check logs

To verify the configuration reload was successful we can check `/var/log/faucet/faucet.log` and make sure faucet successfully loaded the configuration we can check the faucet log file `/var/log/faucet/faucet.log`:

Listing 7: `/var/log/faucet/faucet.log`

```
faucet INFO      Loaded configuration from /etc/faucet/faucet.yaml
faucet INFO      Add new datapath DPID 1 (0x1)
faucet INFO      Add new datapath DPID 2 (0x2)
faucet INFO      configuration /etc/faucet/faucet.yaml changed, analyzing_
↔differences
faucet INFO      Reconfiguring existing datapath DPID 1 (0x1)
faucet.valve INFO      DPID 1 (0x1) skipping configuration because datapath not up
faucet INFO      Deleting de-configured DPID 2 (0x2)
```

If there were any issues (say faucet wasn't able to find a valid pathway from the old config to the new config) we could issue a faucet restart now which will cause a cold restart of the network.

## Configure gauge

We will not need to edit the default gauge configuration that is shipped with faucet as it will be good enough to complete the rest of this tutorial. If you did need to modify it the path is `/etc/faucet/gauge.yaml` and the default configuration looks like:

Listing 8: gauge.yaml

```

# Recommended configuration is Prometheus for all monitoring, with all_dps: True
faucet_configs:
  - '/etc/faucet/faucet.yaml'
watchers:
  port_status_poller:
    type: 'port_state'
    all_dps: True
    #dps: ['sw1', 'sw2']
    db: 'prometheus'
  port_stats_poller:
    type: 'port_stats'
    all_dps: True
    #dps: ['sw1', 'sw2']
    interval: 10
    db: 'prometheus'
    #db: 'influx'
  flow_table_poller:
    type: 'flow_table'
    all_dps: True
    interval: 60
    db: 'prometheus'
dbs:
  prometheus:
    type: 'prometheus'
    prometheus_addr: '0.0.0.0'
    prometheus_port: 9303
  ft_file:
    type: 'text'
    compress: True
    file: 'flow_table.yaml.gz'
  influx:
    type: 'influx'
    influx_db: 'faucet'
    influx_host: 'influxdb'
    influx_port: 8086
    influx_user: 'faucet'
    influx_pwd: 'faucet'
    influx_timeout: 10

```

This default configuration will setup a prometheus exporter listening on port 0.0.0.0:9303 and write all the different kind of gauge metrics to this exporter.

We will however need to restart the current gauge instance so it can pick up our new faucet configuration:

```
sudo systemctl restart gauge
```

## Connect your first datapath

Now that we've set up all the different components let's connect our first switch (which we call a datapath) to faucet. We will be using [Open vSwitch](#) for this which is a production-grade software switch with very good OpenFlow support.

1. Add WAND Open vSwitch repo

The bundled version of Open vSwitch in Ubuntu 16.04 is quite old so we will use [WAND's package](#)

repo to install a newer version (if you're using a more recent debian or ubuntu release you can skip this step).

---

**Note:** If you're using a more recent debian or ubuntu release you can skip this step

---

```
sudo apt-get install apt-transport-https
echo "deb https://packages.wand.net.nz $(lsb_release -sc) main" | sudo_
↪tee /etc/apt/sources.list.d/wand.list
sudo curl https://packages.wand.net.nz/keyring.gpg -o /etc/apt/trusted.
↪gpg.d/wand.gpg
sudo apt-get update
```

### 2. Install Open vSwitch

```
sudo apt-get install openvswitch-switch
```

### 3. Add network namespaces to simulate hosts

We will use two linux network namespaces to simulate hosts and this will allow us to generate some traffic on our network.

First let's define some useful bash functions by copying and pasting the following definitions into our bash terminal:

```
# Create network namespace
create_ns () {
    NAME=$1
    IP=$2
    NETNS=faucet-${NAME}
    sudo ip netns add ${NETNS}
    sudo ip link add dev veth-${NAME} type veth peer name veth0 netns
↪${NETNS}
    sudo ip link set dev veth-${NAME} up
    sudo ip netns exec ${NETNS} ip link set dev veth0 up
    sudo ip netns exec ${NETNS} ip addr add dev veth0 $IP
    sudo ip netns exec ${NETNS} ip link set dev lo up
}
```

```
# Run command inside network namespace
as_ns () {
    NAME=$1
    NETNS=faucet-${NAME}
    shift
    sudo ip netns exec $NETNS $@
}
```

Now we will create host1 and host2 and assign them some IPs:

```
create_ns host1 192.168.0.1/24
create_ns host2 192.168.0.2/24
```

### 2. Configure Open vSwitch

We will now configure a single Open vSwitch bridge (which will act as our datapath) and add two ports to this bridge:



```

sudo ovs-vsctl add-br br0 \
-- set bridge br0 other-config:datapath-id=0000000000000001 \
-- set bridge br0 other-config:disable-in-band=true \
-- set bridge br0 fail_mode=secure \
-- add-port br0 veth-host1 -- set interface veth-host1 ofport_request=1 \
-- add-port br0 veth-host2 -- set interface veth-host2 ofport_request=2 \
-- set-controller br0 tcp:127.0.0.1:6653 tcp:127.0.0.1:6654

```

The [Open vSwitch documentation](#) is very good if you wish to find out more about configuring Open vSwitch.

### 3. Verify datapath is connected to faucet

At this point everything should be working, we just need to verify that is the case. If we now load up some of the grafana dashboards we imported earlier, we should see the datapath is now listed in the `Faucet Inventory` dashboard.

If you don't see the new datapath listed you can look at the faucet log files `/var/log/faucet/faucet.log` or the Open vSwitch log `/var/log/openvswitch/ovs-vswitchd.log` for clues.

### 4. Generate traffic between virtual hosts

With `host1` and `host2` we can now test our network works and start generating some traffic which will show up in grafana.

Let's start simple with a ping:

```
as_ns host1 ping 192.168.0.2
```

If this test is successful this shows our Open vSwitch is forwarding traffic under faucet control, `/var/log/faucet/faucet.log` should now indicate those two hosts have been learnt:

Listing 9: `/var/log/faucet/faucet.log`

```

faucet.valve INFO      DPID 1 (0x1) L2 learned 22:a6:c7:20:ff:3b (L2 type_
↪0x0806, L3 src 192.168.0.1, L3 dst 192.168.0.2) on Port 1 on VLAN 100_
↪(1 hosts total)
faucet.valve INFO      DPID 1 (0x1) L2 learned 36:dc:0e:b2:a3:4b (L2 type_
↪0x0806, L3 src 192.168.0.2, L3 dst 192.168.0.1) on Port 2 on VLAN 100_
↪(2 hosts total)

```

We can also use `iperf` to generate a large amount of traffic which will show up on the `Port Statistics` dashboard in grafana, just select `sw1` as the `Datapath Name` and `All` for the `Port`.

```

sudo apt-get install iperf3
as_ns host1 iperf3 --server --pidfile /run/iperf3-host1.pid --daemon
as_ns host2 iperf3 --client 192.168.0.1

```

## Further steps

Now that you know how to setup and run faucet in a self-contained virtual environment you can build on this tutorial and start to make more interesting topologies by adding more Open vSwitch bridges, ports and network namespaces. Check out the faucet [Configuration](#) document for more information on features you can turn on and off. In future we will publish additional tutorials on layer 3 routing, inter-VLAN routing, ACLs.

You can also easily add real hardware into the mix as well instead of using a software switch. See the [Vendor-specific Documentation](#) section for information on how to configure a wide variety of different vendor devices for faucet.

### 1.2.2 ACL tutorial

In the *Installing faucet for the first time* tutorial we covered how to install and set-up Faucet. Next we are going to introduce Access Control Lists (ACLs).

ETA: ~25 minutes.

#### Prerequisites

- Install Faucet - *Package installation* steps 1 & 2
- Install Open vSwitch - *Connect your first datapath* steps 1 & 2
- Useful Bash Functions - Copy and paste the following definitions into your bash terminal, or to make them persistent between sessions add them to the bottom of your `.bashrc` and run `source .bashrc`.

```
# Create network namespace
create_ns () {
    NAME=$1
    IP=$2
    NETNS=faucet-{$NAME}
    sudo ip netns add {$NETNS}
    sudo ip link add dev veth-{$NAME} type veth peer name veth0 netns
    ↪{$NETNS}
    sudo ip link set dev veth-{$NAME} up
    sudo ip netns exec {$NETNS} ip link set dev veth0 up
    sudo ip netns exec {$NETNS} ip addr add dev veth0 $IP
    sudo ip netns exec {$NETNS} ip link set dev lo up
}
```

```
# Run command inside network namespace
as_ns () {
    NAME=$1
    NETNS=faucet-{$NAME}
    shift
    sudo ip netns exec $NETNS $@
}
```

**Note:** If not continuing on from the ‘Installing Faucet for first time tutorial’ to setup the hosts and switch run:

```
create_ns host1 192.168.0.1/24
create_ns host2 192.168.0.2/24
sudo ovs-vsctl add-br br0 \
-- set bridge br0 other-config:datapath-id=0000000000000001 \
-- set bridge br0 other-config:disable-in-band=true \
-- set bridge br0 fail_mode=secure \
-- add-port br0 veth-host1 -- set interface veth-host1 ofport_request=1 \
-- add-port br0 veth-host2 -- set interface veth-host2 ofport_request=2 \
-- set-controller br0 tcp:127.0.0.1:6653 tcp:127.0.0.1:6654
```

And the `faucet.yaml` configuration file looks like:

Listing 10: `/etc/faucet/faucet.yaml`

```
vlan:
  office:
```

(continues on next page)

(continued from previous page)

```

    vid: 100
    description: "office network"

dps:
  sw1:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        name: "host1"
        description: "host2 network namespace"
        native_vlan: office
      2:
        name: "host2"
        description: "host2 network namespace"
        native_vlan: office

```

## Overview

Faucet ACLs are made up of lists of rules. The order of the rules in the list denote the priority with the first rules being highest and last lowest. Each of these lists has a name (e.g. 'block-ping'), and can be used on multiple port or VLAN 'acls\_in' fields. Again these are applied in order so all of 'block-ping' rules will be higher than 'allow-all'.

Each rule contains two main items 'matches' and 'actions'. Matches are any packet field such as MAC/IP/transport source/destination fields. For a full list visit the [Ryu documentation](#).

Actions are used to control what the packet does, for example normal L2 forwarding ('allow'). Apply a 'meter' to rate limit traffic, and manipulation of the packet contents and output. Full list is available in the [Meters](#) section of the documentation.

The example below has defined two ACLs 'block-ping' & 'allow-all' these can be used on any and multiple ports or VLANs (more on VLANs later) using the 'acls\_in' key. The block-ping ACL has two rules, one to block ICMP on IPv4 and another for ICMPv6 on IPv6. The allow-all ACL has one rule, which specifies no match fields, and therefore matches all packets, and the action 'allow'. The 'allow' action is a boolean, if it's True allow the packet to continue through the Faucet pipeline, if False drop the packet. 'allow' can be used in conjunction with the other actions to let the traffic flow with the expected layer 2 forwarding behaviour AND be mirrored to another port.

## Network setup

We are going to create the following network:

First we will add two new hosts to our network:

```

create_ns host3 192.168.0.3/24
create_ns host4 192.168.0.4/24

```

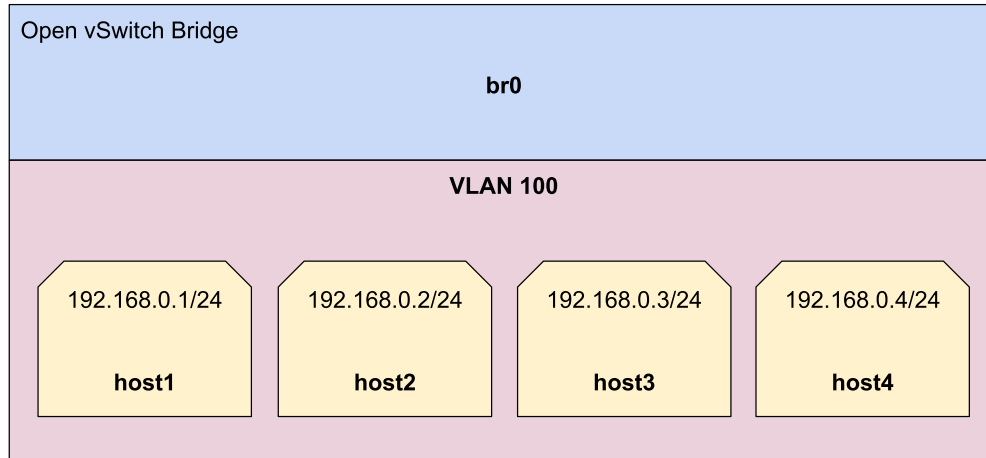
And connect them to br0

```

sudo ovs-vsctl add-port br0 veth-host3 -- set interface veth-host3 ofport_request=3 \
  -- add-port br0 veth-host4 -- set interface veth-host4 ofport_request=4

```

The configuration below will block ICMP on traffic coming in on port 3, and allow everything else. Add this to /etc/faucet/faucet.yaml below the 'dps'.



Listing 11: /etc/faucet/faucet.yaml

```

3:
  name: "host3"
  native_vlan: office
  acls_in: [block-ping, allow-all]
4:
  name: "host4"
  native_vlan: office
acls:
  block-ping:
    - rule:
      dl_type: 0x800      # IPv4
      ip_proto: 1       # ICMP
      actions:
        allow: False
    - rule:
      dl_type: 0x86dd    # IPv6
      ip_proto: 58      # ICMPv6
      actions:
        allow: False
  allow-all:
    - rule:
      actions:
        allow: True

```

Now tell Faucet to reload its configuration, this can be done by restarting the application. But a better way is to send Faucet a SIGHUP signal.

```
check_faucet_config /etc/faucet/faucet.yaml
```

```
sudo systemctl reload faucet
```

Pings to/from host3 should now fail:

```
as_ns host1 ping 192.168.0.3
```

But the other three hosts should be fine:

```
as_ns host1 ping 192.168.0.2
as_ns host1 ping 192.168.0.4
```

## ACL actions

### Mirroring

Mirroring traffic is useful if we want to send it to an out of band NFV service (e.g. Intrusion Detection System, packet capture a port or VLAN). To do this Faucet provides two ACL actions: mirror & output.

The mirror action copies the packet, before any modifications, to the specified port.

**Note:** Mirroring is done in input direction only.

Let's add the mirror action to our block-ping ACL `/etc/faucet/faucet.yaml`

Listing 12: `/etc/faucet/faucet.yaml`

```
...
  block-ping:
    - rule:
      dl_type: 0x800
      ip_proto: 1
      actions:
        allow: False
        mirror: 4
    - rule:
      dl_type: 0x86dd
      ip_proto: 58
      actions:
        allow: False
        mirror: 4
```

And again send the sighup signal to Faucet

```
sudo systemctl reload faucet
```

To check this we will ping from host1 to host3, while performing a tcpdump on host4 who should receive the ping replies. It is a good idea to run each from a different terminal (screen, tmux, ...)

```
as_ns host4 tcpdump -l -e -n -i veth0
```

```
as_ns host1 ping 192.168.0.3
```

Ping should have 100% packet loss.

```
$ as_ns host4 tcpdump -l -e -n -i veth0
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on veth0, link-type EN10MB (Ethernet), capture size 262144 bytes
13:24:36.848331 2e:d4:1a:ca:54:4b > 06:5f:14:fc:47:02, ethertype IPv4 (0x0800),
↪length 98: 192.168.0.3 > 192.168.0.1: ICMP echo reply, id 23660, seq 16, length 64
13:24:37.857024 2e:d4:1a:ca:54:4b > 06:5f:14:fc:47:02, ethertype IPv4 (0x0800),
↪length 98: 192.168.0.3 > 192.168.0.1: ICMP echo reply, id 23660, seq 17, length 64
```

(continues on next page)

(continued from previous page)

```

13:24:38.865005 2e:d4:1a:ca:54:4b > 06:5f:14:fc:47:02, ethertype IPv4 (0x0800),
↪length 98: 192.168.0.3 > 192.168.0.1: ICMP echo reply, id 23660, seq 18, length 64
13:24:39.873377 2e:d4:1a:ca:54:4b > 06:5f:14:fc:47:02, ethertype IPv4 (0x0800),
↪length 98: 192.168.0.3 > 192.168.0.1: ICMP echo reply, id 23660, seq 19, length 64
13:24:40.881129 2e:d4:1a:ca:54:4b > 06:5f:14:fc:47:02, ethertype IPv4 (0x0800),
↪length 98: 192.168.0.3 > 192.168.0.1: ICMP echo reply, id 23660, seq 20, length 64

```

## Output

There is also the ‘output’ action which can be used to achieve the same thing.

Listing 13: /etc/faucet/faucet.yaml

```

block-ping:
  - rule:
    dl_type: 0x800
    ip_proto: 1
    actions:
      allow: False
      output:
        port: 4
  - rule:
    dl_type: 0x86dd
    ip_proto: 58
    actions:
      allow: False
      output:
        port: 4

```

The output action also allows us to change the packet by setting fields (mac/ip addresses, ...), VLAN operations (push/pop/swap VLANs). It can be used in conjunction with the other actions, e.g. output directly but do not allow through the Faucet pipeline (allow: false).

Let’s create a new ACL for host2’s port that will change the MAC source address.

Listing 14: /etc/faucet/faucet.yaml

```

dps:
  sw1:
    ...
    2:
      name: "host2"
      description: "host2 network namespace"
      native_vlan: office
      acls_in: [rewrite-mac, allow-all]
    ...
acls:
  rewrite-mac:
    - rule:
      actions:
        allow: True
        output:
          set_fields:
            - eth_src: "00:00:00:00:00:02"
    ...

```

Again reload Faucet.

Start tcpdump on host1

```
as_ns host1 tcpdump -l -e -n -i veth0
```

Ping host1 from host2

```
as_ns host2 ping 192.168.0.1
```

Here we can see ICMP echo requests are coming from the MAC address “00:00:00:00:00:02” that we set in our output ACL. (The reply is destined to the actual MAC address of host2 thanks to ARP).

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on veth0, link-type EN10MB (Ethernet), capture size 262144 bytes
13:53:41.248235 00:00:00:00:00:02 > 06:5f:14:fc:47:02, ethertype IPv4 (0x0800),
↳length 98: 192.168.0.2 > 192.168.0.1: ICMP echo request, id 23711, seq 1, length 64
13:53:41.248283 06:5f:14:fc:47:02 > ce:bb:23:ce:d5:a0, ethertype IPv4 (0x0800),
↳length 98: 192.168.0.1 > 192.168.0.2: ICMP echo reply, id 23711, seq 1, length 64
13:53:42.247106 00:00:00:00:00:02 > 06:5f:14:fc:47:02, ethertype IPv4 (0x0800),
↳length 98: 192.168.0.2 > 192.168.0.1: ICMP echo request, id 23711, seq 2, length 64
13:53:42.247154 06:5f:14:fc:47:02 > ce:bb:23:ce:d5:a0, ethertype IPv4 (0x0800),
↳length 98: 192.168.0.1 > 192.168.0.2: ICMP echo reply, id 23711, seq 2, length 64
13:53:43.249726 00:00:00:00:00:02 > 06:5f:14:fc:47:02, ethertype IPv4 (0x0800),
↳length 98: 192.168.0.2 > 192.168.0.1: ICMP echo request, id 23711, seq 3, length 64
13:53:43.249757 06:5f:14:fc:47:02 > ce:bb:23:ce:d5:a0, ethertype IPv4 (0x0800),
↳length 98: 192.168.0.1 > 192.168.0.2: ICMP echo reply, id 23711, seq 3, length 64
13:53:44.248713 00:00:00:00:00:02 > 06:5f:14:fc:47:02, ethertype IPv4 (0x0800),
↳length 98: 192.168.0.2 > 192.168.0.1: ICMP echo request, id 23711, seq 4, length 64
13:53:44.248738 06:5f:14:fc:47:02 > ce:bb:23:ce:d5:a0, ethertype IPv4 (0x0800),
↳length 98: 192.168.0.1 > 192.168.0.2: ICMP echo reply, id 23711, seq 4, length 64
```

With the output action we could also use it to mirror traffic to a NFV server (like our fake mirror output action above), and use a VLAN tag to identify what port the traffic originated on on the switch. To do this we will use both the ‘port’ & ‘vlan\_vid’ output fields.

Listing 15: /etc/faucet/faucet.yaml

```
block-ping:
- rule:
  dl_type: 0x800
  ip_proto: 1
  actions:
    allow: False
    output:
      vlan_vid: 3
      port: 4
- rule:
  dl_type: 0x86dd
  ip_proto: 58
  actions:
    allow: False
    output:
      vlan_vid: 3
      port: 4
```

Again reload Faucet, start a tcpdump on host4, and ping from host1 to host3. Ping should still not be allowed through and the tcpdump output should be similar to below (Note the 802.1Q tag and VLAN 3):

```
$ as_ns host4 tcpdump -l -e -n -i veth0

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on veth0, link-type EN10MB (Ethernet), capture size 262144 bytes
14:14:15.285329 2e:d4:1a:ca:54:4b > 06:5f:14:fc:47:02, ethertype 802.1Q (0x8100),
↳ length 102: vlan 3, p 0, ethertype IPv4, 192.168.0.3 > 192.168.0.1: ICMP echo reply,
↳ id 23747, seq 1, length 64
14:14:16.293016 2e:d4:1a:ca:54:4b > 06:5f:14:fc:47:02, ethertype 802.1Q (0x8100),
↳ length 102: vlan 3, p 0, ethertype IPv4, 192.168.0.3 > 192.168.0.1: ICMP echo reply,
↳ id 23747, seq 2, length 64
14:14:17.300898 2e:d4:1a:ca:54:4b > 06:5f:14:fc:47:02, ethertype 802.1Q (0x8100),
↳ length 102: vlan 3, p 0, ethertype IPv4, 192.168.0.3 > 192.168.0.1: ICMP echo reply,
↳ id 23747, seq 3, length 64
```

### 1.2.3 VLAN tutorial

Next we are going to introduce VLANs.

ETA: ~30 mins.

#### Prerequisites

- Install Faucet - *Package installation* steps 1 & 2
- Install Open vSwitch - *Connect your first datapath* steps 1 & 2
- Useful Bash Functions - Copy and paste the following definitions into your bash terminal, or to make them persistent between sessions add them to the bottom of your `.bashrc` and run `source .bashrc`.

```
# Create network namespace
create_ns () {
    NAME=$1
    IP=$2
    NETNS=faucet- $\{NAME\}$ 
    sudo ip netns add  $\{NETNS\}$ 
    sudo ip link add dev veth- $\{NAME\}$  type veth peer name veth0 netns
↳  $\{NETNS\}$ 
    sudo ip link set dev veth- $\{NAME\}$  up
    sudo ip netns exec  $\{NETNS\}$  ip link set dev veth0 up
    sudo ip netns exec  $\{NETNS\}$  ip addr add dev veth0  $\{IP\}$ 
    sudo ip netns exec  $\{NETNS\}$  ip link set dev lo up
}
```

```
# Run command inside network namespace
as_ns () {
    NAME=$1
    NETNS=faucet- $\{NAME\}$ 
    shift
    sudo ip netns exec  $\{NETNS\}$   $\@$ 
}
```

```
# Clean up namespaces, bridges and processes created during faucet_
↳ tutorial
cleanup () {
    for NETNS in  $\$(sudo ip netns list | grep "faucet-" | awk '{print $1}$ 
↳ '); do
```

(continues on next page)



(continued from previous page)

```

[ -n "$NETNS" ] || continue
NAME=${NETNS#faucet-}
if [ -f "/run/dhclient-${NAME}.pid" ]; then
    # Stop dhclient
    sudo pkill -F "/run/dhclient-${NAME}.pid"
fi
if [ -f "/run/iperf3-${NAME}.pid" ]; then
    # Stop iperf3
    sudo pkill -F "/run/iperf3-${NAME}.pid"
fi
if [ -f "/run/bird-${NAME}.pid" ]; then
    # Stop bird
    sudo pkill -F "/run/bird-${NAME}.pid"
fi
# Remove netns and veth pair
sudo ip link delete veth-${NAME}
sudo ip netns delete $NETNS
done
for DNSMASQ in /run/dnsmasq-vlan*.pid; do
    [ -e "$DNSMASQ" ] || continue
    # Stop dnsmasq
    sudo pkill -F "$DNSMASQ"
done
# Remove faucet dataplane connection
sudo ip link delete veth-faucet 2>/dev/null || true
# Remove openvswitch bridge
sudo ovs-vsctl del-br br0
}

# Add tagged VLAN interface to network namespace
add_tagged_interface () {
    NAME=$1
    IP=$2
    VLAN=$3
    NETNS=faucet-${NAME}
    sudo ip netns exec ${NETNS} ip link add link veth0 name veth0.${VLAN}
↪ type vlan id $VLAN
    sudo ip netns exec ${NETNS} ip link set dev veth0.${VLAN} up
    sudo ip netns exec ${NETNS} ip addr flush dev veth0
    sudo ip netns exec ${NETNS} ip addr add dev veth0.${VLAN} $IP
}

```

## Overview

In this tutorial we will look at how to do the following tasks using Faucet:

- Use VLANs to segment traffic.
- Create VLAN Trunks.
- Apply an ACL to an entire VLAN.

---

**Note:** We cover *Routing between VLANs* in a later tutorial.

---

A port can be in several VLAN modes:

1. Native - where packets come into the switch with no 802.1Q tag.
2. Tagged - where packets come into the switch with a 802.1Q tag.
3. Mixed - where both native and tagged packets appear on the same port.

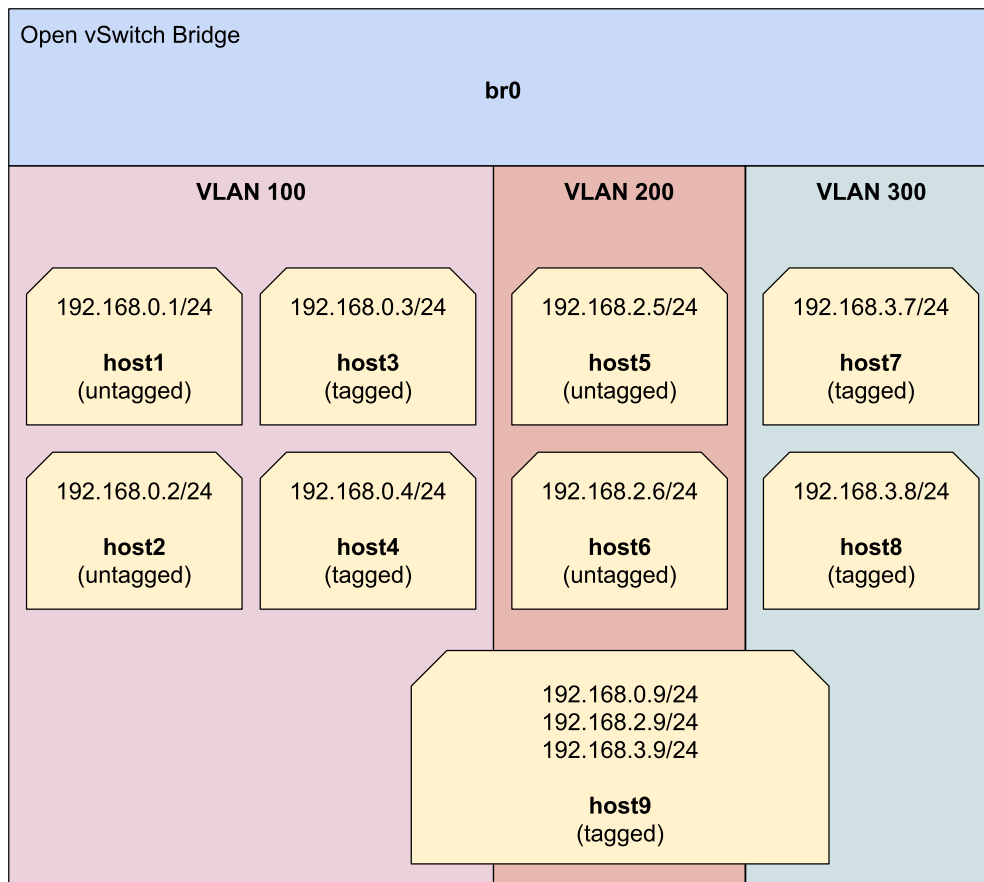
If a packet comes in with a tag for a VLAN that the port is not configured for it will be dropped.

### Configuring VLANs

To demonstrate these tasks we will use a demo network where a single switch br0 connects to 9 hosts.

Ports 1, 2, 5, 6 will be native (untagged) ports. While ports 3, 4, 7, 8, and 9 will be tagged ports.

Here is the structure of the demo setup.



---

**Tip:** Keep this diagram nearby to simplify following the rest of the tutorial.

---

### Network setup

Let's start. Keep host1, host2 on the native VLAN 100 (office VLAN) as in the first and second tutorials.

---

**Note:** To create the hosts and switch again run

```

cleanup
create_ns host1 192.168.0.1/24
create_ns host2 192.168.0.2/24
sudo ovs-vsctl add-br br0 \
-- set bridge br0 other-config:datapath-id=0000000000000001 \
-- set bridge br0 other-config:disable-in-band=true \
-- set bridge br0 fail_mode=secure \
-- add-port br0 veth-host1 -- set interface veth-host1 ofport_request=1 \
-- add-port br0 veth-host2 -- set interface veth-host2 ofport_request=2 \
-- set-controller br0 tcp:127.0.0.1:6653 tcp:127.0.0.1:6654

```

Then add the following hosts with the corresponding VLAN:

- Assign host3 and host4 a VLAN interface (vid:100) as they are on a tagged port.

```

create_ns host3 0.0.0.0
create_ns host4 0.0.0.0
add_tagged_interface host3 192.168.0.3/24 100
add_tagged_interface host4 192.168.0.4/24 100

```

- Assign host5 and host6 an IP address from the VLAN 200 range.

```

create_ns host5 192.168.2.5/24
create_ns host6 192.168.2.6/24

```

- Assign host7 and host8 a VLAN interface (vid:300) as they are on a tagged port.

```

create_ns host7 0.0.0.0
create_ns host8 0.0.0.0
add_tagged_interface host7 192.168.3.7/24 300
add_tagged_interface host8 192.168.3.8/24 300

```

- Add host9 to all VLANs (100, 200, 300) to work as a NFV host.

```

create_ns host9 0.0.0.0
add_tagged_interface host9 192.168.0.9/24 100
add_tagged_interface host9 192.168.2.9/24 200
add_tagged_interface host9 192.168.3.9/24 300

```

Then connect all the hosts to the switch (br0)

```

sudo ovs-vsctl add-port br0 veth-host3 -- set interface veth-host3 ofport_request=3 \
-- add-port br0 veth-host4 -- set interface veth-host4 ofport_request=4 \
-- add-port br0 veth-host5 -- set interface veth-host5 ofport_request=5 \
-- add-port br0 veth-host6 -- set interface veth-host6 ofport_request=6 \
-- add-port br0 veth-host7 -- set interface veth-host7 ofport_request=7 \
-- add-port br0 veth-host8 -- set interface veth-host8 ofport_request=8 \
-- add-port br0 veth-host9 -- set interface veth-host9 ofport_request=9

```

Now we have everything to start working with faucet through its configuration file. Each time we will only need to change the configuration file and restart faucet (or send it HUP signal to reload the configuration file).

## Basic VLAN settings

Change `/etc/faucet/faucet.yaml` to reflect our setting.

Listing 16: /etc/faucet/faucet.yaml

```
vlans:
  vlan100:
    vid: 100
  vlan200:
    vid: 200
  vlan300:
    vid: 300
dps:
  sw1:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        name: "host1"
        description: "host2 network namespace"
        native_vlan: vlan100
      2:
        name: "host2"
        description: "host2 network namespace"
        native_vlan: vlan100
      3:
        name: "host3"
        tagged_vlans: [vlan100]
      4:
        name: "host4"
        tagged_vlans: [vlan100]
      5:
        name: "host5"
        native_vlan: vlan200
      6:
        name: "host6"
        native_vlan: vlan200
      7:
        name: "host7"
        tagged_vlans: [vlan300]
      8:
        name: "host8"
        tagged_vlans: [vlan300]
      9:
        name: "host9"
        tagged_vlans: [vlan100, vlan200, vlan300]
```

Send SIGHUP signal to reload the configuration file, and check how its log the new configuration in /var/log/faucet/faucet.log

```
sudo systemctl reload faucet
cat /var/log/faucet/faucet.log
```

Let's do the following simple tests:

1. Ping between hosts in the same VLAN (all should work)

```
as_ns host1 ping 192.168.0.2
as_ns host3 ping 192.168.0.4
as_ns host5 ping 192.168.2.6
as_ns host7 ping 192.168.3.8
```

2. Ping between hosts in the same VLAN where the one host is native and the other is tagged should work also. In particular between host1 (native VLAN 100) to host3 (tagged VLAN 100).

```
as_ns host1 ping 192.168.0.3
```

3. Ping between hosts in different VLANs should fail. To test that let's add the IP address 192.168.0.5 to host5 (native VLAN 200) and try to ping it from host1 (native VLAN 100).

```
as_ns host5 ip address add 192.168.0.5 dev veth0
as_ns host1 ping 192.168.0.5
```

4. Now we can test the trunk link to host9 from different VLANs (all should work)

```
as_ns host1 ping 192.168.0.9
as_ns host3 ping 192.168.0.9
as_ns host5 ping 192.168.2.9
as_ns host7 ping 192.168.3.9
```

## VLAN ACL

Let's apply an ACL on a particular VLAN (e.g. VLAN 300). We will block any ICMP packets on VLAN 300. First create an ACL to block the ping. Open `/etc/faucet/faucet.yaml` and add the 'acls' section.

Listing 17: `/etc/faucet/faucet.yaml`

```
acls:
  block-ping:
    - rule:
      dl_type: 0x800      # IPv4
      ip_proto: 1        # ICMP
      actions:
        allow: False
    - rule:
      dl_type: 0x86dd    # IPv6
      ip_proto: 58       # ICMPv6
      actions:
        allow: False
```

Then apply this ACL on VLAN 300.

Listing 18: `/etc/faucet/faucet.yaml`

```
vlan:
  vlan100:
    vid: 100
  vlan200:
    vid: 200
  vlan300:
    vid: 300
    acls_in: [block-ping] # Apply ACL only on vlan300
```

Just before we reload the configuration file. Let's verify that pinging is working between hosts in VLAN 300.

```
as_ns host7 ping 192.168.3.8
```

Now let's apply the configuration, send SIGHUP signal to reload the configuration file.

```
sudo systemctl reload faucet
```

Now if you try to ping from host7 and host8, it will not work as it is specified by their VLAN ACL.

```
as_ns host7 ping 192.168.3.8
```

### 1.2.4 Routing tutorial

This tutorial will cover routing with Faucet.

There are three types of routing we can use.

- Inter-VLAN routing
- Static routing
- BGP via an external application (Quagga, Bird, EXABGP, etc)

#### Prerequisites

- Install Faucet - *Package installation* steps 1 & 2
- Install Open vSwitch - *Connect your first datapath* steps 1 & 2
- Useful Bash Functions - Copy and paste the following definitions into your bash terminal, or to make them persistent between sessions add them to the bottom of your `.bashrc` and run `source .bashrc`.

```
# Create network namespace
create_ns () {
    NAME=$1
    IP=$2
    NETNS=faucet-${NAME}
    sudo ip netns add ${NETNS}
    sudo ip link add dev veth-${NAME} type veth peer name veth0 netns
↪${NETNS}
    sudo ip link set dev veth-${NAME} up
    sudo ip netns exec ${NETNS} ip link set dev veth0 up
    sudo ip netns exec ${NETNS} ip addr add dev veth0 $IP
    sudo ip netns exec ${NETNS} ip link set dev lo up
}
```

```
# Run command inside network namespace
as_ns () {
    NAME=$1
    NETNS=faucet-${NAME}
    shift
    sudo ip netns exec $NETNS $@
}
```

```
# Clean up namespaces, bridges and processes created during faucet_
↪tutorial
cleanup () {
    for NETNS in $(sudo ip netns list | grep "faucet-" | awk '{print $1}
↪'); do
        [ -n "$NETNS" ] || continue
        NAME=${NETNS#faucet-}
```

(continues on next page)

(continued from previous page)

```

if [ -f "/run/dhclient-${NAME}.pid" ]; then
    # Stop dhclient
    sudo pkill -F "/run/dhclient-${NAME}.pid"
fi
if [ -f "/run/iperf3-${NAME}.pid" ]; then
    # Stop iperf3
    sudo pkill -F "/run/iperf3-${NAME}.pid"
fi
if [ -f "/run/bird-${NAME}.pid" ]; then
    # Stop bird
    sudo pkill -F "/run/bird-${NAME}.pid"
fi
# Remove netns and veth pair
sudo ip link delete veth-${NAME}
sudo ip netns delete $NETNS
done
for DNSMASQ in /run/dnsmasq-vlan*.pid; do
    [ -e "$DNSMASQ" ] || continue
    # Stop dnsmasq
    sudo pkill -F "$DNSMASQ"
done
# Remove faucet dataplane connection
sudo ip link delete veth-faucet 2>/dev/null || true
# Remove openvswitch bridge
sudo ovs-vsctl del-br br0
}

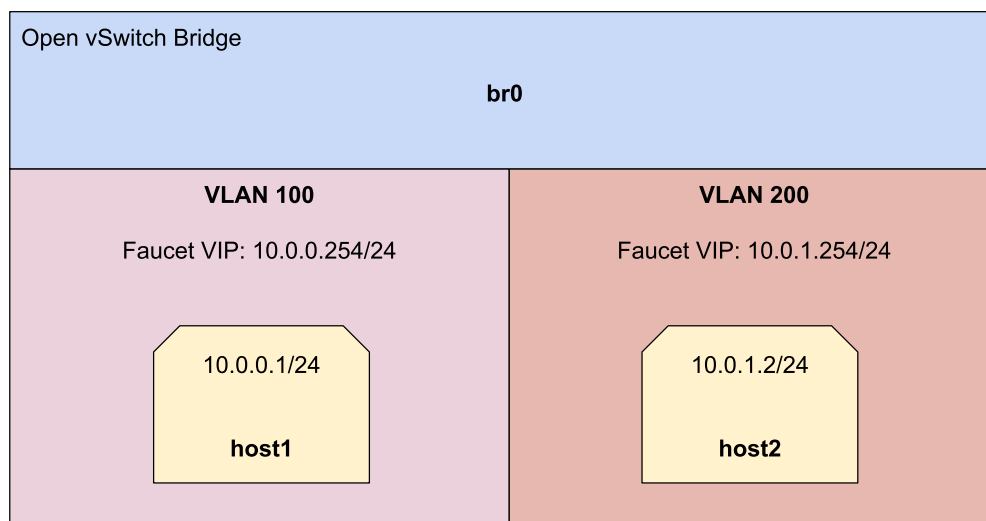
```

- Run the cleanup script to remove old namespaces and switches:

```
cleanup
```

## Routing between VLANs

Let's start with a single switch connected to two hosts in two different VLANs.



```
create_ns host1 10.0.0.1/24
create_ns host2 10.0.1.2/24
sudo ovs-vsctl add-br br0 \
-- set bridge br0 other-config:datapath-id=0000000000000001 \
-- set bridge br0 other-config:disable-in-band=true \
-- set bridge br0 fail_mode=secure \
-- add-port br0 veth-host1 -- set interface veth-host1 ofport_request=1 \
-- add-port br0 veth-host2 -- set interface veth-host2 ofport_request=2 \
-- set-controller br0 tcp:127.0.0.1:6653 tcp:127.0.0.1:6654
```

In this section we will be using faucet as a gateway for our two hosts and using faucet to route between them. To do this we are going to need to give faucet an IP address on the network. This is accomplished with by using two new options that we haven't seen before:

faucet_vips	The IP address for Faucet's routing interface on this VLAN. Multiple IP addresses (IPv4 & IPv6) can be used.
faucet_mac	The MAC address of Faucet's routing interface on this VLAN. If we do not set faucet_mac for each VLAN, routed packets will be dropped unless 'drop_spoofed_faucet_mac' is set to false.

Let's add the following faucet configuration which makes use of these options.

Listing 19: /etc/faucet/faucet.yaml

```
vlan:
  vlan100:
    vid: 100
    faucet_vips: ["10.0.0.254/24"] # Faucet's virtual IP address for vlan100
    faucet_mac: "00:00:00:00:00:11"
  vlan200:
    vid: 200
    faucet_vips: ["10.0.1.254/24"] # Faucet's virtual IP address for vlan200
    faucet_mac: "00:00:00:00:00:22"
dps:
  sw1:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        name: "host1"
        description: "host1 network namespace"
        native_vlan: vlan100
      2:
        name: "host2"
        description: "host2 network namespace"
        native_vlan: vlan200
```

Now lets signal faucet to reload the configuration file.

```
sudo systemctl reload faucet
```

Add a default route on each host to set the gateway to the value we used for faucet\_vips above.

```
as_ns host1 ip route add default via 10.0.0.254 dev veth0
as_ns host2 ip route add default via 10.0.1.254 dev veth0
```

By default traffic between our two hosts will be dropped since they are in different VLANs with different subnets. We can show that by doing the following:



```
as_ns host1 ping 10.0.1.2
```

We can change this by enabling inter-VLAN routing between these two VLANs. In faucet you do this by creating a router and specifying which VLANs can route between each other.

In our case we to enable routing between VLAN 100 and VLAN 200 so we add the following to our configuration file.

Listing 20: /etc/faucet/faucet.yaml

```

routers:
  router-1:                                # Router name
    vlans: [vlan100, vlan200]             # Names of vlans to allow routing between

```

Reload faucet to enable inter-VLAN routing.

```
sudo systemctl reload faucet
```

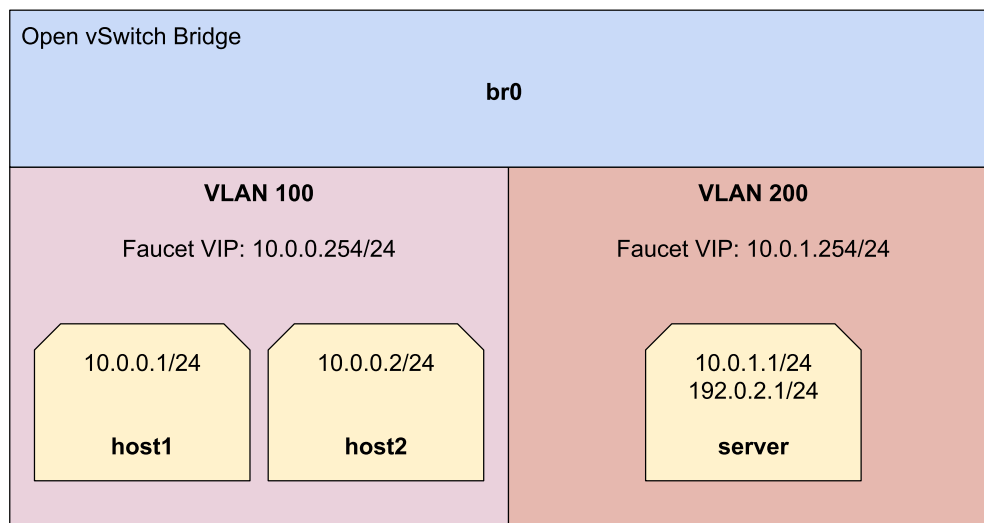
Our ping before from host1 to host2 should now work (the first few packets may get lost as faucet needs to resolve the MAC address of the next hop with ARP).

```
as_ns host1 ping 10.0.1.2
```

Inter-VLAN routing by default will allow all traffic to pass between VLANs, if we wanted to change this and restrict communication to a few different IP addresses or TCP/UDP ports, we could apply a VLAN ACL to each VLAN to limit the types of traffic that may pass and what should be dropped.

## Static routing

For this we will set-up a Faucet switch with three hosts. One of these hosts will act like a server.



Run the cleanup script to remove old namespaces and switches.

```
cleanup
```

Create 3 hosts, in 2 different subnets:

```
create_ns host1 10.0.0.1/24
create_ns host2 10.0.0.2/24
create_ns server 10.0.1.1/24
```

Add a default route for each host to the gateway which is faucet's virtual IP address.

```
as_ns host1 ip route add default via 10.0.0.254
as_ns host2 ip route add default via 10.0.0.254
as_ns server ip route add default via 10.0.1.254
```

Create the bridge and add host1, host2 and the server to br0.

```
sudo ovs-vsctl add-br br0 \
-- set bridge br0 other-config:datapath-id=0000000000000001 \
-- set bridge br0 other-config:disable-in-band=true \
-- set bridge br0 fail_mode=secure \
-- add-port br0 veth-host1 -- set interface veth-host1 ofport_request=1 \
-- add-port br0 veth-host2 -- set interface veth-host2 ofport_request=2 \
-- add-port br0 veth-server -- set interface veth-server ofport_request=3 \
-- set-controller br0 tcp:127.0.0.1:6653 tcp:127.0.0.1:6654
```

For this Faucet configuration we will start from scratch. First we need to define 2 VLANs one for hosts and one for servers. We will also note that inside the configuration for the servers VLAN we see a static route that routes the subnet 192.0.2.0/24 to the server namespace (10.0.1.1).

Listing 21: /etc/faucet/faucet.yaml

```
vlan:
  hosts:
    vid: 100
    description: "vlan for clients"
    faucet_mac: "00:00:00:00:00:11"
    faucet_vips: ["10.0.0.254/24"]

  servers:
    vid: 200
    description: "vlan for servers"
    faucet_mac: "00:00:00:00:00:22"
    faucet_vips: ["10.0.1.254/24"]
    routes:
      - route:
          ip_dst: "192.0.2.0/24"
          ip_gw: '10.0.1.1'

routers:
  router-hosts-servers:
    vlans: [hosts, servers]

dps:
  br0:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        name: "host1"
        description: "host1 network namespace"
        native_vlan: hosts
      2:
        name: "host2"
```

(continues on next page)

(continued from previous page)

```

        description: "host2 network namespace"
        native_vlan: hosts
    3:
        name: "server"
        description: "server network namespace"
        native_vlan: servers

```

Reload Faucet to apply the new configuration.

```
sudo systemctl reload faucet
```

We can verify the inter-VLAN Routing is working by pinging the IP address of the server namespace:

```
as_ns host1 ping 10.0.1.1
```

We also need to add an additional IP alias to server to test the static route works.

```
as_ns server ip address add 192.0.2.1/24 dev veth0
```

And we should now be able to ping our IP alias.

```
as_ns host1 ping 192.0.2.1
```

## BGP routing

For this section we are going to change our static routes from above into BGP routes.

BGP (and other routing) is provided by a NFV service, here we will use **BIRD**. Other applications such as ExaBGP & Quagga could be used. Faucet imports all routes provided by this NFV service. This means we can use our service for other routing protocols (OSPF, RIP, etc) and apply filtering using the service's policy language.

## Setup

Our data plane will end up looking like below, you may notice how we have the Faucet application connected to the control plane and dataplane.

Remove the following lines from `/etc/faucet/faucet.yaml` to remove the static route from faucet:

Listing 22: `/etc/faucet/faucet.yaml`

```

routes:
  - route:
      ip_dst: "192.0.2.0/24"
      ip_gw: '10.0.1.1'

```

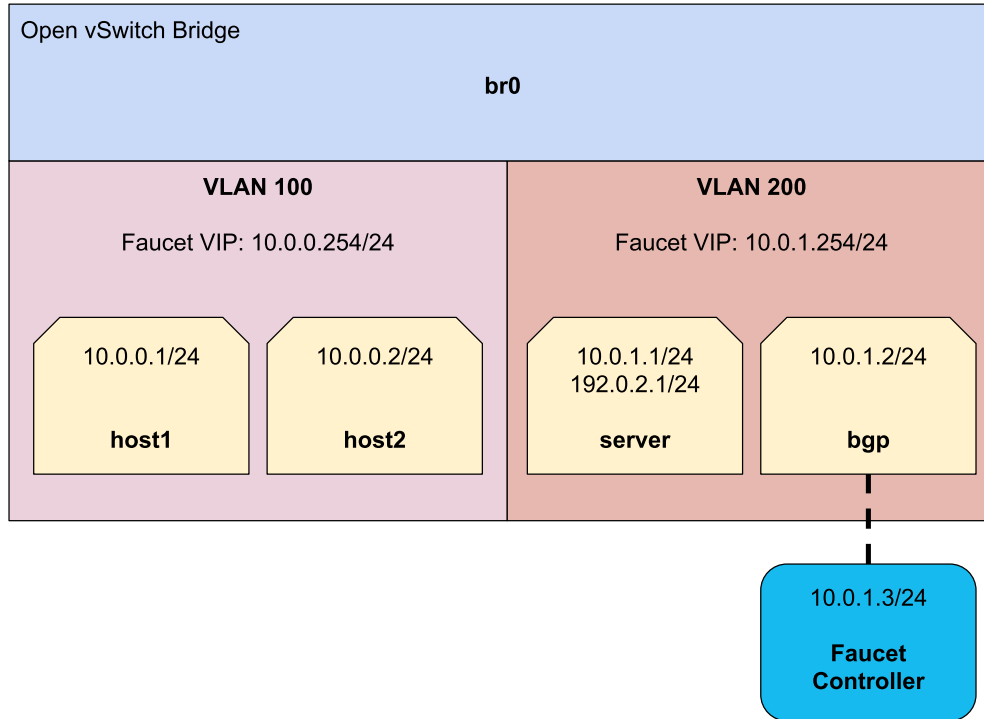
Reload Faucet

```
sudo systemctl reload faucet
```

Verify that we can no longer ping the address we were previously static routing.

```
as_ns host1 ping 192.0.2.1
```

Let's add a new network namespace to run BIRD



```
create_ns bgp 10.0.1.2/24
sudo ovs-vsctl add-port br0 veth-bgp -- set interface veth-bgp ofport_request=4
```

Next we will add a dataplane connection for Faucet so that it can communicate with BIRD running on the bgp namespace.

```
sudo ip link add veth-faucet type veth peer name veth-faucet-ovs
sudo ovs-vsctl add-port br0 veth-faucet-ovs -- set interface veth-faucet-ovs ofport_
->request=5
sudo ip addr add 10.0.1.3/24 dev veth-faucet
sudo ip link set veth-faucet up
sudo ip link set veth-faucet-ovs up
```

Now install BIRD on the system and stop it from running:

```
sudo apt-get install bird
sudo systemctl stop bird
sudo systemctl stop bird6
```

To configure BIRD add the following to `/etc/bird/bird.conf`, this will create a simple routing setup where BIRD originates a static route for 192.0.2.0/24 and sends this to faucet over BGP.

Listing 23: `/etc/bird/bird.conf`

```
protocol kernel {
    scan time 60;
    import none;
}

protocol device {
```

(continues on next page)

(continued from previous page)

```

    scan time 60;
}

# Generate static route inside bird
protocol static {
    route 192.0.2.0/24 via 10.0.1.1;
}

# BGP peer with faucet
# Import all routes and export our static route
protocol bgp faucet {
    local as 65001;
    neighbor 10.0.1.3 port 9179 as 65000;
    export all;
    import all;
}

```

We can now start BIRD inside the bgp namespace:

```
as_ns bgp bird -P /run/bird-bgp.pid
```

We'll configure Faucet to talk to BIRD by adding BGP configuration to `/etc/faucet/faucet.yaml`. Change the servers VLAN to look like the configuration below, leaving all other VLANs alone, and add a Faucet router.

Listing 24: `/etc/faucet/faucet.yaml`

```

routers:
  bird:
    vlans: servers
    bgp:
      as: 65000 # Faucet's AS number
      port: 9179 # BGP port for Faucet to listen on.
      routerid: '10.0.1.3' # Faucet's Unique ID.
      server_addresses: ['10.0.1.3'] # Faucet's listen IP for BGP
      neighbor_addresses: ['10.0.1.2'] # Neighbouring IP addresses (IPv4/
↔ IPv6)
      neighbor_as: 65001 # Neighbour's AS number
  vlans:
    servers:
      vid: 200
      description: "vlan for gw port"
      faucet_mac: "00:00:00:00:00:22"
      faucet_vips: ["10.0.1.254/24"]
    ...

```

And finally add the port configuration for the Faucet data plane interface (veth-faucet0).

Listing 25: `/etc/faucet/facuet.yaml`

```

dps:
  br0:
    ...
    interfaces:
      ...
      4:
        name: "bgp"

```

(continues on next page)

(continued from previous page)

```

description: "BIRD BGP router"
native_vlan: servers
5:
name: "faucet"
description: "faucet dataplane connection"
native_vlan: servers

```

Now reload Faucet.

```
sudo systemctl reload faucet
```

We can use the command line tool `birdc` to query the status of our peering connection, we should see that it is now established:

```

as_ns bgp birdc show protocols all faucet

name      proto   table   state  since      info
faucet    BGP     master  up     13:25:38   Established
Preference: 100
Input filter: ACCEPT
Output filter: ACCEPT
Routes:      1 imported, 1 exported, 1 preferred
Route change stats:      received  rejected  filtered  ignored  accepted
Import updates:          1          0         0         0         1
Import withdraws:        0          0        ---         0         0
Export updates:           2          1         0        ---         1
Export withdraws:        0          ---        ---        ---         0
BGP state:      Established
Neighbor address: 10.0.1.3
Neighbor AS:      65000
Neighbor ID:      10.0.1.3
Neighbor caps:    AS4
Session:          external AS4
Source address:   10.0.1.2
Hold timer:       185/240
Keepalive timer:  57/80

```

Using `birdc` we can also check what routes are being exported to faucet:

```

as_ns bgp birdc show route export faucet

192.0.2.0/24      via 10.0.1.1 on veth0 [static1 13:25:34] * (200)

```

And which routes bird receives from faucet:

```

as_ns bgp birdc show route protocol faucet

10.0.1.0/24      via 10.0.1.254 on veth0 [faucet 13:25:38 from 10.0.1.3] * (100) [i]

```

In `/var/log/faucet/faucet.log` we should now see log messages relating to BGP:

Listing 26: `/var/log/faucet/faucet.log`

```

Jan 16 13:25:17 faucet      INFO      Reloading configuration
Jan 16 13:25:17 faucet      INFO      configuration /etc/faucet/faucet.yaml changed, ↵
↵analyzing differences

```

(continues on next page)

(continued from previous page)

```

Jan 16 13:25:17 faucet      INFO      Add new datapath DPID 1 (0x1)
Jan 16 13:25:17 faucet      INFO      Adding BGP speaker key DP ID: 1, VLAN VID: 200,
↪IP version: 4 for VLAN servers vid:200 untagged: Port 3,Port 4,Port 5
Jan 16 13:25:38 faucet      INFO      BGP peer router ID 10.0.1.2 AS 65001 up
Jan 16 13:25:38 faucet      INFO      BGP add 192.0.2.0/24 nexthop 10.0.1.1
Jan 16 13:25:42 faucet.valve INFO      DPID 1 (0x1) br0 resolving 10.0.1.1 (1 flows)
↪on VLAN 200
Jan 16 13:25:42 faucet.valve INFO      DPID 1 (0x1) br0 Adding new route 192.0.2.0/24
↪via 10.0.1.1 (aa:97:cd:33:74:a9) on VLAN 200

```

Once confirming the BGP connection is up between BIRD and faucet and the correct routes are being advertised, we should now be able to ping the IP alias on the server namespace again:

```
as_ns host1 ping 192.0.2.1
```

## 1.2.5 NFV services tutorial

This tutorial will cover using faucet with Network Function Virtualisation (NFV) style services.

NFV services that will be demonstrated in this tutorial are:

- DHCP/DNS server
- [Zeek](#) (formerly known as Bro) Intrusion Detection System (IDS)

This tutorial demonstrates how the previous topics in this tutorial series can be combined to run real world services on our network.

### Prerequisites

- A good understanding of the previous tutorial topics ([ACL tutorial](#), [VLAN tutorial](#), [Routing tutorial](#))
- Install Faucet - [Package installation](#) steps 1 & 2
- Install Open vSwitch - [Connect your first datapath](#) steps 1 & 2
- Useful Bash Functions - Copy and paste the following definitions into your bash terminal, or to make them persistent between sessions add them to the bottom of your `.bashrc` and run `'source .bashrc'`.

```

# Create network namespace
create_ns () {
    NAME=$1
    IP=$2
    NETNS=faucet- $\{NAME\}$ 
    sudo ip netns add  $\{NETNS\}$ 
    sudo ip link add dev veth- $\{NAME\}$  type veth peer name veth0 netns
    ↪ $\{NETNS\}$ 
    sudo ip link set dev veth- $\{NAME\}$  up
    sudo ip netns exec  $\{NETNS\}$  ip link set dev veth0 up
    sudo ip netns exec  $\{NETNS\}$  ip addr add dev veth0  $\{IP\}$ 
    sudo ip netns exec  $\{NETNS\}$  ip link set dev lo up
}

```

```

# Run command inside network namespace
as_ns () {
    NAME=$1

```

(continues on next page)

(continued from previous page)

```

NETNS=faucet- $\{NAME\}$ 
shift
sudo ip netns exec  $\$NETNS$   $\$@$ 
}

```

```

# Clean up namespaces, bridges and processes created during faucet_
↳ tutorial
cleanup () {
    for NETNS in  $\$(sudo ip netns list | grep "faucet-" | awk '{print $1}$ 
↳ '); do
        [ -n " $\$NETNS$ " ] || continue
        NAME= $\{NETNS\}$ #faucet-
        if [ -f "/run/dhclient- $\{NAME\}$ .pid" ]; then
            # Stop dhclient
            sudo pkill -F "/run/dhclient- $\{NAME\}$ .pid"
        fi
        if [ -f "/run/iperf3- $\{NAME\}$ .pid" ]; then
            # Stop iperf3
            sudo pkill -F "/run/iperf3- $\{NAME\}$ .pid"
        fi
        if [ -f "/run/bird- $\{NAME\}$ .pid" ]; then
            # Stop bird
            sudo pkill -F "/run/bird- $\{NAME\}$ .pid"
        fi
        # Remove netns and veth pair
        sudo ip link delete veth- $\{NAME\}$ 
        sudo ip netns delete  $\$NETNS$ 
    done
    for DNSMASQ in /run/dnsmasq-vlan*.pid; do
        [ -e " $\$DNSMASQ$ " ] || continue
        # Stop dnsmasq
        sudo pkill -F " $\{DNSMASQ\}$ "
    done
    # Remove faucet dataplane connection
    sudo ip link delete veth-faucet 2>/dev/null || true
    # Remove openvswitch bridge
    sudo ovs-vsctl del-br br0
}

```

```

# Add tagged VLAN interface to network namespace
add_tagged_interface () {
    NAME= $\$1$ 
    IP= $\$2$ 
    VLAN= $\$3$ 
    NETNS=faucet- $\{NAME\}$ 
    sudo ip netns exec  $\{NETNS\}$  ip link add link veth0 name veth0. $\{VLAN\}$ 
↳ type vlan id  $\$VLAN$ 
    sudo ip netns exec  $\{NETNS\}$  ip link set dev veth0. $\{VLAN\}$  up
    sudo ip netns exec  $\{NETNS\}$  ip addr flush dev veth0
    sudo ip netns exec  $\{NETNS\}$  ip addr add dev veth0. $\{VLAN\}$   $\$IP$ 
}

```

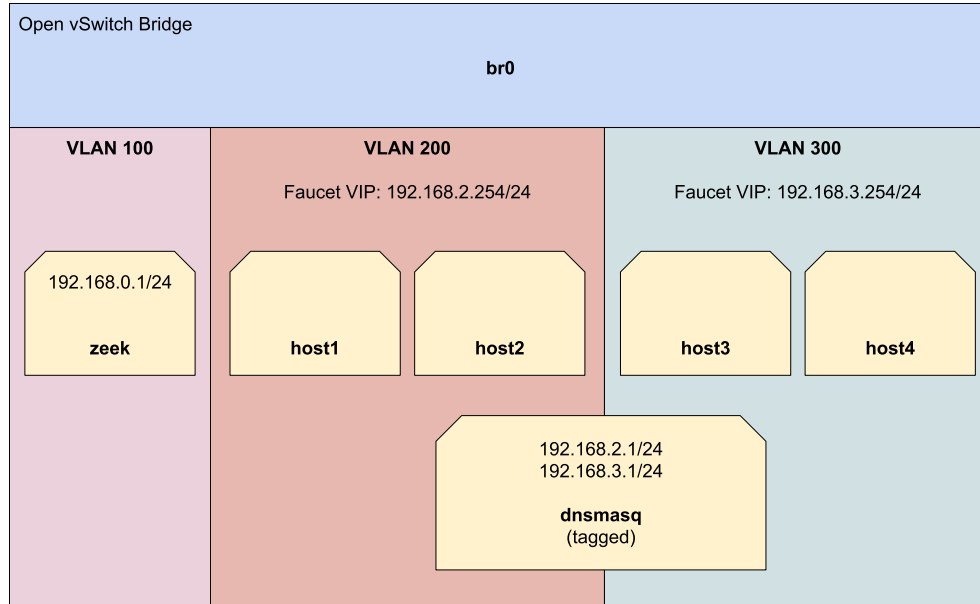
- Run the cleanup script to remove old namespaces and switches:

```
cleanup
```



## Network setup

The network will be divided into three VLANs, two of which are client VLANs (200 & 300), with two clients in each and a DHCP/DNS server. There is also a separate VLAN 100 for the Zeek server which we will mirror traffic two from the other two VLANs.



To start, let's create our hosts and dnsmasq namespaces.

```
# DHCP/DNS server
create_ns dnsmasq 0.0.0.0
add_tagged_interface dnsmasq 192.168.2.1/24 200 # to serve VLAN 200
add_tagged_interface dnsmasq 192.168.3.1/24 300 # to serve VLAN 300

# VLAN 200 hosts
create_ns host1 0.0.0.0
create_ns host2 0.0.0.0
# VLAN 300 hosts
create_ns host3 0.0.0.0
create_ns host4 0.0.0.0
```

Then create an Open vSwitch bridge and connect all hosts to it.

```
sudo ovs-vsctl add-br br0 \
-- set bridge br0 other-config:datapath-id=0000000000000001 \
-- set bridge br0 other-config:disable-in-band=true \
-- set bridge br0 fail_mode=secure \
-- add-port br0 veth-host1 -- set interface veth-host1 ofport_request=1 \
-- add-port br0 veth-host2 -- set interface veth-host2 ofport_request=2 \
-- add-port br0 veth-host3 -- set interface veth-host3 ofport_request=3 \
-- add-port br0 veth-host4 -- set interface veth-host4 ofport_request=4 \
-- add-port br0 veth-dnsmasq -- set interface veth-dnsmasq ofport_request=5 \
-- set-controller br0 tcp:127.0.0.1:6653 tcp:127.0.0.1:6654
```

### Dnsmasq setup

We will use `dnsmasq` to assign IP addresses to our host namespaces via the DHCP protocol. It will also serve as our DNS resolver for the hosts.

First install `dnsmasq`:

```
sudo apt-get install dnsmasq
sudo systemctl stop dnsmasq
```

Run the following two commands to start two `dnsmasq` instances inside the `dnsmasq` namespace. One instance will serve hosts on VLAN 200 and the other VLAN 300. We will be providing DHCP leases in the supplied ranges, the lease will set the gateway for each host to point at faucet's virtual IP and set `dnsmasq` as the DNS resolver. We also provide a fake `does.it.work` DNS name which we will later use to demonstrate DNS is working as expected.

```
# 192.168.2.0/24 for VLAN 200
as_ns dnsmasq dnsmasq \
    --dhcp-range=192.168.2.10,192.168.2.20 \
    --dhcp-sequential-ip \
    --dhcp-option=option:router,192.168.2.254 \
    --no-resolv \
    --txt-record=does.it.work,yes \
    --bind-interfaces \
    --except-interface=lo --interface=veth0.200 \
    --dhcp-leasefile=/tmp/nfv-dhcp-vlan200.leases \
    --log-facility=/tmp/nfv-dhcp-vlan200.log \
    --pid-file=/run/dnsmasq-vlan200.pid \
    --conf-file=

# 192.168.3.0/24 for VLAN 300
as_ns dnsmasq dnsmasq \
    --dhcp-range=192.168.3.10,192.168.3.20 \
    --dhcp-sequential-ip \
    --dhcp-option=option:router,192.168.3.254 \
    --no-resolv \
    --txt-record=does.it.work,yes \
    --bind-interfaces \
    --except-interface=lo --interface=veth0.300 \
    --dhcp-leasefile=/tmp/nfv-dhcp-vlan300.leases \
    --log-facility=/tmp/nfv-dhcp-vlan300.log \
    --pid-file=/run/dnsmasq-vlan300.pid \
    --conf-file=
```

Now let's configure `faucet.yaml`.

Listing 27: `/etc/faucet/faucet.yaml`

```
vlan:
  vlan200:
    vid: 200
    description: "192.168.2.0/24 network"
    faucet_vips: ["192.168.2.254/24"]
    faucet_mac: "00:00:00:00:00:22"
  vlan300:
    vid: 300
    description: "192.168.3.0/24 network"
    faucet_vips: ["192.168.3.254/24"]
    faucet_mac: "00:00:00:00:00:33"
```

(continues on next page)

(continued from previous page)

```
dps:
  sw1:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        name: "host1"
        description: "host1 network namespace"
        native_vlan: vlan200
      2:
        name: "host2"
        description: "host2 network namespace"
        native_vlan: vlan200
      3:
        name: "host3"
        description: "host3 network namespace"
        native_vlan: vlan300
      4:
        name: "host4"
        description: "host4 network namespace"
        native_vlan: vlan300
      5:
        name: "dnsmasq"
        description: "dnsmasq server network namespace"
        tagged_vlans: [vlan200, vlan300]
```

Now reload faucet configuration file.

```
sudo systemctl reload faucet
```

Use dhclient to configure host1 to host4 using DHCP (it may take a few seconds, but should return when successful).

```
as_ns host1 dhclient -v -pf /run/dhclient-host1.pid -lf /run/dhclient-host1.leases_
↪veth0
as_ns host2 dhclient -v -pf /run/dhclient-host2.pid -lf /run/dhclient-host2.leases_
↪veth0
as_ns host3 dhclient -v -pf /run/dhclient-host3.pid -lf /run/dhclient-host3.leases_
↪veth0
as_ns host4 dhclient -v -pf /run/dhclient-host4.pid -lf /run/dhclient-host4.leases_
↪veth0
```

If dhclient is unable to obtain an address you can check `/tmp/nfv-dhcp-vlan<vlanid>.log` (e.g `/tmp/nfv-dhcp-vlan300.leases`) to check the log messages from dnsmasq.

To look up the address for each namespace we can run the following commands:

```
as_ns host1 ip address show dev veth0
as_ns host2 ip address show dev veth0
as_ns host3 ip address show dev veth0
as_ns host4 ip address show dev veth0
```

If the hosts have IPs then great our DHCP server works.

At the moment we should be able to ping inside VLAN 200 and VLAN 300:

```
as_ns host1 ping <ip of host2> # both in VLAN 200 should work
as_ns host3 ping <ip of host4> # both in VLAN 300 should work
```

Pinging between VLANs will not currently work as we didn't turn on inter-VLAN routing in our faucet configuration.

### DNS

We can use faucet to enforce where protocols such as DNS go on the network. In this section we will use a faucet ACL to rewrite DNS packets to allow our dnsmasq namespace to answer DNS queries for any IP address.

Firstly, we can see that our dnsmasq server is correctly responding to DNS requests by manually querying them:

```
as_ns host1 host -t txt does.it.work 192.168.2.1
as_ns host3 host -t txt does.it.work 192.168.3.1
```

Both commands should return:

```
does.it.work descriptive text "yes"
```

But if we tried to query say 8.8.8.8 we would see this fail:

```
as_ns host1 host -t txt does.it.work 8.8.8.8
```

To make this work we first need the MAC address of the dnsmasq container:

```
as_ns dnsmasq cat /sys/class/net/veth0/address

00:11:22:33:44:55
```

We now replace our previous faucet configuration with the configuration below which adds an ACL that rewrites the MAC address of all DNS packets from the host namespaces and sends these to our dnsmasq namespace. Make sure to update the example MAC address of 00:11:22:33:44:55 with the one you get from running the previous command.

Listing 28: /etc/faucet/faucet.yaml

```
vlan:
  vlan200:
    vid: 200
    description: "192.168.2.0/24 network"
    faucet_vips: ["192.168.2.254/24"]
    faucet_mac: "00:00:00:00:00:22"
  vlan300:
    vid: 300
    description: "192.168.3.0/24 network"
    faucet_vips: ["192.168.3.254/24"]
    faucet_mac: "00:00:00:00:00:33"
dps:
  sw1:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        name: "host1"
        description: "host1 network namespace"
        native_vlan: vlan200
        acls_in: [nfv-dns, allow-all]
      2:
        name: "host2"
        description: "host2 network namespace"
```

(continues on next page)

(continued from previous page)

```

        native_vlan: vlan200
        acls_in: [nfv-dns, allow-all]
    3:
        name: "host3"
        description: "host3 network namespace"
        native_vlan: vlan300
        acls_in: [nfv-dns, allow-all]
    4:
        name: "host4"
        description: "host4 network namespace"
        native_vlan: vlan300
        acls_in: [nfv-dns, allow-all]
    5:
        name: "dnsmasq"
        description: "dnsmasq server network namespace"
        tagged_vlans: [vlan200, vlan300]
acls:
  nfv-dns:
    # Force UDP DNS to our DNS server
    - rule:
        dl_type: 0x800      # ipv4
        nw_proto: 17       # udp
        udp_dst: 53        # dns
        actions:
          output:
            set_fields:
              - eth_dst: "00:11:22:33:44:55" # MAC address of dnsmasq_
↪namespace
          allow: True
    # Force TCP DNS to our DNS server
    - rule:
        dl_type: 0x800      # ipv4
        nw_proto: 6        # tcp
        tcp_dst: 53        # dns
        actions:
          output:
            set_fields:
              - eth_dst: "00:11:22:33:44:55" # MAC address of dnsmasq_
↪namespace
          allow: True
  allow-all:
    - rule:
        actions:
          allow: True

```

As usual reload faucet configuration file.

```
sudo systemctl reload faucet
```

The next step is to configure the namespace to be able to handle incoming DNS packets with any IP, this can be done by adding some rules to iptables that will NAT all DNS traffic to the IP address of the VLAN interface:

```
as_ns dnsmasq iptables -t nat -A PREROUTING -i veth0.200 -p udp --dport 53 -j DNAT --
↪to-destination 192.168.2.1
as_ns dnsmasq iptables -t nat -A PREROUTING -i veth0.200 -p tcp --dport 53 -j DNAT --
↪to-destination 192.168.2.1

```

(continues on next page)

(continued from previous page)

```
as_ns dnsmasq iptables -t nat -A PREROUTING -i veth0.300 -p udp --dport 53 -j DNAT --
↳to-destination 192.168.3.1
as_ns dnsmasq iptables -t nat -A PREROUTING -i veth0.300 -p tcp --dport 53 -j DNAT --
↳to-destination 192.168.3.1
```

Now we should be able to query any IP address from the hosts and get a valid DNS response:

```
as_ns host1 host -t txt does.it.work 8.8.8.8
as_ns host2 host -t txt does.it.work 8.8.4.4
```

## Zeek IDS

We will now add an IDS to our network on it's on separate VLAN and use faucet to mirror packets from VLAN 200 and 300 to the IDS VLAN.

## Zeek installation

We need first to install Zeek (formerly known as Bro).

```
sudo apt-get install bro broctl
```

## Configure Zeek

In `/etc/bro/node.cfg`, set `veth0` as the interface to monitor

Listing 29: `/etc/bro/node.cfg`

```
[bro]
type=standalone
host=localhost
interface=veth0
```

Comment out `MailTo` in `/etc/bro/broctl.cfg`

Listing 30: `/etc/bro/broctl.cfg`

```
# Recipient address for all emails sent out by bro and BroControl.
# MailTo = root@localhost
```

## Run Zeek

Firstly, let's create a namespace to run Zeek inside:

```
create_ns zeek 192.168.0.1
sudo ovs-vsctl add-port br0 veth-zeek -- set interface veth-zeek ofport_request=6
```

Since this is the first-time use of the Zeek command shell application, perform an initial installation of the BroControl configuration:

```
as_ns zeek broctl install
```

Then start Zeek instant

```
as_ns zeek broctl start
```

Check Zeek status

```
as_ns zeek broctl status
```

Name	Type	Host	Status	Pid	Started
bro	standalone	localhost	running	15052	07 May 09:03:59

Now let's add a mirror ACL so all VLAN 200 & VLAN 300 traffic is sent to Zeek.

We will use a VLAN ACLs similar to the previous VLAN tutorial. Copy and paste the entire configuration below into faucet.yaml.

Listing 31: /etc/faucet/faucet.yaml

```

acls:
  mirror-acl:
    - rule:
      actions:
        allow: true
        mirror: zeek
vlans:
  zeek-vlan:
    vid: 100
    description: "Zeek IDS network"
  vlan200:
    vid: 200
    description: "192.168.2.0/24 network"
    faucet_vips: ["192.168.2.254/24"]
    faucet_mac: "00:00:00:00:00:22"
    acls_in: [mirror-acl]
  vlan300:
    vid: 300
    description: "192.168.3.0/24 network"
    faucet_vips: ["192.168.3.254/24"]
    faucet_mac: "00:00:00:00:00:33"
    acls_in: [mirror-acl]
dps:
  sw1:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        name: "host1"
        description: "host1 network namespace"
        native_vlan: vlan200
      2:
        name: "host2"
        description: "host2 network namespace"
        native_vlan: vlan200
      3:
        name: "host3"
        description: "host3 network namespace"

```

(continues on next page)

(continued from previous page)

```

        native_vlan: vlan300
4:
    name: "host4"
    description: "host4 network namespace"
    native_vlan: vlan300
5:
    name: "dnsmasq"
    description: "dnsmasq server network namespace"
    tagged_vlans: [vlan200, vlan300]
6:
    name: "zeek"
    description: "Zeek network namespace"
    native_vlan: zeek-vlan

```

As usual reload faucet configuration file.

```
sudo systemctl reload faucet
```

If we generate some DNS traffic on either of the hosts VLANs

```
as_ns host4 host -t txt does.it.work 192.168.3.1
```

Then if we inspect the Zeek logs for DNS `/var/log/bro/current/dns.log`, we should see that Zeek has seen the DNS queries and logged these.

Listing 32: `/var/log/bro/current/dns.log`

```

#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path dns
#open 2019-01-17-17-43-56
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p
↪ proto trans_id rtt query qclass qclass_name qtype qtype_
↪ name rcode rcode_name AA TC RD RA Z answers_
↪ TTls rejected
#types time string addr port addr port enum count interval
↪ string count string count string count string bool bool bool
↪ bool count vector[string] vector[interval] bool
1547700236.794299 CsulWM1Px7fIyPpCVi 192.168.3.10 43428 192.168.3.1
↪53 udp 14288 0.006973does.it.work 1 C_INTERNET 16 TXT
↪ 0 NOERROR T F T T 2 TXT 3 yes 0.
↪000000 F
1547700379.311319 CZalloBd3CgWBmgS8 192.168.3.11 45089 192.168.3.1
↪53 udp 64001 0.000336does.it.work 1 C_INTERNET 16 TXT
↪ 0 NOERROR T F T T 0 TXT 3 yes 0.
↪000000 F

```

You can also check if the traffic is being mirrored as expected using `tcpdump` in the `zeek` network namespace:

```
as_ns zeek sudo tcpdump -i veth0 -n -l
```

in one window, and then generating some more DNS traffic, eg:

```
as_ns host4 host -t txt does.it.work 192.168.3.1
```



then you should see something like:

Listing 33: zeek namespace tcpdump output

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on veth0, link-type EN10MB (Ethernet), capture size 262144 bytes
12:19:24.624244 IP 192.168.3.13.38174 > 192.168.3.1.53: 64571+ TXT? does.it.work. (30)
12:19:24.625109 IP 192.168.3.1.53 > 192.168.3.13.38174: 64571* 1/0/0 TXT "yes" (46)
```

## 1.3 Installation

We recommend installing faucet with apt for first time users and provide a *Installing faucet for the first time* tutorial which walks you through all the required steps for setting up faucet and gauge for the first time.

Once installed, see *Configuration* for documentation on how to configure faucet. Also, see *Vendor-specific Documentation* for documentation on how to configure your switch.

More advanced methods of installing faucet are also available here:

1. *Installation using APT*
2. *Installation with Docker*
3. *Installation with Pip*
4. *Installing on Raspberry Pi*
5. *Installing with Virtual Machine image*

### 1.3.1 Installation using APT

We maintain a apt repo for installing faucet and its dependencies on Debian-based Linux distributions.

Here is a list of packages we supply:

Package	Description
python3-faucet	Install standalone faucet/gauge python3 library
faucet	Install python3 library, systemd service and default config files
gauge	Install python3 library, systemd service and default config files
faucet-all-in-one	Install faucet, gauge, prometheus and grafana. Easy to use and good for testing faucet for the first time.

### Installation on Debian/Raspbian 9+ and Ubuntu 16.04+

```
sudo apt-get install curl gnupg apt-transport-https lsb-release
echo "deb https://packagecloud.io/faucetsdn/faucet/$(lsb_release -si | awk '{print_
↳tolower($0)}')/ $(lsb_release -sc) main" | sudo tee /etc/apt/sources.list.d/faucet.
↳list
curl -L https://packagecloud.io/faucetsdn/faucet/gpgkey | sudo apt-key add -
sudo apt-get update
```

Then to install all components for a fully functioning system on a single machine:

```
sudo apt-get install faucet-all-in-one
```

or you can install the individual components:

```
sudo apt-get install faucet
sudo apt-get install gauge
```

### 1.3.2 Installation with Docker

We provide official automated builds on [Docker Hub](#) so that you can easily run Faucet and its components in a self-contained environment without installing on the main host system.

#### Installing docker

We recommend installing Docker Community Edition (CE) according to the official [docker engine installation guide](#).

#### Configuring dockers

First, we need to create some configuration files on our host to mount inside the docker containers to configure faucet and gauge:

```
sudo mkdir -p /etc/faucet
sudo vi /etc/faucet/faucet.yaml
sudo vi /etc/faucet/gauge.yaml
```

See the [Configuration](#) section for configuration options.

#### Starting dockers

We use Docker tags to differentiate between versions of Faucet. The latest tag will always point to the latest stable release of Faucet. All tagged versions of Faucet in git are also available to use, for example using the `faucet/faucet:1.8.0` Docker will run the released version 1.8.0 of Faucet.

By default the Faucet and Gauge images are run as the `faucet` user under UID 0, GID 0. If you need to change that it can be overridden at runtime with the Docker flags: `-e LOCAL_USER_ID` and `-e LOCAL_GROUP_ID`.

To pull and run the latest version of Faucet:

```
mkdir -p /var/log/faucet/
docker pull faucet/faucet:latest
docker run -d \
  --name faucet \
  --restart=always \
  -v /etc/faucet:/etc/faucet/ \
  -v /var/log/faucet:/var/log/faucet/ \
  -p 6653:6653 \
  -p 9302:9302 \
  faucet/faucet
```

Port 6653 is used for OpenFlow, port 9302 is used for Prometheus - port 9302 may be omitted if you do not need Prometheus.

To pull and run the latest version of Gauge:

```
mkdir -p /var/log/faucet/gauge/
docker pull faucet/gauge:latest
docker run -d \
  --name gauge \
  --restart=always \
  -v /etc/faucet:/etc/faucet/ \
  -v /var/log/faucet:/var/log/faucet/ \
  -p 6654:6653 \
  -p 9303:9303 \
  faucet/gauge
```

Port 6654 is used for OpenFlow, port 9303 is used for Prometheus - port 9303 may be omitted if you do not need Prometheus.

### Additional arguments

You may wish to run faucet under docker with additional arguments, for example: setting certificates for an encrypted control channel. This can be done by overriding the docker entrypoint like so:

```
docker run -d \
  --name faucet \
  --restart=always \
  -v /etc/faucet:/etc/faucet/ \
  -v /etc/ryu/ssl:/etc/ryu/ssl/ \
  -v /var/log/faucet:/var/log/faucet/ \
  -p 6653:6653 \
  -p 9302:9302 \
  faucet/faucet \
  faucet \
  --ctl-privkey /etc/ryu/ssl/ctrlr.key \
  --ctl-cert /etc/ryu/ssl/ctrlr.cert \
  --ca-certs /etc/ryu/ssl/sw.cert
```

You can get a list of all additional arguments faucet supports by running:

```
docker run -it faucet/faucet faucet --help
```

### Docker compose

This is an example docker-compose file that can be used to set up gauge to talk to Prometheus and InfluxDB with a Grafana instance for dashboards and visualisations.

It can be run with:

```
docker-compose pull
docker-compose up
```

The time-series databases with the default settings will write to `/opt/prometheus/ /opt/influxdb/ shared/data/db` you can edit these locations by modifying the `docker-compose.yaml` file.

On OSX, some of the default shared paths are not accessible, so to overwrite the location that volumes are written to on your host, export an environment variable name `FAUCET_PREFIX` and it will get prepended to the host paths. For example:

```
export FAUCET_PREFIX=/opt/faucet
```

When all the docker containers are running we will need to configure Grafana to talk to Prometheus and InfluxDB. First login to the Grafana web interface on port 3000 (e.g <http://localhost:3000>) using the default credentials of `admin:admin`.

Then add two data sources. Use the following settings for prometheus:

```
Name: Prometheus
Type: Prometheus
Url: http://prometheus:9090
```

And the following settings for InfluxDB:

```
Name: InfluxDB
Type: InfluxDB
Url: http://influxdb:8086
With Credentials: true
Database: faucet
User: faucet
Password: faucet
```

Check the connection using test connection.

From here you can add a new dashboard and a graphs for pulling data from the data sources. Hover over the + button on the left sidebar in the web interface and click `Import`.

We will import the following dashboards, just download the following links and upload them through the grafana dashboard import screen:

- [Instrumentation](#)
- [Inventory](#)
- [Port Statistics](#)

### 1.3.3 Installation with Pip

You can install the latest pip package, or you can install directly from git via pip.

#### Installing faucet

First, install some python dependencies:

```
apt-get install python3-dev python3-pip
pip3 install setuptools
pip3 install wheel
```

Then install the latest stable release of faucet from pypi, via pip:

```
pip3 install faucet
```

Or, install the latest development code from git, via pip:

```
pip3 install git+https://github.com/faucetsdn/faucet.git
```

## Starting faucet manually

Faucet includes a start up script for starting Faucet and Gauge easily from the command line.

To run Faucet manually:

```
faucet --verbose
```

To run Gauge manually:

```
gauge --verbose
```

There are a number of options that you can supply the start up script for changing various options such as OpenFlow port and setting up an encrypted control channel. You can find a list of the additional arguments by running:

```
faucet --help
```

## Starting faucet With systemd

Systemd can be used to start Faucet and Gauge at boot automatically:

```
$EDITOR /etc/systemd/system/faucet.service
$EDITOR /etc/systemd/system/gauge.service
systemctl daemon-reload
systemctl enable faucet.service
systemctl enable gauge.service
systemctl restart faucet
systemctl restart gauge
```

/etc/systemd/system/faucet.service should contain:

Listing 34: faucet.service

```
[Unit]
Description="Faucet OpenFlow switch controller"
After=network-online.target
Wants=network-online.target

[Service]
EnvironmentFile=/etc/default/faucet
User=faucet
Group=faucet
ExecStart=/usr/local/bin/faucet --ryu-config-file=${FAUCET_RYU_CONF} --ryu-ofp-tcp-
↳listen-port=${FAUCET_LISTEN_PORT}
ExecReload=/bin/kill -HUP $MAINPID
Restart=always

[Install]
WantedBy=multi-user.target
```

/etc/systemd/system/gauge.service should contain:

Listing 35: gauge.service

```
[Unit]
Description="Gauge OpenFlow statistics controller"
```

(continues on next page)

(continued from previous page)

```
After=network-online.target
Wants=network-online.target

[Service]
EnvironmentFile=/etc/default/gauge
User=faucet
Group=faucet
ExecStart=/usr/local/bin/gauge --ryu-config-file=${GAUGE_RYU_CONF} --ryu-ofp-tcp-
↳listen-port=${GAUGE_LISTEN_PORT} --ryu-wsapi-host=${WSAPI_LISTEN_HOST} --ryu-
↳app=ryu.app.ofctl_rest
Restart=always

[Install]
WantedBy=multi-user.target
```

### 1.3.4 Installing on Raspberry Pi

We provide a Raspberry Pi image running FAUCET which can be retrieved from the [latest faucet release page](#) on GitHub. Download the `faucet_VERSION_raspbian-lite.zip` file.

The image can then be copied onto an SD card following the same steps from the official [Raspberry Pi installation guide](#).

Once you have booted up the Raspberry Pi and logged in using the default credentials you can follow through the *Installing faucet for the first time* tutorial starting from *Configure prometheus* to properly configure each component.

---

**Note:** It is strongly recommended to use a Raspberry Pi 3 or better.

---

### 1.3.5 Installing with Virtual Machine image

We provide a VM image for running FAUCET for development and learning purposes. The VM comes pre-installed with FAUCET, GAUGE, prometheus and grafana.

Openstack's [diskimage-builder](#) (DIB) is used to build the VM images in many formats (qcow2,tgz,squashfs,vhd,raw).

#### Downloading pre-built images

Pre-built images are available on github, see the [latest faucet release page](#) on GitHub and download the `faucet-amd64-VERSION.qcow2` file.

#### Building the images

If you don't want to use our pre-built images, you can build them yourself:

1. Install the latest `disk-image-builder`
2. Install a patched `vhd-util`
3. Run `build-faucet-vm.sh`

## Security considerations

This VM is not secure by default, it includes no firewall and has a number of network services listening on all interfaces with weak passwords. It also includes a backdoor user (faucet) with weak credentials.

### Services

The VM exposes a number of ports listening on all interfaces by default:

Service	Port
SSH	22
Faucet OpenFlow Channel	6653
Gauge OpenFlow Channel	6654
Grafana Web Interface	3000
Prometheus Web Interface	9090

### Default Credentials

Service	Username	Password
VM TTY Console	faucet	faucet
SSH	faucet	faucet
Grafana Web Interface	admin	admin

## Post-install steps

Grafana comes installed but unconfigured, you will need to login to the grafana web interface at `http://VM_IP:3000` and configure a data source and some dashboards.

After logging in with the default credentials shown above, the first step is to add a [prometheus data source](#), use the following settings then click `Save & Test`:

```
Name: Prometheus
Type: Prometheus
URL: http://localhost:9090
```

Next we want to add some dashboards so that we can later view the metrics from faucet.

Hover over the `+` button on the left sidebar in the web interface and click `Import`.

We will import the following dashboards, just download the following links and upload them through the grafana dashboard import screen:

- [Instrumentation](#)
- [Inventory](#)
- [Port Statistics](#)

You will need to supply your own `faucet.yaml` and `gauge.yaml` configuration in the VM. There are samples provided at `/etc/faucet/faucet.yaml` and `/etc/faucet/gauge.yaml`.

Finally you will need to point one of the supported OpenFlow vendors at the controller VM, port 6653 is the Faucet OpenFlow control channel and 6654 is the Gauge OpenFlow control channel for monitoring.

## 1.4 Configuration

### 1.4.1 Faucet configuration

Faucet is configured with a YAML-based configuration file, `faucet.yaml`. The following is example demonstrating a few common features:

Listing 36: `faucet.yaml`

```
include:
  - accls.yaml

vlangs:
  office:
    vid: 100
    description: "office network"
    accls_in: [office-vlan-protect]
    faucet_mac: "0e:00:00:00:10:01"
    faucet_vips: ['10.0.100.254/24', '2001:100::1/64', 'fe80::c00:00ff:fe00:1001/
↵64']
    routes:
      - route:
          ip_dst: '192.168.0.0/24'
          ip_gw: '10.0.100.2'
  guest:
    vid: 200
    description: "guest network"
    faucet_mac: "0e:00:00:00:20:01"
    faucet_vips: ['10.0.200.254/24', '2001:200::1/64', 'fe80::c00:00ff:fe00:2001/
↵64']

routers:
  router-office-guest:
    vlangs: [office, guest]

dps:
  sw1:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        name: "h1"
        description: "host1 container"
        native_vlan: office
        accls_in: [access-port-protect]
      2:
        name: "h2"
        description: "host2 container"
        native_vlan: office
        accls_in: [access-port-protect]
      3:
        name: "g1"
        description: "guest1 container"
        native_vlan: guest
        accls_in: [access-port-protect]
      4:
        name: "s1"
```

(continues on next page)



(continued from previous page)

```

        description: "services container"
        native_vlan: office
        acls_in: [service-port-protect]
    5:
        name: "trunk"
        description: "VLAN trunk to sw2"
        tagged_vlans: [office]
        acls_in: [access-port-protect]
sw2:
    dp_id: 0x2
    hardware: "Allied-Telesis"
    interfaces:
        1:
            name: "pi"
            description: "raspberrry pi"
            native_vlan: office
            acls_in: [access-port-protect]
        2:
            name: "laptop"
            description: "guest laptop"
            native_vlan: guest
            acls_in: [access-port-protect]
        4:
            name: "s1"
            description: "services Laptop"
            native_vlan: guest
            acls_in: [access-port-protect]
        24:
            name: "trunk"
            description: "VLAN trunk to sw1"
            tagged_vlans: [office, guest]

```

Listing 37: acs.yaml

```

acsl:
    office-vlan-protect:
        # Prevent IPv4 communication between Office/Guest networks
        - rule:
            dl_type: 0x800          # ipv4
            ipv4_src: 10.0.100.0/24
            ipv4_dst: 10.0.200.0/24
            actions:
                allow: 0          # drop
        - rule:
            actions:
                allow: 1          # allow

    access-port-protect:
        # Drop dhcp servers
        - rule:
            dl_type: 0x800          # ipv4
            nw_proto: 17           # udp
            udp_src: 67            # bootps
            udp_dst: 68           # bootpc
            actions:
                allow: 0          # drop

```

(continues on next page)

```

# Drop dhcpv6 servers
- rule:
    dl_type: 0x86dd      # ipv6
    nw_proto: 17        # udp
    udp_src: 547        # dhcpv6-server
    udp_dst: 546        # dhcpv6-client
    actions:
        allow: 0        # drop
# Drop icmpv6 RAs
- rule:
    dl_type: 0x86dd      # ipv6
    nw_proto: 58        # icmpv6
    icmpv6_type: 134    # router advertisement
    actions:
        allow: 0        # drop
# Drop SMTP
- rule:
    dl_type: 0x800      # ipv4
    nw_proto: 6         # tcp
    tcp_dst: 25         # smtp
    actions:
        allow: 0        # drop
# Force DNS to our DNS server
- rule:
    dl_type: 0x800      # ipv4
    nw_proto: 17        # udp
    udp_dst: 53         # dns
    actions:
        output:
            set_fields:
                - eth_dst: "72:b8:3c:4c:dc:4d"
            port: "s1" # s1 container
# Force DNS to our DNS server
- rule:
    dl_type: 0x800      # ipv4
    nw_proto: 6         # tcp
    tcp_dst: 53         # dns
    actions:
        output:
            set_fields:
                - eth_dst: "72:b8:3c:4c:dc:4d"
            port: "s1" # s1 container
- rule:
    actions:
        allow: 1        # allow

service-port-protect:
# Drop icmpv6 RAs
- rule:
    dl_type: 0x86dd      # ipv6
    nw_proto: 58        # icmpv6
    icmpv6_type: 134    # router advertisement
    actions:
        allow: 0        # drop
# Drop SMTP
- rule:
    dl_type: 0x800      # ipv4

```

(continues on next page)

(continued from previous page)

```

    nw_proto: 6          # tcp
    tcp_dst: 25         # smtp
    actions:
      allow: 0          # drop
- rule:
  actions:
    allow: 1           # allow

```

The datapath ID may be specified as an integer or hex string (beginning with 0x).

A port not explicitly defined in the YAML configuration file will be left down and will drop all packets.

## Configuration options

### Top Level

Table 1: Faucet.yaml

Attribute	Type	Default	Description
acls	dictionary	{}	Configuration specific to acls. The keys are names of each acl, and the values are config dictionaries holding the acl's configuration (see below).
dps	dictionary	{}	Configuration specific to datapaths. The keys are names or dp_ids of each datapath, and the values are config dictionaries holding the datapath's configuration (see below).
meters	dictionary	{}	Configuration specific to meters. The keys are names of each meter, and the values are config dictionaries holding the meter's configuration (see below).
routers	dictionary	{}	Configuration specific to routers. The keys are names of each router, and the values are config dictionaries holding the router's configuration (see below).
version	integer	2	The config version. 2 is the only supported version.
vlangs	dictionary	{}	Configuration specific to vlans. The keys are names or vids of each vlan, and the values are config dictionaries holding the vlan's configuration (see below).

## DP

DP configuration is entered in the 'dps' configuration block. The 'dps' configuration contains a dictionary of configuration blocks each containing the configuration for one datapath. The keys can either be string names given to the datapath, or the OFP datapath id.

Table 2: dps: &lt;dp name or id&gt;: {}

Attribute	Type	Default	Description
advertise_interval	integer	30	How often to advertise (eg. IPv6 RAs)
arp_neighbor_timeout	integer	250	ARP and neighbour timeout in seconds
description	string	name	Description of this datapath, strictly informational
dot1x	dictionary	{}	802.1X configuration (see below)

Continued on next page

Table 2 – continued from previous page

Attribute	Type	Default	Description
dp_id	integer	The configuration key	The OFP datapath-id of this datapath
drop_bpdu	boolean	True	If True, Faucet will drop all STP BPDUs arriving at the datapath. NB: Faucet does not handle BPDUs itself, if you disable this then you either need to configure an ACL to catch BPDUs or Faucet will forward them as though they were normal traffic.
drop_broadcast_source_address	boolean	True	If True, Faucet will drop any packet from a broadcast source address
drop_lldp	boolean	True	If True, Faucet will drop all LLDP packets arriving at the datapath.
drop_spoofed_faucet_mac	boolean	True	If True, Faucet will drop any packet it receives with an ethernet source address equal to a MAC address that Faucet is using.
group_table	boolean	False	If True, Faucet will use the OpenFlow Group tables to flood packets. This is an experimental feature that is not fully supported by all devices and may not interoperate with all features of faucet.
hardware	string	“Open vSwitch”	The hardware model of the datapath. Defaults to “Open vSwitch”. Other options can be seen in the documentation for valve.py
high_priority	integer	low_priority + 1 (9001)	The high priority value.
highest_priority	integer	high_priority + 98 (9099)	The highest priority number to use.
ignore_learn_ins	integer	10	Ignore every approx nth packet for learning. 2 will ignore 1 out of 2 packets; 3 will ignore 1 out of 3 packets. This limits control plane activity when learning new hosts rapidly. Flooding will still be done by the dataplane even with a packet is ignored for learning purposes.
interfaces	dictionary	{}	Configuration block for interface specific config (see below)

Continued on next page

Table 2 – continued from previous page

Attribute	Type	Default	Description
interface_ranges	dictionary	{}	Contains the config blocks for sets of multiple interfaces. The configuration entered here will be used as the defaults for these interfaces. The defaults can be overwritten by configuring the interfaces individually, which will also inherit all defaults not specifically configured. For example, if the range specifies tagged_vlans: [1, 2, 3], and the individual interface specifies tagged_vlans: [4], the result will be tagged_vlans: [4]. The format for the configuration key is a comma separated string. The elements can either be the name or number of an interface or a range of port numbers eg: "1-6,8,port9".
learn_ban_timeout	integer	10	When a host is rapidly moving between ports Faucet will stop learning mac addresses on one of the ports for this number of seconds.
learn_jitter	integer	10	In order to reduce load on the controller Faucet will randomly vary the timeout for learnt mac addresses by up to this number of seconds.
lldp_beacon	dictionary	{}	Configuration block for LLDP beacons
low_priority	integer	low_priority + 9000 (9000)	The low priority value.
lowest_priority	integer	priority_offset (0)	The lowest priority number to use.
max_host_fib_retry_count	integer	10	Limit the number of times Faucet will attempt to resolve a next-hop's l2 address.
max_hosts_per_resolve_cycle	integer	5	Limit the number of hosts resolved per cycle.
max_resolve_backoff_time	integer	32	When resolving next hop l2 addresses, Faucet will back off exponentially until it reaches this value.
metrics_rate_limit_sec	integer	0	Rate limit metric updates - don't update metrics if last update was less than this many seconds ago.
name	string	The configuration key	A name to reference the datapath by.
ofchannel_log	string	None	Name of logfile for openflow logs
packetin_pps	integer	None	Ask switch to rate limit packet pps.
priority_offset	integer	0	Shift all priority values by this number.
proactive_learn_v4	boolean	True	Whether proactive learning is enabled for IPv4 nexthops
proactive_learn_v6	boolean	True	Whether proactive learning is enabled for IPv6 nexthops
stack	dictionary	{}	Configuration block for stacking config, for loop protection (see below)

Continued on next page

Table 2 – continued from previous page

Attribute	Type	Default	Description
timeout	integer	300	Timeout for MAC address learning
use_idle_timeout	boolean	False	Turn on/off the use of idle timeout for src_table, default OFF.
table_sizes	dictionary	{}	For TFM based switches, size of each FAUCET table (all must be specified)
global_vlan	int	2**11-1	When global routing is enabled, FIB VID used internally

## Stacking (DP)

Stacking is configured in the dp configuration block and in the interface configuration block. At the dp level the following attributes can be configured withing the configuration block 'stack':

Table 3: dps: <dp name or id>: stack: {}

Attribute	Type	Default	Description
priority	integer	0	Setting any value for stack priority indicates that this datapath should be the root for the stacking topology.

## LLDP (DP)

LLDP beacons are configured in the dp and interface configuration blocks.

LLDP beacons can be used to, among other things, facilitate physical troubleshooting (e.g. so that a standard cable tester can display port information), verify FAUCET stacking topology, and cue a phone to use the right voice VLAN.

NOTE: while FAUCET can receive and log LLDP from other devices, FAUCET does not do spanning tree those LLDP packets will have no influence on FAUCET's forwarding decisions.

The following attributes can be configured withing the 'lldp\_beacon' configuration block at the dp level:

Table 4: dps: <dp name or id>: lldp\_beacon: {}

Attribute	Type	Default	Description
system_name	string	The datapath name	System name inside LLDP packet
send_interval	integer	None	Seconds between sending beacons
max_per_interval	integer	None	The maximum number of beacons, across all ports to send each interval

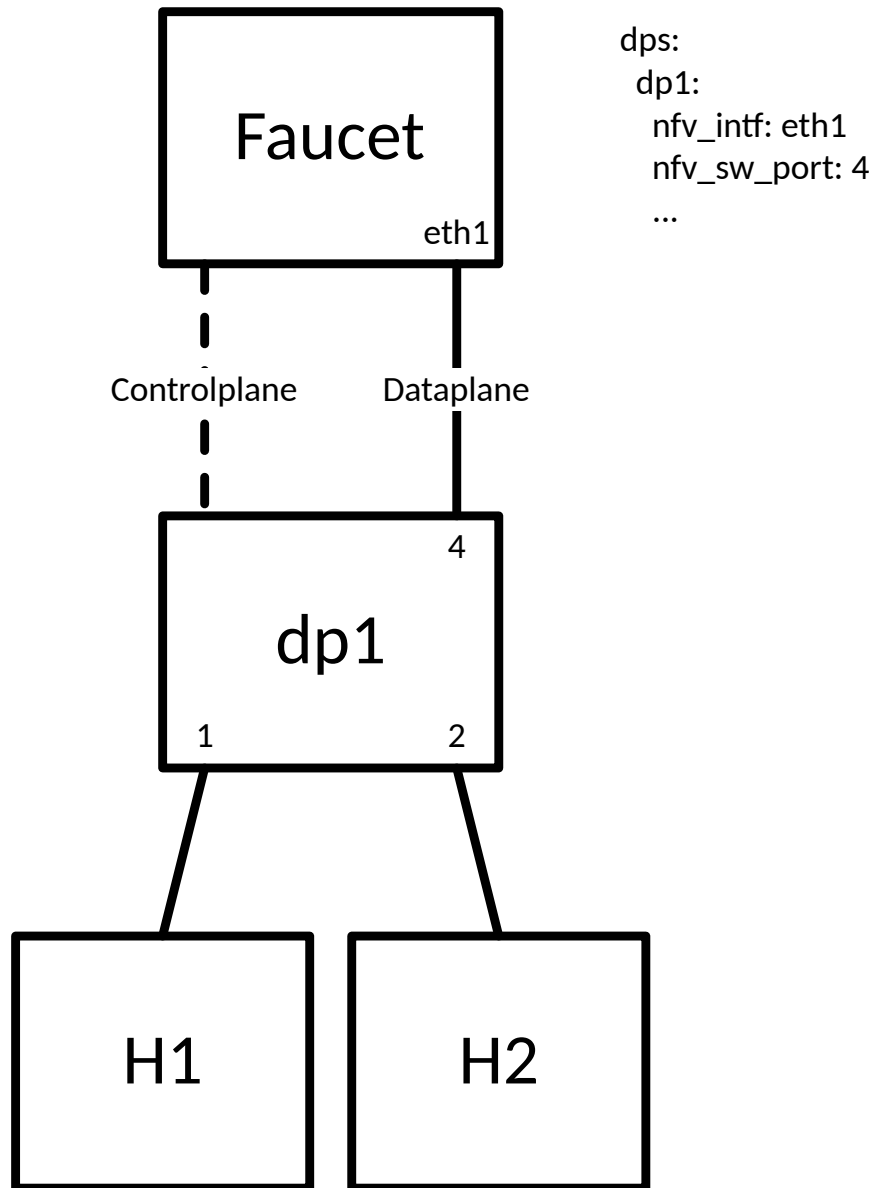
## 802.1X (DP)

**Note:** 802.1X support is experimental, and there may be incomplete features or bugs. If you find an issue please email the mailing list or create an Github issue.

Faucet implements 802.1X by forwarding EAPOL packets on the dataplane to a socket it is listening on. These packets are then passed through to a RADIUS server which performs the authentication and generates the reply message.

For each instance of Faucet there is only one 802.1X speaker. This 802.1X speaker is configured by the options below. Except for the 'nfv\_sw\_port' option, the configuration for the speaker is configured using the first dp's dot1x config dictionary. For all other dps only the 'nfv\_sw\_port' option is required with the others ignored.

A basic network and configuration with two hosts may look like:



A brief overview of the current state of the implementation:

Implemented:

- EAP Types: MD5, PEAP, TLS, TTLS.
- Authentication session expiry default 3600 seconds. (configurable (per authentication) via returning the Session-Timeout attribute in the RADIUS Access-Accept message).

- Faucet connects to a single RADIUS server, and passes through all EAP messages.
- Client can end session with EAP-Logoff.

Not Supported (yet):

- RADIUS Accounting.
- Multiple RADIUS Servers.
- Other EAP types. E.g. FAST, ...
- Dynamic assignment of VLAN/ACL.

802.1X port authentication is configured in the dp configuration block and in the interface configuration block. At the dp level the following attributes can be configured with the configuration block 'dot1x':

Table 5: dps: <dp name or id>: dot1x: {}

Attribute	Type	Default	Description
nfv_intf	str		The interface for Faucet to listen for EAP packets from the dataplane. - NOTE: Faucet will only use the config from the first dp
nfv_sw_port	int		Switch port number that connects to the Faucet server's nfv_intf
radius_ip	str		IP address of RADIUS Server the 802.1X speaker will authenticate with. - NOTE: Faucet will only use the config from the first dp
radius_port	int	1812	UDP port of RADIUS Server the 802.1X speaker will authenticate with. - NOTE: Faucet will only use the config from the first dp
radius_secret	str		Shared secret used by the RADIUS server and the 802.1X speaker. - NOTE: Faucet will only use the config from the first dp

## Interfaces

Configuration for each interface is entered in the 'interfaces' configuration block withing the config for the datapath. Each interface configuration block is a dictionary keyed by the interface name.

Defaults for groups of interfaces can also be configured under the 'interface-ranges' attribute within the datapath configuration block. These provide default values for a number of interfaces which can be overwritten with the config block for an individual interface. These are keyed with a string containing a comma separated list of OFP port numbers, interface names or with OFP port number ranges (eg. 1-6).



Table 6: dps: &lt;dp name or id&gt;: interfaces: &lt;interface name or OFP port number&gt;: {}

Attribute	Type	Default	Description
acl_in	integer or string	None	Deprecated, replaced by acls_in which accepts a list. The acl that should be applied to all packets arriving on this port. referenced by name or list index
acls_in	a list of ACLs, as integers or strings	None	A list of ACLs that should be applied to all packets arriving on this port. referenced by name or list index. ACLs listed first take priority over those later in the list.
description	string	Name (which defaults to the configuration key)	Description, purely informational
dot1x	boolean	False	Enable 802.1X port authentication
enabled	boolean	True	Allow packets to be forwarded through this port.
hairpin	boolean	False	If True it allows packets arriving on this port to be output to this port. This is necessary to allow routing between two vlans on this port, or for use with a WIFI radio port.
lldp_beacon	dictionary	{}	Configuration block for lldp configuration
loop_protect	boolean	False	if True, do simple (host/access port) loop protection on this port.
loop_protect_external	boolean	False	if True, do external (other switch) loop protection on this port.
max_hosts	integer	255	the maximum number of mac addresses that can be learnt on this port.
mirror	a list of integers or strings	None	Mirror all allowed packets recieved from (subject to ACLs), and all packets transmitted to, the ports specified (by name or by port number), to this port. If mirroring of denied by ACL packets is desired, use the ACL rule mirror option.
name	string	The configuration key.	a name to reference this port by.
native_vlan	integer	None	The vlan associated with untagged packets arriving and leaving this interface.
number	integer	The configuration key.	The OFP port number for this port.
opstatus_reconf	boolean	True	If True, FAUCET will reconfigure the pipeline based on operational status of the port.
output_only	boolean	False	If True, no packets will be accepted from this port.
override_output_port	integer	None	If set, packets are sent to this other port.
permanent_learn	boolean	False	When True Faucet will only learn the first MAC address on this interface. All packets with an ethernet src address not equal to that MAC address will be dropped.
stack	dictionary	None	configuration block for interface level stacking configuration
<b>1.4. Configuration</b>			<b>61</b>
tagged_vlans	list of integers or strings	None	The vlans associated with tagged packets arriving and leaving this interfaces.
unicast_flood	boolean	True	If False unicast packets will not be flooded to this port

## Stacking (Interfaces)

Stacking port configuration indicates how datapaths are connected when using stacking. The configuration is found under the 'stack' attribute of an interface configuration block. The following attributes can be configured:

Table 7: dps: <dp name or id>; interfaces: <interface name or port number>; stack: {}

Attribute	Type	Default	Description
dp	integer or string	None	The name of dp_id of the dp connected to this port
port	integer or string	None	The name or OFP port number of the interface on the remote dp connected to this interface.

## LLDP (Interfaces)

Interface specific configuration for LLDP.

Table 8: dps: <dp name or id>; interfaces: <interface name or port number>; lldp\_beacon: {}

Attribute	Type	Default	Description
enable	boolean	False	Enable sending lldp beacons from this interface
org_tlvs	list	[]	Definitions of Organisational TLVs to add to LLDP beacons
port_descr	string	Interface description	Port description to use in beacons from this interface
system_name	string	lldp_beacon (dp) system name	The System Name to use in beacons from this interface

## LLDP Organisational TLVs (Interfaces)

Faucet allows defining organisational TLVs for LLDP beacons. These are configured in a list under lldp\_beacons/org\_tlvs at the interfaces level of configuration.

Each list element contains a dictionary with the following elements:

Table 9: dps: <dp name or id>; interfaces: <interface name or port number>; lldp\_beacon: org\_tlvs: - {}

Attribute	Type	Default	Description
info	string	None	The info field of the tlv, as a hex string
oui	integer	None	The Organisationally Unique Identifier
subtype	integer	None	The organizationally defined subtype

## Router

Routers config is used to allow routing between VLANs, and optionally BGP. Routers configuration is entered in the 'routers' configuration block at the top level of the faucet configuration file. Configuration for each router is an entry

in the routers dictionary and is keyed by a name for the router. The following attributes can be configured:

Table 10: routers: <router name>: {}

Attribute	Type	Default	Description
vlan	list of integers or strings	None	Enables inter-vlan routing on the given VLANs.
bgp	BGP configuration.	None	See below for BGP configuration.

## BGP

Routers config to enable BGP routing.

Table 11: routers: <router name>: {}

Attribute	Type	Default	Description
as	integer	None	The local AS number to used when speaking BGP
connect_mode	string	“passive”	Must be “passive”
neighbor_addresses	list of strings (IP addresses)	None	The list of BGP neighbours
neighbor_as	integer	None	The AS Number for the BGP neighbours
routerid	string (IP address)	None	BGP router ID.
server_addresses	list of strings (IP addresses)	None	IP addresses for FAUCET to listen for incoming BGP addresses.
port	integer	None	Port to use for BGP sessions
vlan	string	None	The VLAN to add/remove BGP routes from.

## VLAN

VLANs are configured in the ‘vlans’ configuration block at the top level of the faucet config file. The config for each vlan is an entry keyed by its vid or a name. The following attributes can be configured:

Table 12: vlans: <vlan name or vid>: { }

Attribute	Type	Default	Description
acl_in	string or integer	None	Deprecated, replaced by acls_in which accepts a list. The acl to be applied to all packets arriving on this vlan.
acls_in	a list of ACLs, as integers or strings	None	The acl to be applied to all packets arriving on this vlan. ACLs listed first take priority over those later in the list.
description	string	None	Strictly informational
faucet_vips	list of strings (IP address prefixes)	None	The IP Address for Faucet's routing interface on this vlan
faucet_mac	string (MAC address)	None	Set MAC for FAUCET VIPs on this VLAN
max_hosts	integer	255	The maximum number of hosts that can be learnt on this vlan.
minimum_ip_size_check	boolean	True	If False, don't check that IP packets have a payload (must be False for OVS trace/tutorial to work)
name	string	the configuration key	A name that can be used to refer to this vlan.
proactive_arp_limit	integer	2052	Do not proactively ARP for hosts once this value has been reached (set to None for unlimited)
proactive_nd_limit	integer	2052	Don't proactively discover IPv6 hosts once this value has been reached (set to None for unlimited)
routes	list of routes	None	Static routes configured on this vlan (see below)
targeted_gw_resolution	boolean	False	If True, and a gateway has been resolved, target the first re-resolution attempt to the same port rather than flooding.
unicast_flood	boolean	True	If False packets to unknown ethernet destination MAC addresses will be dropped rather than flooded.
vid	integer	the configuration key	The vid for the vlan.

## Static Routes

Static routes are given as a list. Each entry in the list contains a dictionary keyed with the keyword 'route' and contains a dictionary configuration block as follows:

Table 13: vlans: <vlan name or vid>: routes: - route: { }

Attribute	Type	Default	Description
ip_dst	string (IP subnet)	None	The destination subnet.
ip_gw	string (IP address)	None	The next hop for this route

## Meters

**Note:** Meters are platform dependent and not all functions may be available.

Meters are configured under the ‘meters’ configuration block. The meters block contains a dictionary of individual meters each keyed by its name.

Table 14: meters: <meter name>:

Attribute	Type	Default	Description
meter_id	int		Unique identifier.
entry	dictionary		Defines the meter actions. Details Below.

Table 15: : meters: <meter name>: entry:

Attribute	Type	Default	Description
flags	string or list of strings	KBPS	Possible values are ‘KBPS’ (Rate value in kb/s (kilo-bit per second).), ‘PKTPS’ (Rate value in packet/sec.), ‘BURST’ (Do burst size), ‘STATS’ (Collect statistics)
bands	list of bands (which are dictionaries, see below)		

Table 16: : meters: <meter name>: entry: bands:

Attribute	Type	Default	Description
type	string		‘DROP’ - drop packets when the band rate is exceeded, or ‘DSCP_REMARK’ - use a simple DiffServ policer to remark the DSCP field in the IP header of packets that exceed the band rate.
rate	int		Rate for dropping or remarking packets, depending on the above type. Value is in KBPS or PKTPS flag depending on the flag set.
burst_size	int		Only used if flags includes BURST. Indicates the length of packet or byte burst to consider for applying the meter.
prec_level	int		Only used if type is DSCP_REMARK. The amount by which the drop precedence should be increased.

## ACLs

ACLs are configured under the ‘acls’ configuration block. The acls block contains a dictionary of individual acls each keyed by its name.

Each acl contains a list of rules: a packet will have the first matching rule applied to it.

Each rule is a dictionary containing the single key 'rule' with matches and actions. Matches are key/values based on the ryu RESTful API. Actions is a dictionary of actions to apply upon match.

Table 17: : acls: <acl name>: - rule: actions: { }

Attribute	Type	Default	Description
allow	boolean	False	If True allow the packet to continue through the Faucet pipeline, if False drop the packet.
force_port_vlan	boolean	False	if True, don't verify VLAN/port association.
cookie	int, 0-2**16	defaults to datapath cookie value	If set, cookie on this flow will be set to this value.
meter	string	None	Meter to apply to the packet
mirror	string or integer	None	Copy the packet, before any modifications, to the specified port (NOTE: ACL mirroring is done in input direction only)
output	dictionary	None	Used to output a packet directly. Details below.

The output action contains a dictionary with the following elements:

Table 18: : acls: <acl name>: - rule: actions: output: { }

Attribute	Type	Default	Description
set_fields	list of dictionaries	None	A list of fields to set with values, eg. eth_dst: "1:2:3:4:5:6"
port	integer or string	None	The port to output the packet to.
ports	list of [ integer or string ]	None	The list of ports the packet will be output through.
pop_vlans	boolean	False	Pop vlan tag before output.
vlan_vid	integer	False	Push vlan tag before output.
swap_vid	integer	None	Rewrite the vlan vid of the packet when outputting
vlan_vids	list of [ integer or {vid: integer, eth_type: integer} ]	None	Push vlan tags on output, with optional eth_type.
failover	dictionary	None	Output with a failover port (see below).

Failover is an experimental option, but can be configured as follows:

Table 19: : acls: <acl name>: - rule: actions: output: failover: { }

Attribute	Type	Default	Description
group_id	integer	None	The OFP group id to use for the failover group
ports	list	None	The list of ports the packet can be output through.

### 1.4.2 Gauge configuration

Gauge is configured similarly with, gauge.yaml. The following is an example demonstrating a few common features:

Listing 38: gauge.yaml

```
# Recommended configuration is Prometheus for all monitoring, with all_dps: True
faucet_configs:
  - '/etc/faucet/faucet.yaml'
watchers:
  port_status_poller:
    type: 'port_state'
    all_dps: True
    #dps: ['sw1', 'sw2']
    db: 'prometheus'
  port_stats_poller:
    type: 'port_stats'
    all_dps: True
    #dps: ['sw1', 'sw2']
    interval: 10
    db: 'prometheus'
    #db: 'influx'
  flow_table_poller:
    type: 'flow_table'
    all_dps: True
    interval: 60
    db: 'prometheus'
dbs:
  prometheus:
    type: 'prometheus'
    prometheus_addr: '0.0.0.0'
    prometheus_port: 9303
  ft_file:
    type: 'text'
    compress: True
    file: 'flow_table.yaml.gz'
  influx:
    type: 'influx'
    influx_db: 'faucet'
    influx_host: 'influxdb'
    influx_port: 8086
    influx_user: 'faucet'
    influx_pwd: 'faucet'
    influx_timeout: 10
```

### 1.4.3 Verifying configuration

You can verify that your configuration is correct with the `check_faucet_config` script:

```
check_faucet_config /etc/faucet/faucet.yaml
```

### 1.4.4 Configuration examples

For complete working examples of configuration features, see the unit tests, `tests/faucet_mininet_test.py`. For example, `FaucetUntaggedACLTest` shows how to configure an ACL to block a TCP port, `FaucetTaggedIPv4RouteTest` shows how to configure static IPv4 routing.

### 1.4.5 Applying configuration updates

You can update FAUCET's configuration by sending it a HUP signal. This will cause it to apply the minimum number of flow changes to the switch(es), to implement the change.

```
pkill -HUP -f faucet.faucet
```

### 1.4.6 Configuration in separate files

Extra DP, VLAN or ACL data can also be separated into different files and included into the main configuration file, as shown below. The `include` field is used for configuration files which are required to be loaded, and Faucet will log an error if there was a problem while loading a file. Files listed on `include-optional` will simply be skipped and a warning will be logged instead.

Files are parsed in order, and both absolute and relative (to the configuration file) paths are allowed. DPs, VLANs or ACLs defined in subsequent files overwrite previously defined ones with the same name.

faucet.yaml

```
include:
  - /etc/faucet/dps.yaml
  - /etc/faucet/vlans.yaml

include-optional:
  - acls.yaml
```

dps.yaml

```
# Recursive include is allowed, if needed.
# Again, relative paths are relative to this configuration file.
include-optional:
  - override.yaml

dps:
  test-switch-1:
    ...
  test-switch-2:
    ...
```

### 1.4.7 Environment variables

You can use environment variables to override default behaviour of faucet such as paths for configuration files and port numbers.



Environment Variable	Type	Default	Description
FAUCET_CONFIG	Colon-separated list of file paths	<code>/etc/faucet/faucet.yaml:</code>  <code>/etc/ryu/faucet/faucet.yaml</code>	Faucet will load its configuration from the first valid file in list
FAUCET_CONFIG_STAT_RELOAD	boolean	False	If true, faucet will automatically reload itself and apply new configuration when FAUCET_CONFIG changes
FAUCET_LOG_LEVEL	Python log level	INFO	Log verbosity
FAUCET_LOG	File path or STDOUT or STDERR	<code>/var/log/faucet/</code> <code>faucet.log</code>	Location for faucet to log messages to, can be special values STDOUT or STDERR
FAUCET_EXCEPTION_LOG	File path or STDOUT or STDERR	<code>/var/log/faucet/</code> <code>faucet_exception.log</code>	Location for faucet log to log exceptions to, can be special values STDOUT or STDERR
FAUCET_EVENT_SOCKET	Socket path		Location to a UNIX socket where faucet will write events to, or empty to disable events
FAUCET_PROMETHEUS_PORT	Port	9302	TCP port to listen on for faucet prometheus client
FAUCET_PROMETHEUS_ADDR	IP address	0.0.0.0	IP address to listen on for faucet prometheus client
GAUGE_CONFIG	Colon-separated list of file paths	<code>/etc/faucet/gauge.yaml:</code>  <code>/etc/ryu/faucet/gauge.yaml</code>	Gauge will load its configuration from the first valid file in list
GAUGE_CONFIG_STAT_RELOAD	boolean	False	If true, gauge will automatically reload itself and apply new configuration when GAUGE_CONFIG changes
GAUGE_LOG_LEVEL	Python log level	INFO	Log verbosity
GAUGE_LOG	File path or STDOUT or STDERR	<code>/var/log/faucet/</code> <code>gauge.log</code>	Location for gauge to log messages to, can be special values STDOUT or STDERR
GAUGE_EXCEPTION_LOG	File path or STDOUT or STDERR	<code>/var/log/faucet/</code> <code>gauge_exception.log</code>	Location for faucet log to log exceptions to, can be special values STDOUT or STDERR
GAUGE_PROMETHEUS_ADDR	IP address	0.0.0.0	IP address to listen on for gauge prometheus client

## 1.5 Monitoring

Faucet can be monitored in a number of ways. Both the faucet and gauge services export instrumentation data via a built-in Prometheus exporter which can be consumed by [Prometheus](#). By default the Prometheus exporter is available on port 9302, this can be changed with *Environment variables* (FAUCET\_PROMETHEUS\_PORT and FAUCET\_PROMETHEUS\_ADDR).

Gauge also collects conventional switch statistics (port counters, port state, etc) and can export these to a number of different databases (including Prometheus). For information on configuring gauge see the *Gauge configuration* section.

### 1.5.1 Prometheus metrics

Below is a list of the metrics exported by faucet and gauge.

#### Exported by faucet

Table 20: Faucet prometheus metrics

Metric	Type	Description
faucet_pbr_version	gauge	Faucet PBR version
faucet_config_reload_requests_total	counter	number of config reload requests
faucet_config_load_error	gauge	1 if last attempt to re/load config failed
faucet_event_id	gauge	highest/most recent event ID to be sent
faucet_config_reload_warm_total	counter	number of warm, differences only config reloads executed
faucet_config_reload_cold_total	counter	number of cold, complete reprovision config reloads executed
of_ignored_packet_ins_total	counter	number of OF packet_ins received but ignored from DP (due to rate limiting)
of_unexpected_packet_ins_total	counter	number of OF packet_ins received that are unexpected from DP (e.g. for VLAN that is not configured)
of_packet_ins_total	counter	number of OF packet_ins received from DP
of_non_vlan_packet_ins_total	counter	number of OF packet_ins received from DP, not associated with a FAUCET VLAN
of_vlan_packet_ins_total	counter	number of OF packet_ins received from DP, associated with a FAUCET VLAN
of_flowmsgs_sent_total	counter	number of OF flow messages (and packet outs) sent to DP
of_errors_total	counter	number of OF errors received from DP
of_dp_connections_total	counter	number of OF connections from a DP
of_dp_disconnections_total	counter	number of OF connections from a DP
vlan_hosts_learned	gauge	number of hosts learned on a VLAN
port_vlan_hosts_learned	gauge	number of hosts learned on a port and VLAN
vlan_neighbors	gauge	number of L3 neighbors on a VLAN (whether resolved to L2 addresses, or not)
vlan_learn_bans	gauge	number of times learning was banned on a VLAN
faucet_config_table_names	gauge	number to names map of FAUCET pipeline tables
faucet_packet_in_secs	histogram	FAUCET packet in processing time
faucet_valve_service_secs	histogram	FAUCET valve service processing time
bgp_neighbor_uptime	gauge	BGP neighbor uptime in seconds
bgp_neighbor_routes	gauge	BGP neighbor route count

Continued on next page

Table 20 – continued from previous page

Metric	Type	Description
learned_macs	gauge	MAC address stored as 64bit number to DP ID, port, VLAN, and n (discrete index)
port_status	gauge	status of switch ports
port_stack_state	gauge	state of stacking on a port
port_learn_bans	gauge	number of times learning was banned on a port
port_lacp_status	gauge	status of LACP on port
dp_status	gauge	status of datapaths
of_dp_desc_stats	gauge	DP description (OFPPDescStatsReply)
stack_cabling_errors_total	counter	number of cabling errors detected in all FAUCET stacks
stack_probes_received_total	counter	number of stacking messages received
dp_dot1x_success_total	counter	number of successful authentications on dp
dp_dot1x_failure_total	counter	number of authentications attempts failed on dp
dp_dot1x_logoff_total	counter	number of eap-logoff events on dp
port_dot1x_success_total	counter	number of successful authentications on port
port_dot1x_failure_total	counter	number of authentications attempts failed on port
port_dot1x_logoff_total	counter	number of eap-logoff events on port

## Exported by gauge

Table 21: Gauge prometheus metrics

Metric	Type	Description
faucet_pbr_version	gauge	Faucet PBR version
dp_status	gauge	status of datapaths
of_port_tx_packets	gauge	
of_port_rx_packets	gauge	
of_port_tx_bytes	gauge	
of_port_rx_bytes	gauge	
of_port_tx_dropped	gauge	
of_port_rx_dropped	gauge	
of_port_rx_errors	gauge	
of_port_reason	gauge	
of_port_state	gauge	
of_port_curr_speed	gauge	
of_port_max_speed	gauge	

## 1.6 Configuration Recipe Book

In this section we will cover some common network configurations and how you would configure these with the Faucet YAML configuration format.

### 1.6.1 Forwarding

### 1.6.2 Routing

### 1.6.3 Policy

## 1.7 Vendor-specific Documentation

### 1.7.1 Faucet on Allied Telesis products

#### Introduction

Allied Telesis has a wide portfolio of OpenFlow enabled switches that all support the Faucet pipeline. These OpenFlow enabled switches come in various port configurations of 10/18/28/52/96 with POE+ models as well. Here is a list of some of our most popular switches:

- AT-x930
- SBx908Gen2
- AT-x950
- AT-x510
- AT-x230

#### Setup

#### Switch

#### OpenFlow supported Firmware

OpenFlow has been supported since AlliedWarePlus version 5.4.6 onwards. To inquire more about compatibility of versions, you can contact our [customer support team](#).

#### OpenFlow configuration

For a **Pure OpenFlow** deployment, we recommend the following configurations on the switch. Most of these configuration steps will be shown with an example.

```
/* Create an OpenFlow native VLAN */
awplus (config)# vlan database
awplus (config-vlan)# vlan 4090

/* Set an IP address for Control Plane(CP)
 * Here we will use vlan1 for Management/Control Plane */
awplus (config)# interface vlan1
awplus (config-if)# ip address 192.168.1.1/24

/* Configure the FAUCET controller
 * Let's use TCP port 6653 for connection to Faucet */
awplus (config)# openflow controller tcp 192.168.1.10 6653

/* (OPTIONAL) Configure GAUGE controller
 * Let's use TCP port 6654 for connection to Gauge */
awplus (config)# openflow controller tcp 192.168.1.10 6654
```

(continues on next page)

(continued from previous page)

```

/* NOTE - Starting from AlliedWarePlus version 5.4.8-2, we have added support for ↵
↵controller name.
 * You can specify a controller name with the optional <name> parameter.
 * Users can still use the previous controller commands (without the name parameter) ↵
↵and the switch will auto-generate
 * a suitable name (starting with "oc") in that case.
 * Here is an example to add a controller with name 'faucet' using TCP port 6653 */
awplus (config)# openflow controller faucet tcp 192.168.1.10 6653

/* User must set a dedicated native VLAN for OpenFlow ports
 * OpenFlow native VLAN MUST be created before it is set!
 * VLAN ID for this native VLAN must be different from the native VLAN for control ↵
↵plane */
awplus (config)# openflow native vlan 4090

/* Enable OpenFlow on desired ports */
awplus (config)# interface port1.0.1-1.0.46
awplus (config-if)# openflow

/* Disable Spanning Tree Globally */
awplus (config)# no spanning-tree rstp enable

/* Disable Loop protection detection Globally */
awplus (config)# no loop-protection loop-detect

/* OpenFlow requires that ports under its control do not send any control traffic
 * So it is better to disable RSTP and IGMP Snooping TCN Query Solicitation.
 * Disable IGMP Snooping TCN Query Solicitation on the OpenFlow native VLAN */
awplus (config)# interface vlan4090
awplus (config-if)# no ip igmp snooping tcn query solicit

```

Once OpenFlow is up and running and connected to Faucet/Gauge controller, you should be able to verify the operation using some of our show commands.

```

/* To check contents of the DP flows */
awplus# show openflow flows

/* To check the actual rules as pushed by the controller */
awplus# show openflow rules

/* To check the OpenFlow configuration and other parameters */
awplus# show openflow status
awplus# show openflow config
awplus# show openflow coverage

```

Some other OPTIONAL configuration commands, that may be useful to modify some parameters, if needed.

```

/* Set the OpenFlow version other than default version(v1.3) */
awplus (config)# openflow version 1.0

/* Set IPv6 hardware filter size
 * User needs to configure the following command if a packet needs to be forwarded by ↵
↵IPv6 address matching! */
awplus (config)# platform hwfilter-size ipv4-full-ipv6

```

(continues on next page)

(continued from previous page)

```
/* Set the datapath ID(DPID)
 * By default, we use the switch MAC address for datapath-ID.
 * To change the DPID to a hex value 0x1, use the following */
awplus (config)# openflow datapath-id 1

/* NOTE - For all software versions prior to 5.4.7, all data VLAN(s) must be included
↳in the vlan database config
 * on the switch before they can be used by OpenFlow.
 * Here is an example to create DP VLANs 2-100 */
awplus (config)# vlan database
awplus (config-vlan)# vlan 2-100

/* NOTE - Starting from software version 5.4.8-2, in order to negate a controller,
↳you need to specify the controller name.
 * In case you add the controller the legacy way (without the name), the newer
↳software will auto-generate a name which can be
 * used to delete the controller.
 * Here is an example to delete a controller with auto-generated name ocl */
awplus (config)# no openflow controller ocl
```

### Useful Switch related configurations

---

**Note:** If the Openflow controller is located in a different VLAN or Network segment, routing needs to be configured so that the switch can talk to the controller.

---

```
/* To set Timezone: Codes - https://www.timeanddate.com/time/zones/ */
/* For US Pacific Time zone */
awplus (config)# clock timezone NAPST minus 8

/* To set DNS, say a local Gateway also acting as a DNS forwarder 10.20.0.1 */
awplus (config)# ip name-server 10.20.0.1

/* To make sure that DNS and routing correctly work, Gateway address needs to be set.
 * Here, Gateway is set only to the management VLAN, vlan1; 255 is the max depth
↳allowed */
awplus (config)# ip route 0.0.0.0/0 vlan1 255
awplus (config)# ip route 0.0.0.0/0 10.20.0.1

/* To see the configured Route database */
awplus# show ip route database

/* To test routing, ping Google.com - note the name to ip resolution */
awplus# ping google.com
```

### Setting up PKI Certs for secure connectivity between Switch and Openflow Controller

---

**Note:** There are many ways to get the keys and certificates into the box. Here, both private key (unencrypted PEM formatted) and corresponding Certificate (PEM) as trusted by the Openflow Controller is provided to the Switch Admin for installation.

---

Getting keys into the Switch flash partition

```

/* Here SCP is used to copy. TFTP, USB, etc are other supported methods */
awplus# copy scp://user@10.20.5.5/home/user/switch-cert.pem switch-cert.pem
awplus# copy scp://user@10.20.5.5/home/user/switch-key_nopass.pem switch-key_nopass.
↪pem

/* Showing only relevant files */
awplus# dir
    1679 -rw- Dec 20 2017 09:04:35  switch-key_nopass.pem
    11993 -rw- Dec 20 2017 09:04:03  switch-cert.pem

```

### Setting up Trustpoint for SSL connectivity to Openflow Controller

```

/* Create a local trustpoint */
awplus (config)# crypto pki trustpoint local

/* Point the switch to the OF controller */
awplus (config)# openflow controller ssl 192.168.1.10 6653

/* Allow OpenFlow to use local trustpoint */
awplus (config)# openflow ssl trustpoint local

/* Copy the new key and pvt keys to local trustpoint directory */
awplus# copy switch-key_nopass.pem .certs/pki/local/akey.pem

Overwrite flash:/.certs/pki/local/akey.pem (y/n)[n]:y
Copying...
Successful operation

awplus# copy switch-cert.pem .certs/pki/local/cacert.pem

Overwrite flash:/.certs/pki/local/cacert.pem (y/n)[n]:y
Copying...
Successful operation

```

### Enabling SNMP for monitoring Management/Control Plane Port

Openflow enabled ports are monitored via Openflow Stats request/response protocol. This means that Management port (and if Openflow control channel port is separate), are not monitored on the switch. Hence, SNMP is used to monitor the same. SNMP v2 is the most widely used. As an example below, let us assume NMS is @ 10.20.30.71

```

/* Check contents of existing access-list */
awplus# show access-list

/* Enable the SNMP agent and enable the generation of authenticate
 * failure traps to monitor unauthorized SNMP access. */
awplus (config)# snmp-server enable trap auth

/* Creating a write access community called sfractalonpremlrw for use by
 * the central network management station at 10.20.30.71 */
awplus (config)# access-list 96 permit 10.20.30.71
awplus (config)# snmp-server community sfractalonpremlrw rw view atview 96

/* Enable link traps on VLANs or specific interfaces (in our case management port) */
awplus (config)# interface port1.0.1
awplus (config-if)# snmp trap link-status

/* Configuring Trap Hosts */

```

(continues on next page)

(continued from previous page)

```
awplus (config)# snmp-server host 10.20.30.71 version 2c sfractalonpremlrw

/* Confirm all SNMP settings */
awplus# show snmp-server
SNMP Server ..... Enabled
IP Protocol ..... IPv4, IPv6
SNMP Startup Trap Delay ..... 30 Seconds
SNMPv3 Engine ID (configured name) ... Not set
SNMPv3 Engine ID (actual) ..... 0x80001f8880a2977c410e3bb658

awplus# show snmp-server community
SNMP community information:
  Community Name ..... sfractalonpremlrw
  Access ..... Read-write
  View ..... atview

awplus# show run snmp
snmp-server
snmp-server enable trap auth
snmp-server community sfractalonpremlrw rw view atview 96
snmp-server host 10.20.30.71 version 2c sfractalonpremlrw
!

/* Check if the interface is configured for SNMP */
awplus# show interface port1.0.1
Interface port1.0.1
  Scope: both
  Link is UP, administrative state is UP
  Thrash-limiting
    Status Not Detected, Action learn-disable, Timeout 1(s)
  Hardware is Ethernet, address is 001a.eb96.6ef2
  index 5001 metric 1 mru 1500
  current duplex full, current speed 1000, current polarity mdi
  configured duplex auto, configured speed auto, configured polarity auto
  <UP,BROADCAST,RUNNING,MULTICAST>
  SNMP link-status traps: Sending (suppressed after 20 traps in 60 sec)
    Link-status trap delay: 0 sec
    input packets 14327037, bytes 3727488153, dropped 0, multicast packets 440768
    output packets 11172202, bytes 2028940085, multicast packets 233192 broadcast_
->packets 1889
    Time since last state change: 40 days 00:48:38

awplus# show access-list
Standard IP access list 96
  10 permit 10.20.30.71
```

### Enabling sFlow for monitoring Management/Control Port

Openflow enabled ports are monitored via Openflow Stats request/response protocol. This means that Management port (and if Openflow control channel port is separate), are not monitored on the switch. Hence, sFlow is used to monitor the same. At this time, no TLS/SSL support is seen on the sFlow Controller channel.

```
/* Check for any existing sFlow configuration */
awplus# show running-config sflow
!

/* Enable sFlow globally */
```

(continues on next page)



(continued from previous page)

```

awplus (config)# sflow enable
% INFO: sFlow will not function until collector address is non-zero
% INFO: sFlow will not function until agent address is set
awplus# show running-config sflow
!
sflow enable
!

/* Confirm the new sFlow settings */
awplus# show sflow
sFlow Agent Configuration:                Default Values
sFlow Admin Status ..... Enabled        [Disabled]
sFlow Agent Address ..... [not set]      [not set]
Collector Address ..... 0.0.0.0          [0.0.0.0]
Collector UDP Port ..... 6343            [6343]
Tx Max Datagram Size ..... 1400         [1400]

sFlow Agent Status:
Polling/sampling/Tx ..... Inactive because:
                                     - Agent Addr is not set
                                     - Collector Addr is 0.0.0.0
                                     - Polling & sampling disabled on all ports

/* Agent IP MUST be the IP address of the management port of this switch */
awplus (config)# sflow agent ip 192.0.2.23

/* Default sFlow UDP collector port is 6343 */
awplus (config)# sflow collector ip 192.0.2.25 port 6343
awplus (config)# interface port1.0.1
awplus (config-if)# sflow polling-interval 120
awplus (config-if)# sflow sampling-rate 512

awplus# show running-config sflow
!
sflow agent ip 192.0.2.23
sflow collector ip 192.0.2.25
sflow enable
!
interface port1.0.1
  sflow polling-interval 120
  sflow sampling-rate 512
!
awplus#

```

## Faucet

Edit the faucet configuration file (`/etc/faucet/faucet.yaml`) to add the datapath of the switch you wish to be managed by faucet. This yaml file also contains the interfaces that need to be seen by Faucet as openflow ports. The device type (hardware) should be set to `Allied-Telesis` in the configuration file.

```

:caption: /etc/faucet/faucet.yaml
:name: allied-telesis/faucet.yaml

dps:
  allied-telesis:

```

(continues on next page)

(continued from previous page)

```
dp_id: 0x0000eccd6d123456
hardware: "Allied-Telesis"
interfaces:
  1:
    native_vlan: 100
    name: "port1.0.1"
  2:
    tagged_vlans: [2001,2002,2003]
    name: "port1.0.2"
    description: "windscale"
```

## References

- Allied Telesis x930
- OpenFlow Configuration Guide
- Chapter 61 (SNMP)
- SNMP Feature Guide

## 1.7.2 Faucet on HPE-Aruba Switches

### Introduction

All the Aruba's v3 generation of wired switches support the FAUCET pipeline. These switches include:

- 5400R
- 3810
- 2930F

The FAUCET pipeline is only supported from 16.03 release of the firmware onwards. HPE Aruba recommends use of the latest available firmware, which can be downloaded from [HPE Support](#).

For any queries, please post your question on HPE's [SDN forum](#).

### Setup

#### System & Network Requirements

- NOTE: Do not configure IPv6 management of the switch! This will cause IPv6 under OpenFlow control to break.
- Use Serial Console cable to login to the box.
- Use `minicom` for serial terminal @ 115Kbps. Minicom is available on Linux and MacOS (macports) systems.
- Connected Port 1 of Switch to Top of the Rack (TOR) switch which had DHCP and DNS enabled. Mac Address was programmed into DNS/DHCP Server so that IP address of 10.20.5.11 was provided to this box.
- Need a TFTP Server on the network with write access so that we can store system software for upgrade and also certificates. The switch can copy files from a TFTP Server.

**Tip:** How to clear the password settings

Simultaneously press “Reset” and “Clear” buttons using a paper clip. Release “Reset” button only first. Once the orange power light comes up (after ~5 seconds), release the “Clear” button.

**Switch****Chassis configuration**

Skip this step if you have a fixed configuration system (2930 or 3810). On a chassis system with insertable cards (5400R) new cards are configured to work in a backwards-compatible way (with reduced functionality) unless older cards are disabled in the chassis. To disable older (V2) cards and enable all functionality necessary to operate FAUCET, put the chassis into a mode where only V3 cards are allowed.

- *Chassis system (5400R)*

```
// Disable backwards compatibility, enable full Openflow flexibility
switch (config)# no allow-v2-modules
```

**VLAN/PORT configuration**

To ensure any port/vlan configuration specified in the *faucet.yaml* file works, one needs to pre-configure all vlans on the switch. Every dataplane port on the switch is made a tagged member of every vlan. This permits FAUCET to perform flow matching and packet-out on any port/vlan combination. The control-plane port (either OOBM or a front-panel port) is kept separate, so that FAUCET does not attempt to modify the control-plane port state.

- *Using OOBM control-plane (3810, 5400R)*

```
// Increase the maximum number of allowed VLANs on the box and save the configuration.
switch (config)# max-vlans 4094
switch (config)# write mem

// Reboot the box for the new max-vlan configuration to take affect.
switch (config)# boot system

// Configure the control-plane IP address
switch (config)# oobm ip address 20.0.0.1/24

// Create maximum number of VLANs and tag every dataplane port available to each vlan.
→ Takes up to 30 minutes.
switch (config)# vlan 2-4094 tagged all
```

- *Using VLAN control-plane (2930)*

```
// Increase the maximum number of allowed VLANs on the box and save the configuration.
switch (config)# max-vlans 2048
switch (config)# write mem

// Reboot the box for the new max-vlan configuration to take affect.
switch (config)# boot system

// If you have mixed both management and control-plane vlan to a single port (port 1)
switch (config)# vlan 2048 untagged 1

// Alternatively, you can have a separate port for control plane traffic
```

(continues on next page)

(continued from previous page)

```
// Create a control-plane vlan and add a single control-plane port (port 48)
switch (config)# vlan 2048 untagged 48

// Configure the control-plane IP address
// May Not be needed if you have port 1 set to DHCP/Bootp/DNS IP address of 10.20.5.11
switch (config)# vlan 2048 ip address 10.20.5.11/16

// Alternatively, to configure only the control-plane IP address
switch (config)# vlan 2048 ip address 20.0.0.1/24

// Create maximum number of VLANs and tag every dataplane port available to each vlan,
// except for the control-plane vlan (above). Note that the command below assumes it
// is run on a 52-port switch, with port 48 as the control-plane. Takes up to 20
↳minutes.
switch (config)# vlan 2-2047 tagged 1-47,49-52

// Configure DNS. Here DNS is set to a local LAN DNS server
switch (config)# ip dns server-address priority 1 10.20.0.1
```

### OpenFlow configuration

Aruba switches reference a controller by ID, so first configure the controllers which will be used. The controller-interface matches the control-plane configuration above.

- *Using OOBM control-plane (3810, 5400R)*

```
// Enter OpenFlow context
switch (config)# openflow

// Configure an OpenFlow controller connection for FAUCET over tcp-port 6653
switch(openflow)# controller-id 1 ip 20.0.0.2 port 6653 controller-interface oobm

// Faucet Controller name can be FQDN
switch(openflow)# controller-id 1 hostname controller-1.tenant1.tenants.
↳servicefractal.com port 6653 controller-interface oobm

// Configure an OpenFlow controller connection for Gauge over tcp-port 6654
switch(openflow)# controller-id 2 ip 20.0.0.2 port 6654 controller-interface oobm

// Gauge Controller name can be FQDN
switch(openflow)# controller-id 2 hostname controller-1.tenant1.tenants.
↳servicefractal.com port 6654 controller-interface oobm
```

- *Using VLAN control-plane (2930)*

```
// Enter OpenFlow context
switch (config)# openflow

// Configure an OpenFlow controller connection for FAUCET over tcp-port 6653
switch(openflow)# controller-id 1 ip 20.0.0.2 port 6653 controller-interface vlan 2048

// Faucet Controller name can be FQDN
switch(openflow)# controller-id 1 hostname controller-1.tenant1.tenants.
↳servicefractal.com port 6653 controller-interface vlan 2048

// Configure an OpenFlow controller connection for Gauge over tcp-port 6654
switch(openflow)# controller-id 2 ip 20.0.0.2 port 6654 controller-interface vlan 2048
```

(continues on next page)

(continued from previous page)

```
// Gauge Controller name can be FQDN
switch(openflow)# controller-id 2 hostname controller-1.tenant1.tenants.
↪servicefractal.com port 6654 controller-interface vlan 2048
```

Aruba switches support two OpenFlow instance types:

- **Aggregate** - Every VLAN on the switch apart from the controller/management VLANs are OpenFlow managed.
- **Virtualization** - A set of VLANs configured as members are OpenFlow managed.

Since FAUCET is designed for a pure OpenFlow environment, we choose the “**aggregate**” instance type.

```
// Enter the OpenFlow instance context
switch(openflow)# instance aggregate

// Associate the controllers to the instance
switch(of-inst-aggregate)# controller-id 1
switch(of-inst-aggregate)# controller-id 2

// Associate the controllers in secure mode to the instance
switch(of-inst-aggregate)# controller-id 1 secure
switch(of-inst-aggregate)# controller-id 2 secure

// Configure the OpenFlow version to be 1.3
switch(of-inst-aggregate)# version 1.3 only

// Configure the pipeline model type of the instance. It is a must to set it to
↪custom.
switch(of-inst-aggregate)# pipeline-model custom

// Configure the payload in the packet-ins message to be sent in its original form.
switch(of-inst-aggregate)# packet-in vlan-tagging input-form

// Ensure the switch re-attempts an OpenFlow connection at least once
// every 10 seconds when connection is dropped/inactive.
switch(of-inst-aggregate)# max-backoff-interval 10

// Allow OpenFlow to override some protocols which are otherwise excluded from
↪OpenFlow processing in switch CPU.
switch(of-inst-aggregate)# override-protocol all
WARNING: Overriding the protocol can also potentially lead to control packets
of the protocol to bypass any of the security policies like ACL(s).
Continue (y/n)? y

// Enable the instance
switch(of-inst-aggregate)# enable
switch(of-inst-aggregate)# exit

// Enable OpenFlow globally
switch(openflow)# enable
switch(openflow)# exit

// To save the Configuration
switch# save
switch# write mem
```

(continues on next page)

(continued from previous page)

```
// Show running Configuration
switch# show running-config

// Check the OpenFlow instance configuration (includes Datapath ID associated)
switch# show openflow instance aggregate
...

// Easier way to get the Datapath ID associated with the OpenFlow instance
switch# show openflow instance aggregate | include Datapath ID
          Datapath ID                : 00013863bbc41800
```

At this point, OpenFlow is enabled and running on the switch. If the FAUCET controller is running and has connected to the switch successfully, you should see the FAUCET pipeline programmed on the switch.

```
switch# show openflow instance aggregate flow-table

OpenFlow Instance Flow Table Information

Table
ID      Table Name          Flow      Miss
Count   Count                    Goto Table
-----
0       Port ACL                  5         0         1, 2, 3, 4...
1       VLAN                    10        0         2, 3, 4, 5...
2       VLAN ACL                 1         0         3, 4, 5, 6...
3       Ethernet Source         2         0         4, 5, 6, 7, 8
4       IPv4 FIB                 1         0         5, 6, 7, 8
5       IPv6 FIB                 1         0         6, 7, 8
6       VIP                     1         0         7, 8
7       Ethernet Destination    2         0         8
8       Flood                   21        0         *
```

Table ID	Table Name	Available	Free Flow Count
0	Port ACL	Ports 1-52	: 46
1	VLAN	Ports 1-52	: 91
2	VLAN ACL	Ports 1-52	: 50
3	Ethernet Source	Ports 1-52	: 99
4	IPv4 FIB	Ports 1-52	: 100
5	IPv6 FIB	Ports 1-52	: 100
6	VIP	Ports 1-52	: 20
7	Ethernet Destination	Ports 1-52	: 99
8	Flood	Ports 1-52	: 280

```
* Denotes that the pipeline could end here.

switch# show openflow instance aggregate
          Configured OF Version      : 1.3 only
          Negotiated OF Version      : 1.3
          Instance Name              : aggregate
          Data-path Description      : aggregate
          Administrator Status       : Enabled
          Member List                : VLAN 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
↪11, 12,
          .....
```

(continues on next page)

(continued from previous page)

```

.....

Controller Id Connection Status Connection State Secure Role
-----
1 Connected Active Yes Equal
2 Connected Active Yes Equal

// To just get openflow controllers
switch (openflow)# show openflow controllers

Controller Information

Controller Id IP Address Hostname Port Interface
-----
↪- 1 0.0.0.0 controller-1.t... 6653 VLAN 2048
2 0.0.0.0 controller-1.t... 6654 VLAN 2048

// Copy Running Config to a TFTP Server
// (first enable TFTP client)
switch (config)# tftp client

```

## PKI Setup on switch

**Note:** Root certificate container supports only one root certificate not a chain. So, install the one that the CSR (Certificate Signing Request) is signed with.

```

switch# show crypto pki application

Certificate Extension Validation :

Application SAN/CN
-----
openflow Disabled
syslog Disabled

// Here, we create Service Fractal CA profile
switch (config)# crypto pki ta-profile SERVICEFRACTAL_CA

// Copy the root certificate for the SERVICEFRACTAL_CA from a tftp server
switch# copy tftp ta-certificate SERVICEFRACTAL_CA 10.10.22.15 tenant1.tenants.
↪servicefractal.com.cert.pem

switch# show crypto pki ta-profile SERVICEFRACTAL_CA
Profile Name Profile Status CRL Configured OCSP Configured
-----
SERVICEFRACTAL_CA 1 certificate installed No No

Trust Anchor:
Version: 3 (0x2)
Serial Number: 4096 (0x1000)
Signature Algorithm: sha256withRSAEncryption

```

(continues on next page)

(continued from previous page)

```

...
.....

// Now we are ready to create a CSR so that a switch identity certificate_
↳that is accepted by the controller can be setup.

switch (config)# crypto pki identity-profile hpe_sf_switch1 subject common-name_
↳myswitch.tenant1.tenants.servicefractal.com org ServiceFractal org-unit vendor-test_
↳locality MyCity state CA country US

switch (config)# show crypto pki identity-profile
Switch Identity:
  ID Profile Name      : hpe_sf_switch1
  Common Name (CN)    : myswitch.tenant1.tenants.servicefractal.com
  Org Unit (OU)       : vendor-test
  Org Name (O)        : ServiceFractal
  Locality (L)        : MyCity
  State (ST)          : CA
  Country (C)         : US

// Generate CSR
switch (config)# crypto pki create-csr certificate-name hpeswt_switch1.crt ta-profile_
↳SERVICEFRACTAL_CA usage openflow

// Copy the printed CSR request and send it to "SERVICEFRACTAL_CA"

switch (config)# show crypto pki local-certificate summary
Name                Usage            Expiration        Parent / Profile
-----
hpeswt_switch1.crt  Openflow         CSR                SERVICEFRACTAL_CA

// Once the signed certificate is received, copy the same to switch.
switch (config)# copy tftp local-certificate 10.10.22.15 myswitch.tenant1.tenants.
↳servicefractal.com.cert.pem
000M Transfer is successful

switch (config)# show crypto pki local-certificate summary
Name                Usage            Expiration        Parent / Profile
-----
hpeswt_switch1.crt  Openflow         2019/01/02       SERVICEFRACTAL_CA

```

## Faucet

On the FAUCET configuration file (/etc/faucet/faucet.yaml), add the datapath of the switch you wish to be managed by FAUCET. The device type (hardware) should be set to Aruba in the configuration file.

Listing 39: /etc/faucet/faucet.yaml

```

dps:
  aruba-3810:
    dp_id: 0x00013863bbc41800
    hardware: "Aruba"
    interfaces:
      1:
        native_vlan: 100

```

(continues on next page)



(continued from previous page)

```
    name: "port1"
  2:
    native_vlan: 100
    name: "port2"
```

## Limitations

- Aruba switches currently does not support all the IPv6 related functionality inside FAUCET
- Aruba switches currently does not support the OFPAT\_DEC\_NW\_TTL action (so when routing, TTL will not be decremented).

## Debug

If you encounter a failure or unexpected behavior, it may help to enable debug output on Aruba switches. Debug output displays information about what OpenFlow is doing on the switch at message-level granularity.

```
switch# debug openflow
switch# debug destination session
switch# show debug

Debug Logging

Source IP Selection: Outgoing Interface
Origin identifier: Outgoing Interface IP
Destination:
Session

Enabled debug types:
openflow
openflow packets
openflow events
openflow errors
openflow packets tx
openflow packets rx
openflow packets tx pkt_in
openflow packets rx pkt_out
openflow packets rx flow_mod
```

## References

- [Aruba OpenFlow Administrator Guide \(16.03\)](#)
- [Aruba OS version as of Dec 2017 is 16.05](#)
- [Aruba Switches](#)
- [FAUCET](#)
- [Model 2390F Product Site](#)
- [2930F top level documentation](#)
- [Password settings](#)
- [PKI Setup](#)

### 1.7.3 Faucet on Lagopus

#### Introduction

Lagopus is a software OpenFlow 1.3 switch, that also supports DPDK.

FAUCET is supported as of Lagopus 0.2.11 (<https://github.com/lagopus/lagopus/issues/107>).

#### Setup

##### Lagopus install on a supported Linux distribution

Install Lagopus according to the [quickstart guide](#). You don't need to install Ryu since we will be using FAUCET and FAUCET's installation takes care of that dependency.

These instructions are for Ubuntu 16.0.4 (without DPDK). In theory any distribution, with or without DPDK, that Lagopus supports will work with FAUCET.

##### Create lagopus.dsl configuration file

In this example, Lagopus is controlling two ports, enp1s0f0 and enp1s0f1, which will be known as OpenFlow ports 1 and 2 on DPID 0x1. FAUCET and Lagopus are running on the same host (though of course, they don't need to be).

Listing 40: /usr/local/etc/lagopus/lagopus.dsl

```
channel channel01 create -dst-addr 127.0.0.1 -protocol tcp

controller controller01 create -channel channel01 -role equal -connection-type main

interface interface01 create -type ethernet-rawsock -device enp1s0f0

interface interface02 create -type ethernet-rawsock -device enp1s0f1

port port01 create -interface interface01

port port02 create -interface interface02

bridge bridge01 create -controller controller01 -port port01 1 -port port02 2 -dpid_
↪0x1
bridge bridge01 enable
```

##### Create faucet.yaml

Listing 41: /etc/faucet/faucet.yaml

```
vlan:
  100:
    name: "test"
dps:
  lagopus-1:
    dp_id: 0x1
    hardware: "Lagopus"
    interfaces:
```

(continues on next page)

(continued from previous page)

```

1:
   native_vlan: 100
2:
   native_vlan: 100

```

## Start Lagopus

Start in debug mode, in a dedicated terminal.

```
lagopus -d
```

## Run FAUCET

```
faucet --verbose --ryu-ofp-listen-host=127.0.0.1
```

## Test connectivity

Host(s) on enp1s0f0 and enp1s0f1 in the same IP subnet, should now be able to communicate, and FAUCET's log file should indicate learning is occurring:

Listing 42: /var/log/faucet/faucet.log

```

May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Configuring DP
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Delete VLAN vid:100 ports:1,2
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) VLANs changed/added: [100]
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Configuring VLAN vid:100 ports:1,
↪2
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Configuring VLAN vid:100 ports:1,
↪2
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Port 1 added
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Sending config for port 1
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Port 2 added
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Sending config for port 2
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Packet_in src:00:16:41:6d:87:28_
↪in_port:1 vid:100
May 11 13:04:57 faucet.valve INFO      learned 1 hosts on vlan 100
May 11 13:04:57 faucet.valve INFO      DPID 1 (0x1) Packet_in src:00:16:41:32:87:e0_
↪in_port:2 vid:100
May 11 13:04:57 faucet.valve INFO      learned 2 hosts on vlan 100

```

## 1.7.4 Faucet on ZodiacFX

### Introduction

ZodiacFX is a small 4 port multi table OF1.3 switch from [Northbound Networks](#).



(continued from previous page)

```
 / /_ / /_ / /_ / /_ / /_ / /_ / |
 /_ /_ /_ /_ /_ /_ /_ /_ /_ /_ /_ |
                               by Northbound Networks

Type 'help' for a list of available commands

Zodiac_FX#
Zodiac_FX# found initial prompt
config
Zodiac_FX(config)# setting ethertype-filter
set ethertype-filter enable
EtherType Filtering Enabled
Zodiac_FX(config)# setting of-portset of-port 6653
OpenFlow Port set to 6653
Zodiac_FX(config)# save
Writing Configuration to EEPROM (197 bytes)
Zodiac_FX(config)# exit
Zodiac_FX# restart
Restarting the Zodiac FX, please reopen your terminal application.
```

## 1.7.5 Faucet on ZodiacGX

### Introduction

ZodiacGX is a small 5 port multi table OF1.3 switch from [Northbound Networks](#). Please see the [documentation](#) for configuring OpenFlow on the switch, and use ZodiacGX as the FAUCET hardware type.

### Caveats

- The default OF port is 6633; it is recommended to use 6653.
- Minimum firmware required is v1.01

## 1.7.6 Faucet on NoviFlow

### Introduction

NoviFlow provide a range of switches known to work with FAUCET.

These instructions have been tested on NS1248, NS1132, NS2116, NS2128, NS2122, NS2150, NS21100 switches, using NoviWare versions starting from NW400.5.4, running with FAUCET v1.8.14.

Compared to older versions of NoviWare and Faucet, where manual pipeline configuration was required, it is possible to use the GenericTFM Hardware type to make Faucet automatically program the tables based on the needs of its current configuration.

### Setup

### Configure the CPN on the switch

The only configuration required in the switch is the definition of the IP and ports on which the Faucet controller must be reached. Optionally it is also possible to change the switch DPID. In this example, the server running FAUCET is 10.0.1.8; configuration for CPN interfaces is not shown.

```
set config controller controllergroup faucet controllerid 1 priority 1 ipaddr 10.0.1.  
↪ port 6653 security none  
set config controller controllergroup gauge controllerid 1 priority 1 ipaddr 10.0.1.8  
↪ port 6654 security none  
set config switch dpid 0x1
```

### Create faucet.yaml

In order to exploit the automatic pipeline configuration, the hardware specified in `faucet.yaml` must be `GenericTFM`

```
vlans:  
  100:  
    name: "test"  
dps:  
  noviflow-1:  
    dp_id: 0x1  
    hardware: "GenericTFM"  
    interfaces:  
      1:  
        native_vlan: 100  
      2:  
        native_vlan: 100  
    etc...
```

### Run FAUCET

```
faucet --verbose
```

### Using Older Faucet and NoviWare versions

Before the introduction of `GenericTFM`, Faucet used a static pipeline which needed to be configured in the switch before connecting to the controller. The following match configuration is known to pass the unit tests using `NW400.4.3` with `FAUCET 1.6.18`, but take care to adjust ACL tables matches based on the type of ACL rules defined in the configuration file. Different `FAUCET` releases may also use different match fields in the other tables.

```
set config pipeline tablesizes 1524 1024 1024 5000 3000 1024 1024 5000 1024  
↪ tablewidths 80 40 40 40 40 40 40 40 40  
set config table tableid 0 matchfields 0 3 4 5 6 10 11 12 13 14 23 29 31  
set config table tableid 1 matchfields 0 3 4 5 6  
set config table tableid 2 matchfields 0 5 6 10 11 12 14  
set config table tableid 3 matchfields 0 3 4 5 6 10  
set config table tableid 4 matchfields 5 6 12  
set config table tableid 5 matchfields 5 6 27  
set config table tableid 6 matchfields 3 5 10 23 29
```

(continues on next page)

(continued from previous page)

```
set config table tableid 7 matchfields 3 6
set config table tableid 8 matchfields 0 3 6
```

Note that this table configuration will allow most of the automated test cases to pass, except `FaucetIPv6TupleTest` (which requires IPv6 Src and Dst matching in the ACL table). In order to run this test, table 0 must be configured as follows:

```
set config table tableid 0 matchfields 0 5 6 10 26 27 13 14
```

## 1.7.7 Faucet on Cisco Switches (Beta)

### Introduction

Cisco supports Openflow with FAUCET pipeline on the Catalyst 9000 Series switches.

The solution support is currently in beta on the following models:

- C9300-48P-A
- C9410R with SUP1
- C9500-48X-A

For access to the beta image and for solution support, please send an email to [cat9k-openflow-triage\(mailer list\)](mailto:cat9k-openflow-triage@redhat.com).

### Setup

#### Boot up in Openflow Mode

The Catalyst 9K will be in traditional switching mode by default. The below command will enable Openflow mode on the switch.

```
Switch-C9300#
Switch-C9300#configure terminal
Switch-C9300(config)#boot mode ?
openflow  openflow forwarding mode

Switch-C9300(config)#boot mode openflow
Changes to the boot mode preferences have been stored,
but it cannot take effect until the next reload.
Use "show boot mode" to check the boot mode currently
active.
Switch-C9300(config)#end

Switch-C9300#show boot mode
System initialized in normal switching mode
System configured to boot in openflow forwarding mode

Reload required to boot switch in configured boot mode.

Switch-C9300#reload
```

### Configure Openflow

\*\* Configure the Management interface communicate with controller. \*\*

```
Switch-C9300#
Switch-C9300#configure terminal
Switch-C9300(config)#interface GigabitEthernet0/0
Switch-C9300(config-if)#vrf forwarding Mgmt-vrf
Switch-C9300(config-if)#ip address 192.168.0.41 255.255.255.0
Switch-C9300(config-if)#negotiation auto
Switch-C9300(config-if)#end
Switch-C9300#
```

\*\* Configure the Openflow feature and controller connectivity \*\*

```
Switch-C9300#
Switch-C9300#configure terminal
Switch-C9300(config)#feature openflow
Switch-C9300(config)#openflow
Switch-C9300(config-openflow)#switch 1 pipeline 1
Switch-C9300(config-openflow-switch)#controller ipv4 192.168.0.91 port 6653 vrf Mgmt-
↪vrf security none
Switch-C9300(config-openflow-switch)#controller ipv4 192.168.0.91 port 6654 vrf Mgmt-
↪vrf security none
Switch-C9300(config-openflow-switch)#datapath-id 0xABCDEF1234
Switch-C9300(config-openflow-switch)#end
Switch-C9300#
```

### Faucet

On the FAUCET configuration file (/etc/faucet/faucet .yaml), add the datapath of the switch you wish to be managed by FAUCET. The device type (hardware) should be set to CiscoC9K in the configuration file.

```
:caption: /etc/faucet/faucet.yaml
:name: cisco/faucet.yaml

dps:
  Cisco-C9K:
    dp_id: 0xABCDEF1234
    hardware: "CiscoC9K"
    interfaces:
      1:
        native_vlan: 100
        name: "port1"
      2:
        native_vlan: 100
        name: "port2"
```

### Troubleshooting

Command to check overall openflow configuration

```
Switch-C9300#
Switch-C9300#show openflow switch 1
```

(continues on next page)



(continued from previous page)

```

Logical Switch Context
  Id: 1
  Switch type: Forwarding
  Pipeline id: 1
  Data plane: secure
  Table-Miss default: drop
  Configured protocol version: Negotiate
  Config state: no-shutdown
  Working state: enabled
  Rate limit (packet per second): 0
  Burst limit: 0
  Max backoff (sec): 8
  Probe interval (sec): 5
  TLS local trustpoint name: not configured
  TLS remote trustpoint name: not configured
  Logging flow changes: Disabled
  Stats collect interval (sec): 5
  Stats collect Max flows: 9216
  Stats collect period (sec): 1
  Minimum flow idle timeout (sec): 10
  OFA Description:
    Manufacturer: Cisco Systems, Inc.
    Hardware: C9300-48P
    Software: Cisco IOS Software [Fuji], Catalyst L3 Switch Software (CAT9K_
↪IOSXE), Version 16.8.1GO3, RELEASE SOFTWARE (fc1)| openvswitch 2.1
    Serial Num: FCW2145L0FP
    DP Description: Faucet-C9300:sw1
  OF Features:
    DPID: 0x000000ABCDEF1234
    Number of tables: 9
    Number of buffers: 256
    Capabilities: FLOW_STATS TABLE_STATS PORT_STATS
  Controllers:
    192.168.0.91:6653, Protocol: TCP, VRF: Mgmt-vrf
    192.168.0.91:6654, Protocol: TCP, VRF: Mgmt-vrf
  Interfaces:
    GigabitEthernet1/0/1
    GigabitEthernet1/0/2
    ....

```

**Command to check the openflow flows installed**

```

Switch-C9300#
Switch-C9300#show openflow switch 1 flow list
  Logical Switch Id: 1
  Total flows: 9

  Flow: 1 Match: any Actions: drop, Priority: 0, Table: 0, Cookie: 0x0, Duration: ↪
↪33812.029s, Packets: 46853, Bytes: 3636857
  ...

```

**Command to check the state of the port status**

```

Switch-C9300#
Switch-C9300#show openflow switch 1 ports
  Logical Switch Id: 1
  Port      Interface Name  Config-State  Link-State  Features

```

(continues on next page)

(continued from previous page)

1	Gi1/0/1	PORT_UP	LINK_UP	1GB-HD
2	Gi1/0/2	PORT_UP	LINK_DOWN	1GB-HD
3	Gi1/0/3	PORT_UP	LINK_DOWN	1GB-HD
4	Gi1/0/4	PORT_UP	LINK_DOWN	1GB-HD

### Command to check the status of the controller

```
Switch-C9300#
Switch-C9300#show openflow switch 1 controller
Logical Switch Id: 1
Total Controllers: 2

Controller: 1
  192.168.0.91:6653
  Protocol: tcp
  VRF: Mgmt-vrf
  Connected: Yes
  Role: Equal
  Negotiated Protocol Version: OpenFlow 1.3
  Last Alive Ping: 2018-10-03 18:43:07 NZST
  state: ACTIVE
  sec_since_connect: 13150

Controller: 2
  192.16.0.91:6654
  Protocol: tcp
  VRF: Mgmt-vrf
  Connected: Yes
  Role: Equal
  Negotiated Protocol Version: OpenFlow 1.3
  Last Alive Ping: 2018-10-03 18:43:07 NZST
  state: ACTIVE
  sec_since_connect: 12960
```

### Command to check controller statistics

```
Switch-C9300#
Switch-C9300#show openflow switch 1 controller stats
Logical Switch Id: 1
Total Controllers: 2

Controller: 1
  address                : tcp:192.168.0.91:6653%Mgmt-vrf
  connection attempts    : 165
  successful connection attempts : 61
  flow adds               : 1286700
  flow mods               : 645
  flow deletes            : 909564
  flow removals           : 0
  flow errors              : 45499
  flow unencodable errors : 0
  total errors            : 45499
  echo requests           : rx: 842945, tx:205
  echo reply              : rx: 140, tx:842945
  flow stats              : rx: 0, tx:0
  barrier                  : rx: 8324752, tx:8324737
  packet-in/packet-out    : rx: 29931732, tx:8772758
```

(continues on next page)

(continued from previous page)

```
Controller: 2
  address                : tcp:192.168.0.91:6654%Mgmt-vrf
  connection attempts   : 11004
  successful connection attempts : 3668
  flow adds              : 0
  flow mods              : 0
  flow deletes           : 0
  flow removals         : 0
  flow errors            : 0
  flow unencodable errors : 0
  total errors           : 0
  echo requests         : rx: 946257, tx:1420
  echo reply             : rx: 1420, tx:946257
  flow stats             : rx: 47330, tx:57870
  barrier                : rx: 0, tx:0
  packet-in/packet-out  : rx: 377, tx:0
```

## References

- [Catalyst 9K at-a-glance](#)
- [Catalyst 9400 SUP1](#)
- [Catalyst 9400 Linecard](#)

## 1.7.8 Faucet on OVS with DPDK

### Introduction

Open vSwitch is a software OpenFlow switch, that supports DPDK. It is also the reference switching platform for FAUCET.

### Setup

#### Install OVS on a supported Linux distribution

Install OVS and DPDK per the [official OVS instructions](#), including enabling DPDK at compile time and in OVS's initial configuration.

These instructions are known to work for Ubuntu 16.0.4, with OVS 2.7.0 and DPDK 16.11.1, kernel 4.4.0-77. In theory later versions of these components should work without changes. A multiport NIC was used, based on the Intel 82580 chipset.

#### Bind NIC ports to DPDK

---

**Note:** If you have a multiport NIC, you must bind all the ports on the NIC to DPDK, even if you do not use them all.

---

From the DPDK source directory, determine the relationship between the interfaces you want to use with DPDK and their PCI IDs:

```
export DPDK_DIR=`pwd`
$DPDK_DIR/tools/dpdk-devbind.py --status
```

In this example, we want to use `enp1s0f0` and `enp1s0f1`.

```
$ ./tools/dpdk-devbind.py --status

Network devices using DPDK-compatible driver
=====
<none>

Network devices using kernel driver
=====
0000:01:00.0 '82580 Gigabit Network Connection' if=enp1s0f0 drv=igb unused=
0000:01:00.1 '82580 Gigabit Network Connection' if=enp1s0f1 drv=igb unused=
0000:01:00.2 '82580 Gigabit Network Connection' if=enp1s0f2 drv=igb unused=
0000:01:00.3 '82580 Gigabit Network Connection' if=enp1s0f3 drv=igb unused=
```

Still from the DPDK source directory:

```
export DPDK_DIR=`pwd`
modprobe vfio-pci
chmod a+x /dev/vfio
chmod 0666 /dev/vfio/*
$DPDK_DIR/tools/dpdk-devbind.py --bind=vfio-pci 0000:01:00.0 0000:01:00.1 0000:01:00.
→2 0000:01:00.3
$DPDK_DIR/tools/dpdk-devbind.py --status
```

### Confirm OVS has been configured to use DPDK

```
$ sudo /usr/local/share/openvswitch/scripts/ovs-ctl stop
* Exiting ovs-vswitchd (20510)
* Exiting ovssdb-server (20496)
$ sudo /usr/local/share/openvswitch/scripts/ovs-ctl start
* Starting ovssdb-server
* system ID not configured, please use --system-id
* Configuring Open vSwitch system IDs
EAL: Detected 4 lcore(s)
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL: PCI device 0000:01:00.0 on NUMA socket -1
EAL:   probe driver: 8086:150e net_e1000_igb
EAL:   using IOMMU type 1 (Type 1)
EAL: PCI device 0000:01:00.1 on NUMA socket -1
EAL:   probe driver: 8086:150e net_e1000_igb
EAL: PCI device 0000:01:00.2 on NUMA socket -1
EAL:   probe driver: 8086:150e net_e1000_igb
EAL: PCI device 0000:01:00.3 on NUMA socket -1
EAL:   probe driver: 8086:150e net_e1000_igb
EAL: PCI device 0000:02:00.0 on NUMA socket -1
EAL:   probe driver: 8086:150e net_e1000_igb
EAL: PCI device 0000:02:00.1 on NUMA socket -1
EAL:   probe driver: 8086:150e net_e1000_igb
EAL: PCI device 0000:02:00.2 on NUMA socket -1
EAL:   probe driver: 8086:150e net_e1000_igb
```

(continues on next page)

(continued from previous page)

```
EAL: PCI device 0000:02:00.3 on NUMA socket -1
EAL: probe driver: 8086:150e net_e1000_igb
Zone 0: name:<rte_eth_dev_data>, phys:0x7ffced40, len:0x30100, virt:0x7f843ffced40,
↳socket_id:0, flags:0
* Starting ovs-vswitchd
* Enabling remote OVSDB managers
```

## Configure an OVS bridge with the DPDK ports

```
ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev protocols=OpenFlow13
ovs-vsctl add-port br0 dpdk0 -- set interface enp1s0f0 type=dpdk options:dpdk-
↳devargs=0000:01:00.0
ovs-vsctl add-port br0 dpdk1 -- set interface enp1s0f1 type=dpdk options:dpdk-
↳devargs=0000:01:00.1
ovs-vsctl set-fail-mode br0 secure
ovs-vsctl set-controller br0 tcp:127.0.0.1:6653
ovs-ofctl show br0
ovs-vsctl get bridge br0 datapath_id
```

## Create faucet.yaml

**Note:** Change `dp_id`, to the value reported above, prefaced with “0x”.

Listing 44: `/etc/faucet/faucet.yaml`

```
vlan:
  100:
    name: "test"
dps:
  ovswdpdk-1:
    dp_id: 0x000090e2ba7e7564
    hardware: "Open vSwitch"
    interfaces:
      1:
        native_vlan: 100
      2:
        native_vlan: 100
```

## Run FAUCET

```
faucet --verbose --ryu-ofp-listen-host=127.0.0.1
```

## Test connectivity

Host(s) on `enp1s0f0` and `enp1s0f1` in the same IP subnet, should now be able to communicate, and FAUCET’s log file should indicate learning is occurring:

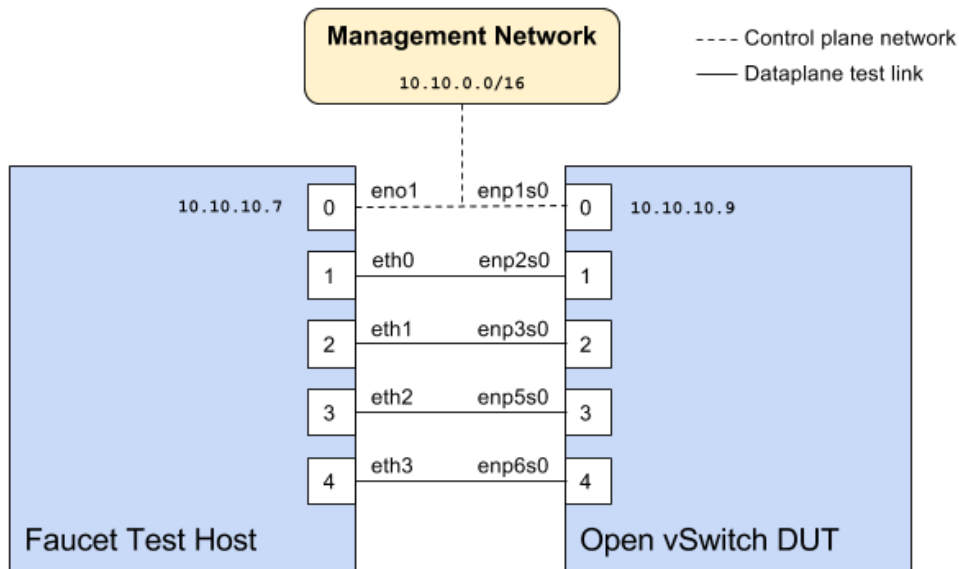
Listing 45: /var/log/faucet/faucet.log

```

May 11 14:53:32 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) ↵
↪Configuring DP
May 11 14:53:32 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) Delete_↵
↪VLAN vid:100 ports:1,2
May 11 14:53:32 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) VLANs_↵
↪changed/added: [100]
May 11 14:53:32 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) ↵
↪Configuring VLAN vid:100 ports:1,2
May 11 14:53:32 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) ↵
↪Configuring VLAN vid:100 ports:1,2
May 11 14:53:32 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) Port 1_↵
↪added
May 11 14:53:32 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) Sending_↵
↪config for port 1
May 11 14:53:32 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) Port 2_↵
↪added
May 11 14:53:32 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) Sending_↵
↪config for port 2
May 11 14:53:33 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) Packet_↵
↪in src:00:16:41:6d:87:28 in_port:1 vid:100
May 11 14:53:33 faucet.valve INFO learned 1 hosts on vlan 100
May 11 14:53:33 faucet.valve INFO DPID 159303465858404 (0x90e2ba7e7564) Packet_↵
↪in src:00:16:41:32:87:e0 in_port:2 vid:100
May 11 14:53:33 faucet.valve INFO learned 2 hosts on vlan 100
    
```

### 1.7.9 Faucet Testing with OVS on Hardware

#### Setup



## Faucet configuration file

Listing 46: /etc/faucet/hw\_switch\_config.yaml

```
# Faucet Configuration file: /etc/faucet/hw_switch_config.yaml
#
# If hw_switch value set to True, map a hardware OpenFlow switch to ports on this_
↪machine.
# Otherwise, run tests against OVS locally.
hw_switch: True
hardware: 'Open vSwitch'
dp_ports:
  1: eth0
  2: eth1
  3: eth2
  4: eth3

# Hardware switch's DPID
dpid: 0xacd28f18b
cpn_intf: eno1
of_port: 6636
gauge_of_port: 6637
```

## Hardware

1. For Network Interface Cards (NICs), prefer Intel branded models.
2. I have also used Hi-Speed USB to dual Ethernet which works great

## Software

1. Ubuntu 16.04 Xenial
2. Open vSwitch 2.7.2+

## Commands

Commands to be executed on each side - **Faucet Test host** and **Open vSwitch**.

### Commands on Faucet Test Host

Run these commands as root on the Ubuntu system (v16.04 used)

```
$ sudo mkdir -p /usr/local/src/
$ sudo mkdir -p /etc/faucet/
$ sudo cd /usr/local/src/
$ sudo git clone https://github.com/faucetsdn/faucet.git
$ cd faucet
$ sudo ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default_
↪qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
```

(continues on next page)

(continued from previous page)

```

valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default_
↳qlen 1000
link/ether b4:96:91:00:88:a4 brd ff:ff:ff:ff:ff:ff
inet6 fe80::b696:91ff:fe00:88a4/64 scope link
valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default_
↳qlen 1000
link/ether b4:96:91:00:88:a5 brd ff:ff:ff:ff:ff:ff
inet6 fe80::b696:91ff:fe00:88a5/64 scope link
valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default_
↳qlen 1000
link/ether b4:96:91:00:88:a6 brd ff:ff:ff:ff:ff:ff
inet6 fe80::b696:91ff:fe00:88a6/64 scope link
valid_lft forever preferred_lft forever
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default_
↳qlen 1000
link/ether b4:96:91:00:88:a7 brd ff:ff:ff:ff:ff:ff
inet6 fe80::b696:91ff:fe00:88a7/64 scope link
valid_lft forever preferred_lft forever
6: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default_
↳qlen 1000
link/ether 00:1e:67:ff:f6:80 brd ff:ff:ff:ff:ff:ff
inet 10.10.10.7/16 brd 10.20.255.255 scope global eno1
valid_lft forever preferred_lft forever
inet6 cafe:babe::21e:67ff:feff:f680/64 scope global mngtmpaddr dynamic
valid_lft 86398sec preferred_lft 14398sec
inet6 fe80::21e:67ff:feff:f680/64 scope link
valid_lft forever preferred_lft forever

```

**Tip:** To locate the corresponding physical port, you can make the port LED blink with *Ethtool*.

## Commands on Open vSwitch

Login as root on the Ubuntu system and install OpenvSwitch and start openvswitch-switch service

```

$ sudo apt-get install openvswitch-switch
$ sudo systemctl status openvswitch-switch.service
$ sudo ovs-vsctl add-br ovs-br0
$ sudo ovs-vsctl add-port ovs-br0 enp2s0 -- set Interface enp2s0 ofport_request=1
$ sudo ovs-vsctl add-port ovs-br0 enp3s0 -- set Interface enp3s0 ofport_request=2
$ sudo ovs-vsctl add-port ovs-br0 enp5s0 -- set Interface enp5s0 ofport_request=3
$ sudo ovs-vsctl add-port ovs-br0 enp6s0 -- set Interface enp6s0 ofport_request=4
$ sudo ovs-vsctl set-fail-mode ovs-br0 secure
$ sudo ovs-vsctl set bridge ovs-br0 protocols=OpenFlow13
$ sudo ovs-vsctl set-controller ovs-br0 tcp:10.10.10.7:6636 tcp:10.10.10.7:6637
$ sudo ovs-vsctl get bridge ovs-br0 datapath_id
$ sudo ovs-vsctl show
308038ec-495d-412d-9b13-fe95bda4e176
    Bridge "ovs-br0"

```

(continues on next page)



(continued from previous page)

```

Controller "tcp:10.10.10.7:6636"
Controller "tcp:10.10.10.7:6637"
Port "enp3s0"
  Interface "enp3s0"
  Port "enp2s0"
    Interface "enp2s0"
  Port "enp6s0"
    Interface "enp6s0"
Port "ovs-br0"
  Interface "ovs-br0"
    type: internal
Port "enp5s0"
  Interface "enp5s0"
    type: system
ovs_version: "2.7.0"

$ sudo ovs-vsctl -- --columns=name,ofport list Interface
name          : "ovs-br0"
ofport        : 65534

name          : "enp5s0"
ofport        : 3

name          : "enp2s0"
ofport        : 1

name          : "enp6s0"
ofport        : 4

name          : "enp3s0"
ofport        : 2

```

**Tip:** To locate the corresponding physical port, you can make the port LED blink with *Ethtool*.

Check port speed information to make sure that they are at least 1Gbps

```

$ sudo ovs-ofctl -O OpenFlow13 dump-ports-desc ovs-br0
OFPST_PORT_DESC reply (OF1.3) (xid=0x2):
  1(enp2s0): addr:00:0e:c4:ce:77:25
    config:      0
    state:       0
    current:     1GB-FD COPPER AUTO_NEG
    advertised:  10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG AUTO_
↔PAUSE
    supported:   10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG AUTO_
↔PAUSE
    speed:      1000 Mbps now, 1000 Mbps max
  2(enp3s0): addr:00:0e:c4:ce:77:26
    config:      0
    state:       0
    current:     1GB-FD COPPER AUTO_NEG
    advertised:  10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG AUTO_
↔PAUSE
    supported:   10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG AUTO_
↔PAUSE

```

(continues on next page)

(continued from previous page)

```

    speed: 1000 Mbps now, 1000 Mbps max
3 (enp5s0): addr:00:0e:c4:ce:77:27
    config:      0
    state:      0
    current:    1GB-FD COPPER AUTO_NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG AUTO_
↪PAUSE
    supported:  10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG AUTO_
↪PAUSE
    speed: 1000 Mbps now, 1000 Mbps max
4 (enp6s0): addr:00:0a:cd:28:f1:8b
    config:      0
    state:      0
    current:    1GB-FD COPPER AUTO_NEG
    advertised: 10MB-HD COPPER AUTO_NEG AUTO_PAUSE AUTO_PAUSE_ASYM
    supported:  10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-HD 1GB-FD COPPER AUTO_NEG
    speed: 1000 Mbps now, 1000 Mbps max
LOCAL(ovs-br0): addr:00:0a:cd:28:f1:8b
    config:      PORT_DOWN
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max

```

## Running the tests

Edit the `/etc/faucet/hw_switch_config.yaml` file as shown earlier in this document setting `hw_switch=False` initially for testing.

```

$ sudo cp /usr/local/src/faucet/hw_switch_config.yaml /etc/faucet/hw_switch_config.
↪yaml
$ sudo $EDITOR /etc/faucet/hw_switch_config.yaml
$ cd /usr/local/src/faucet/

```

Install docker by following the *Installing docker* section and then run the hardware based tests by following the *Running the tests* section.

Once the above minitest version is successful with `hw_switch=False`, then edit the `/etc/faucet/hw_switch_config.yaml` file and set `hw_switch=True`.

Run tests again, verify they all pass.

## Debugging

### TCPDump

Many times, we want to know what is coming in on a port. To check on interface `enp2s0`, for example, use

```
$ sudo tcpdump -A -w enp2s0_all.pcap -i enp2s0
```

Or

```
$ sudo tcpdump -A -w enp2s0_all.pcap -i enp2s0 'dst host <controller-ip-address> and
↪port 6653'
```

To read the pcap file, use

```
$ sudo tcpdump -r enp2s0_all.pcap
```

More detailed examples are available @ [https://www.wains.be/pub/networking/tcpdump\\_advanced\\_filters.txt](https://www.wains.be/pub/networking/tcpdump_advanced_filters.txt)

**Note:** On which machine should one run tcpdump?

Depends, if you want to examine the packet\_ins that are sent from switch to controller, run on the switch listening on the interface that is talking to the controller. If you are interested on what is coming in on a particular test port, then run it on the Test Host on that interface.

## Ethtool

To locate a physical port say `enp2s0`, make the LED blink for 5 seconds:

```
$ sudo ethtool -p enp2s0 5
```

To figure out speed on the interface. Note that if Speed on the interface is at least not 1G, then tests may not run correctly.

```
$ sudo ethtool enp2s0  
$ sudo ethtool enp2s0 | grep Speed
```

## References

<https://www.garron.me/en/linux/ubuntu-network-speed-duplex-lan.html>

## 1.8 External Resources

### 1.8.1 Online Tutorials

- <http://docs.openvswitch.org/en/latest/tutorials/faucet/>
- <http://costiser.ro/2017/03/07/sdn-lesson-2-introducing-faucet-as-an-openflow-controller/>
- <https://inside-openflow.com/openflow-tracks/faucet-controller-application-technical-track/>
- <https://blog.cyberreboot.org/building-a-software-defined-network-with-raspberry-pis-and-a-zodiac-fx-switch-97184032cdc1>

### 1.8.2 Tutorial Videos

- <https://www.youtube.com/watch?v=fuqzzjmcwII>



## 2.1 Developer Guide

This file contains an overview of architecture, coding design/practices, testing and style.

### 2.1.1 Before submitting a PR

- If you have general questions, feel free to reach out to the faucet-dev mailing list.
- If you are new to FAUCET, or are contemplating a major change, it's recommended to open a github issue with the proposed change. This will enable broad understanding of your work including being able to catch any potential snags very early (for example, adding new dependencies). Architectural and approach questions are best settled at this stage before any code is written.
- Please send relatively small, tightly scoped PRs (approx 200-300 LOC or less). This makes review and analysis easier and lowers risk, including risk of merge conflicts with other PRs. Larger changes must be refactored into incremental changes.
- You must add a test if FAUCET's functionality changes (ie. a new feature, or correcting a bug).
- All unit and integration tests must pass (please use the docker based tests; see *Software switch testing with docker*). Where hardware is available, please also run the hardware based integration tests also.
- In order to speed up acceptance of your PR we recommend enabling TravisCI on your own github repo, and linking the test results in the body of the PR. This enables the maintainers to quickly verify that your changes pass all tests in a pristine environment while conserving our TravisCI resources on the main branch (by minimizing resources used on potentially failing test runs which could be caught before opening a PR on the main branch).
- You must use the github feature branches (see <https://gist.github.com/vlandham/3b2b79c40bc7353ae95a>), for your change and squash commits (<https://blog.github.com/2016-04-01-squash-your-commits/>) when creating the PR.
- Please use the supplied git pre-commit hook (see `../git-hook/pre-commit`), to automatically run the unit tests and pylint for you at git commit time, which will save you TravisCI resources also.

- pylint must show no new errors or warnings.
- Code must conform to the style guide (see below).

### 2.1.2 PR handling guidelines

This section documents general guidelines for the maintainers in handling PRs. The overall intent is, to enable quality contributions with as low overhead as possible, maximizing the use of tools such as static analysis and unit/integration testing, and supporting rapid and safe advancement of the overall project.

In addition to the above PR submission guidelines, above:

- PRs require a positive review per github's built in gating feature. The approving reviewer executes the merge.
- PRs that should not be merged until some other criteria are met (e.g. not until release day) must include DO NOT MERGE in the title, with the details in PR comments.
- A typical PR review/adjust/merge cycle should be 2-3 days (timezones, weekends, etc permitting). If a PR upon review appears too complex or requires further discussion it is recommended it be refactored into smaller PRs or discussed in another higher bandwidth forum (e.g. a VC) as appropriate.
- A PR can be submitted at any time, but to simplify release logistics PR merges might not be done before release, on release days.

### 2.1.3 Code style

Please use the coding style documented at <https://github.com/google/styleguide/blob/gh-pages/pyguide.md>. Existing code not using this style will be incrementally migrated to comply with it. New code should comply.

### 2.1.4 Faucet Development Environment

A common way of developing faucet is inside a `virtualenv` with an IDE such as `PyCharm`.

Instructions on setting up PyCharm for developing faucet are below.

If you would rather develop on the command line directly, a short summary of the command line setup for development in a `venv` with Python 3.6+ is included after the PyCharm instructions.

#### Create a new project in PyCharm

Set the `Location` of the project to the directory where a checked out copy of the faucet code from git is, for this tutorial I will assume the path is `/Dev/faucet/`.

Ignore the `Project Interpreter` settings for now, we will set those up after the project is created.

Click `Create` when you have completed these steps.

When asked `Would you like to create a project from existing sources instead?` click `Yes`.

#### Create virtual environment

Now that the project is created and source code imported, click the `File -> Settings` menu. In the dialog box that opens click the `Project: faucet -> Project Interpreter` sub menu.

Click the cog and select `Add . . .`

Under `Virtualenv Environment` you want to select `New environment` and select a `Location` for the `virtualenv` (which can be inside the directory where the `faucet` code lives, e.g `/Dev/faucet/venv`).

The `Base interpreter` should be set to `/usr/bin/python3`.

Click `Ok` which will create the `virtualenv`.

Now while that `virtualenv` builds and we still have the settings dialog open we will tweak a few project settings to make them compatible with our code style. Click on the `Tools -> Python Integrated Tools` menu and change the `Docstring format` to `Google`.

Finally, click `Ok` again to get back to the main screen of `PyCharm`.

## Install requirements

Inside the `PyCharm` editor window if we open one of the code files for `faucet` (e.g. `faucet/faucet.py`) we should now get a bar at the top of the window telling us of missing package requirements, click the `Install requirements` option to install the dependencies for `faucet`.

## Create log and configuration directories

Now we need to create a log and configuration directory so that `faucet` can start:

```
mkdir -p /Dev/faucet/venv/var/log/faucet/
mkdir -p /Dev/faucet/venv/etc/faucet/
```

Copy the sample `faucet` configuration file from `/Dev/faucet/etc/faucet/faucet.yaml` to `/Dev/faucet/venv/etc/faucet/` and edit this configuration file as necessary.

Copy the sample `gauge` configuration file from `/Dev/faucet/etc/faucet/gauge.yaml` to `/Dev/faucet/venv/etc/faucet/` and edit this configuration file as necessary.

If you are using the sample configuration “as is” you will also need to copy `/Dev/faucet/etc/faucet/acls.yaml` to `/Dev/faucet/venv/etc/faucet/` as that included by the sample `faucet.yaml` file, and without it the sample `faucet.yaml` file cannot be loaded.

You may also wish to copy `/Dev/faucet/etc/faucet/ryu.conf` to `/Dev/faucet/venv/etc/faucet/` as well so everything can be referenced in one directory inside the `Python` virtual environment.

## Configure PyCharm to run faucet and gauge

Now we need to configure `PyCharm` to run `faucet`, `gauge` and the unit tests.

First, click the `Run -> Run..` menu, then select the `Edit Configurations...` option to get to the build settings dialog.

We will now add run configuration for starting `faucet` and `gauge`. Click the `+` button in the top left hand corner of the window. First, change the name from `Unnamed` to `faucet`. Change the `Script path` to point to `ryu-manager` inside the `virtualenv`, for me this was `../venv/bin/ryu-manager`. Then set the `Parameters` to `faucet.faucet`. Make sure the working directory is set to `/Dev/faucet/faucet/`.

We will use the same steps as above to add a run configuration for `gauge`. Changing the `Script path` to `../venv/bin/ryu-manager` and setting the `Parameters` this time to `faucet.gauge`. Make sure the working directory is set to `/Dev/faucet/faucet/`.

### Configure PyCharm to run unit tests

For running tests we need a few additional dependencies installed, I couldn't work out how to do this through PyCharm so run this command from a terminal window to install the correct dependencies inside the virtualenv:

```
/Dev/faucet/venv/bin/pip3 install -r /Dev/faucet/test-requirements.txt
```

To add the test run configuration we will again click the + button in the top left hand corner, select Python tests -> Unittests. You can provide a Name of Faucet Unit Tests for the run configuration. For Target select Script path and enter the path /Dev/faucet/tests/unit/faucet. For Pattern enter test\_\*.py.

We will also add test run configuration for gauge using the same steps as above. Use Gauge Unit Tests as the Name and for Target select Script path and enter the path /Dev/faucet/tests/unit/gauge. For Pattern enter test\_\*.py.

You can click Apply and Close now that we've added all our new run configuration.

Now that everything is setup you can run either the faucet controller, gauge controller and test suite from the Run menu.

### Developing with a Python 3.6+ venv

If you would prefer not to use PyCharm and are comfortable developing Python directly on the command line, these steps should get you started. They have been tested with Ubuntu 18.04 LTS, which includes Python 3.6, but similar instructions should work on other platforms that include Python 3.6+.

Install C/C++ compilers and Python development environment packages:

```
sudo apt-get install python3-venv libpython3.6-dev gcc g++ make
```

If you have not already, clone the faucet git repository:

```
git clone https://github.com/faucetsdn/faucet.git
```

Then create a Python venv environment within it:

```
cd faucet
python3 -m venv "${PWD}/venv"
```

and activate that virtual environment for all following steps:

```
. venv/bin/activate
```

Ensure that the faucet config is present within the virtual environment, copying from the default config files if required:

```
mkdir -p "${VIRTUAL_ENV}/var/log/faucet"
mkdir -p "${VIRTUAL_ENV}/etc/faucet"

for FILE in {acls,faucet,gauge}.yaml ryu.conf; do
  if [ -f "${VIRTUAL_ENV}/etc/faucet/${FILE}" ]; then
    echo "Preserving existing ${FILE}"
  else
    echo "Installing template ${FILE}"
    cp -p "etc/faucet/${FILE}" "${VIRTUAL_ENV}/etc/faucet/${FILE}"
  fi
done
```

Then install the runtime and development requirements



```
"${VIRTUAL_ENV}/bin/pip3" install wheel # For bdist_wheel targets
"${VIRTUAL_ENV}/bin/pip3" install -r "${VIRTUAL_ENV}/../test-requirements.txt
↪"
```

Finally install faucet in an editable form:

```
pip install -e .
```

And then confirm that you can run the unit tests:

```
pytest tests/unit/faucet/
pytest tests/unit/gauge/
```

## 2.1.5 Makefile

Makefile is provided at the top level of the directory. Output of make is normally stored in `dist` directory. The following are the targets that can be used:

- **uml**: Uses `pyreverse` to provide code class diagrams.
- **codefmt**: Provides command line usage to “Code Style” the Python file
- **codeerrors**: Uses `pylint` on all Python files to generate a code error report and is placed in `dist` directory.
- **stats**: Provides a list of all commits since the last release tag.
- **release**: Used for releasing FAUCET to the next version, Requires `version` and `next_version` variables.

To *directly install* faucet from the cloned git repo, you could use `sudo python setup.py install` command from the root of the directory.

To *build pip installable package*, you could use `python setup.py sdist` command from the root of the directory.

To *remove* any temporarily created directories and files, you could use `rm -rf dist *egg-info` command.

## Building Documentation

The documentation is built with Sphinx, from within the `docs` directory.

To be able to build the documentation ensure you have the relevant packages installed:

```
cd docs
sudo apt-get install librsvg2-bin make
pip3 install -r requirements.txt
```

and then you can build HTML documentation with:

```
cd docs
make html
```

and the documentation will be found under `_build/html` in the `docs` directory.

## 2.1.6 Key architectural concepts/assumptions:

FAUCET’s architecture depends on key assumptions, which must be kept in mind at all times.

- FAUCET is the only controller for the switch, that can add or remove flows.
- All supported dataplanes must implement OpenFlow functionally (hardware, software or both) identically. No TTP or switch specific drivers.

In addition:

- FAUCET provisions default deny flows (all traffic not explicitly programmed is dropped).
- Use of packet in is minimized.

FAUCET depends upon these assumptions to guarantee that the switch is always in a known and consistent state, which in turn is required to support high availability (FAUCET provides high availability, through multiple FAUCET controllers using the same version of configuration - any FAUCET can give the switch a consistent response - no state sharing between controllers is required). The FAUCET user can program customized flows to be added to the switch using FAUCET ACLs (see below).

FAUCET also programs the dataplane to do flooding (where configured). This minimizes the use of packet in. This is necessary to reduce competition between essential control plane messages (adding and removing flows), and traffic from the dataplane on the limited bandwidth OpenFlow control channel. Unconstrained packet in messages impact the switch CPU, may overwhelm the OpenFlow control channel, and will expose the FAUCET controller to unvalidated dataplane packets, all of which are security and reliability concerns. In future versions, packet in will be eliminated altogether. The FAUCET user is expected to use policy based forwarding (eg ACLs that redirect traffic of interest to high performance dataplane ports for NFV offload), not packet in.

FAUCET requires all supported dataplanes to implement OpenFlow (specifically, a subset of OpenFlow 1.3) in a functionally identical way. This means that there is no switch-specific driver layer - the exact same messages are sent, whether the switch is OVS or hardware. While this does prevent some earlier generation OpenFlow switches from being supported, commercially available current hardware does not have as many restrictions, and eliminating the need for a switch-specific (or TTP) layer greatly reduces implementation complexity and increases controller programmer productivity.

## 2.2 Architecture

### 2.2.1 Faucet Design and Architecture

Faucet enables practical SDN for the masses (see <http://queue.acm.org/detail.cfm?id=3015763>).

- Drop in/replacement for non-SDN L2/L3 IPv4/IPv6 switch/router (easy migration)
- Packet forwarding/flooding/multicasting done entirely by switch hardware (controller only notified on topology change)
- BGP and static routing (other routing protocols provided by NFV)
- Multi vendor/platform support using OpenFlow 1.3 multi table
- Multi switch, vendor neutral “stacking” (Faucet distributed switching, loop free topology without spanning tree)
- ACLs, as well as allow/drop, allow packets to be copied/rewritten for external NFV applications
- Monitored with Prometheus
- Small code base with high code test coverage and automated testing both hardware and software

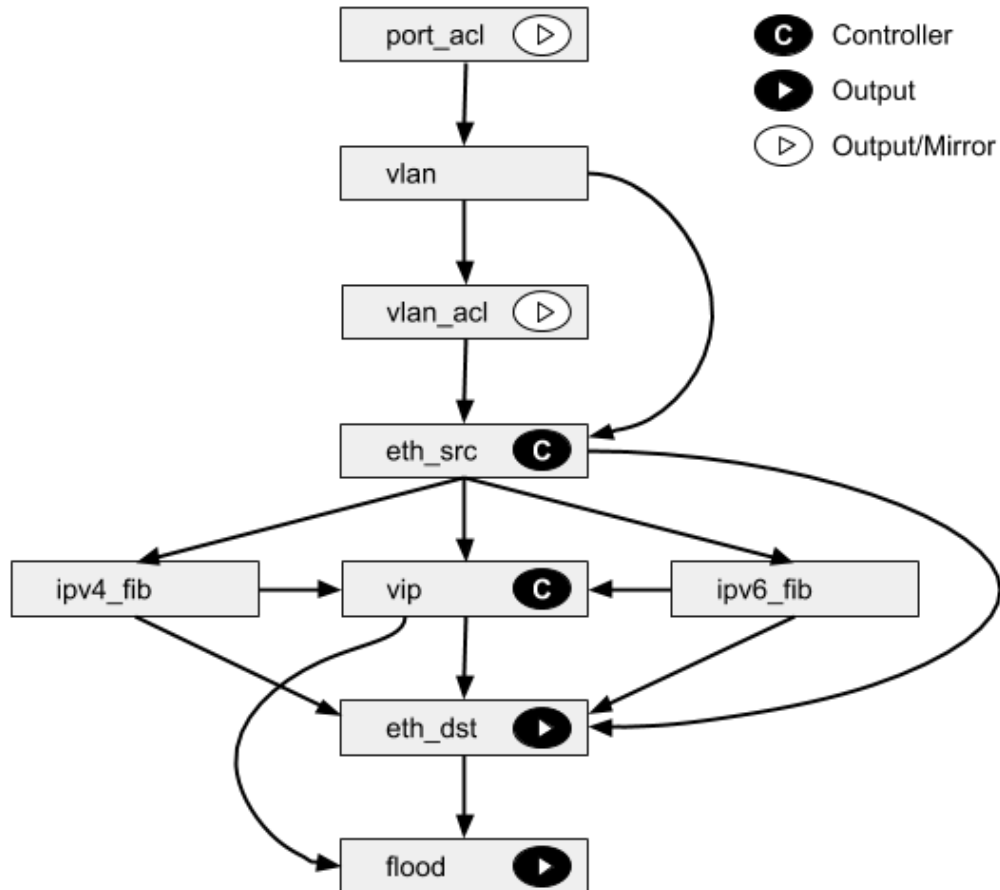
See unit and integration tests for working configuration examples.

## 2.2.2 Faucet Openflow Switch Pipeline

This summarizes the global FAUCET pipeline; however, certain tables may be omitted if the functionality is not required. For example, if routing is not configured, neither FIB table nor the VIP table will be provisioned.

Usually the OpenFlow table IDs will be allocated sequentially for the tables actually used, so tables should be referenced by their name rather than the table ID in this diagram.

See also canonical pipeline definitions in `faucet_pipeline.py`.



### PORT\_ACL Table

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

### VLAN Table

- Match fields: `eth_dst`, `eth_type`, `in_port`, `vlan_vid`
- **Operations:**
  - Drop unwanted L2 protocol traffic (and spoofing of Faucet’s virtual MAC)
  - **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
- Interception of L2 control traffic (e.g. LACP, LLDP if configured).
- Unknown traffic is dropped

### VLAN\_ACL Table

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

### ETH\_SRC Table

- Match fields: `eth_dst`, `eth_src`, `eth_type`, `in_port`, `vlan_vid`
- **Operations:**
  - For IPv4/IPv6 traffic where Faucet is the next hop, send to IPV4\_FIB or IPV6\_FIB (route)
  - For known source MAC, send to ETH\_DST (switch)
  - For unknown source MACs, copy header to controller via packet in (for learning) and send to FLOOD

### IPV4\_FIB Table

- Match fields: `eth_type`, `ipv4_dst`, `vlan_vid`
- **Operations:**
  - Route IPv4 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/HAIRPIN/VIP table
  - Unknown traffic is dropped

### IPV6\_FIB Table

- Match fields: `eth_type`, `ipv6_dst`, `vlan_vid`
- **Operations:**
  - Route IPv4 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/HAIRPIN/VIP table
  - Unknown traffic is dropped

## VIP Table

- Match fields: `arp_tpa`, `eth_dst`, `eth_type`, `icmpv6_type`, `ip_proto`
- **Operations:**
  - Send traffic destined for FAUCET VIPs including IPv4 ARP and IPv6 ND to the controller, and traffic for unresolved hosts in connected IP subnets (if proactively learning).
  - IPv4 ARP/IPv6 ND traffic may be flooded also (sent to FLOOD)

## ETH\_DST\_HAIRPIN Table

- Exact match (no wildcards)
- Match fields: `eth_dst`, `in_port`, `vlan_vid`
- **Operations:**
  - For destination MAC addresses we have learned output packet towards that host (popping VLAN frame if we are outputting on an untagged port), and where hairpinning is desired (e.g. routing between hosts on the same port, but different VLANS).
  - Unknown traffic is sent to ETH\_DST table.

## ETH\_DST Table

- Exact match (no wildcards)
- Match fields: `eth_dst`, `vlan_vid`
- **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

## FLOOD Table

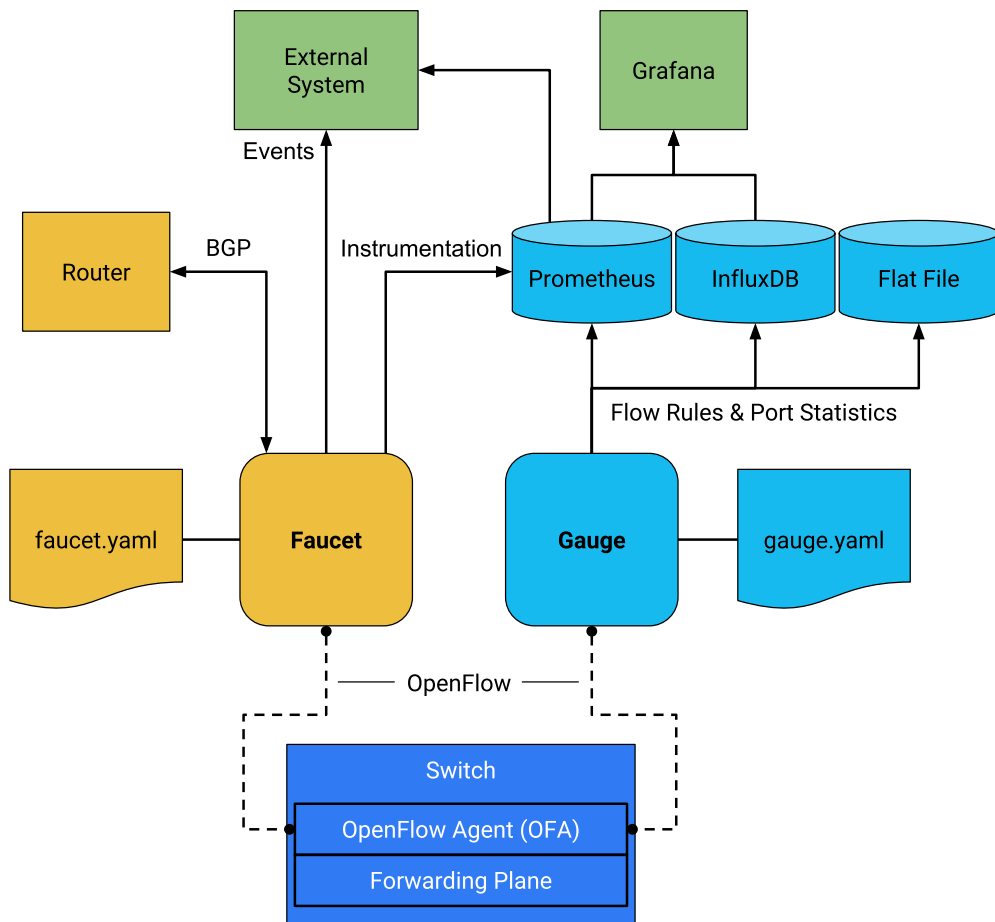
- Match fields: `eth_dst`, `in_port`, `vlan_vid`
- **Operations:**
  - Flood broadcast within VLAN
  - Flood multicast within VLAN
  - Unknown traffic is flooded within VLAN

## 2.2.3 Faucet Architecture

## 2.3 Testing

### 2.3.1 Installing docker

First, get yourself setup with docker based on our *Installing docker* documentation.



### 2.3.2 Software switch testing with docker

First prime the pip cache with the following command (re-run this when upgrading FAUCET also).

```
sudo ./docker/pip_deps.sh
```

Then you can build and run the mininet tests from the docker entry-point:

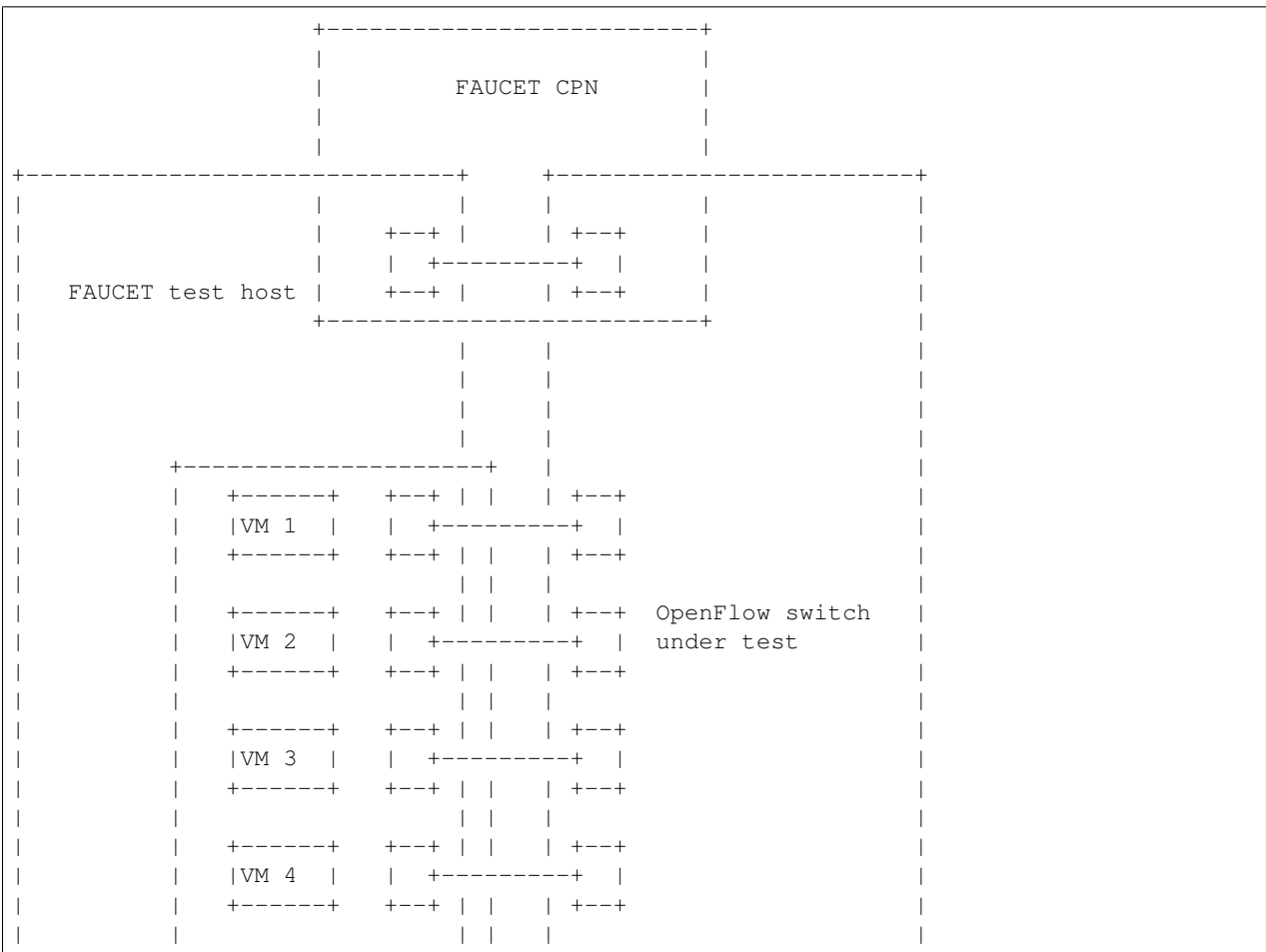
```
sudo docker build --pull -t faucet/tests -f Dockerfile.tests .
sudo apparmor_parser -R /etc/apparmor.d/usr.sbin.tcpdump
sudo modprobe openvswitch
sudo docker run --sysctl net.ipv6.conf.all.disable_ipv6=0 --privileged --rm \
    -v /var/local/lib/docker:/var/lib/docker \
    -v $HOME/.cache/pip:/var/tmp/pip-cache \
    -ti faucet/tests
```

The apparmor command is currently required on Ubuntu hosts to allow the use of tcpdump inside the container.

If you need to use a proxy, the following to your docker run command.

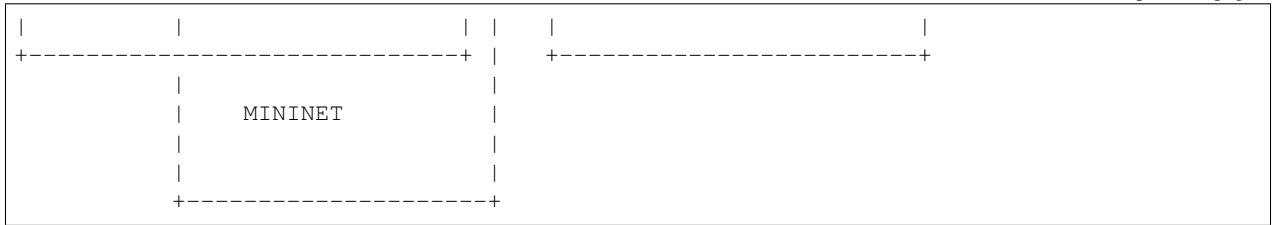
```
--build-arg http_proxy=http://your.proxy:port
```

### 2.3.3 Hardware switch testing with docker



(continues on next page)

(continued from previous page)



## Requirements

Your test host, requires at least 5 interfaces. 4 interfaces to connect to the dataplane, and one for the CPN for OpenFlow. You will need to assign an IP address to the CPN interface on the host, and configure the switch with a CPN IP address and establish that they can reach each other (eg via ping).

You will need to configure the switch with two OpenFlow controllers, both with the host’s CPN IP address, but with different ports (defaults are given below for *of\_port* and *gauge\_of\_port*).

---

**Note:** It is very important to disable any process that could cause any traffic on the dataplane test interfaces, and the test interfaces should have all IPv4/IPv6 dynamic address assignment disabled. To achieve this, on Ubuntu for example, you can set the interfaces to “unmanaged” in Network Manager, and make sure processes like [Avahi](#) ignores the test interfaces.

---



---

**Note:** Hardware tests must not be run from virtualized hosts (such as under VMware). The tests need to control physical port status, and need low level L2 packet access (eg. to rewrite Ethernet source and destination addresses) which virtualization may interfere with.

---



---

**Note:** Hardware tests require the test switch to have all non-OpenFlow switching/other features (eg. RSTP, DHCP) disabled on the dataplane test interfaces. These features will conflict with the functions FAUCET itself provides (and in turn the tests).

---

It is assumed that you execute all following commands from your FAUCET source code directory (eg one you have git cloned).

## Test configuration

Create a directory for the test configuration:

```
mkdir -p /etc/faucet
$EDITOR /etc/faucet/hw_switch_config.yaml
```

hw\_switch\_config.yaml should contain the correct configuration for your switch:

```
hw_switch: True
hardware: 'Open vSwitch'
# Map ports on the hardware switch, to physical ports on this machine.
dp_ports:
  1: enp1s0f0
  2: enp1s0f1
```

(continues on next page)



(continued from previous page)

```

3: enpls0f2
4: enpls0f3
# Hardware switch's DPID
dpid: 0xeccd6d9936ed
# Port on this machine that connects to hardware switch's CPN port.
# Hardware switch must use IP address of this port as controller IP.
cpn_intf: enp5s0
# There must be two controllers configured on the hardware switch,
# with same IP (see cpn_intf), but different ports - one for FAUCET,
# one for Gauge.
of_port: 6636
gauge_of_port: 6637
# If you wish to test OF over TLS to the hardware switch,
# set the following parameters per Ryu documentation.
# https://github.com/osrg/ryu/blob/master/doc/source/tls.rst
# ctl_privkey: ctl-privkey.pem
# ctl_cert: ctl-cert.pem
# ca_certs: /usr/local/var/lib/openvswitch/pki/switchca/cacert.pem

```

## Running the tests

Before starting the hardware test suite for the first time, you will need to install ebttables on the host machine:

```
sudo apt-get install ebttables
```

After every reboot of your host machine you will also need to manually load the `openvswitch` and `ebttables` kernel modules. If using `apparmor` you will also need to disable the profile for `tcpdump`:

```
sudo modprobe openvswitch
sudo modprobe ebttables
sudo apparmor_parser -R /etc/apparmor.d/usr.sbin.tcpdump
```

Then you can build and run the test suite:

```
sudo docker build --pull -t faucet/tests -f Dockerfile.tests .
sudo docker run --privileged --rm --net=host --cap-add=NET_ADMIN \
    -v /var/local/lib/docker:/var/lib/docker \
    -v $HOME/.cache/pip:/var/tmp/pip-cache \
    -v /etc/faucet:/etc/faucet \
    -v /var/tmp:/var/tmp \
    -ti faucet/tests
```

### 2.3.4 Test suite options

In both the software and hardware version of the test suite we can provide flags inside the `FAUCET_TESTS` environment variable to run specific parts of the test suite.

---

**Note:** Multiple flags can be added to `FAUCET_TESTS`, below are just some examples of how individual flags work.

---

### Running specific integration tests

If specific test names are listed in the `FAUCET_TESTS` environment then only these integration tests will be run and all others skipped.

If we add the following to either of the previous docker run commands then only the `FaucetUntaggedTest` will be run.

```
-e FAUCET_TESTS="FaucetUntaggedTest "
```

### Running only the integration tests

Sometimes you will want to skip the pytype, linting and documentation tests in order to complete a faucet test suite run against hardware quicker.

```
-e FAUCET_TESTS="-i "
```

### Skip code checks

Sometimes you will want to skip the pytype, linting and documentation tests.

This can be done with with the `-n` flag:

```
-e FAUCET_TESTS="-n "
```

### Skip unit tests

Sometimes you will want to skip the unit tests which are small tests that verify small chunks of the code base return the correct values. If these are skipped the integration tests (which spin up virtual networks and tests faucet controllers under different configurations) will still be run.

This can be done with with the `-u` flag:

```
-e FAUCET_TESTS="-u "
```

### Checking test results

If a test fails, you can look in `/var/tmp` - there will be subdirectories created for each test, which will contain all the logs and debug information (including tcpdumps).

By default the test suite cleans up these files but if we use the `-k` flag the test suite will keep these files.

```
-e FAUCET_TESTS="-k "
```

## 2.4 Fuzzing

### 2.4.1 Fuzzing faucet config with docker

First, get yourself setup with docker based on our *Installing docker* documentation.

Then you can build and run the afl-fuzz tests:

```
docker build -t faucet/config-fuzzer -f Dockerfile.fuzz-config .

docker run -d \
  -u $(id -u $USER) \
  --name config-fuzzer \
  -v /var/log/afl:/var/log/afl/ \
  faucet/config-fuzzer
```

AFL then will run indefinitely. You can find the output in `/var/log/afl/`. You will then need to run the output configs with faucet to see the error produced.

## 2.4.2 Fuzzing faucet packet handling with docker

Build and run the afl-fuzz tests:

```
docker build -t faucet/packet-fuzzer -f Dockerfile.fuzz-packet .

docker run -d \
  -u $(id -u $USER) \
  --name packet-fuzzer \
  -v /var/log/afl:/var/log/afl/ \
  -v /var/log/faucet:/var/log/faucet/ \
  -p 6653:6653 \
  -p 9302:9302 \
  faucet/packet-fuzzer
```

AFL will then fuzz the packet handling indefinitely. The afl output can be found in `/var/log/afl/`. To check the error produced by an afl crash file use `display_packet_crash`:

```
python3 tests/fuzzer/display_packet_crash.py /var/log/afl/crashes/X
```

Where X is the name of the crash file. The output can then be found in the faucet logs (`/var/log/faucet/`).

## 2.5 Source Code

### 2.5.1 faucet

#### faucet package

#### Submodules

#### faucet.acl module

Configuration for ACLs.

```
class faucet.acl.ACL(_id, dp_id, conf)
```

Bases: `faucet.conf.Conf`

Contains the state for an ACL, including the configuration.

ACL Config

ACLs are configured under the 'acls' configuration block. The acls block contains a dictionary of individual acls each keyed by its name.

Each acl contains a list of rules, a packet will have the first matching rule applied to it.

Each rule is a dictionary containing the single key 'rule' with the value the matches and actions for the rule.

The matches are key/values based on the ryu RESTful API. The key 'actions' contains a dictionary with keys/values as follows:

- allow (int): if 1 allow the packet to continue through the Faucet pipeline, if 0 drop the packet.
- force\_port\_vlan (int): if 1, do not verify the VLAN/port association for this packet and override any VLAN ACL on the forced VLAN.
- meter (str): meter to apply to the packet
- output (dict): used to output a packet directly. details below.
- cookie (int): set flow cookie to this value on this flow

The output action contains a dictionary with the following elements:

- tunnel (dict): the tunnel formation, creates a tunnel from the applied port(s) to the specified destination
- port (int or string): the port to output the packet to
- ports (list): a list of the ports (int or string) to output the packet to
- set\_fields (list): a list of fields to set with values
- pop\_vlans: (int): pop the packet vlan before outputting
- vlan\_vid: (int): push the vlan vid on the packet when outputting
- vlan\_vids: (list): push the list of vlans on the packet when outputting, with option eth\_type
- swap\_vid (int): rewrite the vlan vid of the packet when outputting
- failover (dict): Output with a failover port (experimental)

```
actions_types = {'allow': <class 'int'>, 'force_port_vlan': <class 'int'>, 'meter':
```

```
build (meters, vid, port_num)
```

Check that ACL can be built from config.

```
check_config ()
```

Check config at instantiation time for errors, typically via assert.

```
defaults = {'exact_match': False, 'rules': None}
```

```
defaults_types = {'exact_match': <class 'bool'>, 'rules': <class 'list'>}
```

```
get_in_port_match (tunnel_id)
```

Returns a port number of the src\_port of the tunnel that the ingress tunnel ACL will need to match to.  
:param tunnel\_id: tunnel identifier to obtain the src\_port :type tunnel\_id: int

**Returns** src\_port number if it exists, None otherwise

**Return type** int OR None

```
get_meters ()
```

```
get_mirror_destinations ()
```

```
get_tunnel_id (rule_index)
```

Gets the tunnel ID for the rule :param rule\_index: Index of the tunnel rule in the self.rules list :type rule\_index: int

**Returns** Identifier for the tunnel

**Return type** tunnel\_id (int)

```

get_tunnel_rule_indices ()
    Get the rules from the rule conf that contain tunnel outputs :returns: list of integer indices into the self.rules
    rule list that contain tunnel information :rtype: rules (list)

output_actions_types = {'failover': <class 'dict'>, 'pop_vlans': <class 'int'>, 'por

remove_non_tunnel_rules ()
    Removes all non-tunnel rules from the ACL and removes all match fields and non-tunnel required actions
    from the tunnel rules

resolve_ports (resolve_port_cb, resolve_tunnel_objects)

rule_types = {'actions': <class 'dict'>, 'arp_op': (<class 'str'>, <class 'int'>), '

tunnel_types = {'dp': <class 'str'>, 'port': (<class 'str'>, <class 'int'>), 'tunnel

unpack_tunnel (tunnel_id)
    Retrieves the information from the tunnel dict for the tunnel with id :param tunnel_id: Identifier for the
    tunnel :type tunnel_id: int

    Returns Tunnel information

    Return type (src_dp, src_port, dst_dp, dst_port)

update_tunnel_acl_conf (dp)
    Update the ACL rule conf if the DP is in the path :param dp: The dp that this tunnel acl object belongs to
    :type dp: DP

    Returns True if any value was updated

    Return type bool

verify_tunnel_compatibility_rules (dp)
    Verify the actions in the tunnel ACL to by making sure the user hasn't specified an action/match that will
    create a clash. :param dp: The dp that this tunnel acl object belongs to :type dp: DP

    TODO: Choose what combinations of matches & actions to disallow with a tunnel

```

### faucet.check\_faucet\_config module

Standalone script to check FAUCET configuration, return 0 if provided config OK.

```
faucet.check_faucet_config.check_config(conf_files, debug_level, check_output_file)
    Return True and successful config dict, if all config can be parsed.
```

```
faucet.check_faucet_config.main()
    Mainline.
```

### faucet.conf module

Base configuration implementation.

```
class faucet.conf.Conf (_id, dp_id, conf=None)
    Bases: object

    Base class for FAUCET configuration.

    check_config ()
        Check config at instantiation time for errors, typically via assert.
```

**conf\_diff** (*other*)  
Return text diff between two Confs.

**conf\_hash** (*dyn=False, subconf=True, ignore\_keys=None*)  
Return hash of keys configurably filtering attributes.

**defaults = None**

**defaults\_types = None**

**dyn\_finalized = False**

**dyn\_hash = None**

**finalize** ()  
Configuration parsing marked complete.

**ignore\_subconf** (*other, ignore\_keys=None*)  
Return True if this config same as other, ignoring sub config.

**merge\_dyn** (*other\_conf*)  
Merge dynamic state from other conf object.

**mutable\_attrs = frozenset ()**

**set\_defaults** (*defaults=None, conf=None*)  
Set default values and run any basic sanity checks.

**to\_conf** ()  
Return configuration as a dict.

**update** (*conf*)  
Parse supplied YAML config and sanity check.

**exception** `faucet.conf.InvalidConfigError`  
Bases: `Exception`

This error is thrown when the config file is not valid.

`faucet.conf.test_config_condition` (*cond, msg*)  
Evaluate condition and raise `InvalidConfigError` if condition not True.

### faucet.config\_parser module

Implement configuration file parsing.

`faucet.config_parser.dp_parser` (*config\_file, logname*)  
Parse a config file into DP configuration objects with hashes of config include/files.

`faucet.config_parser.get_config_for_api` (*valves*)  
Return config as dict for all DPs.

`faucet.config_parser.watcher_parser` (*config\_file, logname, prom\_client*)  
Return Watcher instances from config.

### faucet.config\_parser\_util module

Utility functions supporting FAUCET/Gauge config parsing.

**class** `faucet.config_parser_util.UniqueKeyLoader` (*stream*)  
Bases: `yaml.loader.Loader`

**construct\_mapping** (*node, deep=False*)

Check for duplicate YAML keys.

`faucet.config_parser_util.config_changed` (*top\_config\_file, new\_top\_config\_file, config\_hashes*)

Return True if configuration has changed.

#### Parameters

- **top\_config\_file** (*str*) – name of FAUCET config file
- **new\_top\_config\_file** (*str*) – name, possibly new, of FAUCET config file.
- **config\_hashes** (*dict*) – map of config file/includes and hashes of contents.

**Returns** True if the file, or any file it includes, has changed.

**Return type** bool

`faucet.config_parser_util.config_file_hash` (*config\_file\_name*)

Return hash of YAML config file contents.

`faucet.config_parser_util.dp_config_path` (*config\_file, parent\_file=None*)

Return full path to config file.

`faucet.config_parser_util.dp_include` (*config\_hashes, config\_file, logname, top\_confs*)

Handles including additional config files

`faucet.config_parser_util.get_logger` (*logname*)

Return logger instance for config parsing.

`faucet.config_parser_util.read_config` (*config\_file, logname*)

Return a parsed YAML config file or None.

## faucet.dp module

Configuration for a datapath.

**class** `faucet.dp.DP` (*\_id, dp\_id, conf*)

Bases: `faucet.conf.Conf`

Stores state related to a datapath controlled by Faucet, including configuration.

**add\_acl** (*acl\_ident, acl*)

Add an ACL to this DP.

**add\_port** (*port*)

Add a port to this DP.

**add\_router** (*router\_ident, router*)

Add a router to this DP.

**classmethod** **add\_stack\_link** (*graph, dp, port*)

Add a stack link to the stack graph.

**base\_prom\_labels** ()

Return base Prometheus labels for this DP.

**bgp\_routers** ()

Return list of routers with BGP enabled.

**check\_config** ()

Check config at instantiation time for errors, typically via assert.

**classification\_table()**

Returns classification table

**default\_table\_sizes\_types** = {'classification': <class 'int'>, 'eth\_dst': <class 'int'

defaults = {'advertise\_interval': 30, 'arp\_neighbor\_timeout': 30, 'cache\_update\_guar

defaults\_types = {'advertise\_interval': <class 'int'>, 'arp\_neighbor\_timeout': <clas

dot1x\_defaults\_types = {'nfv\_intf': <class 'str'>, 'nfv\_sw\_port': <class 'int'>, 'ra

**dot1x\_ports()**

Return list of ports with 802.1x enabled.

**finalize\_config(dps)**

Perform consistency checks after initial config parsing.

**finalize\_tunnel\_acls(dps)**

Turn off ACLs not in use and resolve the src dp & port for relevant ACLs :param dps: :type dps: list

**get\_config\_changes(logger, new\_dp)**

Detect any config changes.

#### Parameters

- **logger** (*ValveLogger*) – logger instance
- **new\_dp** (*DP*) – new dataplane configuration.

#### Returns

changes tuple containing:

deleted\_ports (set): deleted port numbers. changed\_ports (set): changed/added port numbers. changed\_acl\_ports (set): changed ACL only port numbers. deleted\_vlans (set): deleted VLAN IDs. changed\_vlans (set): changed/added VLAN IDs. all\_ports\_changed (bool): True if all ports changed.

#### Return type

 (tuple)

**get\_config\_dict()**

Return DP config as a dict for API call.

**get\_native\_vlan(port\_num)**

Return native VLAN for a port by number, or None.

**get\_tables()**

Return tables as dict for API call.

**in\_port\_tables()**

Return list of tables that specify in\_port as a match.

**is\_in\_path(src\_dp, dst\_dp)**

Returns true if the current DP is in the path from src\_dp to dst\_dp :param src\_dp: :type src\_dp: DP :param dst\_dp: :type dst\_dp: DP

#### Returns

**True if self is in the path from the src\_dp to the dst\_dp** False otherwise

#### Return type

 bool

**is\_stack\_edge()**

Return True if this DP is a stack edge.



**is\_stack\_root** ()  
Return True if this DP is the root of the stack.

**lldp\_beacon\_defaults\_types** = {'max\_per\_interval': <class 'int'>, 'send\_interval': <c

**lldp\_beacon\_send\_ports** (*now*)  
Return list of ports to send LLDP packets; stacked ports always send LLDP.

**match\_tables** (*match\_type*)  
Return list of tables with matches of a specific match type.

**static modify\_stack\_topology** (*graph, dp, port, add=True*)  
Add/remove an edge to the stack graph which originates from this dp and port.

**mutable\_attrs** = frozenset({'vlans', 'stack'})

**output\_table** ()  
Returns first output table

**output\_tables** ()  
Return tables that cause a packet to be forwarded.

**peer\_stack\_up\_ports** (*peer\_dp*)  
Return list of stack ports that are up towards a peer.

**port\_labels** (*port\_no*)  
Return port name and description labels for a port number.

**port\_no\_valid** (*port\_no*)  
Return True if supplied port number valid on this datapath.

**classmethod remove\_stack\_link** (*graph, dp, port*)  
Remove a stack link to the stack graph.

**reset\_refs** (*vlans=None, root\_dp=None*)  
Resets vlan references

**resolve\_port** (*port\_name*)  
Resolve a port by number or name.

**resolve\_stack\_topology** (*dps*)  
Resolve inter-DP config for stacking.

**set\_defaults** ()  
Set default values and run any basic sanity checks.

**shortest\_path** (*dest\_dp, src\_dp=None*)  
Return shortest path to a DP, as a list of DPs.

**shortest\_path\_port** (*dest\_dp*)  
Return first port on our DP, that is the shortest path towards dest DP.

**shortest\_path\_to\_root** ()  
Return shortest path to root DP, as list of DPs.

**stack\_defaults\_types** = {'priority': <class 'int'>}

**table\_by\_id** (*table\_id*)  
Gets first table with table id

## faucet.faucet module

RyuApp shim between Ryu and Valve.

```
class faucet.faucet.EventFaucetAdvertise
    Bases: ryu.controller.event.EventBase

    Event used to trigger periodic network advertisements (eg IPv6 RAs).

class faucet.faucet.EventFaucetExperimentalAPIRegistered
    Bases: ryu.controller.event.EventBase

    Event used to notify that the API is registered with Faucet.

class faucet.faucet.EventFaucetFastAdvertise
    Bases: ryu.controller.event.EventBase

    Event used to trigger periodic fast network advertisements (eg LACP).

class faucet.faucet.EventFaucetFastStateExpire
    Bases: ryu.controller.event.EventBase

    Event used to trigger fast expiration of state in controller.

class faucet.faucet.EventFaucetMetricUpdate
    Bases: ryu.controller.event.EventBase

    Event used to trigger update of metrics.

class faucet.faucet.EventFaucetResolveGateways
    Bases: ryu.controller.event.EventBase

    Event used to trigger gateway re/resolution.

class faucet.faucet.EventFaucetStateExpire
    Bases: ryu.controller.event.EventBase

    Event used to trigger expiration of state in controller.

class faucet.faucet.Faucet (*args, **kwargs)
    Bases: faucet.valve_ryuapp.RyuAppBase

    A RyuApp that implements an L2/L3 learning VLAN switch.

    Valve provides the switch implementation; this is a shim for the Ryu event handling framework to interface with Valve.

    bgp = None

    desc_stats_reply_handler (ryu_event)
        Handle OFPDescStatsReply from datapath.

        Parameters ryu_event (ryu.controller.ofp_event.EventOFPDescStatsReply) – trigger.

    error_handler (ryu_event)
        Handle an OFPError from a datapath.

        Parameters ryu_event (ryu.controller.ofp_event.EventOFPErrorMsg) – trigger

    exc_logname = 'faucet.exception'

    features_handler (ryu_event)
        Handle receiving a switch features message from a datapath.

        Parameters ryu_event (ryu.controller.ofp_event.EventOFPStateChange) – trigger.
```

**flowremoved\_handler** (*ryu\_event*)  
 Handle a flow removed event.

**Parameters** **ryu\_event** (*ryu.controller.ofp\_event.EventOFPPFlowRemoved*)  
     – trigger.

**get\_config** ()  
 FAUCET experimental API: return config for all Valves.

**get\_tables** (*dp\_id*)  
 FAUCET experimental API: return config tables for one Valve.

**logname** = 'faucet'

**metric\_update** (\_)  
 Handle a request to update metrics in the controller.

**metrics** = None

**notifier** = None

**packet\_in\_handler** (*ryu\_event*)  
 Handle a packet in event from the dataplane.

**Parameters** **ryu\_event** (*ryu.controller.event.EventReplyBase*) – packet in  
     message.

**port\_status\_handler** (*ryu\_event*)  
 Handle a port status change event.

**Parameters** **ryu\_event** (*ryu.controller.ofp\_event.EventOFPPortStatus*) –  
     trigger.

**reload\_config** (*ryu\_event*)  
 Handle a request to reload configuration.

**start** ()  
 Start controller.

**valves\_manager** = None

### faucet.faucet\_bgp module

BGP implementation for FAUCET.

**class** faucet.faucet\_bgp.**BgpSpeakerKey** (*dp\_id, vlan\_vid, ipv*)

Bases: object

Uniquely describe a BGP speaker.

**class** faucet.faucet\_bgp.**FaucetBgp** (*logger, exc\_logname, metrics, send\_flow\_msgs*)

Bases: object

Wrapper for Ryu BGP speaker.

**exc\_logname** = None

**reset** (*valves*)

Set up a BGP speaker for every VLAN that requires it.

**shutdown\_bgp\_speakers** ()

Shutdown any active BGP speakers.

**update\_metrics** (*\_now*)  
Update BGP metrics.

### faucet.faucet\_dot1x module

802.1x implementation for FAUCET.

**class** faucet.faucet\_dot1x.**FaucetDot1x** (*logger, metrics, send\_flow\_msgs*)  
Bases: object

Wrapper for experimental Chewie 802.1x authenticator.

**auth\_handler** (*address, port\_id*)  
Callback for when a successful auth happens.

**create\_flow\_pair** (*dp\_id, dot1x\_port, nfv\_sw\_port, acl\_manager*)  
Creates the pair of flows that redirects the eapol packets to/from the supplicant and nfv port

#### Parameters

- **dp\_id** (*int*) –
- **dot1x\_port** (*Port*) –
- **nfv\_sw\_port** (*Port*) –
- **acl\_manager** (*ValveAclManager*) –

**Returns** list

**failure\_handler** (*address, port\_id*)  
Callback for when a EAP failure happens.

**get\_valve\_and\_port** (*port\_id*)  
Finds the valve and port that this address corresponds to :param port\_id: is a macaddress string

**logoff\_handler** (*address, port\_id*)  
Callback for when an EAP logoff happens.

**nfv\_sw\_port\_up** (*dp\_id, dot1x\_ports, nfv\_sw\_port, acl\_manager*)  
Setup the dot1x forward port acls when the nfv\_sw\_port comes up. :param dp\_id: :type dp\_id: int :param dot1x\_ports: :type dot1x\_ports: Iterable of Port objects :param nfv\_sw\_port: :type nfv\_sw\_port: Port :param acl\_manager: :type acl\_manager: ValveAclManager

**Returns** list of flowmods

**port\_down** (*dp\_id, dot1x\_port, nfv\_sw\_port, acl\_manager*)  
Remove the acls added by FaucetDot1x.get\_port\_acls :param dp\_id: :type dp\_id: int :param dot1x\_port: :type dot1x\_port: Port :param nfv\_sw\_port: :type nfv\_sw\_port: Port :param acl\_manager: :type acl\_manager: ValveAclManager

**Returns** list of flowmods

**port\_up** (*dp\_id, dot1x\_port, nfv\_sw\_port, acl\_manager*)  
Setup the dot1x forward port acls. :param dp\_id: :type dp\_id: int :param dot1x\_port: :type dot1x\_port: Port :param nfv\_sw\_port: :type nfv\_sw\_port: Port :param acl\_manager: :type acl\_manager: ValveAclManager

**Returns** list of flowmods

**reset** (*valves*)  
Set up a dot1x speaker.

**set\_mac\_str** (*valve, valve\_index, port\_num*)

**Parameters**

- **valve** (*Valve*) –
- **valve\_index** (*int*) –
- **port\_num** (*int*) –

**Returns** *str*

`faucet.faucet_dot1x.get_mac_str (valve_index, port_num)`

Gets the mac address string for the valve/port combo :param valve\_index: The internally used id of the valve.  
:type valve\_index: int :param port\_num: port number :type port\_num: int

**Returns** *str*

### faucet.faucet\_experimental\_api module

Implement experimental API.

**class** `faucet.faucet_experimental_api.FaucetExperimentalAPI (*_args, **_kwargs)`

Bases: `object`

An experimental API for communicating with Faucet.

Contains methods for interacting with a running Faucet controller from within a RyuApp. This app should be run together with Faucet in the same ryu-manager process.

**add\_port\_acl** (*port, acl*)

Add an ACL to a port.

**add\_vlan\_acl** (*vlan, acl*)

Add an ACL to a VLAN.

**delete\_port\_acl** (*port, acl*)

Delete an ACL from a port.

**delete\_vlan\_acl** (*vlan, acl*)

Delete an ACL from a VLAN.

**get\_config** ()

Get the current running config of Faucet as a python dictionary.

**get\_tables** (*dp\_id*)

Get current FAUCET tables as a dict of table name: table no.

**is\_registered** ()

Return True if registered and ready to serve API requests.

**push\_config** (*config*)

Push supplied config to FAUCET.

**reload\_config** ()

Reload config from config file in FAUCET\_CONFIG env variable.

### faucet.faucet\_experimental\_event module

Experimental FAUCET event notification.

```
class faucet.faucet_experimental_event.FaucetExperimentalEventNotifier (socket_path,  
metrics,  
logger)
```

Bases: object

Event notification, via Unix domain socket.

```
check_path (socket_path)  
    Check that socket_path is valid.
```

```
notify (dp_id, dp_name, event_dict)  
    Notify of an event.
```

```
start ()  
    Start socket server.
```

```
class faucet.faucet_experimental_event.NonBlockLock
```

Bases: object

Non blocking lock that can be used as a context manager.

```
acquire_nonblock ()  
    Attempt to acquire a lock.
```

```
release ()  
    Release lock when done.
```

### faucet.faucet\_metadata module

This module contains code relating to the use of OpenFlow Metadata within Faucet.

```
faucet.faucet_metadata.get_egress_metadata (port_num, vid)  
    Return the metadata value to output a packet to port port_num on vlan vid
```

### faucet.faucet\_metrics module

Implement Prometheus statistics.

```
class faucet.faucet_metrics.FaucetMetrics (reg=None)
```

Bases: *faucet.prom\_client.PromClient*

Container class for objects that can be exported to Prometheus.

```
inc_var (var, labels, val=1)
```

```
reset_dp_id (dp_labels)  
    Set all DPID-only counter/gauges to 0.
```

### faucet.faucet\_pipeline module

Standard FAUCET pipeline.

```
class faucet.faucet_pipeline.ValveTableConfig(name, table_id, exact_match=None,
                                             meter=None, output=True,
                                             miss_goto=None, size=None,
                                             match_types=None, set_fields=None,
                                             dec_ttl=None, vlan_port_scale=None,
                                             next_tables=None, metadata_match=0,
                                             metadata_write=0)
```

Bases: object

Configuration for a single table.

## faucet.fctl module

Report state based on FAUCET/Gauge/Prometheus variables.

```
faucet.fctl.decode_value(metric_name, value)
    Convert values to human readable format based on metric name
```

```
faucet.fctl.get_samples(endpoints, metric_name, label_matches, nonzero_only=False, retries=3)
    return a list of Prometheus samples for a given metric
```

Prometheus Sample objects are named tuples with the fields: name, labels, value, timestamp, exemplar.

### Parameters

- **endpoints** (*list of strings*) – the prometheus endpoints to query
- **metric\_name** (*string*) – the metric to retrieve
- **label\_matches** (*dict*) – filters results by label
- **nonzero\_only** (*bool*) – only return samples with non-zero values
- **retries** (*int*) – number of retries when querying

**Returns** list of Prometheus Sample objects

```
faucet.fctl.main()
```

```
faucet.fctl.parse_args(sys_args)
    Parse and return CLI args.
```

```
faucet.fctl.report_label_match_metrics(report_metrics, metrics, display_labels=None,
                                       nonzero_only=False, delim='\t', label_matches=None)
    Text report on a list of Prometheus metrics.
```

```
faucet.fctl.scrape_prometheus(endpoints, retries=3, err_output_file=<_io.TextIOWrapper
                              name='<stdout>' mode='w' encoding='UTF-8'>)
    Scrape a list of Prometheus/FAUCET/Gauge endpoints and aggregate results.
```

## faucet.gauge module

RyuApp shim between Ryu and Gauge.

```
class faucet.gauge.Gauge(*args, **kwargs)
    Bases: faucet.valve_ryuapp.RyuAppBase
```

Ryu app for polling Faucet controlled datapaths for stats/state.

It can poll multiple datapaths. The configuration files for each datapath should be listed, one per line, in the file set as the environment variable GAUGE\_CONFIG. It logs to the file set as the environment variable GAUGE\_LOG,

```
exc_logname = 'gauge.exception'
```

```
logname = 'gauge'
```

```
prom_client = None
```

```
reload_config(ryu_event)
    Handle request for Gauge config reload.
```

```
update_watcher_handler(ryu_event)
    Handle port status change event.
```

Parameters *ryu\_event* (*ryu.controller.event.EventReplyBase*) – port status change event.

### faucet.gauge\_influx module

Library for interacting with InfluxDB.

```
class faucet.gauge_influx.GaugeFlowTableInfluxDBLogger(conf, logname, prom_client)
    Bases: faucet.gauge_pollers.GaugeFlowTablePoller, faucet.gauge_influx.InfluxShipper
```

### Example

```
> use faucet
Using database faucet
> show series where table_id = '0' and in_port = '2'
key
---
flow_byte_count,dp_name=windscale-faucet-1,eth_type=2048,in_port=2,ip_proto=17,
↳priority=9099,table_id=0,udp_dst=53
flow_byte_count,dp_name=windscale-faucet-1,eth_type=2048,in_port=2,ip_proto=6,
↳priority=9098,table_id=0,tcp_dst=53
flow_byte_count,dp_name=windscale-faucet-1,in_port=2,priority=9097,table_id=0
flow_packet_count,dp_name=windscale-faucet-1,eth_type=2048,in_port=2,ip_proto=17,
↳priority=9099,table_id=0,udp_dst=53
flow_packet_count,dp_name=windscale-faucet-1,eth_type=2048,in_port=2,ip_proto=6,
↳priority=9098,table_id=0,tcp_dst=53
flow_packet_count,dp_name=windscale-faucet-1,in_port=2,priority=9097,table_id=0
> select * from flow_byte_count where table_id = '0' and in_port = '2' and ip_
↳proto = '17' and time > now() - 5m
name: flow_byte_count
time                arp_tpa dp_name                eth_dst eth_src eth_type icmpv6_
↳type in_port ip_proto ipv4_dst ipv6_dst priority table_id tcp_dst udp_dst value_
↳vlan_vid
-----
↳
↳
1501154797000000000    windscale-faucet-1                2048
↳ 2 17 9099 0 53 9414
1501154857000000000    windscale-faucet-1                2048
↳ 2 17 9099 0 53 10554
```

(continues on next page)



(continued from previous page)

1501154917000000000	windscale-faucet-1	2048			
↪ 2 17	9099 0	53	10554		↪
1501154977000000000	windscale-faucet-1	2048			
↪ 2 17	9099 0	53	12164		↪
1501155037000000000	windscale-faucet-1	2048			
↪ 2 17	9099 0	53	12239		↪

**update** (*rcv\_time*, *dp\_id*, *msg*)

Handle the responses to requests.

Called when a reply to a stats request sent by this object is received by the controller.

It should acknowledge the receipt by setting `self.reply_pending` to false.**Parameters**

- **rcv\_time** – the time the response was received
- **dp\_id** – DP ID
- **msg** – the stats reply message

**class** `faucet.gauge_influx.GaugePortStateInfluxDBLogger` (*conf*, *logname*, *prom\_client*)Bases: `faucet.gauge_pollers.GaugePortStatePoller`, `faucet.gauge_influx.InfluxShipper`**Example**

```
> use faucet
Using database faucet
> precision rfc3339
> select * from port_state_reason where port_name = 'port1.0.1' order by time_
↪desc limit 10;
name: port_state_reason
-----
time                dp_name                port_name                value
2017-02-21T02:12:29Z windscale-faucet-1     port1.0.1                2
2017-02-21T02:12:25Z windscale-faucet-1     port1.0.1                2
2016-07-27T22:05:08Z windscale-faucet-1     port1.0.1                2
2016-05-25T04:33:00Z windscale-faucet-1     port1.0.1                2
2016-05-25T04:32:57Z windscale-faucet-1     port1.0.1                2
2016-05-25T04:31:21Z windscale-faucet-1     port1.0.1                2
2016-05-25T04:31:18Z windscale-faucet-1     port1.0.1                2
2016-05-25T04:27:07Z windscale-faucet-1     port1.0.1                2
2016-05-25T04:27:04Z windscale-faucet-1     port1.0.1                2
2016-05-25T04:24:53Z windscale-faucet-1     port1.0.1                2
```

**update** (*rcv\_time*, *dp\_id*, *msg*)

Handle the responses to requests.

Called when a reply to a stats request sent by this object is received by the controller.

It should acknowledge the receipt by setting `self.reply_pending` to false.**Parameters**

- **rcv\_time** – the time the response was received
- **dp\_id** – DP ID

- **msg** – the stats reply message

```
class faucet.gauge_influx.GaugePortStatsInfluxDBLogger (conf, logname, prom_client)
  Bases: faucet.gauge_pollers.GaugePortStatsPoller, faucet.gauge_influx.InfluxShipper
```

Periodically sends a port stats request to the datapath and parses and outputs the response.

### Example

```
> use faucet
Using database faucet
> show measurements
name: measurements
-----
bytes_in
bytes_out
dropped_in
dropped_out
errors_in
packets_in
packets_out
port_state_reason
> precision rfc3339
> select * from packets_out where port_name = 'port1.0.1' order by time desc,
↳limit 10;
name: packets_out
-----
time                dp_name                port_name                value
2017-03-06T05:21:42Z  windscale-faucet-1    port1.0.1                76083431
2017-03-06T05:21:33Z  windscale-faucet-1    port1.0.1                76081172
2017-03-06T05:21:22Z  windscale-faucet-1    port1.0.1                76078727
2017-03-06T05:21:12Z  windscale-faucet-1    port1.0.1                76076612
2017-03-06T05:21:02Z  windscale-faucet-1    port1.0.1                76074546
2017-03-06T05:20:52Z  windscale-faucet-1    port1.0.1                76072730
2017-03-06T05:20:42Z  windscale-faucet-1    port1.0.1                76070528
2017-03-06T05:20:32Z  windscale-faucet-1    port1.0.1                76068211
2017-03-06T05:20:22Z  windscale-faucet-1    port1.0.1                76065982
2017-03-06T05:20:12Z  windscale-faucet-1    port1.0.1                76063941
```

```
update (rcv_time, dp_id, msg)
```

Handle the responses to requests.

Called when a reply to a stats request sent by this object is received by the controller.

It should acknowledge the receipt by setting `self.reply_pending` to false.

#### Parameters

- **rcv\_time** – the time the response was received
- **dp\_id** – DP ID
- **msg** – the stats reply message

```
class faucet.gauge_influx.InfluxShipper
  Bases: object
```

Convenience class for shipping values to InfluxDB.

Inheritors must have a `WatcherConf` object as `conf`.

```

conf = None

logger = None

static make_point (tags, rcv_time, stat_name, stat_val)
    Make an InfluxDB point.

make_port_point (dp_name, port_name, rcv_time, stat_name, stat_val)
    Make an InfluxDB point about a port measurement.

ship_error_prefix = 'error shipping points: '

ship_points (points)
    Make a connection to InfluxDB and ship points.

```

### faucet.gauge\_pollers module

Library for polling dataplanes for statistics.

```

class faucet.gauge_pollers.GaugeFlowTablePoller (conf, logname, prom_client)
    Bases: faucet.gauge_pollers.GaugeThreadPoller

    Periodically dumps the current datapath flow table as a yaml object.

    Includes a timestamp and a reference ($DATAPATHNAME-flowtables). The flow table is dumped as an OF-
    FlowStatsReply message (in yaml format) that matches all flows.

    no_response ()
        Called when a polling cycle passes without receiving a response.

    send_req ()
        Send a stats request to a datapath.

class faucet.gauge_pollers.GaugePoller (conf, logname, prom_client)
    Bases: object

    Abstraction for a poller for statistics.

    static is_active ()
        Return True if the poller is controlling the request loop for its stat

    no_response ()
        Called when a polling cycle passes without receiving a response.

    report_dp_status (dp_status)
        Report DP status.

    running ()
        Return True if the poller is running.

    send_req ()
        Send a stats request to a datapath.

    start (ryudp, active)
        Start the poller.

    stop ()
        Stop the poller.

    update (rcv_time, dp_id, msg)
        Handle the responses to requests.

        Called when a reply to a stats request sent by this object is received by the controller.

```

It should acknowledge the receipt by setting `self.reply_pending` to false.

### Parameters

- `rcv_time` – the time the response was received
- `dp_id` – DP ID
- `msg` – the stats reply message

**class** `faucet.gauge_pollers.GaugePortStatePoller` (*conf, logname, prom\_client*)

Bases: `faucet.gauge_pollers.GaugePoller`

Abstraction for port state poller.

**no\_response** ()

Called when a polling cycle passes without receiving a response.

**send\_req** ()

Send a stats request to a datapath.

**class** `faucet.gauge_pollers.GaugePortStatsPoller` (*conf, logname, prom\_client*)

Bases: `faucet.gauge_pollers.GaugeThreadPoller`

Periodically sends a port stats request to the datapath and parses and outputs the response.

**no\_response** ()

Called when a polling cycle passes without receiving a response.

**send\_req** ()

Send a stats request to a datapath.

**class** `faucet.gauge_pollers.GaugeThreadPoller` (*conf, logname, prom\_client*)

Bases: `faucet.gauge_pollers.GaugePoller`

A ryu thread object for sending and receiving OpenFlow stats requests.

The thread runs in a loop sending a request, sleeping then checking a response was received before sending another request.

The methods `send_req`, `update` and `no_response` should be implemented by subclasses.

**is\_active** ()

Return True if the poller is controlling the request loop for its stat

**no\_response** ()

Called when a polling cycle passes without receiving a response.

**send\_req** ()

Send a stats request to a datapath.

**start** (*ryudp, active*)

Start the poller.

**stop** ()

Stop the poller.

## faucet.gauge\_prom module

Prometheus for Gauge.

**class** `faucet.gauge_prom.GaugeFlowTablePrometheusPoller` (*conf, logname, prom\_client*)

Bases: `faucet.gauge_pollers.GaugeFlowTablePoller`

Export flow table entries to Prometheus.

**update** (*rcv\_time, dp\_id, msg*)

Handle the responses to requests.

Called when a reply to a stats request sent by this object is received by the controller.

It should acknowledge the receipt by setting `self.reply_pending` to false.

#### Parameters

- **rcv\_time** – the time the response was received
- **dp\_id** – DP ID
- **msg** – the stats reply message

**class** `faucet.gauge_prom.GaugePortStatePrometheusPoller` (*conf, logname, prom\_client*)

Bases: `faucet.gauge_pollers.GaugePortStatePoller`

Export port state changes to Prometheus.

**update** (*rcv\_time, dp\_id, msg*)

Handle the responses to requests.

Called when a reply to a stats request sent by this object is received by the controller.

It should acknowledge the receipt by setting `self.reply_pending` to false.

#### Parameters

- **rcv\_time** – the time the response was received
- **dp\_id** – DP ID
- **msg** – the stats reply message

**class** `faucet.gauge_prom.GaugePortStatsPrometheusPoller` (*conf, logger, prom\_client*)

Bases: `faucet.gauge_pollers.GaugePortStatsPoller`

Exports port stats to Prometheus.

**update** (*rcv\_time, dp\_id, msg*)

Handle the responses to requests.

Called when a reply to a stats request sent by this object is received by the controller.

It should acknowledge the receipt by setting `self.reply_pending` to false.

#### Parameters

- **rcv\_time** – the time the response was received
- **dp\_id** – DP ID
- **msg** – the stats reply message

**class** `faucet.gauge_prom.GaugePrometheusClient` (*reg=None*)

Bases: `faucet.prom_client.PromClient`

Wrapper for Prometheus client that is shared between all pollers.

**reregister\_flow\_vars** (*table\_name, table\_tags*)

Register the flow variables needed for this client

### faucet.meter module

Configure meters.

```
class faucet.meter.Meter (_id, dp_id, conf)
    Bases: faucet.conf.Conf

    Implement FAUCET configuration for an OpenFlow meter.

    defaults = {'entry': None, 'meter_id': None}
    defaults_types = {'entry': <class 'dict'>, 'meter_id': <class 'int'>}
    entry = None
    entry_msg = None
    meter_id = None
```

### faucet.port module

Port configuration.

```
class faucet.port.Port (_id, dp_id, conf=None)
    Bases: faucet.conf.Conf

    Stores state for ports, including the configuration.

    check_config ()
        Check config at instantiation time for errors, typically via assert.

    defaults = {'acl_in': None, 'acls_in': None, 'description': None, 'dot1x': False,
    defaults_types = {'acl_in': (<class 'str'>, <class 'int'>), 'acls_in': <class 'list'>,
    finalize ()
        Configuration parsing marked complete.

    hosts (vlans=None)
        Return all host cache entries this port has learned (on all or specified VLANs).

    hosts_count (vlans=None)
        Return count of all hosts this port has learned (on all or specified VLANs).

    is_stack_admin_down ()
        Return True if port is in ADMIN_DOWN state.

    is_stack_down ()
        Return True if port is in DOWN state.

    is_stack_init ()
        Return True if port is in INIT state.

    is_stack_up ()
        Return True if port is in UP state.

    lldp_beacon_defaults_types = {'enable': <class 'bool'>, 'org_tlvs': <class 'list'>,
    lldp_beacon_enabled ()
        Return True if LLDP beacon enabled on this port.

    lldp_org_tlv_defaults_types = {'info': (<class 'str'>, <class 'bytearray'>), 'oui':
```

```

mirror_actions ()
    Return OF actions to mirror this port.

running ()
    Return True if port enabled and up.

set_defaults ()
    Set default values and run any basic sanity checks.

stack_admin_down ()
    Change the current stack state to ADMIN_DOWN.

stack_defaults_types = {'dp': <class 'str'>, 'port': (<class 'str'>, <class 'int'>)}

stack_down ()
    Change the current stack state to DOWN.

stack_init ()
    Change the current stack state to INIT_DOWN.

stack_up ()
    Change the current stack state to UP.

vlangs ()
    Return all VLANs this port is in.

```

### faucet.prom\_client module

Implement Prometheus client.

```

class faucet.prom_client.PromClient (reg=None)
    Bases: object

    Prometheus client.

    REQUIRED_LABELS = ['dp_id', 'dp_name']

    server = None

    start (prom_port, prom_addr, use_test_thread=False)
        Start webserver.

    thread = None

```

```

faucet.prom_client.make_wsgi_app (registry)
    Create a WSGI app which serves the metrics from a registry.

```

### faucet.router module

Configure routing between VLANs.

```

class faucet.router.Router (_id, dp_id, conf)
    Bases: faucet.conf.Conf

    Implement FAUCET configuration for a router.

    bgp_as ()
        Return BGP AS.

    bgp_connect_mode ()
        Return BGP connect mode.

```

**bgp\_defaults\_types** = {'as': <class 'int'>, 'connect\_mode': <class 'str'>, 'neighbor\_

**bgp\_ipvs** ()  
Return list of IP versions for BGP configured on this VLAN.

**bgp\_neighbor\_addresses** ()  
Return BGP neighbor addresses.

**bgp\_neighbor\_addresses\_by\_ipv** (*ipv*)  
Return BGP neighbor addresses with specified IP version on this VLAN.

**bgp\_neighbor\_as** ()  
Return BGP neighbor AS number.

**bgp\_port** ()  
Return BGP port.

**bgp\_routerid** ()  
Return BGP router ID.

**bgp\_server\_addresses** ()  
Return BGP server addresses.

**bgp\_server\_addresses\_by\_ipv** (*ipv*)  
Return BGP server addresses with specified IP version on this VLAN.

**bgp\_vlan** ()  
Return BGP VLAN.

**check\_config** ()  
Check config at instantiation time for errors, typically via assert.

**defaults** = {'bgp': {}, 'vlans': None}

**defaults\_types** = {'bgp': <class 'dict'>, 'vlans': <class 'list'>}

**finalize** ()  
Configuration parsing marked complete.

**ipaddress\_fields** = ('neighbor\_addresses', 'server\_addresses')

**set\_bgp\_vlan** (*vlan*)  
Set BGP VLAN.

**set\_defaults** (*defaults=None, conf=None*)  
Set default values and run any basic sanity checks.

**vip\_map** (*ipa*)  
Return VIP for IP address, if any.

### faucet.tfm\_pipeline module

Configure switch tables with TFM messages.

`faucet.tfm_pipeline.load_tables` (*dp, valve\_cl*)  
Configure switch tables with TFM messages.

### faucet.valve module

Implementation of Valve learning layer 2/3 switch.



---

```

class faucet.valve.AlliedTelesis (dp, logname, metrics, notifier, dot1x)
    Bases: faucet.valve.OVSVValve

    Valve implementation for AT.

    DEC_TTL = False

class faucet.valve.ArubaValve (dp, logname, metrics, notifier, dot1x)
    Bases: faucet.valve.TfmValve

    Valve implementation for Aruba.

    DEC_TTL = False

class faucet.valve.CiscoC9KValve (dp, logname, metrics, notifier, dot1x)
    Bases: faucet.valve.TfmValve

    Valve implementation for C9K.

class faucet.valve.NoviFlowValve (dp, logname, metrics, notifier, dot1x)
    Bases: faucet.valve.Valve

    Valve implementation for NoviFlow with static pipeline.

    STATIC_TABLE_IDS = True

    USE_BARRIERS = True

class faucet.valve.OVSVValve (dp, logname, metrics, notifier, dot1x)
    Bases: faucet.valve.Valve

    Valve implementation for OVS.

    USE_BARRIERS = False

class faucet.valve.TfmValve (dp, logname, metrics, notifier, dot1x)
    Bases: faucet.valve.Valve

    Valve implementation that uses OpenFlow send table features messages.

class faucet.valve.Valve (dp, logname, metrics, notifier, dot1x)
    Bases: object

    Generates the messages to configure a datapath as a l2 learning switch.

    Vendor specific implementations may require sending configuration flows. This can be achieved by inheriting
    from this class and overwriting the function switch_features.

    DEC_TTL = True

    GROUPS = True

    STATIC_TABLE_IDS = False

    USE_BARRIERS = True

    acl_manager

    add_authed_mac (port_num, mac)
        Add authed mac address

    add_route (vlan, ip_gw, ip_dst)
        Add route to VLAN routing table.

    advertise (now, _other_values)
        Called periodically to advertise services (eg. IPv6 RAs).

```

### `close_logs ()`

Explicitly close any active loggers.

### `datapath_connect (now, discovered_up_ports)`

Handle Ryu datapath connection event and provision pipeline.

#### Parameters

- `now (float)` – current epoch time.
- `discovered_up_ports (set)` – datapath port numbers that are up.

**Returns** OpenFlow messages to send to datapath.

**Return type** list

### `datapath_disconnect ()`

Handle Ryu datapath disconnection event.

### `del_authed_mac (port_num, mac=None)`

### `del_route (vlan, ip_dst)`

Delete route from VLAN routing table.

### `dot1x`

### `dp`

### `dp_init ()`

Initialize datapath state at connection/re/config time.

### `fast_advertise (now, _other_valves)`

Called periodically to send LLDP/LACP packets.

### `fast_state_expire (now, other_valves)`

Called periodically to verify the state of stack ports.

### `flood_manager`

### `flow_timeout (now, table_id, match)`

Call flow timeout message handler:

#### Parameters

- `now (float)` – current epoch time.
- `table_id (int)` – ID of table where flow was installed.
- `match (dict)` – match conditions for expired flow.

**Returns** OpenFlow messages, if any.

**Return type** list

### `get_config_dict ()`

Return datapath config as a dict for experimental API.

### `get_tunnel_flowmods ()`

Returns flowmods for the tunnels

### `host_manager`

### `lACP_down (port, cold_start=False)`

Return OpenFlow messages when LACP is down on a port.

**lACP\_handler** (*now, pkt\_meta*)

Handle a LACP packet.

We are a currently a passive, non-aggregateable LACP partner.

**Parameters**

- **now** (*float*) – current epoch time.
- **pkt\_meta** (*PacketMeta*) – packet for control plane.

**Returns** OpenFlow messages, if any by Valve.

**Return type** dict

**lACP\_up** (*port*)

Return OpenFlow messages when LACP is up on a port.

**lldp\_handler** (*now, pkt\_meta, other\_valves*)

Handle an LLDP packet.

**Parameters** **pkt\_meta** (*PacketMeta*) – packet for control plane.

**logger**

**logname**

**metrics**

**notifier**

**ofchannel\_log** (*ofmsgs*)

Log OpenFlow messages in text format to debugging log.

**ofchannel\_logger**

**ofdescstats\_handler** (*body*)

Handle OF DP description.

**oferror** (*msg*)

Correlate OFError message with flow we sent, if any.

**Parameters** **msg** (*ryu.controller.ofp\_event.EventOFMsgBase*) – message from datapath.

**parse\_pkt\_meta** (*msg*)

Parse OF packet-in message to PacketMeta.

**parse\_rcv\_packet** (*in\_port, vlan\_vid, eth\_type, data, orig\_len, pkt, eth\_pkt, vlan\_pkt*)

Parse a received packet into a PacketMeta instance.

**Parameters**

- **in\_port** (*int*) – port packet was received on.
- **vlan\_vid** (*int*) – VLAN VID of port packet was received on.
- **eth\_type** (*int*) – Ethernet type of packet.
- **data** (*bytes*) – Raw packet data.
- **orig\_len** (*int*) – Original length of packet.
- **pkt** (*ryu.lib.packet.packet*) – parsed packet received.
- **eth\_pkt** (*ryu.lib.packet.ethernet*) – parsed Ethernet header.
- **vlan\_pkt** (*ryu.lib.packet.vlan*) – parsed VLAN Ethernet header.

**Returns** PacketMeta instance.

### **pipeline**

#### **port\_add** (*port\_num*)

Handle addition of a single port.

**Parameters** **port\_num** (*list*) – list of port numbers.

**Returns** OpenFlow messages, if any.

**Return type** list

#### **port\_delete** (*port\_num*)

Return flow messages that delete port from pipeline.

#### **port\_status\_handler** (*port\_no, reason, state, \_other\_valves*)

Return OpenFlow messages responding to port operational status change.

#### **ports\_add** (*port\_nums, cold\_start=False, log\_msg='up'*)

Handle the addition of ports.

##### **Parameters**

- **port\_num** (*list*) – list of port numbers.
- **cold\_start** (*bool*) – True if configuring datapath from scratch.

**Returns** OpenFlow messages, if any.

**Return type** list

#### **ports\_delete** (*port\_nums, log\_msg='down'*)

Handle the deletion of ports.

**Parameters** **port\_nums** (*list*) – list of port numbers.

**Returns** OpenFlow messages, if any.

**Return type** list

#### **prepare\_send\_flows** (*flow\_msgs*)

Prepare to send flows to datapath.

**Parameters** **flow\_msgs** (*list*) – OpenFlow messages to send.

#### **rate\_limit\_packet\_ins** (*now*)

Return True if too many packet ins this second.

#### **rcv\_packet** (*now, other\_valves, pkt\_meta*)

Handle a packet from the dataplane (eg to re/learn a host).

The packet may be sent to us also in response to FAUCET initiating IPv6 neighbor discovery, or ARP, to resolve a nexthop.

##### **Parameters**

- **other\_valves** (*list*) – all Valves other than this one.
- **pkt\_meta** ([PacketMeta](#)) – packet for control plane.

**Returns** OpenFlow messages, if any by Valve.

**Return type** dict

#### **recent\_ofmsgs**

**reload\_config** (*\_now, new\_dp*)

Reload configuration new\_dp.

**Following config changes are currently supported:**

- **Port config: support all available configs** (e.g. native\_vlan, acl\_in) & change operations (add, delete, modify) a port
- **ACL config:support any modification, currently reload all** rules belonging to an ACL
- VLAN config: enable, disable routing, etc...

**Parameters**

- **now** (*float*) – current epoch time.
- **new\_dp** (*DP*) – new dataplane configuration.

**Returns** OpenFlow messages.

**Return type** ofmsgs (list)

**resolve\_gateways** (*now, \_other\_valves*)

Call route managers to re/resolve gateways.

**Returns** OpenFlow messages, if any by Valve.

**Return type** dict

**send\_flows** (*ryu\_dp, flow\_msgs*)

Send flows to datapath (or disconnect an OF session).

**Parameters**

- **ryu\_dp** (*ryu.controller.controller.Datapath*) – datapath.
- **flow\_msgs** (*list*) – OpenFlow messages to send.

**state\_expire** (*now, other\_valves*)

Expire controller caches/state (e.g. hosts learned).

**Parameters**

- **now** (*float*) – current epoch time.
- **other\_valves** (*list*) – all Valves other than this one.

**Returns** OpenFlow messages, if any by Valve.

**Return type** dict

**switch\_features** (*\_msg*)

Send configuration flows necessary for the switch implementation.

**Parameters** **msg** (*OFPSwitchFeatures*) – msg sent from switch.

Vendor specific configuration should be implemented here.

**update\_config\_metrics** ()

Update table names for configuration.

**update\_metrics** (*now, updated\_port=None, rate\_limited=False*)

Update Gauge/metrics.

**update\_tunnel\_flowrules** ()

Update tunnel ACL rules because the stack topology has changed

**class** `faucet.valve.ValveLogger` (*logger, dp\_id, dp\_name*)

Bases: `object`

Logger for a Valve that adds DP ID.

**debug** (*log\_msg*)

Log debug level message.

**error** (*log\_msg*)

Log error level message.

**info** (*log\_msg*)

Log info level message.

**warning** (*log\_msg*)

Log warning level message.

`faucet.valve.valve_factory` (*dp*)

Return a Valve object based dp's hardware configuration field.

**Parameters** `dp` (DP) – DP instance with the configuration for this Valve.

### faucet.valve\_acl module

Compose ACLs on ports.

**class** `faucet.valve_acl.ValveAclManager` (*port\_acl\_table, vlan\_acl\_table, pipeline, meters, dp\_acls=None*)

Bases: `faucet.valve_manager_base.ValveManagerBase`

Handle installation of ACLs on a DP

**add\_authed\_mac** (*port\_num, mac*)

Add authed mac address

**add\_port** (*port*)

Install port acls if configured

**add\_vlan** (*vlan*)

Install vlan acls if configured

**cold\_start\_port** (*port*)

Reload acl for a port by deleting existing rules and calling add\_port

**create\_acl\_tunnel** (*dp*)

Create tunnel acls from ACLs that require applying in DP Returns flowmods for the tunnel :param dp: DP that contains the tunnel acls to build :type dp: DP

**create\_dot1x\_flow\_pair** (*dot1x\_port, nfv\_sw\_port, mac*)

Create dot1x flow pair

**del\_authed\_mac** (*port\_num, mac=None*)

remove authed mac address

**del\_dot1x\_flow\_pair** (*dot1x\_port, nfv\_sw\_port, mac*)

Deletes dot1x flow pair

**initialise\_tables** ()

Install dp acls if configured

`faucet.valve_acl.build_acl_entry` (*acl\_table, rule\_conf, meters, acl\_allow\_inst, acl\_force\_port\_vlan\_inst, port\_num=None, vlan\_vid=None*)

Build flow/groupmods for one ACL rule entry.

`faucet.valve_acl.build_acl_ofmsgs` (*acls, acl\_table, acl\_allow\_inst, acl\_force\_port\_vlan\_inst, highest\_priority, meters, exact\_match, port\_num=None, vlan\_vid=None*)

Build flow/groupmods for all entries in an ACL.

`faucet.valve_acl.build_output_actions` (*acl\_table, output\_dict*)

Implement actions to alter packet/output.

`faucet.valve_acl.push_vlan` (*acl\_table, vlan\_vid*)

Push a VLAN tag with optional selection of eth type.

`faucet.valve_acl.rewrite_vlan` (*acl\_table, output\_dict*)

Implement actions to rewrite VLAN headers.

## faucet.valve\_flood module

Manage flooding to ports on VLANs.

**class** `faucet.valve_flood.ValveFloodManager` (*logger, flood\_table, pipeline, use\_group\_table, groups, combinatorial\_port\_flood*)

Bases: `faucet.valve_manager_base.ValveManagerBase`

Implement dataplane based flooding for standalone dataplanes.

**FLOOD\_DSTS** = ((**True, None, None**), (**False, '01:80:c2:00:00:00', 'ff:ff:ff:00:00:00'**), (

**add\_vlan** (*vlan*)

install flows in response to a new vlan

**build\_flood\_rules** (*vlan, modify=False*)

Add flows to flood packets to unknown destinations on a VLAN.

**static edge\_learn\_port** (*\_other\_valves, pkt\_meta*)

Possibly learn a host on a port.

### Parameters

- **other\_valves** (*list*) – All Valves other than this one.
- **pkt\_meta** (`PacketMeta`) – `PacketMeta` instance for packet received.

**Returns** port to learn host on.

**initialise\_tables** ()

Initialise the flood table with filtering flows.

**static update\_stack\_topo** (*event, dp, port=None*)

Update the stack topology. It has nothing to do for non-stacking DPs.

**class** `faucet.valve_flood.ValveFloodStackManager` (*logger, flood\_table, pipeline, use\_group\_table, groups, combinatorial\_port\_flood, stack, stack\_ports, dp\_shortest\_path\_to\_root, shortest\_path\_port*)

Bases: `faucet.valve_flood.ValveFloodManager`

Implement dataplane based flooding for stacked dataplanes.

**EXT\_PORT\_FLAG** = 1

**NONEXT\_PORT\_FLAG** = 0

**edge\_learn\_port** (*other\_valves, pkt\_meta*)

Possibly learn a host on a port.

### Parameters

- **other\_valves** (*list*) – All Valves other than this one.
- **pkt\_meta** (*PacketMeta*) – PacketMeta instance for packet received.

**Returns** port to learn host on, or None.

**update\_stack\_topo** (*event, dp, port=None*)  
Update the stack topo according to the event.

### faucet.valve\_host module

Manage host learning on VLANs.

```
class faucet.valve_host.ValveHostFlowRemovedManager(logger, ports, vlans,  
eth_src_table, eth_dst_table,  
eth_dst_hairpin_table,  
pipeline, learn_timeout,  
learn_jitter, learn_ban_timeout,  
cache_update_guard_time,  
idle_dst, stack)
```

Bases: *faucet.valve\_host.ValveHostManager*

Trigger relearning on flow removed notifications.

---

**Note:** not currently reliable.

---

**expire\_hosts\_from\_vlan** (*\_vlan, \_now*)  
Expire hosts from VLAN cache.

**flow\_timeout** (*now, table\_id, match*)  
Handle a flow timed out message from dataplane.

**learn\_host\_timeouts** (*port*)  
Calculate flow timeouts for learning on a port.

```
class faucet.valve_host.ValveHostManager(logger, ports, vlans, eth_src_table,  
eth_dst_table, eth_dst_hairpin_table, pipeline,  
learn_timeout, learn_jitter, learn_ban_timeout,  
cache_update_guard_time, idle_dst, stack)
```

Bases: *faucet.valve\_manager\_base.ValveManagerBase*

Manage host learning on VLANs.

**add\_port** (*port*)  
initialise override\_output\_port if necessary

**ban\_rules** (*pkt\_meta*)  
Limit learning to a maximum configured on this port/VLAN.

**Parameters** **pkt\_meta** – PacketMeta instance.

**Returns** OpenFlow messages, if any.

**Return type** list

**del\_port** (*port*)  
delete flows in response to a port removal



**delete\_host\_from\_vlan** (*eth\_src, vlan*)  
Delete a host from a VLAN.

**expire\_hosts\_from\_vlan** (*vlan, now*)  
Expire hosts from VLAN cache.

**flow\_timeout** (*\_now, \_table\_id, \_match*)  
Handle a flow timed out message from dataplane.

**initialise\_tables** ()  
initialise tables controlled by this manager

**learn\_host\_on\_vlan\_port\_flows** (*port, vlan, eth\_src, delete\_existing, refresh\_rules, src\_rule\_idle\_timeout, src\_rule\_hard\_timeout, dst\_rule\_idle\_timeout*)  
Return flows that implement learning a host on a port.

**learn\_host\_on\_vlan\_ports** (*now, port, vlan, eth\_src, delete\_existing=True, last\_dp\_coldstart\_time=None*)  
Learn a host on a port.

**learn\_host\_timeouts** (*port*)  
Calculate flow timeouts for learning on a port.

### faucet.valve\_manager\_base module

Valve Manager base class

**class** faucet.valve\_manager\_base.ValveManagerBase

Bases: object

Base class for ValveManager objects.

Expected to control the installation of flows into datapath tables.

Ideally each datapath table should be controlled by 1 manager only.

**add\_port** (*port*)  
install flows in response to a new port

**add\_vlan** (*vlan*)  
install flows in response to a new vlan

**del\_port** (*port*)  
delete flows in response to a port removal

**del\_vlan** (*vlan*)  
delete flows in response to a vlan removal

**initialise\_tables** ()  
initialise tables controlled by this manager

### faucet.valve\_of module

Utility functions to parse/create OpenFlow messages.

faucet.valve\_of.**apply\_actions** (*actions*)

Return instruction that applies action list.

**Parameters** *actions* (*list*) – list of OpenFlow actions.

**Returns** instruction of actions.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPInstruction`

`faucet.valve_of.apply_meter` (*meter\_id*)

Return instruction to apply a meter.

`faucet.valve_of.barrier` ()

Return OpenFlow barrier request.

**Returns** barrier request.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPBarrierRequest`

`faucet.valve_of.bucket` (*weight=0, watch\_port=4294967295, watch\_group=4294967295, actions=None*)

Return a group action bucket with provided actions.

`faucet.valve_of.build_group_flood_buckets` (*vlan\_flood\_acts*)

Return a list of group buckets to implement flooding on a VLAN.

`faucet.valve_of.build_match_dict` (*in\_port=None, vlan=None, eth\_type=None, eth\_src=None, eth\_dst=None, eth\_dst\_mask=None, icmpv6\_type=None, nw\_proto=None, nw\_dst=None, metadata=None, metadata\_mask=None, vlan\_pcp=None*)

`faucet.valve_of.controller_pps_meteradd` (*datapath=None, pps=0*)

Add a PPS meter towards controller.

`faucet.valve_of.controller_pps_meterdel` (*datapath=None*)

Delete a PPS meter towards controller.

`faucet.valve_of.dec_ip_ttl` ()

Return OpenFlow action to decrement IP TTL.

**Returns** decrement IP TTL.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPActionDecNwTtl`

`faucet.valve_of.dedupe_ofmsgs` (*input\_ofmsgs*)

Return deduplicated ofmsg list.

`faucet.valve_of.dedupe_output_port_acts` (*output\_port\_acts*)

Deduplicate parser.OFPActionOutputs (because Ryu doesn't define `__eq__`).

**Parameters of `ryu.ofproto.ofproto_v1_3_parser.OFPActionOutput`** (*list*) – output to port actions.

**Returns** output to port actions.

**Return type** list of `ryu.ofproto.ofproto_v1_3_parser.OFPActionOutput`

`faucet.valve_of.desc_stats_request` (*datapath=None*)

Query switch description.

`faucet.valve_of.devid_present` (*vid*)

Return VLAN VID without VID\_PRESENT flag set.

**Parameters** *vid* (*int*) – VLAN VID with VID\_PRESENT.

**Returns** VLAN VID.

**Return type** `int`

`faucet.valve_of.faucet_async` (*datapath=None, notify\_flow\_removed=False, packet\_in=True, port\_status=True*)  
Return async message config for FAUCET/Gauge

`faucet.valve_of.faucet_config` (*datapath=None*)  
Return switch config for FAUCET.

`faucet.valve_of.flood_port_outputs` (*tagged\_ports, untagged\_ports, in\_port=None, exclude\_ports=None*)  
Return actions for both tagged and untagged ports.

`faucet.valve_of.flood_tagged_port_outputs` (*ports, in\_port=None, exclude\_ports=None*)  
Return list of actions necessary to flood to list of tagged ports.

`faucet.valve_of.flood_untagged_port_outputs` (*ports, in\_port=None, exclude\_ports=None*)  
Return list of actions necessary to flood to list of untagged ports.

`faucet.valve_of.flowmod` (*cookie, command, table\_id, priority, out\_port, out\_group, match\_fields, inst, hard\_timeout, idle\_timeout, flags=0*)

`faucet.valve_of.goto_table` (*table*)  
Return instruction to goto table.

**Parameters** `table` (`ValveTable`) – table to goto.

**Returns** goto instruction.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPInstruction`

`faucet.valve_of.group_act` (*group\_id*)  
Return an action to run a group.

`faucet.valve_of.groupadd` (*datapath=None, type\_=0, group\_id=0, buckets=None*)  
Add a group.

`faucet.valve_of.groupadd_ff` (*datapath=None, group\_id=0, buckets=None*)  
Add a fast failover group.

`faucet.valve_of.groupdel` (*datapath=None, group\_id=4294967292*)  
Delete a group (default all groups).

`faucet.valve_of.groupmod` (*datapath=None, type\_=0, group\_id=0, buckets=None*)  
Modify a group.

`faucet.valve_of.groupmod_ff` (*datapath=None, group\_id=0, buckets=None*)  
Modify a fast failover group.

`faucet.valve_of.ignore_port` (*port\_num*)  
Return True if FAUCET should ignore this port.

**Parameters** `port_num` (*int*) – switch port.

**Returns** True if FAUCET should ignore this port.

**Return type** `bool`

`faucet.valve_of.is_apply_actions` (*instruction*)  
Return True if an apply action.

**Parameters** `instruction` – OpenFlow instruction.

**Returns** True if an apply action.

**Return type** `bool`

`faucet.valve_of.is_delete` (*ofmsg*)

`faucet.valve_of.is_flowdel` (*ofmsg*)

Return True if flow message is a FlowMod and a delete.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a FlowMod delete/strict.

**Return type** bool

`faucet.valve_of.is_flowmod` (*ofmsg*)

Return True if flow message is a FlowMod.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a FlowMod

**Return type** bool

`faucet.valve_of.is_groupadd` (*ofmsg*)

Return True if OF message is a GroupMod and command is add.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a GroupMod add

**Return type** bool

`faucet.valve_of.is_groupdel` (*ofmsg*)

Return True if OF message is a GroupMod and command is delete.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a GroupMod delete

**Return type** bool

`faucet.valve_of.is_groupmod` (*ofmsg*)

Return True if OF message is a GroupMod.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a GroupMod

**Return type** bool

`faucet.valve_of.is_meter` (*instruction*)

Return True if a meter.

**Parameters** `instruction` – OpenFlow instruction.

**Returns** True if a meter.

**Return type** bool

`faucet.valve_of.is_meteradd` (*ofmsg*)

Return True if OF message is a MeterMod and command is add.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a MeterMod add

**Return type** bool

`faucet.valve_of.is_meterdel` (*ofmsg*)

Return True if OF message is a MeterMod and command is delete.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a MeterMod delete

**Return type** bool

`faucet.valve_of.is_metermod(ofmsg)`

Return True if OF message is a MeterMod.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a MeterMod

**Return type** bool

`faucet.valve_of.is_output(ofmsg)`

Return True if flow message is an action output message.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a OFPActionOutput.

**Return type** bool

`faucet.valve_of.is_packetout(ofmsg)`

Return True if OF message is a PacketOut

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a PacketOut

**Return type** bool

`faucet.valve_of.is_set_field(action)`

`faucet.valve_of.is_table_features_req(ofmsg)`

Return True if flow message is a TFM req.

**Parameters** `ofmsg` – ryu.ofproto.ofproto\_v1\_3\_parser message.

**Returns** True if is a TFM req.

**Return type** bool

`faucet.valve_of.match(match_fields)`

Return OpenFlow matches from dict.

**Parameters** `match_fields` (*dict*) – match fields and values.

**Returns** matches.

**Return type** ryu.ofproto.ofproto\_v1\_3\_parser.OFPMatch

`faucet.valve_of.match_from_dict(match_dict)`

`faucet.valve_of.metadata_goto_table(metadata, mask, table)`

Return instructions to write metadata and goto table.

**Parameters**

- **metadata** (*int*) – metadata to write to packet
- **maks** (*int*) – mask to apply to metadata
- **table** (`ValveTable`) – table to goto.

**Returns** list of OFPInstructions

`faucet.valve_of.meteradd(meter_conf)`

Add a meter based on YAML configuration.

`faucet.valve_of.meterdel(datapath=None, meter_id=4294967295)`

Delete a meter (default all meters).

`faucet.valve_of.output_controller(max_len=128)`

Return OpenFlow action to packet in to the controller.

**Parameters** `max_len(int)` – max number of bytes from packet to output.

**Returns** packet in action.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPActionOutput`

`faucet.valve_of.output_in_port()`

Return OpenFlow action to output out input port.

**Returns** `ryu.ofproto.ofproto_v1_3_parser.OFPActionOutput`.

`faucet.valve_of.output_port(port_num, max_len=0)`

Return OpenFlow action to output to a port.

**Parameters**

- `port_num(int)` – port to output to.
- `max_len(int)` – maximum length of packet to output (default no maximum).

**Returns** output to port action.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPActionOutput`

`faucet.valve_of.packetout(port_num, data)`

Return OpenFlow action to packet out to dataplane from controller.

**Parameters**

- `port_num(int)` – port to output to.
- `data(str)` – raw packet to output.

**Returns** packet out action.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPActionOutput`

`faucet.valve_of.packetouts(port_nums, data)`

Return OpenFlow action to multiply packet out to dataplane from controller.

**Parameters**

- `port_num(list)` – ints, ports to output to.
- `data(str)` – raw packet to output.

**Returns** packet out action.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPActionOutput`

`faucet.valve_of.pop_vlan()`

Return OpenFlow action to pop outermost Ethernet 802.1Q VLAN header.

**Returns** Pop VLAN.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPActionPopVlan`

`faucet.valve_of.port_status_from_state(state)`

Return True if OFPPS\_LINK\_DOWN is not set.

`faucet.valve_of.push_vlan_act(table, vlan_vid, eth_type=33024)`

Return OpenFlow action list to push Ethernet 802.1Q header with VLAN VID.

**Parameters** `vid(int)` – VLAN VID

**Returns** actions to push 802.1Q header with VLAN VID set.

**Return type** list

`faucet.valve_of.set_field(**kws)`

Return action to set any field.

**Parameters** `kws` (*dict*) – exactly one field to set

**Returns** set field action.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPActionSetField`

`faucet.valve_of.table_features(body)`

`faucet.valve_of.valve_flowreorder(input_ofmsgs, use_barriers=True)`

Reorder flows for better OFA performance.

`faucet.valve_of.valve_match_vid(value)`

`faucet.valve_of.vid_present(vid)`

Return VLAN VID with VID\_PRESENT flag set.

**Parameters** `vid` (*int*) – VLAN VID

**Returns** VLAN VID with VID\_PRESENT.

**Return type** `int`

### `faucet.valve_of_old` module

Deprecated OF matches.

### `faucet.valve_packet` module

Utility functions for parsing and building Ethernet packet/contents.

**class** `faucet.valve_packet.PacketMeta` (*data, orig\_len, pkt, eth\_pkt, vlan\_pkt, port, valve\_vlan, eth\_src, eth\_dst, eth\_type*)

Bases: `object`

Original, and parsed Ethernet packet metadata.

`ETH_TYPES_PARSERS = {2048: (4, <function ipv4_parseable>, <class 'ryu.lib.packet.ipv4`

`MAX_ETH_TYPE_PKT_SIZE = {2048: 110, 2054: 64}`

`MIN_ETH_TYPE_PKT_SIZE = {2048: 38, 2054: 46, 34525: 58}`

`data`

`eth_dst`

`eth_pkt`

`eth_src`

`eth_type`

`ip_ver()`

Return IP version number.

`l3_dst`

`l3_pkt`

`l3_src`

**log()**

**orig\_len**

**packet\_complete()**

True if we have the complete packet.

**pkt**

**port**

**reparse(max\_len)**

Reparse packet using data up to the specified maximum length.

**reparse\_all()**

Reparse packet with all available data.

**reparse\_ip(payload=0)**

Reparse packet with specified IP header type and optionally payload.

**vlan**

**vlan\_pkt**

`faucet.valve_packet.arp_reply(vid, eth_src, eth_dst, src_ip, dst_ip)`

Return an ARP reply packet.

### Parameters

- **vid** (*int or None*) – VLAN VID to use (or None).
- **eth\_src** (*str*) – Ethernet source address.
- **eth\_dst** (*str*) – destination Ethernet MAC address.
- **src\_ip** (*ipaddress.IPv4Address*) – source IPv4 address.
- **dst\_ip** (*ipaddress.IPv4Address*) – destination IPv4 address.

**Returns** serialized ARP reply packet.

**Return type** `ryu.lib.packet.arp`

`faucet.valve_packet.arp_request(vid, eth_src, eth_dst, src_ip, dst_ip)`

Return an ARP request packet.

### Parameters

- **vid** (*int or None*) – VLAN VID to use (or None).
- **eth\_src** (*str*) – Ethernet source address.
- **eth\_dst** (*str*) – Ethernet destination address.
- **src\_ip** (*ipaddress.IPv4Address*) – source IPv4 address.
- **dst\_ip** (*ipaddress.IPv4Address*) – requested IPv4 address.

**Returns** serialized ARP request packet.

**Return type** `ryu.lib.packet.arp`

`faucet.valve_packet.build_pkt_header(vid, eth_src, eth_dst, dl_type)`

Return an Ethernet packet header.

### Parameters

- **vid** (*int or None*) – VLAN VID to use (or None).



- **eth\_src** (*str*) – source Ethernet MAC address.
- **eth\_dst** (*str*) – destination Ethernet MAC address.
- **dl\_type** (*int*) – EtherType.

**Returns** Ethernet packet with header.

**Return type** ryu.lib.packet.ethernet

`faucet.valve_packet.echo_reply` (*vid, eth\_src, eth\_dst, src\_ip, dst\_ip, data*)  
Return an ICMP echo reply packet.

**Parameters**

- **vid** (*int or None*) – VLAN VID to use (or None).
- **eth\_src** (*str*) – Ethernet source address.
- **eth\_dst** (*str*) – destination Ethernet MAC address.
- **src\_ip** (*ipaddress.IPv4Address*) – source IPv4 address.
- **dst\_ip** (*ipaddress.IPv4Address*) – destination IPv4 address.

**Returns** serialized ICMP echo reply packet.

**Return type** ryu.lib.packet.icmp

`faucet.valve_packet.faucet_lldp_stack_state_tlvs` (*dp, port*)  
Return a LLDP TLV for state of a stack port.

`faucet.valve_packet.faucet_lldp_tlvs` (*dp*)  
Return LLDP TLVs for a datapath.

`faucet.valve_packet.faucet_oui` (*mac*)  
Return first 3 bytes of MAC address (given as str).

`faucet.valve_packet.faucet_tlvs` (*lldp\_pkt, faucet\_dp\_mac*)  
Return list of TLVs with FAUCET OUI.

`faucet.valve_packet.icmpv6_echo_reply` (*vid, eth\_src, eth\_dst, src\_ip, dst\_ip, hop\_limit, id\_, seq, data*)  
Return IPv6 ICMP echo reply packet.

**Parameters**

- **vid** (*int or None*) – VLAN VID to use (or None).
- **eth\_src** (*str*) – source Ethernet MAC address.
- **eth\_dst** (*str*) – destination Ethernet MAC address.
- **src\_ip** (*ipaddress.IPv6Address*) – source IPv6 address.
- **dst\_ip** (*ipaddress.IPv6Address*) – destination IPv6 address.
- **hop\_limit** (*int*) – IPv6 hop limit.
- **id** (*int*) – identifier for echo reply.
- **seq** (*int*) – sequence number for echo reply.
- **data** (*str*) – payload for echo reply.

**Returns** Serialized IPv6 ICMP echo reply packet.

**Return type** ryu.lib.packet.ethernet

`faucet.valve_packet.int_from_mac` (*mac*)

`faucet.valve_packet.int_in_mac` (*mac*, *to\_int*)

`faucet.valve_packet.ipv4_parseable` (*ip\_header\_data*)  
Return True if an IPv4 packet we could parse.

`faucet.valve_packet.ipv6_link_eth_mcast` (*dst\_ip*)  
Return an Ethernet multicast address from an IPv6 address.

See RFC 2464 section 7.

**Parameters** `dst_ip` (*ipaddress.IPv6Address*) – IPv6 address.

**Returns** Ethernet multicast address.

**Return type** `str`

`faucet.valve_packet.ipv6_solicited_node_from_ucast` (*ucast*)  
Return IPv6 solicited node multicast address from IPv6 unicast address.

See RFC 3513 section 2.7.1.

**Parameters** `ucast` (*ipaddress.IPv6Address*) – IPv6 unicast address.

**Returns** IPv6 solicited node multicast address.

**Return type** `ipaddress.IPv6Address`

`faucet.valve_packet.lacp_reqreply` (*eth\_src*, *actor\_system*, *actor\_key*, *actor\_port*, *actor\_state\_synchronization=0*, *actor\_state\_activity=0*, *partner\_system='00:00:00:00:00:00'*, *partner\_key=0*, *partner\_port=0*, *partner\_system\_priority=0*, *partner\_port\_priority=0*, *partner\_state\_defaulted=0*, *partner\_state\_expired=0*, *partner\_state\_timeout=0*, *partner\_state\_collecting=0*, *partner\_state\_distributing=0*, *partner\_state\_aggregation=0*, *partner\_state\_synchronization=0*, *partner\_state\_activity=0*)

Return a LACP frame.

**Parameters**

- **eth\_src** (*str*) – source Ethernet MAC address.
- **actor\_system** (*str*) – actor system ID (MAC address)
- **actor\_key** (*int*) – actor’s LACP key assigned to this port.
- **actor\_port** (*int*) – actor port number.
- **actor\_state\_synchronization** (*int*) – 1 if we will use this link.
- **actor\_state\_activity** (*int*) – 1 if actively sending LACP.
- **partner\_system** (*str*) – partner system ID (MAC address)
- **partner\_key** (*int*) – partner’s LACP key assigned to this port.
- **partner\_port** (*int*) – partner port number.
- **partner\_system\_priority** (*int*) – partner’s system priority.
- **partner\_port\_priority** (*int*) – partner’s port priority.
- **partner\_state\_defaulted** (*int*) – 1 if partner reverted to defaults.
- **partner\_state\_expired** (*int*) – 1 if partner thinks LACP expired.
- **partner\_state\_timeout** (*int*) – 1 if partner has short timeout.

- **partner\_state\_collecting** (*int*) – 1 if partner receiving on this link.
- **partner\_state\_distributing** (*int*) – 1 if partner transmitting on this link.
- **partner\_state\_aggregation** (*int*) – 1 if partner can aggregate this link.
- **partner\_state\_synchronization** (*int*) – 1 if partner will use this link.
- **partner\_state\_activity** (*int*) – 1 if partner actively sends LACP.

**Returns** Ethernet packet with header.

**Return type** ryu.lib.packet.ethernet

`faucet.valve_packet.lldp_beacon(eth_src, chassis_id, port_id, ttl, org_tlvs=None, system_name=None, port_descr=None)`

Return an LLDP frame suitable for a host/access port.

**Parameters**

- **eth\_src** (*str*) – source Ethernet MAC address.
- **chassis\_id** (*str*) – Chassis ID.
- **port\_id** (*int*) – port ID,
- **TTL** (*int*) – TTL for payload.
- **org\_tlvs** (*list*) – list of tuples of (OUI, subtype, info).

**Returns** Ethernet packet with header.

**Return type** ryu.lib.packet.ethernet

`faucet.valve_packet.mac_addr_is_unicast(mac_addr)`

Returns True if mac\_addr is a unicast Ethernet address.

**Parameters** **mac\_addr** (*str*) – MAC address.

**Returns** True if a unicast Ethernet address.

**Return type** bool

`faucet.valve_packet.mac_byte_mask(mask_bytes=0)`

Return a MAC address mask with n bytes masked out.

`faucet.valve_packet.mac_mask_bits(mac_mask)`

Return number of bits in MAC mask or 0.

`faucet.valve_packet.nd_advert(vid, eth_src, eth_dst, src_ip, dst_ip)`

Return IPv6 neighbor advertisement packet.

**Parameters**

- **vid** (*int or None*) – VLAN VID to use (or None).
- **eth\_src** (*str*) – source Ethernet MAC address.
- **eth\_dst** (*str*) – destination Ethernet MAC address.
- **src\_ip** (*ipaddress.IPv6Address*) – source IPv6 address.
- **dst\_ip** (*ipaddress.IPv6Address*) – destination IPv6 address.

**Returns** Serialized IPv6 neighbor discovery packet.

**Return type** ryu.lib.packet.ethernet

`faucet.valve_packet.nd_request(vid, eth_src, eth_dst, src_ip, dst_ip)`

Return IPv6 neighbor discovery request packet.

### Parameters

- **vid** (*int* or *None*) – VLAN VID to use (or None).
- **eth\_src** (*str*) – source Ethernet MAC address.
- **eth\_dst** (*str*) – Ethernet destination address.
- **src\_ip** (*ipaddress.IPv6Address*) – source IPv6 address.
- **dst\_ip** (*ipaddress.IPv6Address*) – requested IPv6 address.

**Returns** Serialized IPv6 neighbor discovery packet.

**Return type** `ryu.lib.packet.ethernet`

`faucet.valve_packet.parse_eth_pkt(pkt)`

Return parsed Ethernet packet.

**Parameters** **pkt** (*ryu.lib.packet.packet*) – packet received from dataplane.

**Returns** Ethernet packet.

**Return type** `ryu.lib.packet.ethernet`

`faucet.valve_packet.parse_faucet_lldp(lldp_pkt, faucet_dp_mac)`

Parse and return FAUCET TLVs from LLDP packet.

**Parameters** **pkt** (*ryu.lib.packet.packet*) – packet received from dataplane.

**Returns** LACP packet.

**Return type** `ryu.lib.packet.lacp`

`faucet.valve_packet.parse_lldp(pkt)`

Return parsed LLDP packet.

**Parameters** **pkt** (*ryu.lib.packet.packet*) – packet received from dataplane.

**Returns** LLDP packet.

**Return type** `ryu.lib.packet.lldp`

`faucet.valve_packet.parse_packet_in_pkt(data, max_len, eth_pkt=None, vlan_pkt=None)`

Parse a packet received via packet in from the dataplane.

### Parameters

- **data** (*bytearray*) – packet data from dataplane.
- **max\_len** (*int*) – max number of packet data bytes to parse.

**Returns** raw packet `ryu.lib.packet.ethernet`: parsed Ethernet packet. `int`: Ethernet type of packet (inside VLAN) `int`: VLAN VID (or None if no VLAN)

**Return type** `ryu.lib.packet.packet`

`faucet.valve_packet.router_advert(vid, eth_src, eth_dst, src_ip, dst_ip, vips, pi_flags=6)`

Return IPv6 ICMP Router Advert.

### Parameters

- **vid** (*int* or *None*) – VLAN VID to use (or None).
- **eth\_src** (*str*) – source Ethernet MAC address.

- **eth\_dst** (*str*) – dest Ethernet MAC address.
- **src\_ip** (*ipaddress.IPv6Address*) – source IPv6 address.
- **vips** (*list*) – prefixes (*ipaddress.IPv6Address*) to advertise.
- **pi\_flags** (*int*) – flags to set in prefix information field (default set A and L)

**Returns** Serialized IPv6 ICMP RA packet.

**Return type** `ryu.lib.packet.ethernet`

`faucet.valve_packet.tlv_cast` (*tlvs, tlv\_attr, cast\_func*)  
Return cast'd attribute of first TLV or None.

`faucet.valve_packet.tlvs_by_subtype` (*tlvs, subtype*)  
Return list of TLVs with matching type.

`faucet.valve_packet.tlvs_by_type` (*tlvs, tlv\_type*)  
Return list of TLVs with matching type.

## faucet.valve\_pipeline module

Manages movement of packets through the faucet pipeline.

**class** `faucet.valve_pipeline.ValvePipeline` (*dp*)  
Bases: `faucet.valve_manager_base.ValveManagerBase`

Responsible for maintaing the integrity of the Faucet pipeline for a single valve.

Controls what packets a module sees in its tables and how it can pass packets through the pipeline.

Responsible for installing flows in the vlan, egress and classification tables

**accept\_to\_classification** (*actions=None*)  
Get instructions to forward packet through the pipeline to classification table. :param actions: (optional) list of actions to apply to packet.

**Returns** list of instructions

**accept\_to\_l2\_forwarding** (*actions=None*)  
Get instructions to forward packet through the pipeline to l2 forwarding. :param actions: (optional) list of actions to apply to packet.

**Returns** list of instructions

**accept\_to\_vlan** (*actions=None*)  
Get instructions to forward packet through the pipeline to vlan table. :param actions: (optional) list of actions to apply to packet.

**Returns** list of instructions

**add\_port** (*port*)  
install flows in response to a new port

**del\_port** (*port*)  
delete flows in response to a port removal

**filter\_packets** (*match\_dict, priority\_offset=0*)  
get a list of flow modification messages to filter packets from the pipeline. :param match\_dict: a dictionary specifying the match fields :param priority\_offset: used to prevent overlapping entries

**initialise\_tables** ()  
Install rules to initialise the classification\_table

**output** (*port, vlan, hairpin=False, loop\_protect\_field=None*)

Get instructions list to output a packet through the regular pipeline.

### Parameters

- **port** – Port object of port to output packet to
- **vlan** – Vlan object of vlan to output packet on
- **hairpin** – if True, hairpinning is required

**Returns** list of Instructions

**remove\_filter** (*match\_dict, strict=True, priority\_offset=0*)

retrieve flow mods to remove a filter from the classification table

**select\_packets** (*target\_table, match\_dict, actions=None, priority\_offset=0*)

retrieve rules to redirect packets matching match\_dict to table

## faucet.valve\_route module

Valve IPv4/IPv6 routing implementation.

**class** faucet.valve\_route.**AnonVLAN** (*vid*)

Bases: object

**class** faucet.valve\_route.**NextHop** (*eth\_src, port, now*)

Bases: object

Describes a directly connected (at layer 2) nexthop.

**age** (*now*)

Return age of this nexthop.

**cache\_time**

**dead** (*max\_fib\_retries*)

Return True if this nexthop is considered dead.

**eth\_src**

**last\_retry\_time**

**next\_retry** (*now, max\_resolve\_backoff\_time*)

Increment state for next retry.

**next\_retry\_time**

**port**

**resolution\_due** (*now, max\_age*)

Return True if this nexthop is due to be re resolved/retried.

**resolve\_retries**

**class** faucet.valve\_route.**ValveIPv4RouteManager** (*logger, global\_vlan, neighbor\_timeout, max\_hosts\_per\_resolve\_cycle, max\_host\_fib\_retry\_count, max\_resolve\_backoff\_time, proactive\_learn, dec\_ttl, multi\_out, fib\_table, vip\_table, pipeline, routers*)

Bases: *faucet.valve\_route.ValveRouteManager*

Implement IPv4 RIB/FIB.

```

CONTROL_ETH_TYPES = (2048, 2054)
ETH_TYPE = 2048
ICMP_SIZE = 110
ICMP_TYPE = 1
IPV = 4
IP_PKT
    alias of ryu.lib.packet.ipv4.ipv4
advertise (_vlan)
control_plane_handler (now, pkt_meta)
class faucet.valve_route.ValveIPv6RouteManager (logger, global_vlan, neighbor_timeout,
                                                max_hosts_per_resolve_cycle,
                                                max_host_fib_retry_count,
                                                max_resolve_backoff_time, proactive_learn,
                                                dec_ttl, multi_out, fib_table,
                                                vip_table, pipeline, routers)

Bases: faucet.valve_route.ValveRouteManager
Implement IPv6 FIB.
CONTROL_ETH_TYPES = (34525,)
ETH_TYPE = 34525
ICMP_TYPE = 58
IPV = 6
IP_PKT
    alias of ryu.lib.packet.ipv6.ipv6
advertise (vlan)
control_plane_handler (now, pkt_meta)
class faucet.valve_route.ValveRouteManager (logger, global_vlan, neighbor_timeout,
                                                max_hosts_per_resolve_cycle,
                                                max_host_fib_retry_count,
                                                max_resolve_backoff_time, proactive_learn,
                                                dec_ttl, multi_out, fib_table, vip_table,
                                                pipeline, routers)

Bases: faucet.valve_manager_base.ValveManagerBase
Base class to implement RIB/FIB.
CONTROL_ETH_TYPES = ()
ETH_TYPE = None
ICMP_SIZE = 128
ICMP_TYPE = None
IPV = 0
IP_PKT = None
active

```

**add\_host\_fib\_route\_from\_pkt** (*now, pkt\_meta*)

Add a host FIB route given packet from host.

**Parameters**

- **now** (*float*) – seconds since epoch.
- **pkt\_meta** (*PacketMeta*) – received packet.

**Returns** OpenFlow messages.

**Return type** list

**add\_route** (*vlan, ip\_gw, ip\_dst*)

Add a route to the RIB.

**Parameters**

- **vlan** (*vlan*) – VLAN containing this RIB.
- **ip\_gw** (*ipaddress.ip\_address*) – IP address of nexthop.
- **ip\_dst** (*ipaddress.ip\_network*) – destination IP network.

**Returns** OpenFlow messages.

**Return type** list

**add\_vlan** (*vlan*)

install flows in response to a new vlan

**advertise** (*vlan*)

**control\_plane\_handler** (*now, pkt\_meta*)

**dec\_ttl**

**del\_route** (*vlan, ip\_dst*)

Delete a route from the RIB.

Only one route with this exact destination is supported.

**Parameters**

- **vlan** (*vlan*) – VLAN containing this RIB.
- **ip\_dst** (*ipaddress.ip\_network*) – destination IP network.

**Returns** OpenFlow messages.

**Return type** list

**expire\_port\_nexthops** (*port*)

Expire all hosts on a port.

**fib\_table**

**global\_routing**

**global\_vlan**

**logger**

**max\_host\_fib\_retry\_count**

**max\_hosts\_per\_resolve\_cycle**

**max\_resolve\_backoff\_time**

**multi\_out**



**neighbor\_timeout**  
**nexthop\_dead** (*nexthop\_cache\_entry*)  
**pipeline**  
**proactive\_learn**  
**resolve\_expire\_hosts** (*vlan, now, resolve\_all=True*)  
 Re/resolve hosts.

**Parameters**

- **vlan** (*vlan*) – VLAN containing this RIB/FIB.
- **now** (*float*) – seconds since epoch.
- **resolve\_all** (*bool*) – attempt to resolve all unresolved gateways.

**Returns** OpenFlow messages.**Return type** list

**resolve\_gateways** (*vlan, now, resolve\_all=True*)  
 Re/resolve gateways.

**Parameters**

- **vlan** (*vlan*) – VLAN containing this RIB/FIB.
- **now** (*float*) – seconds since epoch.
- **resolve\_all** (*bool*) – attempt to resolve all unresolved gateways.

**Returns** OpenFlow messages.**Return type** list**route\_priority****routers****vip\_table****faucet.valve\_ryuapp module**

RyuApp base class for FAUCET/Gauge.

**class** faucet.valve\_ryuapp.**EventReconfigure**

Bases: ryu.controller.event.EventBase

Event sent to controller to cause config reload.

**class** faucet.valve\_ryuapp.**RyuAppBase** (\*args, \*\*kwargs)

Bases: ryu.base.app\_manager.RyuApp

RyuApp base class for FAUCET/Gauge.

**OFF\_VERSIONS** = [4]**connect\_or\_disconnect\_handler** (*ryu\_event*)

Handle connection or disconnection of a datapath.

**Parameters** **ryu\_event** (*ryu.controller.dpset.EventDP*) – trigger.**exc\_logname** = ''

**get\_setting** (*setting*, *path\_eval=False*)

Return config setting prefaced with logname.

**logname** = ''

**reconnect\_handler** (*ryu\_event*)

Handle reconnection of a datapath.

**Parameters** **ryu\_event** (*ryu.controller.dpset.EventDPReconnected*) – trigger.

**reload\_config** (*\_ryu\_event*)

Handle reloading configuration.

**signal\_handler** (*sigid*, *\_*)

Handle signals.

**Parameters** **sigid** (*int*) – signal received.

**start** ()

Start controller.

**exception** `faucet.valve_ryuapp.ValveDeadThreadException`

Bases: `Exception`

Exception raised when a dead thread is detected.

### faucet.valve\_table module

Abstraction of an OF table.

**class** `faucet.valve_table.ValveGroupEntry` (*table*, *group\_id*, *buckets*)

Bases: `object`

Abstraction for a single OpenFlow group entry.

**add** ()

Return flows to add this entry to the group table.

**delete** ()

Return flow to delete an existing group entry.

**modify** ()

Return flow to modify an existing group entry.

**update\_buckets** (*buckets*)

Update entry with new buckets.

**class** `faucet.valve_table.ValveGroupTable`

Bases: `object`

Wrap access to group table.

**delete\_all** ()

Delete all groups.

**entries** = `None`

**get\_entry** (*group\_id*, *buckets*)

Update entry with *group\_id* with *buckets*, and return the entry.

**static** **group\_id\_from\_str** (*key\_str*)

Return a group ID based on a string key.

**class** `faucet.valve_table.ValveTable` (*name*, *table\_config*, *flow\_cookie*, *notify\_flow\_removed=False*, *next\_tables=None*)

Bases: `object`

Wrapper for an OpenFlow table.

**flowcontroller** (*match=None*, *priority=None*, *inst=None*, *max\_len=96*)  
Add flow outputting to controller.

**flowdel** (*match=None*, *priority=None*, *out\_port=4294967295*, *strict=False*)  
Delete matching flows from a table.

**flowdrop** (*match=None*, *priority=None*, *hard\_timeout=0*)  
Add drop matching flow to a table.

**flowmod** (*match=None*, *priority=None*, *inst=None*, *command=0*, *out\_port=0*, *out\_group=0*, *hard\_timeout=0*, *idle\_timeout=0*, *cookie=None*)  
Helper function to construct a flow mod message with cookie.

**goto** (*next\_table*)  
Add goto next table instruction.

**goto\_miss** (*next\_table*)  
Add miss goto table instruction.

**goto\_this** ()

**static match** (*in\_port=None*, *vlan=None*, *eth\_type=None*, *eth\_src=None*, *eth\_dst=None*, *eth\_dst\_mask=None*, *icmpv6\_type=None*, *nw\_proto=None*, *nw\_dst=None*, *metadata=None*, *metadata\_mask=None*, *vlan\_pcp=None*)  
Compose an OpenFlow match rule.

**set\_field** (\*\**kws*)  
Return set field action.

**set\_vlan\_vid** (*vlan\_vid*)  
Set VLAN VID with VID\_PRESENT flag set.

**Parameters** *vid* (*int*) – VLAN VID

**Returns** set VID with VID\_PRESENT.

**Return type** `ryu.ofproto.ofproto_v1_3_parser.OFPActionSetField`

## faucet.valve\_util module

Utility functions for FAUCET.

`faucet.valve_util.close_logger` (*logger*)  
Close all handlers on logger object.

`faucet.valve_util.dpid_log` (*dpid*)  
Log a DP ID as hex/decimal.

`faucet.valve_util.get_logger` (*logname*, *logfile*, *loglevel*, *propagate*)  
Create and return a logger object.

`faucet.valve_util.get_setting` (*name*, *path\_eval=False*)  
Returns value of specified configuration setting.

`faucet.valve_util.get_sys_prefix` ()  
Returns an additional prefix for log and configuration files when used in a virtual environment

`faucet.valve_util.kill_on_exception` (*logname*)  
decorator to ensure functions will kill ryu when an unhandled exception occurs

`faucet.valve_util.stat_config_files` (*config\_hashes*)  
Return dict of a subset of stat attributes on config files.

`faucet.valve_util.utf8_decode` (*msg\_str*)  
Gracefully decode a possibly UTF-8 string.

### faucet.valves\_manager module

Manage a collection of Valves.

**class** `faucet.valves_manager.ConfigWatcher`

Bases: `object`

Watch config for file or content changes.

**config\_file** = `None`

**config\_file\_stats** = `None`

**config\_hashes** = `None`

**content\_changed** (*new\_config\_file*)  
Return True if config file content actually changed.

**files\_changed** ()  
Return True if any config files changed.

**update** (*new\_config\_file*, *new\_config\_hashes=None*)  
Update state with new config file/hashes.

**class** `faucet.valves_manager.ValvesManager` (*logname*, *logger*, *metrics*, *notifier*, *bgp*, *dot1x*,  
*send\_flows\_to\_dp\_by\_id*)

Bases: `object`

Manage a collection of Valves.

**load\_configs** (*now*, *new\_config\_file*, *delete\_dp=None*)  
Load/apply new config to all Valves.

**new\_valve** (*new\_dp*)

**parse\_configs** (*new\_config\_file*)  
Return parsed configs for Valves, or None.

**port\_status\_handler** (*valve*, *msg*)  
Handle a port status change message.

**request\_reload\_configs** (*now*, *new\_config\_file*, *delete\_dp=None*)  
Process a request to load config changes.

**update\_metrics** (*now*)  
Update metrics in all Valves.

**valve\_flow\_services** (*now*, *valve\_service*)  
Call a method on all Valves and send any resulting flows.

**valve\_packet\_in** (*now*, *valve*, *msg*)  
Time a call to Valve packet in handler.

**valves** = `None`

**faucet.vlan module**

VLAN configuration.

**class** faucet.vlan.**AnyVLAN**

Bases: object

Placeholder any tagged VLAN.

**name** = 'Any VLAN'

**vid** = 4096

**class** faucet.vlan.**HostCacheEntry** (*eth\_src, port, cache\_time*)

Bases: object

Association of a host with a port.

**cache\_time**

**eth\_src**

**eth\_src\_int**

**port**

**class** faucet.vlan.**NullVLAN**

Bases: object

Placeholder null VLAN.

**name** = 'Null VLAN'

**vid** = 0

**class** faucet.vlan.**VLAN** (*\_id, dp\_id, conf=None*)

Bases: *faucet.conf.Conf*

Contains state for one VLAN, including its configuration.

**add\_cache\_host** (*eth\_src, port, cache\_time*)

Add/update a host to the cache on a port at at time.

**add\_route** (*ip\_dst, ip\_gw*)

Add an IP route.

**all\_ip\_gws** (*ipv*)

Return all IP gateways for specified IP version.

**cached\_host** (*eth\_src*)

Return host from cache or None.

**cached\_host\_on\_port** (*eth\_src, port*)

Return host cache entry if host in cache and on specified port.

**cached\_hosts\_count\_on\_port** (*port*)

Return count of all hosts learned on a port.

**cached\_hosts\_on\_port** (*port*)

Return all hosts learned on a port.

**check\_config** ()

Check config at instantiation time for errors, typically via assert.

**clear\_cache\_hosts\_on\_port** (*port*)

Clear all hosts learned on a port.

**defaults** = {'acl\_in': None, 'acls\_in': None, 'description': None, 'faucet\_mac': '0

**defaults\_types** = {'acl\_in': (<class 'int'>, <class 'str'>), 'acls\_in': <class 'list'

**del\_route** (*ip\_dst*)

Delete an IP route.

**exclude\_same\_lag\_member\_ports** (*in\_port=None*)

Ensure output on only one member of a LAG.

**expire\_cache\_host** (*eth\_src*)

Expire a host from caches.

**expire\_cache\_hosts** (*now, learn\_timeout*)

Expire stale host entries.

**faucet\_vips\_by\_ipv** (*ipv*)

Return VIPs with specified IP version on this VLAN.

**flood\_pkt** (*packet\_builder, multi\_out=True, \*args*)

**static\_flood\_ports** (*configured\_ports, exclude\_unicast*)

**from\_connected\_to\_vip** (*src\_ip, dst\_ip*)

Return True if src\_ip in connected network and dst\_ip is a VIP.

### Parameters

- **src\_ip** (*ipaddress.ip\_address*) – source IP.
- **dst\_ip** (*ipaddress.ip\_address*) – destination IP

**Returns** True if local traffic for a VIP.

**get\_ports** ()

Return all ports on this VLAN.

**hairpin\_ports** ()

Return all ports with hairpin enabled.

**hosts\_count** ()

Return number of hosts learned on this VLAN.

**ip\_dsts\_for\_ip\_gw** (*ip\_gw*)

Return list of IP destinations, for specified gateway.

**ip\_in\_vip\_subnet** (*ipa, faucet\_vip=None*)

Return faucet\_vip if IP in same IP network as a VIP on this VLAN.

**ipvs** ()

Return IP versions configured on this VLAN.

**is\_faucet\_vip** (*ipa, faucet\_vip=None*)

Return True if IP is a VIP on this VLAN.

**is\_host\_fib\_route** (*host\_ip*)

Return True if IP destination is a host FIB route.

**Parameters** **host\_ip** – (*ipaddress.ip\_address*): potential host FIB route.

**Returns** True if a host FIB route (and not used as a gateway).

**lACP\_ports** ()

Return ports that have LACP on this VLAN.

**lACP\_up\_ports** ()  
Return ports that have LACP up on this VLAN.

**lags** ()  
Return dict of LAGs mapped to member ports.

**lags\_up** ()  
Return dict of LAGs mapped to member ports that have LACP up.

**link\_and\_other\_vips** (*ipv*)  
Return link local and non-link local VIPs.

**loop\_protect\_external\_ports** ()  
Return ports with external loop protection set.

**loop\_protect\_external\_ports\_up** ()  
Return up ports with external loop protection set.

**mirrored\_ports** ()  
Return ports that are mirrored on this VLAN.

**mutable\_attrs** = frozenset({'tagged', 'untagged'})

**neigh\_cache\_by\_ipv** (*ipv*)  
Return neighbor cache for specified IP version on this VLAN.

**neigh\_cache\_count\_by\_ipv** (*ipv*)  
Return number of hosts in neighbor cache for specified IP version on this VLAN.

**output\_port** (*port*, *hairpin=False*, *output\_table=None*, *loop\_protect\_field=None*)

**pkt\_out\_port** (*packet\_builder*, *port*, *\*args*)

**port\_is\_tagged** (*port*)  
Return True if port number is an tagged port on this VLAN.

**port\_is\_untagged** (*port*)  
Return True if port number is an untagged port on this VLAN.

**reset\_caches** ()  
Reset dynamic caches.

**reset\_ports** (*ports*)  
Reset tagged and untagged port lists.

**route\_count\_by\_ipv** (*ipv*)  
Return route table count for specified IP version on this VLAN.

**routes\_by\_ipv** (*ipv*)  
Return route table for specified IP version on this VLAN.

**set\_defaults** ()  
Set default values and run any basic sanity checks.

**tagged\_flood\_ports** (*exclude\_unicast*)

**untagged\_flood\_ports** (*exclude\_unicast*)

**static\_vid\_valid** (*vid*)  
Return True if VID valid.

**vip\_map** (*ipa*)

### faucet.watcher module

Gauge watcher implementations.

**class** faucet.watcher.**GaugeFlowTableLogger** (*conf, logname, prom\_client*)

Bases: *faucet.gauge\_pollers.GaugeFlowTablePoller*

Periodically dumps the current datapath flow table as a yaml object.

Includes a timestamp and a reference (\$DATAPATHNAME-flowtables). The flow table is dumped as an OF-FlowStatsReply message (in yaml format) that matches all flows.

optionally the output can be compressed by setting compressed: true in the config for this watcher

**update** (*rcv\_time, dp\_id, msg*)

Handle the responses to requests.

Called when a reply to a stats request sent by this object is received by the controller.

It should acknowledge the receipt by setting self.reply\_pending to false.

#### Parameters

- **rcv\_time** – the time the response was received
- **dp\_id** – DP ID
- **msg** – the stats reply message

**class** faucet.watcher.**GaugePortStateLogger** (*conf, logname, prom\_client*)

Bases: *faucet.gauge\_pollers.GaugePortStatePoller*

Abstraction for port state logger.

**static no\_response** ()

Called when a polling cycle passes without receiving a response.

**static send\_req** ()

Send a stats request to a datapath.

**update** (*rcv\_time, dp\_id, msg*)

Handle the responses to requests.

Called when a reply to a stats request sent by this object is received by the controller.

It should acknowledge the receipt by setting self.reply\_pending to false.

#### Parameters

- **rcv\_time** – the time the response was received
- **dp\_id** – DP ID
- **msg** – the stats reply message

**class** faucet.watcher.**GaugePortStatsLogger** (*conf, logname, prom\_client*)

Bases: *faucet.gauge\_pollers.GaugePortStatsPoller*

Abstraction for port statistics logger.

**update** (*rcv\_time, dp\_id, msg*)

Handle the responses to requests.

Called when a reply to a stats request sent by this object is received by the controller.

It should acknowledge the receipt by setting self.reply\_pending to false.

#### Parameters



- **rcv\_time** – the time the response was received
- **dp\_id** – DP ID
- **msg** – the stats reply message

`faucet.watcher.watcher_factory` (*conf*)

Return a Gauge object based on type.

**Parameters** *conf* (*GaugeConf*) – object with the configuration for this valve.

## faucet.watcher\_conf module

Gauge watcher configuration.

**class** `faucet.watcher_conf.WatcherConf` (*\_id*, *dp\_id*, *conf*, *prom\_client*)

Bases: `faucet.conf.Conf`

Stores the state and configuration to monitor a single stat.

Watcher Config

Watchers are configured in the watchers config block in the config for gauge.

The following elements can be configured for each watcher, at the level of `/watchers/<watcher name>/`:

- **type** (string): The type of watcher (IE what stat this watcher monitors). The types are ‘port\_state’, ‘port\_stats’ or ‘flow\_table’.
- **dps** (list): A list of dps that should be monitored with this watcher.
- **db** (string): The db that will be used to store the data once it is retrieved.
- **interval** (int): if this watcher requires polling the switch, it will monitor at this interval.

The config for a db should be created in the gauge config file under the dbs config block.

The following elements can be configured for each db, at the level of `/dbs/<db name>/`:

- **type** (string): the type of db. The available types are ‘text’ and ‘influx’ for port\_state, ‘text’, ‘influx’ and ‘prometheus’ for port\_stats and ‘text’ and flow\_table.

The following config elements then depend on the type. For text:

- **file** (string): the filename of the file to write output to.
- **compress** (bool): compress (with gzip) flow\_table output while writing it

### For influx:

- **influx\_db** (str): The name of the influxdb database. Defaults to ‘faucet’.
- **influx\_host** (str): The host where the influxdb is reachable. Defaults to ‘localhost’.
- **influx\_port** (int): The port that the influxdb host will listen on. Defaults to 8086.
- **influx\_user** (str): The username for accessing influxdb. Defaults to ‘’.
- **influx\_pwd** (str): The password for accessing influxdb. Defaults to ‘’.
- **influx\_timeout** (int): The timeout in seconds for connecting to influxdb. Defaults to 10.
- **influx\_retries** (int): The number of times to retry connecting to influxdb after failure. Defaults to 3.

### For Prometheus:

- **prometheus\_port** (int): The port used to export prometheus data. Defaults to 9303.

- `prometheus_addr` (ip addr str): The address used to export prometheus data. Defaults to '127.0.0.1'.

**add\_db** (*db\_conf*)

Add database config to this watcher.

**add\_dp** (*dp*)

Add a datapath to this watcher.

**check\_config** ()

Check config at instantiation time for errors, typically via assert.

```
db_defaults = {'compress': False, 'file': None, 'influx_db': 'faucet', 'influx_host'
```

```
db_defaults_types = {'compress': <class 'bool'>, 'file': <class 'str'>, 'influx_db':
```

```
defaults = {'all_dps': False, 'db': None, 'db_type': 'text', 'dbs': None, 'dps':
```

```
defaults_types = {'all_dps': <class 'bool'>, 'db': <class 'str'>, 'db_type': <class
```

### Module contents

### 3.1 Frequently Asked Questions

#### 3.1.1 How are packet-ins handled when a message is generated through table-miss flow entry?

Faucet adds explicit rules for unmatched packets.

#### 3.1.2 Are group actions supported in Faucet?

Yes, just not by default currently. Set the `group_table` option to `True` on a datapath to enable group output actions.

#### 3.1.3 Does Faucet send any multi-part requests? If so, please provide sample use cases

Gauge uses multi-part messages for the stats collection (flow table stats and port stats).

#### 3.1.4 Does Faucet install table-miss entry?

Yes.

#### 3.1.5 Does Faucet clear all all switch table entries on connection?

Faucet gives all entries a specific cookie, and it clears all entries with that cookie. I.e., it clears entries added by itself but not anyone else.

### **3.1.6 Does Faucet install fresh set of table entries on connection and re-connection?**

Yes.

### **3.1.7 Does Faucet installed flows support priority? How is this defined - who get higher priority than the other and why?**

Yes, priority is necessary for a number of things. Example: there are higher priority rules for packets with a known source address, and lower ones to send those packets to the controller.

### **3.1.8 Is there a gui for generating a YAML file?**

No.

### **3.1.9 Should Faucet detect Management, OF controller ports and gateway ports on the switch or pure OF only ports where hosts are connected?**

Out of scope for Faucet as it is currently.

### **3.1.10 If another controller is connected to the switch in addition to Faucet, what happens to Faucet?**

Faucet identifies its own flows using a cookie value, if the other controller doesn't use the same cookie value there shouldn't be a problem (provided the rules don't conflict in a problematic way)

### **3.1.11 If another controller connected to switch changes role (master, slave, equal) on the switch, what happens to Faucet?**

Shouldn't be an issue, if another controller is the master then my understanding is Faucet wouldnt be able to install any flows however?

### **3.1.12 Does Faucet send LLDP packets?**

No.

### **3.1.13 Some switches always send VLAN info in packet\_in messages and some don't. How does Faucet handle this?**

Packets should have VLANs pushed before being sent to the controller.

### **3.1.14 Is there a event handler registered to detect if flows on the switch change?**

No.

### **3.1.15 Does Faucet use auxiliary connections?**

No.

### **3.1.16 Does Faucet support L2.5 (MPLS, etc.)?**

No.

### **3.1.17 Stats - what does Faucet collect (flow count, etc)?**

Gauge collects port stats and takes a full flow-table dump periodically.

### **3.1.18 How do I use Gauge?**

Give Gauge a list of Faucet yaml config files and it will poll them for stats (as specified in the config file).



## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`





### f

- faucet, 174
- faucet.acl, 119
- faucet.check\_faucet\_config, 121
- faucet.conf, 121
- faucet.config\_parser, 122
- faucet.config\_parser\_util, 122
- faucet.dp, 123
- faucet.faucet, 125
- faucet.faucet\_bgp, 127
- faucet.faucet\_dot1x, 128
- faucet.faucet\_experimental\_api, 129
- faucet.faucet\_experimental\_event, 129
- faucet.faucet\_metadata, 130
- faucet.faucet\_metrics, 130
- faucet.faucet\_pipeline, 130
- faucet.fctl, 131
- faucet.gauge, 131
- faucet.gauge\_influx, 132
- faucet.gauge\_pollers, 135
- faucet.gauge\_prom, 136
- faucet.meter, 138
- faucet.port, 138
- faucet.prom\_client, 139
- faucet.router, 139
- faucet.tfm\_pipeline, 140
- faucet.valve, 140
- faucet.valve\_acl, 146
- faucet.valve\_flood, 147
- faucet.valve\_host, 148
- faucet.valve\_manager\_base, 149
- faucet.valve\_of, 149
- faucet.valve\_of\_old, 155
- faucet.valve\_packet, 155
- faucet.valve\_pipeline, 161
- faucet.valve\_route, 162
- faucet.valve\_ryuapp, 165
- faucet.valve\_table, 166
- faucet.valve\_util, 167
- faucet.valves\_manager, 168
- faucet.vlan, 169
- faucet.watcher, 172
- faucet.watcher\_conf, 173



## A

- accept\_to\_classification() (*faucet.valve\_pipeline.ValvePipeline* method), 161  
 accept\_to\_l2\_forwarding() (*faucet.valve\_pipeline.ValvePipeline* method), 161  
 accept\_to\_vlan() (*faucet.valve\_pipeline.ValvePipeline* method), 161  
 ACL (class in *faucet.acl*), 119  
 acl\_manager (*faucet.valve.Valve* attribute), 141  
 acquire\_nonblock() (*faucet.faucet\_experimental\_event.NonBlockLock* method), 130  
 actions\_types (*faucet.acl.ACL* attribute), 120  
 active (*faucet.valve\_route.ValveRouteManager* attribute), 163  
 add() (*faucet.valve\_table.ValveGroupEntry* method), 166  
 add\_acl() (*faucet.dp.DP* method), 123  
 add\_authed\_mac() (*faucet.valve.Valve* method), 141  
 add\_authed\_mac() (*faucet.valve\_acl.ValveAclManager* method), 146  
 add\_cache\_host() (*faucet.vlan.VLAN* method), 169  
 add\_db() (*faucet.watcher\_conf.WatcherConf* method), 174  
 add\_dp() (*faucet.watcher\_conf.WatcherConf* method), 174  
 add\_host\_fib\_route\_from\_pkt() (*faucet.valve\_route.ValveRouteManager* method), 163  
 add\_port() (*faucet.dp.DP* method), 123  
 add\_port() (*faucet.valve\_acl.ValveAclManager* method), 146  
 add\_port() (*faucet.valve\_host.ValveHostManager* method), 148  
 add\_port() (*faucet.valve\_manager\_base.ValveManagerBase* method), 149  
 add\_port() (*faucet.valve\_pipeline.ValvePipeline* method), 161  
 add\_port\_acl() (*faucet.faucet\_experimental\_api.FaucetExperimentalAPI* method), 129  
 add\_route() (*faucet.valve.Valve* method), 141  
 add\_route() (*faucet.valve\_route.ValveRouteManager* method), 164  
 add\_route() (*faucet.vlan.VLAN* method), 169  
 add\_router() (*faucet.dp.DP* method), 123  
 add\_stack\_link() (*faucet.dp.DP* class method), 123  
 add\_vlan() (*faucet.valve\_acl.ValveAclManager* method), 146  
 add\_vlan() (*faucet.valve\_flood.ValveFloodManager* method), 147  
 add\_vlan() (*faucet.valve\_manager\_base.ValveManagerBase* method), 149  
 add\_vlan() (*faucet.valve\_route.ValveRouteManager* method), 164  
 add\_vlan\_acl() (*faucet.faucet\_experimental\_api.FaucetExperimentalAPI* method), 129  
 advertise() (*faucet.valve.Valve* method), 141  
 advertise() (*faucet.valve\_route.ValveIPv4RouteManager* method), 163  
 advertise() (*faucet.valve\_route.ValveIPv6RouteManager* method), 163  
 advertise() (*faucet.valve\_route.ValveRouteManager* method), 164  
 age() (*faucet.valve\_route.NextHop* method), 162  
 all\_ip\_gws() (*faucet.vlan.VLAN* method), 169  
 AlliedTelesis (class in *faucet.valve*), 140  
 AnonVLAN (class in *faucet.valve\_route*), 162  
 AnyVLAN (class in *faucet.vlan*), 169  
 apply\_actions() (in module *faucet.valve\_of*), 149  
 apply\_meter() (in module *faucet.valve\_of*), 150  
 arp\_reply() (in module *faucet.valve\_packet*), 156  
 arp\_request() (in module *faucet.valve\_packet*), 156  
 ArubaValve (class in *faucet.valve*), 141  
 auth\_handler() (*faucet.faucet\_dot1x.FaucetDot1x* method), 128

## B

ban\_rules() (*faucet.valve\_host.ValveHostManager method*), 148  
 barrier() (*in module faucet.valve\_of*), 150  
 base\_prom\_labels() (*faucet.dp.DP method*), 123  
 bgp (*faucet.faucet.Faucet attribute*), 126  
 bgp\_as() (*faucet.router.Router method*), 139  
 bgp\_connect\_mode() (*faucet.router.Router method*), 139  
 bgp\_defaults\_types (*faucet.router.Router attribute*), 139  
 bgp\_ipvs() (*faucet.router.Router method*), 140  
 bgp\_neighbor\_addresses() (*faucet.router.Router method*), 140  
 bgp\_neighbor\_addresses\_by\_ipv() (*faucet.router.Router method*), 140  
 bgp\_neighbor\_as() (*faucet.router.Router method*), 140  
 bgp\_port() (*faucet.router.Router method*), 140  
 bgp\_routerid() (*faucet.router.Router method*), 140  
 bgp\_routers() (*faucet.dp.DP method*), 123  
 bgp\_server\_addresses() (*faucet.router.Router method*), 140  
 bgp\_server\_addresses\_by\_ipv() (*faucet.router.Router method*), 140  
 bgp\_vlan() (*faucet.router.Router method*), 140  
 BgpSpeakerKey (*class in faucet.faucet\_bgp*), 127  
 bucket() (*in module faucet.valve\_of*), 150  
 build() (*faucet.acl.ACL method*), 120  
 build\_acl\_entry() (*in module faucet.valve\_acl*), 146  
 build\_acl\_ofmsgs() (*in module faucet.valve\_acl*), 146  
 build\_flood\_rules() (*faucet.valve\_flood.ValveFloodManager method*), 147  
 build\_group\_flood\_buckets() (*in module faucet.valve\_of*), 150  
 build\_match\_dict() (*in module faucet.valve\_of*), 150  
 build\_output\_actions() (*in module faucet.valve\_acl*), 147  
 build\_pkt\_header() (*in module faucet.valve\_packet*), 156

## C

cache\_time (*faucet.valve\_route.NextHop attribute*), 162  
 cache\_time (*faucet.vlan.HostCacheEntry attribute*), 169  
 cached\_host() (*faucet.vlan.VLAN method*), 169  
 cached\_host\_on\_port() (*faucet.vlan.VLAN method*), 169

cached\_hosts\_count\_on\_port() (*faucet.vlan.VLAN method*), 169  
 cached\_hosts\_on\_port() (*faucet.vlan.VLAN method*), 169  
 check\_config() (*faucet.acl.ACL method*), 120  
 check\_config() (*faucet.conf.Conf method*), 121  
 check\_config() (*faucet.dp.DP method*), 123  
 check\_config() (*faucet.port.Port method*), 138  
 check\_config() (*faucet.router.Router method*), 140  
 check\_config() (*faucet.vlan.VLAN method*), 169  
 check\_config() (*faucet.watcher\_conf.WatcherConf method*), 174  
 check\_config() (*in module faucet.check\_faucet\_config*), 121  
 check\_path() (*faucet.faucet\_experimental\_event.FaucetExperimentalEvent method*), 130  
 CiscoC9KValve (*class in faucet.valve*), 141  
 classification\_table() (*faucet.dp.DP method*), 123  
 clear\_cache\_hosts\_on\_port() (*faucet.vlan.VLAN method*), 169  
 close\_logger() (*in module faucet.valve\_util*), 167  
 close\_logs() (*faucet.valve.Valve method*), 141  
 cold\_start\_port() (*faucet.valve\_acl.ValveAclManager method*), 146  
 Conf (*class in faucet.conf*), 121  
 conf (*faucet.gauge\_influx.InfluxShipper attribute*), 134  
 conf\_diff() (*faucet.conf.Conf method*), 121  
 conf\_hash() (*faucet.conf.Conf method*), 122  
 config\_changed() (*in module faucet.config\_parser\_util*), 123  
 config\_file (*faucet.valves\_manager.ConfigWatcher attribute*), 168  
 config\_file\_hash() (*in module faucet.config\_parser\_util*), 123  
 config\_file\_stats (*faucet.valves\_manager.ConfigWatcher attribute*), 168  
 config\_hashes (*faucet.valves\_manager.ConfigWatcher attribute*), 168  
 ConfigWatcher (*class in faucet.valves\_manager*), 168  
 connect\_or\_disconnect\_handler() (*faucet.valve\_ryuapp.RyuAppBase method*), 165  
 construct\_mapping() (*faucet.config\_parser\_util.UniqueKeyLoader method*), 122  
 content\_changed() (*faucet.valves\_manager.ConfigWatcher method*), 168  
 CONTROL\_ETH\_TYPES (*faucet.valve\_route.ValveIPv4RouteManager*

- attribute), 162
- CONTROL\_ETH\_TYPES (faucet.valve\_route.ValveIPv6RouteManager attribute), 163
- CONTROL\_ETH\_TYPES (faucet.valve\_route.ValveRouteManager attribute), 163
- control\_plane\_handler() (faucet.valve\_route.ValveIPv4RouteManager method), 163
- control\_plane\_handler() (faucet.valve\_route.ValveIPv6RouteManager method), 163
- control\_plane\_handler() (faucet.valve\_route.ValveRouteManager method), 164
- controller\_pps\_meteradd() (in module faucet.valve\_of), 150
- controller\_pps\_meterdel() (in module faucet.valve\_of), 150
- create\_acl\_tunnel() (faucet.valve\_acl.ValveAclManager method), 146
- create\_dot1x\_flow\_pair() (faucet.valve\_acl.ValveAclManager method), 146
- create\_flow\_pair() (faucet.faucet\_dot1x.FaucetDot1x method), 128
- ## D
- data (faucet.valve\_packet.PacketMeta attribute), 155
- datapath\_connect() (faucet.valve.Valve method), 142
- datapath\_disconnect() (faucet.valve.Valve method), 142
- db\_defaults (faucet.watcher\_conf.WatcherConf attribute), 174
- db\_defaults\_types (faucet.watcher\_conf.WatcherConf attribute), 174
- dead() (faucet.valve\_route.NextHop method), 162
- debug() (faucet.valve.ValveLogger method), 146
- dec\_ip\_ttl() (in module faucet.valve\_of), 150
- DEC\_TTL (faucet.valve.AlliedTelesis attribute), 141
- DEC\_TTL (faucet.valve.ArubaValve attribute), 141
- DEC\_TTL (faucet.valve.Valve attribute), 141
- dec\_ttl (faucet.valve\_route.ValveRouteManager attribute), 164
- decode\_value() (in module faucet.fctl), 131
- dedupe\_ofmsgs() (in module faucet.valve\_of), 150
- dedupe\_output\_port\_acts() (in module faucet.valve\_of), 150
- default\_table\_sizes\_types (faucet.dp.DP attribute), 124
- defaults (faucet.acl.ACL attribute), 120
- defaults (faucet.conf.Conf attribute), 122
- defaults (faucet.dp.DP attribute), 124
- defaults (faucet.meter.Meter attribute), 138
- defaults (faucet.port.Port attribute), 138
- defaults (faucet.router.Router attribute), 140
- defaults (faucet.vlan.VLAN attribute), 169
- defaults (faucet.watcher\_conf.WatcherConf attribute), 174
- defaults\_types (faucet.acl.ACL attribute), 120
- defaults\_types (faucet.conf.Conf attribute), 122
- defaults\_types (faucet.dp.DP attribute), 124
- defaults\_types (faucet.meter.Meter attribute), 138
- defaults\_types (faucet.port.Port attribute), 138
- defaults\_types (faucet.router.Router attribute), 140
- defaults\_types (faucet.vlan.VLAN attribute), 170
- defaults\_types (faucet.watcher\_conf.WatcherConf attribute), 174
- del\_authed\_mac() (faucet.valve.Valve method), 142
- del\_authed\_mac() (faucet.valve\_acl.ValveAclManager method), 146
- del\_dot1x\_flow\_pair() (faucet.valve\_acl.ValveAclManager method), 146
- del\_port() (faucet.valve\_host.ValveHostManager method), 148
- del\_port() (faucet.valve\_manager\_base.ValveManagerBase method), 149
- del\_port() (faucet.valve\_pipeline.ValvePipeline method), 161
- del\_route() (faucet.valve.Valve method), 142
- del\_route() (faucet.valve\_route.ValveRouteManager method), 164
- del\_route() (faucet.vlan.VLAN method), 170
- del\_vlan() (faucet.valve\_manager\_base.ValveManagerBase method), 149
- delete() (faucet.valve\_table.ValveGroupEntry method), 166
- delete\_all() (faucet.valve\_table.ValveGroupTable method), 166
- delete\_host\_from\_vlan() (faucet.valve\_host.ValveHostManager method), 148
- delete\_port\_acl() (faucet.faucet\_experimental\_api.FaucetExperimentalAPI method), 129
- delete\_vlan\_acl() (faucet.faucet\_experimental\_api.FaucetExperimentalAPI method), 129
- desc\_stats\_reply\_handler() (faucet.faucet.Faucet method), 126
- desc\_stats\_request() (in module

- faucet.valve\_of*), 150
- `devid_present()` (in module *faucet.valve\_of*), 150
- `dot1x` (*faucet.valve.Valve* attribute), 142
- `dot1x_defaults_types` (*faucet.dp.DP* attribute), 124
- `dot1x_ports()` (*faucet.dp.DP* method), 124
- `DP` (class in *faucet.dp*), 123
- `dp` (*faucet.valve.Valve* attribute), 142
- `dp_config_path()` (in module *faucet.config\_parser\_util*), 123
- `dp_include()` (in module *faucet.config\_parser\_util*), 123
- `dp_init()` (*faucet.valve.Valve* method), 142
- `dp_parser()` (in module *faucet.config\_parser*), 122
- `dpid_log()` (in module *faucet.valve\_util*), 167
- `dyn_finalized` (*faucet.conf.Conf* attribute), 122
- `dyn_hash` (*faucet.conf.Conf* attribute), 122
- ## E
- `echo_reply()` (in module *faucet.valve\_packet*), 157
- `edge_learn_port()` (*faucet.valve\_flood.ValveFloodManager* static method), 147
- `edge_learn_port()` (*faucet.valve\_flood.ValveFloodStackManager* method), 147
- `entries` (*faucet.valve\_table.ValveGroupTable* attribute), 166
- `entry` (*faucet.meter.Meter* attribute), 138
- `entry_msg` (*faucet.meter.Meter* attribute), 138
- `error()` (*faucet.valve.ValveLogger* method), 146
- `error_handler()` (*faucet.faucet.Faucet* method), 126
- `eth_dst` (*faucet.valve\_packet.PacketMeta* attribute), 155
- `eth_pkt` (*faucet.valve\_packet.PacketMeta* attribute), 155
- `eth_src` (*faucet.valve\_packet.PacketMeta* attribute), 155
- `eth_src` (*faucet.valve\_route.NextHop* attribute), 162
- `eth_src` (*faucet.vlan.HostCacheEntry* attribute), 169
- `eth_src_int` (*faucet.vlan.HostCacheEntry* attribute), 169
- `eth_type` (*faucet.valve\_packet.PacketMeta* attribute), 155
- `ETH_TYPE` (*faucet.valve\_route.ValveIPv4RouteManager* attribute), 163
- `ETH_TYPE` (*faucet.valve\_route.ValveIPv6RouteManager* attribute), 163
- `ETH_TYPE` (*faucet.valve\_route.ValveRouteManager* attribute), 163
- `ETH_TYPES_PARSERS` (*faucet.valve\_packet.PacketMeta* attribute), 155
- `EventFaucetAdvertise` (class in *faucet.faucet*), 125
- `EventFaucetExperimentalAPIRegistered` (class in *faucet.faucet*), 126
- `EventFaucetFastAdvertise` (class in *faucet.faucet*), 126
- `EventFaucetFastStateExpire` (class in *faucet.faucet*), 126
- `EventFaucetMetricUpdate` (class in *faucet.faucet*), 126
- `EventFaucetResolveGateways` (class in *faucet.faucet*), 126
- `EventFaucetStateExpire` (class in *faucet.faucet*), 126
- `EventReconfigure` (class in *faucet.valve\_ryuapp*), 165
- `exc_logname` (*faucet.faucet.Faucet* attribute), 126
- `exc_logname` (*faucet.faucet\_bgp.FaucetBgp* attribute), 127
- `exc_logname` (*faucet.gauge.Gauge* attribute), 132
- `exc_logname` (*faucet.valve\_ryuapp.RyuAppBase* attribute), 165
- `exclude_same_lag_member_ports()` (*faucet.vlan.VLAN* method), 170
- `expire_cache_host()` (*faucet.vlan.VLAN* method), 170
- `expire_cache_hosts()` (*faucet.vlan.VLAN* method), 170
- `expire_hosts_from_vlan()` (*faucet.valve\_host.ValveHostFlowRemovedManager* method), 148
- `expire_hosts_from_vlan()` (*faucet.valve\_host.ValveHostManager* method), 149
- `expire_port_nexthops()` (*faucet.valve\_route.ValveRouteManager* method), 164
- `EXT_PORT_FLAG` (*faucet.valve\_flood.ValveFloodStackManager* attribute), 147
- ## F
- `failure_handler()` (*faucet.faucet\_dot1x.FaucetDot1x* method), 128
- `fast_advertise()` (*faucet.valve.Valve* method), 142
- `fast_state_expire()` (*faucet.valve.Valve* method), 142
- `Faucet` (class in *faucet.faucet*), 126
- `faucet` (module), 174
- `faucet.acl` (module), 119
- `faucet.check_faucet_config` (module), 121
- `faucet.conf` (module), 121
- `faucet.config_parser` (module), 122
- `faucet.config_parser_util` (module), 122



- faucet.dp (module), 123  
 faucet.faucet (module), 125  
 faucet.faucet\_bgp (module), 127  
 faucet.faucet\_dot1x (module), 128  
 faucet.faucet\_experimental\_api (module), 129  
 faucet.faucet\_experimental\_event (module), 129  
 faucet.faucet\_metadata (module), 130  
 faucet.faucet\_metrics (module), 130  
 faucet.faucet\_pipeline (module), 130  
 faucet.fctl (module), 131  
 faucet.gauge (module), 131  
 faucet.gauge\_influx (module), 132  
 faucet.gauge\_pollers (module), 135  
 faucet.gauge\_prom (module), 136  
 faucet.meter (module), 138  
 faucet.port (module), 138  
 faucet.prom\_client (module), 139  
 faucet.router (module), 139  
 faucet.tfm\_pipeline (module), 140  
 faucet.valve (module), 140  
 faucet.valve\_acl (module), 146  
 faucet.valve\_flood (module), 147  
 faucet.valve\_host (module), 148  
 faucet.valve\_manager\_base (module), 149  
 faucet.valve\_of (module), 149  
 faucet.valve\_of\_old (module), 155  
 faucet.valve\_packet (module), 155  
 faucet.valve\_pipeline (module), 161  
 faucet.valve\_route (module), 162  
 faucet.valve\_ryuapp (module), 165  
 faucet.valve\_table (module), 166  
 faucet.valve\_util (module), 167  
 faucet.valves\_manager (module), 168  
 faucet.vlan (module), 169  
 faucet.watcher (module), 172  
 faucet.watcher\_conf (module), 173  
 faucet\_async () (in module faucet.valve\_of), 150  
 faucet\_config () (in module faucet.valve\_of), 151  
 faucet\_lldp\_stack\_state\_tlvs () (in module faucet.valve\_packet), 157  
 faucet\_lldp\_tlvs () (in module faucet.valve\_packet), 157  
 faucet\_oui () (in module faucet.valve\_packet), 157  
 faucet\_tlvs () (in module faucet.valve\_packet), 157  
 faucet\_vips\_by\_ipv () (faucet.vlan.VLAN method), 170  
 FaucetBgp (class in faucet.faucet\_bgp), 127  
 FaucetDot1x (class in faucet.faucet\_dot1x), 128  
 FaucetExperimentalAPI (class in faucet.faucet\_experimental\_api), 129  
 FaucetExperimentalEventNotifier (class in faucet.faucet\_experimental\_event), 129  
 FaucetMetrics (class in faucet.faucet\_metrics), 130  
 features\_handler () (faucet.faucet.Faucet method), 126  
 fib\_table (faucet.valve\_route.ValveRouteManager attribute), 164  
 files\_changed () (faucet.valves\_manager.ConfigWatcher method), 168  
 filter\_packets () (faucet.valve\_pipeline.ValvePipeline method), 161  
 finalize () (faucet.conf.Conf method), 122  
 finalize () (faucet.port.Port method), 138  
 finalize () (faucet.router.Router method), 140  
 finalize\_config () (faucet.dp.DP method), 124  
 finalize\_tunnel\_acls () (faucet.dp.DP method), 124  
 FLOOD\_DSTS (faucet.valve\_flood.ValveFloodManager attribute), 147  
 flood\_manager (faucet.valve.Valve attribute), 142  
 flood\_pkt () (faucet.vlan.VLAN method), 170  
 flood\_port\_outputs () (in module faucet.valve\_of), 151  
 flood\_ports () (faucet.vlan.VLAN static method), 170  
 flood\_tagged\_port\_outputs () (in module faucet.valve\_of), 151  
 flood\_untagged\_port\_outputs () (in module faucet.valve\_of), 151  
 flow\_timeout () (faucet.valve.Valve method), 142  
 flow\_timeout () (faucet.valve\_host.ValveHostFlowRemovedManager method), 148  
 flow\_timeout () (faucet.valve\_host.ValveHostManager method), 149  
 flowcontroller () (faucet.valve\_table.ValveTable method), 167  
 flowdel () (faucet.valve\_table.ValveTable method), 167  
 flowdrop () (faucet.valve\_table.ValveTable method), 167  
 flowmod () (faucet.valve\_table.ValveTable method), 167  
 flowmod () (in module faucet.valve\_of), 151  
 flowremoved\_handler () (faucet.faucet.Faucet method), 126  
 from\_connected\_to\_vip () (faucet.vlan.VLAN method), 170
- ## G
- Gauge (class in faucet.gauge), 131  
 GaugeFlowTableInfluxDBLogger (class in faucet.gauge\_influx), 132  
 GaugeFlowTableLogger (class in faucet.watcher), 172  
 GaugeFlowTablePoller (class in faucet.gauge\_pollers), 135

- GaugeFlowTablePrometheusPoller (class in *faucet.gauge\_prom*), 136
  - GaugePoller (class in *faucet.gauge\_pollers*), 135
  - GaugePortStateInfluxDBLogger (class in *faucet.gauge\_influx*), 133
  - GaugePortStateLogger (class in *faucet.watcher*), 172
  - GaugePortStatePoller (class in *faucet.gauge\_pollers*), 136
  - GaugePortStatePrometheusPoller (class in *faucet.gauge\_prom*), 137
  - GaugePortStatsInfluxDBLogger (class in *faucet.gauge\_influx*), 134
  - GaugePortStatsLogger (class in *faucet.watcher*), 172
  - GaugePortStatsPoller (class in *faucet.gauge\_pollers*), 136
  - GaugePortStatsPrometheusPoller (class in *faucet.gauge\_prom*), 137
  - GaugePrometheusClient (class in *faucet.gauge\_prom*), 137
  - GaugeThreadPoller (class in *faucet.gauge\_pollers*), 136
  - get\_config() (*faucet.faucet.Faucet* method), 127
  - get\_config() (*faucet.faucet\_experimental\_api.FaucetExperimentalAPI* method), 129
  - get\_config\_changes() (*faucet.dp.DP* method), 124
  - get\_config\_dict() (*faucet.dp.DP* method), 124
  - get\_config\_dict() (*faucet.valve.Valve* method), 142
  - get\_config\_for\_api() (in module *faucet.config\_parser*), 122
  - get\_egress\_metadata() (in module *faucet.faucet\_metadata*), 130
  - get\_entry() (*faucet.valve\_table.ValveGroupTable* method), 166
  - get\_in\_port\_match() (*faucet.acl.ACL* method), 120
  - get\_logger() (in module *faucet.config\_parser\_util*), 123
  - get\_logger() (in module *faucet.valve\_util*), 167
  - get\_mac\_str() (in module *faucet.faucet\_dot1x*), 129
  - get\_meters() (*faucet.acl.ACL* method), 120
  - get\_mirror\_destinations() (*faucet.acl.ACL* method), 120
  - get\_native\_vlan() (*faucet.dp.DP* method), 124
  - get\_ports() (*faucet.vlan.VLAN* method), 170
  - get\_samples() (in module *faucet.fctl*), 131
  - get\_setting() (*faucet.valve\_ryuapp.RyuAppBase* method), 165
  - get\_setting() (in module *faucet.valve\_util*), 167
  - get\_sys\_prefix() (in module *faucet.valve\_util*), 167
  - get\_tables() (*faucet.dp.DP* method), 124
  - get\_tables() (*faucet.faucet.Faucet* method), 127
  - get\_tables() (*faucet.faucet\_experimental\_api.FaucetExperimentalAPI* method), 129
  - get\_tunnel\_flowmods() (*faucet.valve.Valve* method), 142
  - get\_tunnel\_id() (*faucet.acl.ACL* method), 120
  - get\_tunnel\_rule\_indices() (*faucet.acl.ACL* method), 120
  - get\_valve\_and\_port() (*faucet.faucet\_dot1x.FaucetDot1x* method), 128
  - global\_routing (*faucet.valve\_route.ValveRouteManager* attribute), 164
  - global\_vlan (*faucet.valve\_route.ValveRouteManager* attribute), 164
  - goto() (*faucet.valve\_table.ValveTable* method), 167
  - goto\_miss() (*faucet.valve\_table.ValveTable* method), 167
  - goto\_table() (in module *faucet.valve\_of*), 151
  - goto\_this() (*faucet.valve\_table.ValveTable* method), 167
  - group\_act() (in module *faucet.valve\_of*), 151
  - group\_id\_from\_str() (*faucet.valve\_table.ValveGroupTable* static method), 166
  - groupadd() (in module *faucet.valve\_of*), 151
  - groupadd\_ff() (in module *faucet.valve\_of*), 151
  - groupdel() (in module *faucet.valve\_of*), 151
  - groupmod() (in module *faucet.valve\_of*), 151
  - groupmod\_ff() (in module *faucet.valve\_of*), 151
  - GROUPS (*faucet.valve.Valve* attribute), 141
- ## H
- hairpin\_ports() (*faucet.vlan.VLAN* method), 170
  - host\_manager (*faucet.valve.Valve* attribute), 142
  - HostCacheEntry (class in *faucet.vlan*), 169
  - hosts() (*faucet.port.Port* method), 138
  - hosts\_count() (*faucet.port.Port* method), 138
  - hosts\_count() (*faucet.vlan.VLAN* method), 170
- ## I
- ICMP\_SIZE (*faucet.valve\_route.ValveIPv4RouteManager* attribute), 163
  - ICMP\_SIZE (*faucet.valve\_route.ValveRouteManager* attribute), 163
  - ICMP\_TYPE (*faucet.valve\_route.ValveIPv4RouteManager* attribute), 163
  - ICMP\_TYPE (*faucet.valve\_route.ValveIPv6RouteManager* attribute), 163
  - ICMP\_TYPE (*faucet.valve\_route.ValveRouteManager* attribute), 163
  - icmpv6\_echo\_reply() (in module *faucet.valve\_packet*), 157



- ignore\_port() (in module *faucet.valve\_of*), 151
- ignore\_subconf() (*faucet.conf.Conf* method), 122
- in\_port\_tables() (*faucet.dp.DP* method), 124
- inc\_var() (*faucet.faucet\_metrics.FaucetMetrics* method), 130
- InfluxShipper (class in *faucet.gauge\_influx*), 134
- info() (*faucet.valve.ValveLogger* method), 146
- initialise\_tables() (*faucet.valve\_acl.ValveAclManager* method), 146
- initialise\_tables() (*faucet.valve\_flood.ValveFloodManager* method), 147
- initialise\_tables() (*faucet.valve\_host.ValveHostManager* method), 149
- initialise\_tables() (*faucet.valve\_manager\_base.ValveManagerBase* method), 149
- initialise\_tables() (*faucet.valve\_pipeline.ValvePipeline* method), 161
- int\_from\_mac() (in module *faucet.valve\_packet*), 157
- int\_in\_mac() (in module *faucet.valve\_packet*), 158
- InvalidConfigError, 122
- ip\_dsts\_for\_ip\_gw() (*faucet.vlan.VLAN* method), 170
- ip\_in\_vip\_subnet() (*faucet.vlan.VLAN* method), 170
- IP\_PKT (*faucet.valve\_route.ValveIPv4RouteManager* attribute), 163
- IP\_PKT (*faucet.valve\_route.ValveIPv6RouteManager* attribute), 163
- IP\_PKT (*faucet.valve\_route.ValveRouteManager* attribute), 163
- ip\_ver() (*faucet.valve\_packet.PacketMeta* method), 155
- ipaddress\_fields (*faucet.router.Router* attribute), 140
- IPV (*faucet.valve\_route.ValveIPv4RouteManager* attribute), 163
- IPV (*faucet.valve\_route.ValveIPv6RouteManager* attribute), 163
- IPV (*faucet.valve\_route.ValveRouteManager* attribute), 163
- ipv4\_parseable() (in module *faucet.valve\_packet*), 158
- ipv6\_link\_eth\_mcast() (in module *faucet.valve\_packet*), 158
- ipv6\_solicited\_node\_from\_ucast() (in module *faucet.valve\_packet*), 158
- ipvs() (*faucet.vlan.VLAN* method), 170
- is\_active() (*faucet.gauge\_pollers.GaugePoller* static method), 135
- is\_active() (*faucet.gauge\_pollers.GaugeThreadPoller* method), 136
- is\_apply\_actions() (in module *faucet.valve\_of*), 151
- is\_delete() (in module *faucet.valve\_of*), 151
- is\_faucet\_vip() (*faucet.vlan.VLAN* method), 170
- is\_flowdel() (in module *faucet.valve\_of*), 151
- is\_flowmod() (in module *faucet.valve\_of*), 152
- is\_groupadd() (in module *faucet.valve\_of*), 152
- is\_groupdel() (in module *faucet.valve\_of*), 152
- is\_groupmod() (in module *faucet.valve\_of*), 152
- is\_host\_fib\_route() (*faucet.vlan.VLAN* method), 170
- is\_in\_path() (*faucet.dp.DP* method), 124
- is\_meter() (in module *faucet.valve\_of*), 152
- is\_meteradd() (in module *faucet.valve\_of*), 152
- is\_meterdel() (in module *faucet.valve\_of*), 152
- is\_metermod() (in module *faucet.valve\_of*), 153
- is\_output() (in module *faucet.valve\_of*), 153
- is\_packetout() (in module *faucet.valve\_of*), 153
- is\_registered() (*faucet.faucet\_experimental\_api.FaucetExperimental* method), 129
- is\_set\_field() (in module *faucet.valve\_of*), 153
- is\_stack\_admin\_down() (*faucet.port.Port* method), 138
- is\_stack\_down() (*faucet.port.Port* method), 138
- is\_stack\_edge() (*faucet.dp.DP* method), 124
- is\_stack\_init() (*faucet.port.Port* method), 138
- is\_stack\_root() (*faucet.dp.DP* method), 124
- is\_stack\_up() (*faucet.port.Port* method), 138
- is\_table\_features\_req() (in module *faucet.valve\_of*), 153
- ## K
- kill\_on\_exception() (in module *faucet.valve\_util*), 167
- ## L
- l3\_dst (*faucet.valve\_packet.PacketMeta* attribute), 155
- l3\_pkt (*faucet.valve\_packet.PacketMeta* attribute), 155
- l3\_src (*faucet.valve\_packet.PacketMeta* attribute), 155
- lACP\_down() (*faucet.valve.Valve* method), 142
- lACP\_handler() (*faucet.valve.Valve* method), 142
- lACP\_ports() (*faucet.vlan.VLAN* method), 170
- lACP\_reqreply() (in module *faucet.valve\_packet*), 158
- lACP\_up() (*faucet.valve.Valve* method), 143
- lACP\_up\_ports() (*faucet.vlan.VLAN* method), 170
- lags() (*faucet.vlan.VLAN* method), 171
- lags\_up() (*faucet.vlan.VLAN* method), 171
- last\_retry\_time (*faucet.valve\_route.NextHop* attribute), 162

learn\_host\_on\_vlan\_port\_flows() (faucet.valve\_host.ValveHostManager method), 149

learn\_host\_on\_vlan\_ports() (faucet.valve\_host.ValveHostManager method), 149

learn\_host\_timeouts() (faucet.valve\_host.ValveHostFlowRemovedManager method), 148

learn\_host\_timeouts() (faucet.valve\_host.ValveHostManager method), 149

link\_and\_other\_vips() (faucet.vlan.VLAN method), 171

lldp\_beacon() (in module faucet.valve\_packet), 159

lldp\_beacon\_defaults\_types (faucet.dp.DP attribute), 125

lldp\_beacon\_defaults\_types (faucet.port.Port attribute), 138

lldp\_beacon\_enabled() (faucet.port.Port method), 138

lldp\_beacon\_send\_ports() (faucet.dp.DP method), 125

lldp\_handler() (faucet.valve.Valve method), 143

lldp\_org\_tlv\_defaults\_types (faucet.port.Port attribute), 138

load\_configs() (faucet.valves\_manager.ValvesManager method), 168

load\_tables() (in module faucet.tfm\_pipeline), 140

log() (faucet.valve\_packet.PacketMeta method), 156

logger (faucet.gauge\_influx.InfluxShipper attribute), 135

logger (faucet.valve.Valve attribute), 143

logger (faucet.valve\_route.ValveRouteManager attribute), 164

logname (faucet.faucet.Faucet attribute), 127

logname (faucet.gauge.Gauge attribute), 132

logname (faucet.valve.Valve attribute), 143

logname (faucet.valve\_ryuapp.RyuAppBase attribute), 166

logoff\_handler() (faucet.faucet\_dot1x.FaucetDot1x method), 128

loop\_protect\_external\_ports() (faucet.vlan.VLAN method), 171

loop\_protect\_external\_ports\_up() (faucet.vlan.VLAN method), 171

**M**

mac\_addr\_is\_unicast() (in module faucet.valve\_packet), 159

mac\_byte\_mask() (in module faucet.valve\_packet), 159

mac\_mask\_bits() (in module faucet.valve\_packet), 159

main() (in module faucet.check\_faucet\_config), 121

main() (in module faucet.fctl), 131

make\_point() (faucet.gauge\_influx.InfluxShipper static method), 135

make\_port\_point() (faucet.gauge\_influx.InfluxShipper method), 135

make\_wsgi\_app() (in module faucet.prom\_client), 139

match() (faucet.valve\_table.ValveTable static method), 167

match() (in module faucet.valve\_of), 153

match\_from\_dict() (in module faucet.valve\_of), 153

match\_tables() (faucet.dp.DP method), 125

MAX\_ETH\_TYPE\_PKT\_SIZE (faucet.valve\_packet.PacketMeta attribute), 155

max\_host\_fib\_retry\_count (faucet.valve\_route.ValveRouteManager attribute), 164

max\_hosts\_per\_resolve\_cycle (faucet.valve\_route.ValveRouteManager attribute), 164

max\_resolve\_backoff\_time (faucet.valve\_route.ValveRouteManager attribute), 164

merge\_dyn() (faucet.conf.Conf method), 122

metadata\_goto\_table() (in module faucet.valve\_of), 153

Meter (class in faucet.meter), 138

meter\_id (faucet.meter.Meter attribute), 138

meteradd() (in module faucet.valve\_of), 153

meterdel() (in module faucet.valve\_of), 153

metric\_update() (faucet.faucet.Faucet method), 127

metrics (faucet.faucet.Faucet attribute), 127

metrics (faucet.valve.Valve attribute), 143

MIN\_ETH\_TYPE\_PKT\_SIZE (faucet.valve\_packet.PacketMeta attribute), 155

mirror\_actions() (faucet.port.Port method), 138

mirrored\_ports() (faucet.vlan.VLAN method), 171

modify() (faucet.valve\_table.ValveGroupEntry method), 166

modify\_stack\_topology() (faucet.dp.DP static method), 125

multi\_out (faucet.valve\_route.ValveRouteManager attribute), 164

mutable\_attrs (faucet.conf.Conf attribute), 122

mutable\_attrs (faucet.dp.DP attribute), 125

mutable\_attrs (faucet.vlan.VLAN attribute), 171

## N

name (*faucet.vlan.AnyVLAN attribute*), 169  
 name (*faucet.vlan.NullVLAN attribute*), 169  
 nd\_advert () (*in module faucet.valve\_packet*), 159  
 nd\_request () (*in module faucet.valve\_packet*), 159  
 neigh\_cache\_by\_ipv () (*faucet.vlan.VLAN method*), 171  
 neigh\_cache\_count\_by\_ipv () (*faucet.vlan.VLAN method*), 171  
 neighbor\_timeout (*faucet.valve\_route.ValveRouteManager attribute*), 164  
 new\_valve () (*faucet.valves\_manager.ValvesManager method*), 168  
 next\_retry () (*faucet.valve\_route.NextHop method*), 162  
 next\_retry\_time (*faucet.valve\_route.NextHop attribute*), 162  
 NextHop (*class in faucet.valve\_route*), 162  
 nexthop\_dead () (*faucet.valve\_route.ValveRouteManager method*), 165  
 nfv\_sw\_port\_up () (*faucet.faucet\_dot1x.FaucetDot1x method*), 128  
 no\_response () (*faucet.gauge\_pollers.GaugeFlowTablePoller method*), 135  
 no\_response () (*faucet.gauge\_pollers.GaugePoller method*), 135  
 no\_response () (*faucet.gauge\_pollers.GaugePortStatePoller method*), 136  
 no\_response () (*faucet.gauge\_pollers.GaugePortStatsPoller method*), 136  
 no\_response () (*faucet.gauge\_pollers.GaugeThreadPoller method*), 136  
 no\_response () (*faucet.watcher.GaugePortStateLogger static method*), 172  
 NonBlockLock (*class in faucet.faucet\_experimental\_event*), 130  
 NONEXT\_PORT\_FLAG (*faucet.valve\_flood.ValveFloodStackManager attribute*), 147  
 notifier (*faucet.faucet.Faucet attribute*), 127  
 notifier (*faucet.valve.Valve attribute*), 143  
 notify () (*faucet.faucet\_experimental\_event.FaucetExperimentalEventNotifier method*), 130  
 NoviFlowValve (*class in faucet.valve*), 141  
 NullVLAN (*class in faucet.vlan*), 169

## O

ofchannel\_log () (*faucet.valve.Valve method*), 143  
 ofchannel\_logger (*faucet.valve.Valve attribute*), 143  
 ofdescstats\_handler () (*faucet.valve.Valve method*), 143  
 oferror () (*faucet.valve.Valve method*), 143  
 OFP\_VERSIONS (*faucet.valve\_ryuapp.RyuAppBase attribute*), 165

orig\_len (*faucet.valve\_packet.PacketMeta attribute*), 156  
 output () (*faucet.valve\_pipeline.ValvePipeline method*), 161  
 output\_actions\_types (*faucet.acl.ACL attribute*), 121  
 output\_controller () (*in module faucet.valve\_of*), 153  
 output\_in\_port () (*in module faucet.valve\_of*), 154  
 output\_port () (*faucet.vlan.VLAN method*), 171  
 output\_port () (*in module faucet.valve\_of*), 154  
 output\_table () (*faucet.dp.DP method*), 125  
 output\_tables () (*faucet.dp.DP method*), 125  
 OVSVValve (*class in faucet.valve*), 141

## P

packet\_complete () (*faucet.valve\_packet.PacketMeta method*), 156  
 packet\_in\_handler () (*faucet.faucet.Faucet method*), 127  
 PacketMeta (*class in faucet.valve\_packet*), 155  
 packet\_out () (*in module faucet.valve\_of*), 154  
 packetouts () (*in module faucet.valve\_of*), 154  
 parse\_args () (*in module faucet.fctl*), 131  
 parse\_configs () (*faucet.valves\_manager.ValvesManager method*), 168  
 parse\_eth\_pkt () (*in module faucet.valve\_packet*), 160  
 parse\_faucet\_lldp () (*in module faucet.valve\_packet*), 160  
 parse\_lacp\_pkt () (*in module faucet.valve\_packet*), 160  
 parse\_lldp () (*in module faucet.valve\_packet*), 160  
 parse\_packet\_in\_pkt () (*in module faucet.valve\_packet*), 160  
 parse\_pkt\_meta () (*faucet.valve.Valve method*), 143  
 parse\_rcv\_packet () (*faucet.valve.Valve method*), 143  
 peer\_stack\_up\_ports () (*faucet.dp.DP method*), 125  
 pipeline (*faucet.valve.Valve attribute*), 144  
 pipeline (*faucet.valve\_route.ValveRouteManager attribute*), 165  
 pkt (*faucet.valve\_packet.PacketMeta attribute*), 156  
 pkt\_out\_port () (*faucet.vlan.VLAN method*), 171  
 pop\_vlan () (*in module faucet.valve\_of*), 154  
 Port (*class in faucet.port*), 138  
 port (*faucet.valve\_packet.PacketMeta attribute*), 156  
 port (*faucet.valve\_route.NextHop attribute*), 162  
 port (*faucet.vlan.HostCacheEntry attribute*), 169  
 port\_add () (*faucet.valve.Valve method*), 144  
 port\_delete () (*faucet.valve.Valve method*), 144

port\_down() (*faucet.faucet\_dot1x.FaucetDot1x method*), 128  
 port\_is\_tagged() (*faucet.vlan.VLAN method*), 171  
 port\_is\_untagged() (*faucet.vlan.VLAN method*), 171  
 port\_labels() (*faucet.dp.DP method*), 125  
 port\_no\_valid() (*faucet.dp.DP method*), 125  
 port\_status\_from\_state() (*in module faucet.valve\_of*), 154  
 port\_status\_handler() (*faucet.faucet.Faucet method*), 127  
 port\_status\_handler() (*faucet.valve.Valve method*), 144  
 port\_status\_handler() (*faucet.valves\_manager.ValvesManager method*), 168  
 port\_up() (*faucet.faucet\_dot1x.FaucetDot1x method*), 128  
 ports\_add() (*faucet.valve.Valve method*), 144  
 ports\_delete() (*faucet.valve.Valve method*), 144  
 prepare\_send\_flows() (*faucet.valve.Valve method*), 144  
 proactive\_learn (*faucet.valve\_route.ValveRouteManager attribute*), 165  
 prom\_client (*faucet.gauge.Gauge attribute*), 132  
 PromClient (*class in faucet.prom\_client*), 139  
 push\_config() (*faucet.faucet\_experimental\_api.FaucetExperimentalAPI method*), 129  
 push\_vlan() (*in module faucet.valve\_acl*), 147  
 push\_vlan\_act() (*in module faucet.valve\_of*), 154

## R

rate\_limit\_packet\_ins() (*faucet.valve.Valve method*), 144  
 rcv\_packet() (*faucet.valve.Valve method*), 144  
 read\_config() (*in module faucet.config\_parser\_util*), 123  
 recent\_ofmsgs (*faucet.valve.Valve attribute*), 144  
 reconnect\_handler() (*faucet.valve\_ryuapp.RyuAppBase method*), 166  
 release() (*faucet.faucet\_experimental\_event.NonBlockEvent method*), 130  
 reload\_config() (*faucet.faucet.Faucet method*), 127  
 reload\_config() (*faucet.faucet\_experimental\_api.FaucetExperimentalAPI method*), 129  
 reload\_config() (*faucet.gauge.Gauge method*), 132  
 reload\_config() (*faucet.valve.Valve method*), 144  
 reload\_config() (*faucet.valve\_ryuapp.RyuAppBase method*), 166  
 remove\_filter() (*faucet.valve\_pipeline.ValvePipeline method*), 162  
 remove\_non\_tunnel\_rules() (*faucet.acl.ACL method*), 121  
 remove\_stack\_link() (*faucet.dp.DP class method*), 125  
 reparse() (*faucet.valve\_packet.PacketMeta method*), 156  
 reparse\_all() (*faucet.valve\_packet.PacketMeta method*), 156  
 reparse\_ip() (*faucet.valve\_packet.PacketMeta method*), 156  
 report\_dp\_status() (*faucet.gauge\_pollers.GaugePoller method*), 135  
 report\_label\_match\_metrics() (*in module faucet.fctl*), 131  
 request\_reload\_configs() (*faucet.valves\_manager.ValvesManager method*), 168  
 REQUIRED\_LABELS (*faucet.prom\_client.PromClient attribute*), 139  
 reregister\_flow\_vars() (*faucet.gauge\_prom.GaugePrometheusClient method*), 137  
 reset() (*faucet.faucet\_bgp.FaucetBgp method*), 127  
 reset() (*faucet.faucet\_dot1x.FaucetDot1x method*), 128  
 reset\_acl() (*faucet.vlan.VLAN method*), 171  
 reset\_dp\_id() (*faucet.faucet\_metrics.FaucetMetrics method*), 130  
 reset\_ports() (*faucet.vlan.VLAN method*), 171  
 reset\_refs() (*faucet.dp.DP method*), 125  
 resolution\_due() (*faucet.valve\_route.NextHop method*), 162  
 resolve\_expire\_hosts() (*faucet.valve\_route.ValveRouteManager method*), 165  
 resolve\_gateways() (*faucet.valve.Valve method*), 145  
 resolve\_gateways() (*faucet.valve\_route.ValveRouteManager method*), 165  
 resolve\_port() (*faucet.dp.DP method*), 125  
 resolve\_ports() (*faucet.acl.ACL method*), 121  
 resolve\_retries (*faucet.valve\_route.NextHop attribute*), 162  
 resolve\_stack\_topology() (*faucet.dp.DP method*), 125  
 rewrite\_vlan() (*in module faucet.valve\_acl*), 147  
 route\_count\_by\_ipv() (*faucet.vlan.VLAN method*), 171  
 route\_priority (*faucet.valve\_route.ValveRouteManager attribute*), 165  
 Router (*class in faucet.router*), 139  
 router\_advert() (*in module faucet.valve\_packet*),



- 160
- routers (*faucet.valve\_route.ValveRouteManager* attribute), 165
- routes\_by\_ipv() (*faucet.vlan.VLAN* method), 171
- rule\_types (*faucet.acl.ACL* attribute), 121
- running() (*faucet.gauge\_pollers.GaugePoller* method), 135
- running() (*faucet.port.Port* method), 139
- RyuAppBase (class in *faucet.valve\_ryuapp*), 165
- ## S
- scrape\_prometheus() (in module *faucet.fctl*), 131
- select\_packets() (*faucet.valve\_pipeline.ValvePipeline* method), 162
- send\_flows() (*faucet.valve.Valve* method), 145
- send\_req() (*faucet.gauge\_pollers.GaugeFlowTablePoller* method), 135
- send\_req() (*faucet.gauge\_pollers.GaugePoller* method), 135
- send\_req() (*faucet.gauge\_pollers.GaugePortStatePoller* method), 136
- send\_req() (*faucet.gauge\_pollers.GaugePortStatsPoller* method), 136
- send\_req() (*faucet.gauge\_pollers.GaugeThreadPoller* method), 136
- send\_req() (*faucet.watcher.GaugePortStateLogger* static method), 172
- server (*faucet.prom\_client.PromClient* attribute), 139
- set\_bgp\_vlan() (*faucet.router.Router* method), 140
- set\_defaults() (*faucet.conf.Conf* method), 122
- set\_defaults() (*faucet.dp.DP* method), 125
- set\_defaults() (*faucet.port.Port* method), 139
- set\_defaults() (*faucet.router.Router* method), 140
- set\_defaults() (*faucet.vlan.VLAN* method), 171
- set\_field() (*faucet.valve\_table.ValveTable* method), 167
- set\_field() (in module *faucet.valve\_of*), 155
- set\_mac\_str() (*faucet.faucet\_dot1x.FaucetDot1x* method), 128
- set\_vlan\_vid() (*faucet.valve\_table.ValveTable* method), 167
- ship\_error\_prefix (*faucet.gauge\_influx.InfluxShipper* attribute), 135
- ship\_points() (*faucet.gauge\_influx.InfluxShipper* method), 135
- shortest\_path() (*faucet.dp.DP* method), 125
- shortest\_path\_port() (*faucet.dp.DP* method), 125
- shortest\_path\_to\_root() (*faucet.dp.DP* method), 125
- shutdown\_bgp\_speakers() (*faucet.faucet\_bgp.FaucetBgp* method), 127
- signal\_handler() (*faucet.valve\_ryuapp.RyuAppBase* method), 166
- stack\_admin\_down() (*faucet.port.Port* method), 139
- stack\_defaults\_types (*faucet.dp.DP* attribute), 125
- stack\_defaults\_types (*faucet.port.Port* attribute), 139
- stack\_down() (*faucet.port.Port* method), 139
- stack\_init() (*faucet.port.Port* method), 139
- stack\_up() (*faucet.port.Port* method), 139
- start() (*faucet.faucet.Faucet* method), 127
- start() (*faucet.faucet\_experimental\_event.FaucetExperimentalEventNot* method), 130
- start() (*faucet.gauge\_pollers.GaugePoller* method), 135
- start() (*faucet.gauge\_pollers.GaugeThreadPoller* method), 136
- start() (*faucet.prom\_client.PromClient* method), 139
- start() (*faucet.valve\_ryuapp.RyuAppBase* method), 166
- stat\_config\_files() (in module *faucet.valve\_util*), 168
- state\_expire() (*faucet.valve.Valve* method), 145
- STATIC\_TABLE\_IDS (*faucet.valve.NoviFlowValve* attribute), 141
- STATIC\_TABLE\_IDS (*faucet.valve.Valve* attribute), 141
- stop() (*faucet.gauge\_pollers.GaugePoller* method), 135
- stop() (*faucet.gauge\_pollers.GaugeThreadPoller* method), 136
- switch\_features() (*faucet.valve.Valve* method), 145
- ## T
- table\_by\_id() (*faucet.dp.DP* method), 125
- table\_features() (in module *faucet.valve\_of*), 155
- tagged\_flood\_ports() (*faucet.vlan.VLAN* method), 171
- test\_config\_condition() (in module *faucet.conf*), 122
- TfmValve (class in *faucet.valve*), 141
- thread (*faucet.prom\_client.PromClient* attribute), 139
- tlv\_cast() (in module *faucet.valve\_packet*), 161
- tlvs\_by\_subtype() (in module *faucet.valve\_packet*), 161
- tlvs\_by\_type() (in module *faucet.valve\_packet*), 161
- to\_conf() (*faucet.conf.Conf* method), 122
- tunnel\_types (*faucet.acl.ACL* attribute), 121
- ## U
- UniqueKeyLoader (class in

*faucet.config\_parser\_util*), 122  
 unpack\_tunnel() (*faucet.acl.ACL method*), 121  
 untagged\_flood\_ports() (*faucet.vlan.VLAN method*), 171  
 update() (*faucet.conf.Config method*), 122  
 update() (*faucet.gauge\_influx.GaugeFlowTableInfluxDBLogger method*), 133  
 update() (*faucet.gauge\_influx.GaugePortStateInfluxDBLogger method*), 133  
 update() (*faucet.gauge\_influx.GaugePortStatsInfluxDBLogger method*), 134  
 update() (*faucet.gauge\_pollers.GaugePoller method*), 135  
 update() (*faucet.gauge\_prom.GaugeFlowTablePrometheusPoller method*), 137  
 update() (*faucet.gauge\_prom.GaugePortStatePrometheusPoller method*), 137  
 update() (*faucet.gauge\_prom.GaugePortStatsPrometheusPoller method*), 137  
 update() (*faucet.valves\_manager.ConfigWatcher method*), 168  
 update() (*faucet.watcher.GaugeFlowTableLogger method*), 172  
 update() (*faucet.watcher.GaugePortStateLogger method*), 172  
 update() (*faucet.watcher.GaugePortStatsLogger method*), 172  
 update\_buckets() (*faucet.valve\_table.ValveGroupEntry method*), 166  
 update\_config\_metrics() (*faucet.valve.Valve method*), 145  
 update\_metrics() (*faucet.faucet\_bgp.FaucetBgp method*), 127  
 update\_metrics() (*faucet.valve.Valve method*), 145  
 update\_metrics() (*faucet.valves\_manager.ValvesManager method*), 168  
 update\_stack\_topo() (*faucet.valve\_flood.ValveFloodManager static method*), 147  
 update\_stack\_topo() (*faucet.valve\_flood.ValveFloodStackManager method*), 148  
 update\_tunnel\_acl\_conf() (*faucet.acl.ACL method*), 121  
 update\_tunnel\_flowrules() (*faucet.valve.Valve method*), 145  
 update\_watcher\_handler() (*faucet.gauge.Gauge method*), 132  
 USE\_BARRIERS (*faucet.valve.NoviFlowValve attribute*), 141  
 USE\_BARRIERS (*faucet.valve.OVSValve attribute*), 141  
 USE\_BARRIERS (*faucet.valve.Valve attribute*), 141  
 utf8\_decode() (*in module faucet.valve\_util*), 168

## V

Valve (*class in faucet.valve*), 141  
 valve\_factory() (*in module faucet.valve*), 146  
 valve\_flow\_services() (*faucet.valves\_manager.ValvesManager method*), 168  
 valve\_flowreorder() (*in module faucet.valve\_of*), 155  
 valve\_match\_vid() (*in module faucet.valve\_of*), 155  
 valve\_packet\_in() (*faucet.valves\_manager.ValvesManager method*), 168  
 ValveACLManager (*class in faucet.valve\_acl*), 146  
 ValveDeadThreadException, 166  
 ValveFloodManager (*class in faucet.valve\_flood*), 147  
 ValveFloodStackManager (*class in faucet.valve\_flood*), 147  
 ValveGroupEntry (*class in faucet.valve\_table*), 166  
 ValveGroupTable (*class in faucet.valve\_table*), 166  
 ValveHostFlowRemovedManager (*class in faucet.valve\_host*), 148  
 ValveHostManager (*class in faucet.valve\_host*), 148  
 ValveIPv4RouteManager (*class in faucet.valve\_route*), 162  
 ValveIPv6RouteManager (*class in faucet.valve\_route*), 163  
 ValveLogger (*class in faucet.valve*), 145  
 ValveManagerBase (*class in faucet.valve\_manager\_base*), 149  
 ValvePipeline (*class in faucet.valve\_pipeline*), 161  
 ValveRouteManager (*class in faucet.valve\_route*), 163  
 valves (*faucet.valves\_manager.ValvesManager attribute*), 168  
 valves\_manager (*faucet.faucet.Faucet attribute*), 127  
 ValvesManager (*class in faucet.valves\_manager*), 168  
 ValveTable (*class in faucet.valve\_table*), 166  
 ValveTableConfig (*class in faucet.faucet\_pipeline*), 130  
 verify\_tunnel\_compatibility\_rules() (*faucet.acl.ACL method*), 121  
 vid (*faucet.vlan.AnyVLAN attribute*), 169  
 vid (*faucet.vlan.NullVLAN attribute*), 169  
 vid\_present() (*in module faucet.valve\_of*), 155  
 vid\_valid() (*faucet.vlan.VLAN static method*), 171  
 vip\_map() (*faucet.router.Router method*), 140  
 vip\_map() (*faucet.vlan.VLAN method*), 171  
 vip\_table (*faucet.valve\_route.ValveRouteManager attribute*), 165  
 VLAN (*class in faucet.vlan*), 169  
 vlan (*faucet.valve\_packet.PacketMeta attribute*), 156

`vlan_pkt` (*faucet.valve\_packet.PacketMeta* attribute),  
156

`vlangs()` (*faucet.port.Port* method), 139

## W

`warning()` (*faucet.valve.ValveLogger* method), 146

`watcher_factory()` (*in module faucet.watcher*),  
173

`watcher_parser()` (*in module*  
*faucet.config\_parser*), 122

`WatcherConf` (*class in faucet.watcher\_conf*), 173