Software-Defined Enterprise Networks

Brad Cowie
What makes an enterprise network?

- Connects users to services and Internet
- Lots of copper ports and many wireless APs
- Hard to design a standard build-out
  - Too many special cases
  - Odd building layouts
- Often have no control over devices at the access layer
  - BYOD
- Network design has to scale to support all these edge-cases
Problems in Enterprise Networks

- Hand configuration….
- Configuration automation isn’t always the answer
  - Multi-vendor is difficult
  - Can’t define our own learning behaviour
- We want Devops for networks
  - Faucet: Deploying SDN in the Enterprise
    Using OpenFlow and DevOps for rapid development
    [https://queue.acm.org/detail.cfm?id=3015763](https://queue.acm.org/detail.cfm?id=3015763)
What is FAUCET?

- Lightweight Open Source SDN controller
- OpenFlow v1.3
- Multi-vendor (single packet processing pipeline)
- Supports Layer 2 and Layer 3
- Policy driven approach to extensibility
- Monitoring and Instrumentation
- Well-tested
- Production-ready
- Installs in <30 seconds
FAUCET architecture
FAUCET configuration

vlans:
  office:
    vid: 100
    description: "office network"
    faucet_vips: ['10.0.100.254/24']
dps:
  sw1:
    dp_id: 0x1
    hardware: "Open vSwitch"
    interfaces:
      1:
        description: "host1 container"
        native_vlan: office
      2:
        description: "host2 container"
        native_vlan: office
Testing and validation

- FAUCET test suite includes unit tests and integration tests
- Test suite performs 139 different test scenarios
  - Includes real topologies
  - Includes real traffic
- All commits into FAUCET are automatically tested with Travis
- Test suite can also run on physical hardware switches
- We can qualify new network kit with test suite to validate features
- No more attempting to parse vendor documentation
- Automate your RFP process
Push on green

● Track network changes in git
● Integrate test suite with a Continuous Integration tool
● Run test suite on virtual switches against new config
● Run test suite on physical switches against new config
● If tests are passing, push on green!
Network visibility

- Let’s not reinvent the wheel and write our own
- Prometheus (scrapes FAUCET/GAUGE)
  - MAC table
  - Port state
  - Port counters (bytes in/out, packets in/out, errors)
  - OpenFlow channel utilisation
  - Instrumentation
- Grafana provides dashboards & real-time graphs of data in Prometheus
- Prometheus provides alerting
Real-time graphs of host learning
Real-time graphs of OpenFlow control channel

Old Code

New Code
Network visibility - prometheus

- Central database of all knowledge of network
- Where the MAC table?
  - FAUCET includes a centralised, time-series, queryable database of learned MACs
- FCTL tool for querying information
WAND redcables network

- AS 134227
- 192.107.171.0/24
- 192.107.172.0/24
- 2001:df2:9d00::/45
- 248 OpenFlow ports
WAND redcables network

Access

Aggregation
Multiple FAUCET controllers

- Load balancing
- Redundancy
- Separation of duties

```
<table>
<thead>
<tr>
<th>FAUCET L3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenvSwitch/DPDK</td>
</tr>
<tr>
<td>FAUCET L2</td>
</tr>
<tr>
<td>AT-x930-1</td>
</tr>
<tr>
<td>Aruba-2930f-1</td>
</tr>
<tr>
<td>Cisco-3850</td>
</tr>
<tr>
<td>AT-x510-1</td>
</tr>
<tr>
<td>AT-x510-2</td>
</tr>
<tr>
<td>AT-x510-3</td>
</tr>
<tr>
<td>AT-x510-4</td>
</tr>
</tbody>
</table>
```
Ansible for network management

- You can manage a network using ansible without SSH/NetConf/YANG
- We configure every network element with ansible and store in git
- Each commit represents a different network state
- Git makes rolling back & peer review simple
- Perform sanity checks on deploy and roll-back when we find issues
- Use different ansible inventories to separate staging from production
  - Deploy to a “staging” mininet topology first to test network functions
  - Deploy to a canary network first to validate configuration
- Redcables ansible repo is open-source on GitHub
  - [https://github.com/wandsdn/redcables-ansible](https://github.com/wandsdn/redcables-ansible)
Redcables deployment schedule

- Do weekly deployments at 4pm on a Monday
- ~8 seconds of service interruption if we have to restart FAUCET-L3 and flap BGP
- No service interruption for configuration changes
Linux NFV server

- Place services where CPU resources are
- Services are KVM VMs
  - WAND’s AMP (Active network monitoring)
  - Catalyst’s AWDY? (DDoS monitoring)
  - Jool NAT64 (IPv6 only network)
  - isc-dhcp-server (DHCP and DHCPv6)
  - bird (BGP)
- Userspace OpenvSwitch with DPDK acceleration
OpenFlow processing pipeline

(a) Packets are matched against multiple tables in the pipeline

① Find highest-priority matching flow entry

② Apply instructions:
   i. Modify packet & update match fields (apply actions instruction)
   ii. Update action set (clear actions and/or write actions instructions)
   iii. Update metadata

③ Send match data and action set to next table
FAUCET multi-table pipeline
Network functions implemented on OpenvSwitch

- Layer 2 (Forwarding)
- Virtual IPs (VIPs)
- Layer 3 (Routing)
- Policy/ACLs
Forwarding in OpenFlow

Match VLAN:
\[\text{table}=1, \text{priority}=9000, \text{in}\_port=2, \text{dl}\_vlan=250 \quad \text{actions}=\text{goto\_table}:3\]

A. Unknown source & destination:
\[\text{table}=3, \text{priority}=9000 \quad \text{actions}=?\]
\[\text{table}=7, \text{priority}=9000 \quad \text{actions}=\text{goto\_table}:8\]
\[\text{table}=8, \text{priority}=9000, \text{in}\_port=2, \text{dl}\_vlan=250 \quad \text{actions}=\text{output}:3, \text{output}:4, \text{pop}\_vlan, \text{output}:5\]

B. Known source & destination:
\[\text{table}=3, \text{priority}=9098, \text{in}\_port=2, \text{dl}\_vlan=250, \text{dl}\_src=00:0d:88:00:00:aa \quad \text{actions}=\text{goto\_table}:7\]
\[\text{table}=7, \text{priority}=9099, \text{dl}\_vlan=250, \text{dl}\_dst=52:54:00:57:56:78 \quad \text{actions}=\text{pop}\_vlan, \text{output}:10\]
VIPs in OpenFlow

Match VLAN:

\[\text{table}=1, \text{priority}=9000, \text{in\_port}=2, \text{dl\_vlan}=250 \quad \text{actions}=\text{goto\_table}:3\]

A. IPv4:

\[\text{table}=3, \text{priority}=9131, \text{arp}, \text{dl\_vlan}=250 \quad \text{actions}=\text{goto\_table}:6\]

\[\text{table}=6, \text{priority}=9133, \text{arp}, \text{arp\_tpa}=163.7.137.254 \quad \text{actions}=\text{CONTROLLER}:96\]

B. IPv6:

\[\text{table}=1, \text{priority}=9000, \text{in\_port}=2, \text{dl\_vlan}=250 \quad \text{actions}=\text{goto\_table}:3\]

\[\text{table}=3, \text{n\_packets}=2 \quad \text{priority}=9227, \text{ipv6}, \text{dl\_vlan}=250, \text{dl\_dst}=33:33:ff:00:ff:ff \quad \text{actions}=\text{goto\_table}:6\]

\[\text{table}=6, \text{priority}=9227, \text{icmp6}, \text{icmp\_type}=135 \quad \text{actions}=\text{CONTROLLER}:96, \text{goto\_table}:8\]
Routing in OpenFlow

Match VLAN:
\[
table=1, \text{priority}=9000, in\_port=2, dl\_vlan=250 \text{ actions}=goto\_table:3
\]

A. IPv4:
\[
table=3, \text{priority}=9099, ip, dl\_vlan=250, dl\_dst=0e:00:00:00:00:01 \text{ actions}=goto\_table:4
\]
\[
table=4, \text{priority}=9099, ip, dl\_vlan=250
\]
\[
\text{actions}=set\_field:2406->vlan\_vid, set\_field:0e:00:00:24:06->eth\_src, set\_field:f0:1c:2d:7f:a6:f7->eth\_dst, dec\_ttl, goto\_table:7
\]

B. IPv6:
\[
table=3, priority=9099, ipv6, dl\_vlan=250, dl\_dst=0e:00:00:00:00:01 \text{ actions}=goto\_table:5
\]
\[
table=5, priority=9099, ipv6, dl\_vlan=250
\]
\[
\text{actions}=set\_field:2406>\text{vlan\_vid}, set\_field:0e:00:00:24:06>\text{eth\_src}, set\_field:f0:1c:2d:7f:a6:f7->eth\_dst, dec\_ttl, goto\_table:7
\]

Send packet to router:
\[
table=7, \text{priority}=9099, dl\_vlan=2406, dl\_dst=f0:1c:2d:7f:a6:f7 \text{ actions}=output:7
\]
Implementing policy in FAUCET

- Network policy is implemented with FAUCET ACLs
- A FAUCET ACL has a match and action
  - Matches anything OpenFlow can
  - Action can be DROP, ALLOW, OUTPUT, MODIFY
- Port-based ACLs
- VLAN-based ACLs
- Inter-VLAN Routing ACLs
- Policy-based Routing ACLs
Network policy on redcables

- **Port-based ACLs**
  - DHCP and DHCPv6 spoofing protection
  - IPv6 Router Advertisement Guard
  - BCP38
  - NFV offload, output 802.1x EAPOL frames to NAC

- **VLAN-based ACLs**
  - Drop anything other than IPv6 ethertype on our IPv6-only network

- **IVR ACLs**
  - Limit traffic between VLANs

- **PBR ACLs**
  - Assign client subnets to a specific upstream
How do I try out FAUCET?

$ pip install faucet
$ ryu-manager faucet.faucet

or

$ docker run -d --name faucet \
   -v /etc/ryu/faucet/:/etc/ryu/faucet/ \ 
   -v /var/log/ryu/faucet/:/var/log/ryu/faucet/ \ 
   -p 6653:6653 -p 9244:9244 \ 
   faucet/faucet

http://docs.openvswitch.org/en/latest/tutorials/faucet/
Questions?

Contact Me:

- brad@waikato.ac.nz

Contact FAUCET:

- http://faucet.nz
- @faucetsdn
Bonus Slide: What have we learned from this?

● What was harder than expected
  ○ IPv6 Stateless Addressing
  ○ Be careful not to overload switches when reconfiguring 248 ports at same time
  ○ Python and Ryu can be expensive when used incorrectly
  ○ OpenFlow control channel is a scarce resource
  ○ DPDK requires a non-zero amount of tweaking
  ○ If you have humans in the loop they can still make typos (ASN, filters, etc)

● What was easier than expected
  ○ Turning up BGP with REANNZ and announcing network prefixes
  ○ Constantly renumbering network is as easy as running `sed' over git repo
  ○ Debugging issues is as easy as implementing a new test to cover issue
Bonus Slide: Security policy examples

- While I was building redcables there were some large security vulnerabilities
  - Intel AMT
  - WannaCry / SMB 1.0
- I was doing incident response on corporate University network for these
- Central firewall architectures only get you so far
- With FAUCET you can instantly deploy an ACLs to every port to drop these

```
- rule:
  dl_type: 0x800 # ipv4
  nw_proto: 6 # tcp
  tcp_dst: 16992/0x7FFC # intel-amt-http, intel-amt-https, intel-amt-redir
  actions:
    allow: 0 # drop
```
Bonus Slide: Policy based packet inspection

- Carve packets off a (large) link and direct at Endace DAG capture card
- No longer have to inspect entire links
- Distributed packet inspection (steer packets towards nearest DAG)
- Can signal DAG card with metadata about what is being captured

- rule:
  
  dl_type: 0x800  # ipv4
  nw_proto: 6    # tcp
  actions:
     output:
       port: dag  # copy to DAG capture interface