Deploying FAUCET in the Enterprise

Brad Cowie

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

WAND
Network Research Group

What makes an enterprise network?

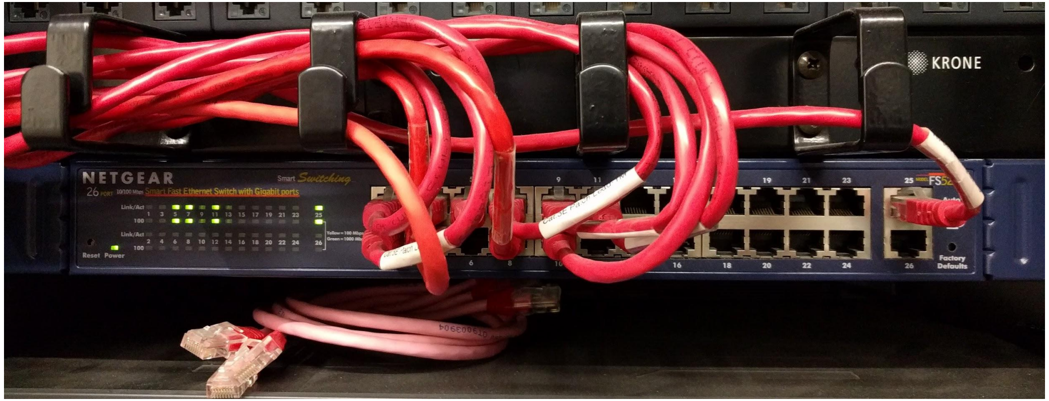
- Network that connects your users to services/WAN
- Lots of copper ports and many wireless APs
- Hard to design a standard build-out
  - Too many special cases
  - Odd building layouts
- Often have no control over devices at the access layer
  - BYOD
- Network design has to scale to support all these edge-cases

Why do I run my own enterprise network?

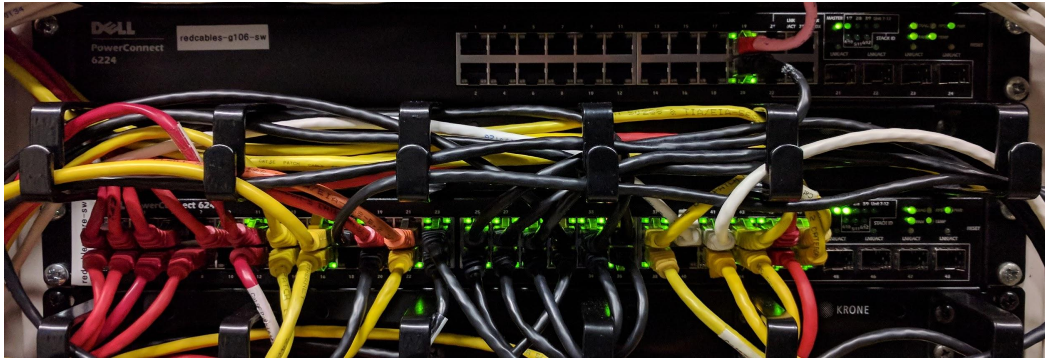
- University network often has restrictions that prevent our research
- University network has long lead times on new services
- We have always maintained our own network for these reasons
- Original drivers for our own network
    - Research traffic
    - BYOD
- In operating an SDN network we've discovered new drivers too

WAND redcables network

- 1st Generation
  - Unmanaged
  - 10/100
  - Linux router/firewall

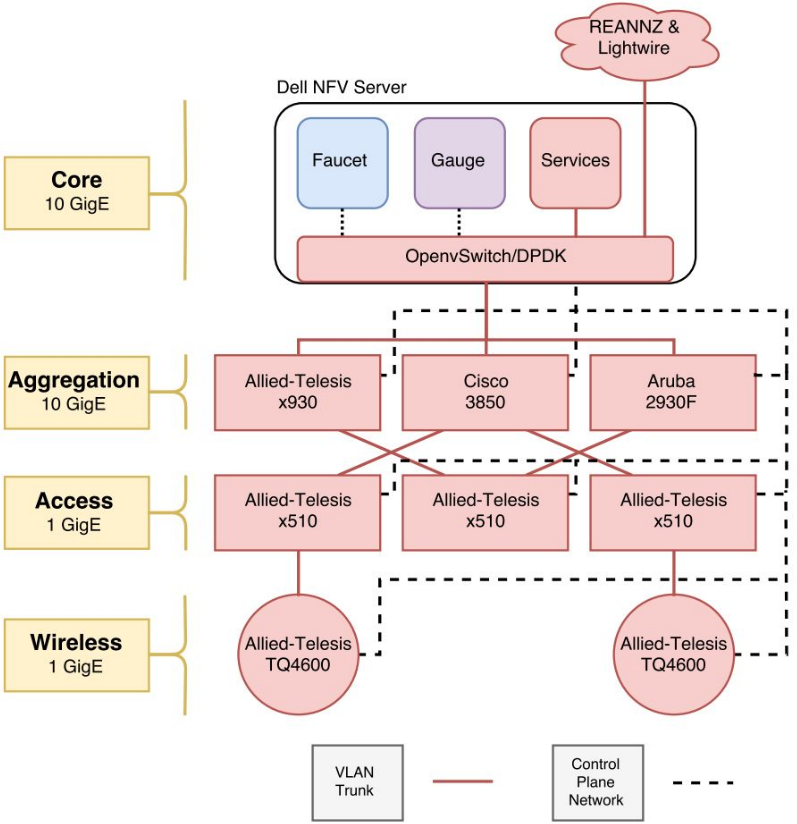

KRONE
NETGEAR
26 PORT 10/100 Mbps Smart Fast Ethernet Switch with Gigabit ports
Smart Switching
Factory Defaults

- 2nd Generation
  - Managed
  - 10/100/1000
  - Linux router/firewall

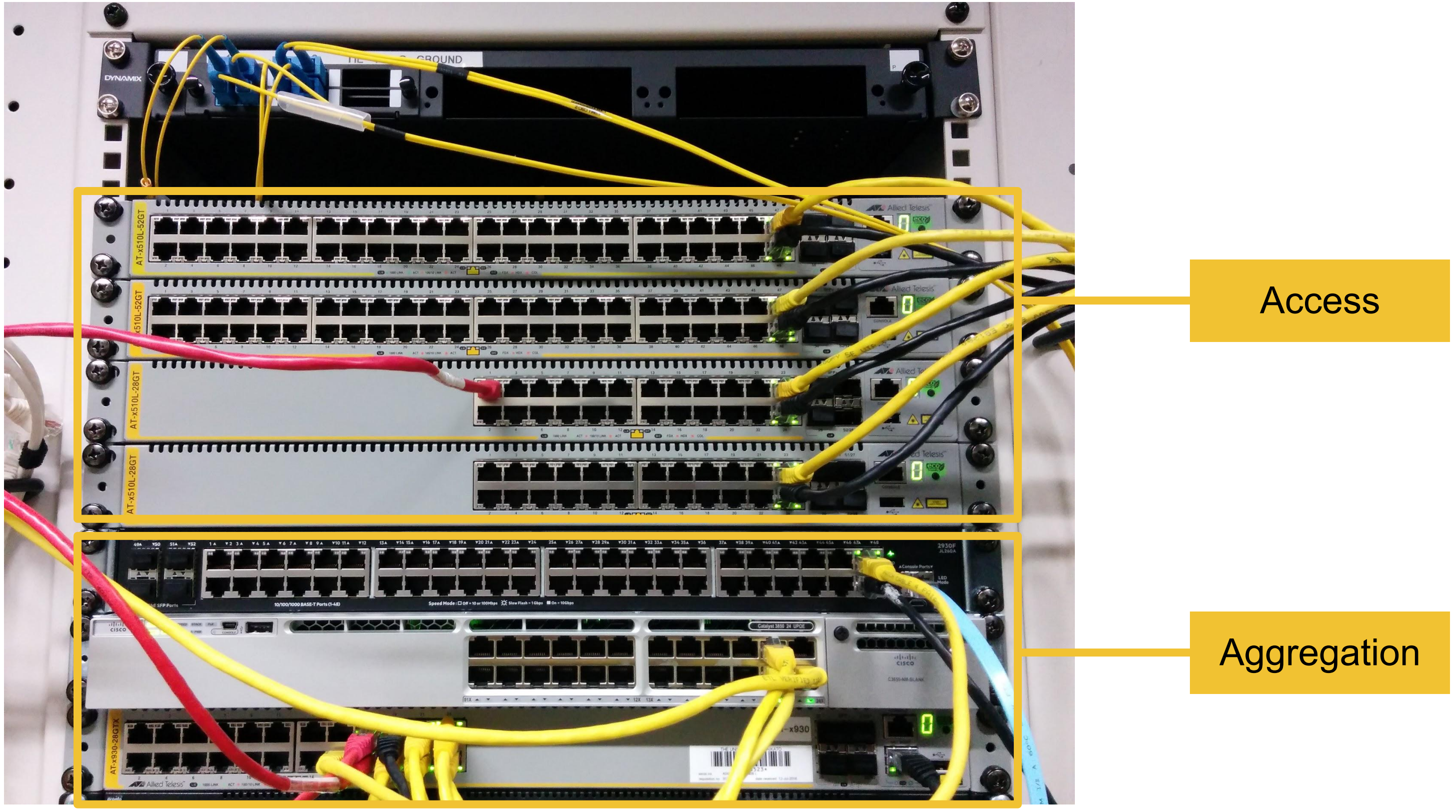


WAND redcables network

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



WAND redcables network

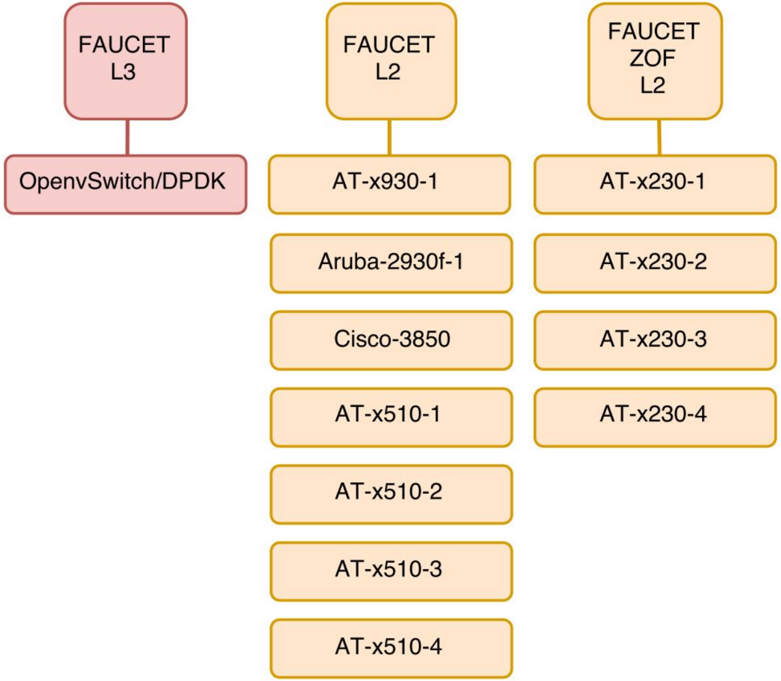


Access

Aggregation

Multiple FAUCET controllers

- Load balancing
- Redundancy
- Separation of duties

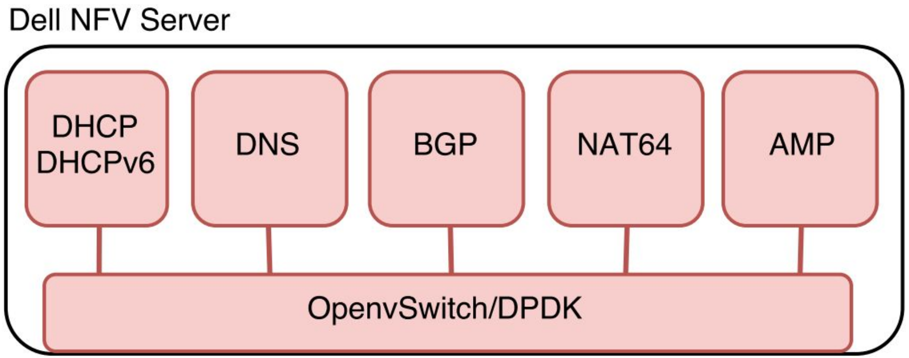


Ansible for network management

- 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
- 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

Network services

- Easily deploy services on network server as VMs
  - WAND’s AMP (Active network monitoring)
  - Catalyst’s Are We DDoS’d Yet? (DDoS monitoring)
  - Jool NAT64 (IPv6 only network)
  - isc-dhcp-server (DHCP and DHCPv6)
  - bird (BGP)



Testing and validation

- FAUCET includes a test suite
- Test suite performs 139 different test scenarios
  - Includes real topologies
  - Includes real traffic
- All commits into FAUCET are automatically tested with Travis
- We implement our own tests for features in use on redcables
- 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

- Do sanity checks as part of ansible playbook, only apply if things look good
- Ideally:
  - Integrate test suite with a Continuous Integration tool
  - Push on green!
- Currently:
  - I am the continuous integration tool
  - 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

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

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
```

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
```

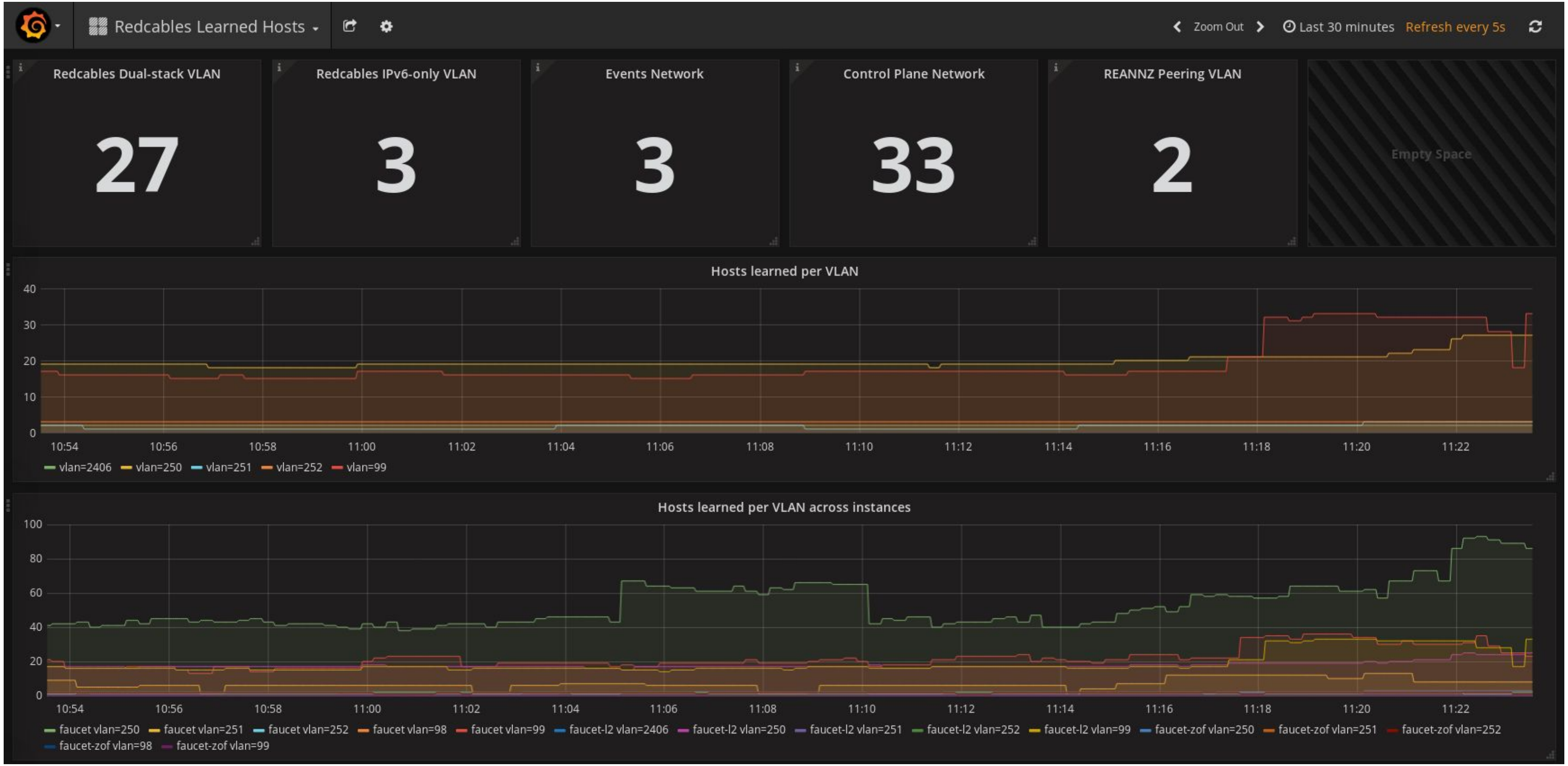
Out-of-line intrusion prevention system

- Use our policy based packet inspection to mirror packets to IPS
- IPS can write FAUCET ACLs to drop traffic
- IPS is no longer bottleneck or single point of failure

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



Redcables Dual-stack VLAN: 27

Redcables IPv6-only VLAN: 3

Events Network: 3

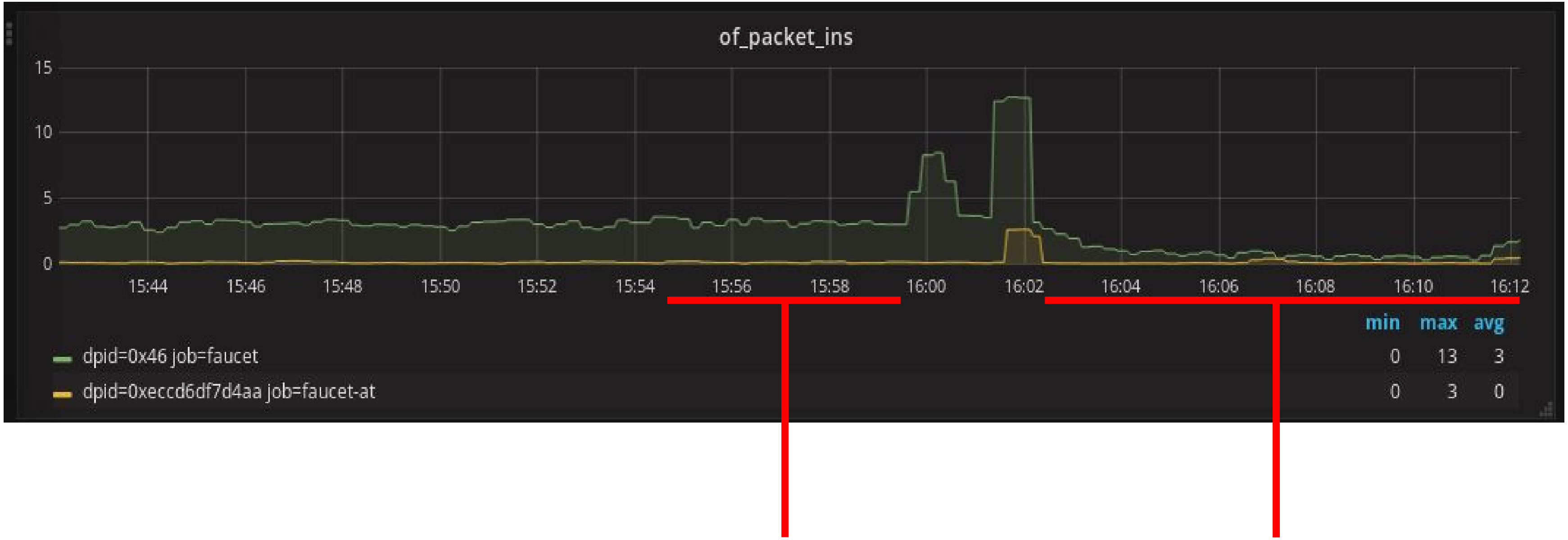
Control Plane Network: 33

REANNZ Peering VLAN: 2

Hosts learned per VLAN

Hosts learned per VLAN across instances

Real-time graphs of OpenFlow control channel



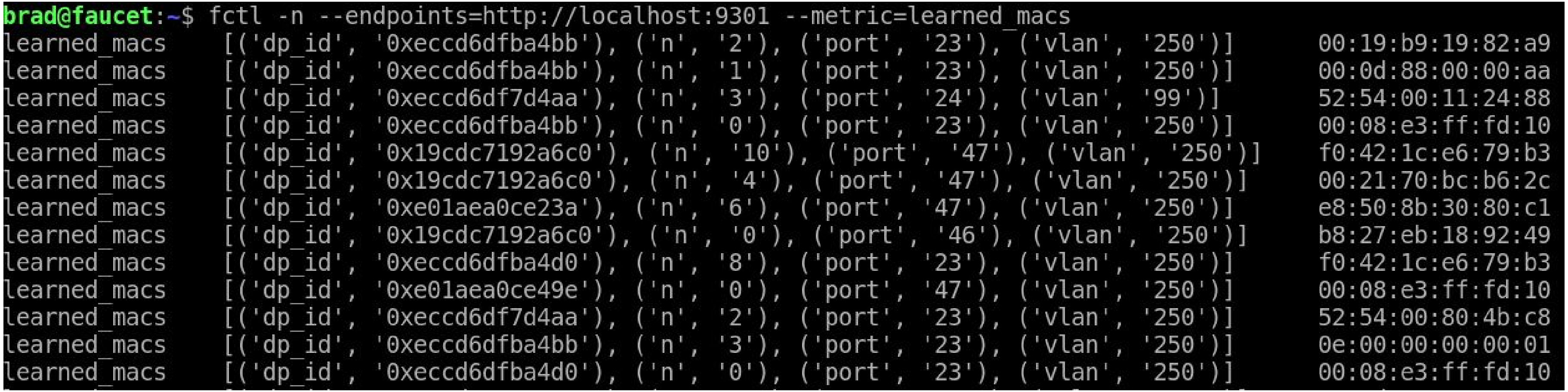
	min	max	avg
🟩 dpid=0x46 job=faucet	0	13	3
🟨 dpid=0xeccd6df7d4aa job=faucet-at	0	3	0

Old Code

New Code

Network visibility - prometheus

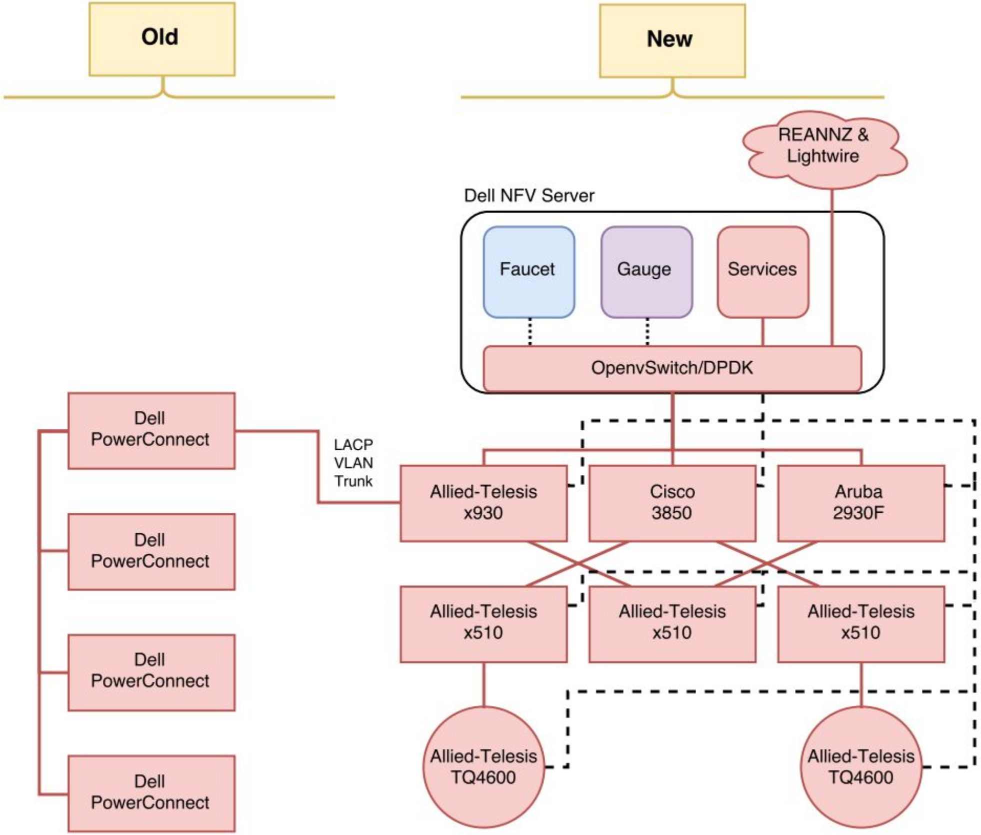
- Central database of all knowledge of network
- Where the MAC table?
    - FAUCET includes a centralised, time-series, queryable database of learned MACs
- FCTL tool for querying information



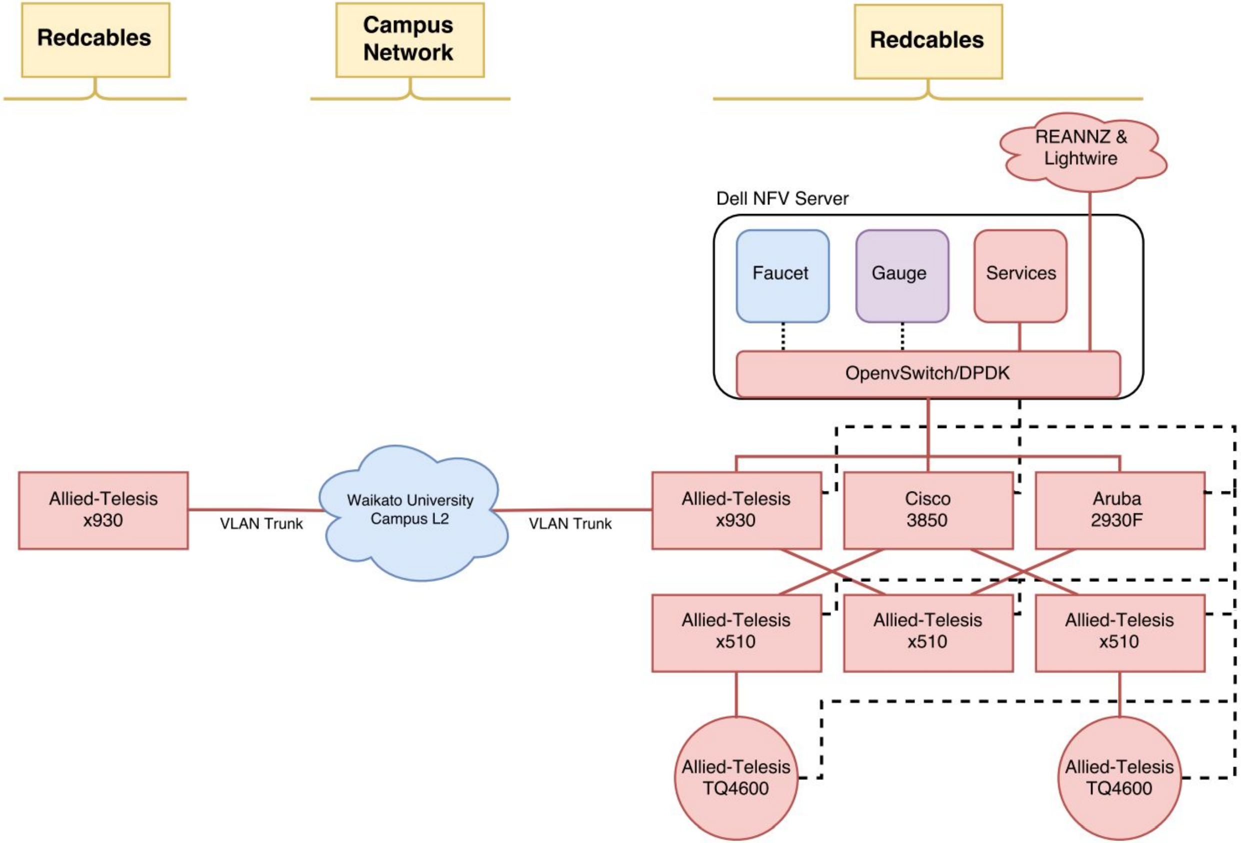
```
brad@faucet:~$ fctl -n --endpoints=http://localhost:9301 --metric=learned_macs
learned_macs    [('dp_id', '0xeccd6dfba4bb'), ('n', '2'), ('port', '23'), ('vlan', '250')]     00:19:b9:19:82:a9
learned_macs    [('dp_id', '0xeccd6dfba4bb'), ('n', '1'), ('port', '23'), ('vlan', '250')]     00:0d:88:00:00:aa
learned_macs    [('dp_id', '0xeccd6df7d4aa'), ('n', '3'), ('port', '24'), ('vlan', '99')]      52:54:00:11:24:88
learned_macs    [('dp_id', '0xeccd6dfba4bb'), ('n', '0'), ('port', '23'), ('vlan', '250')]     00:08:e3:ff:fd:10
learned_macs    [('dp_id', '0x19cdc7192a6c0'), ('n', '10'), ('port', '47'), ('vlan', '250')]   f0:42:1c:e6:79:b3
learned_macs    [('dp_id', '0x19cdc7192a6c0'), ('n', '4'), ('port', '47'), ('vlan', '250')]    00:21:70:bc:b6:2c
learned_macs    [('dp_id', '0xe01aea0ce23a'), ('n', '6'), ('port', '47'), ('vlan', '250')]     e8:50:8b:30:80:c1
learned_macs    [('dp_id', '0x19cdc7192a6c0'), ('n', '0'), ('port', '46'), ('vlan', '250')]    b8:27:eb:18:92:49
learned_macs    [('dp_id', '0xeccd6dfba4d0'), ('n', '8'), ('port', '23'), ('vlan', '250')]     f0:42:1c:e6:79:b3
learned_macs    [('dp_id', '0xe01aea0ce49e'), ('n', '0'), ('port', '47'), ('vlan', '250')]     00:08:e3:ff:fd:10
learned_macs    [('dp_id', '0xeccd6df7d4aa'), ('n', '2'), ('port', '23'), ('vlan', '250')]     52:54:00:80:4b:c8
learned_macs    [('dp_id', '0xeccd6dfba4bb'), ('n', '3'), ('port', '23'), ('vlan', '250')]     0e:00:00:00:00:01
learned_macs    [('dp_id', '0xeccd6dfba4d0'), ('n', '0'), ('port', '23'), ('vlan', '250')]     00:08:e3:ff:fd:10
```

Interoperability - migration

- Old and new networks are connected via VLAN trunk
- Clients can use new network for L3 even on old network



Interoperability - campus network



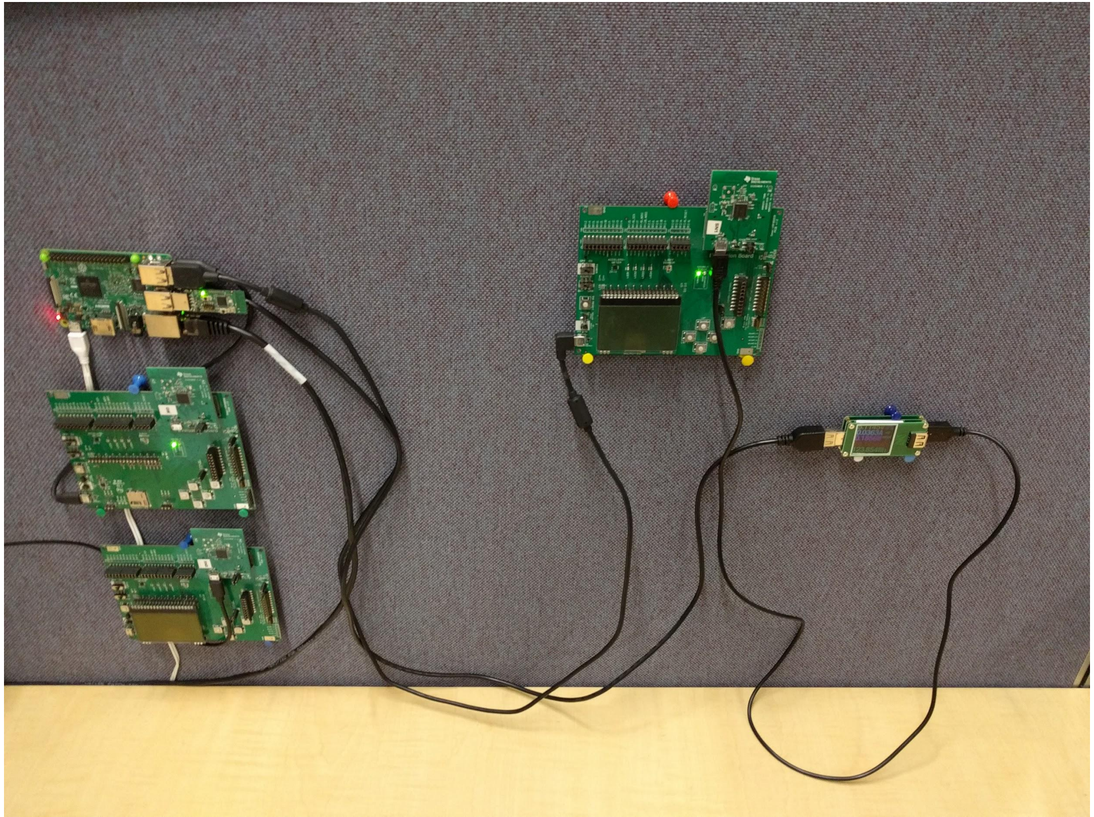
FAUCET makes networking fun

- We discovered we could use this network for new purposes
  - Easily build bespoke deployments
  - Temporary deployments for events becomes easier
- Switches run the exact same config
- We just need to deploy them somewhere and modify the controller

What does a redcables user look like?

802.15.4 IoT testbed

- Route /64 of IPv6 towards a raspberry pi gateway
- TI dev boards running OpenThread

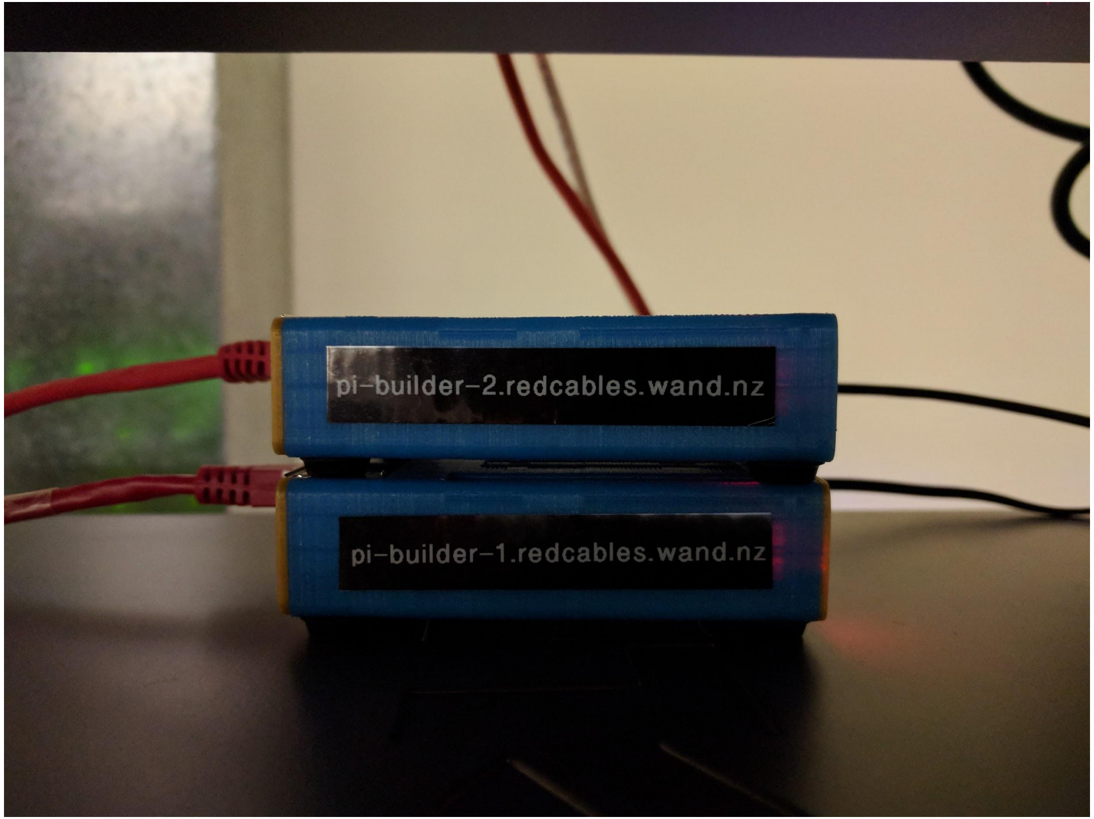


https://github.com/wandsdn/redcables-ansible/commit/edf3ff0da09647978bb91edb5ea33908202d79d1

What does a redcables user look like?

Raspberry Pi Build Farm

- Builds ARM docker containers for FAUCET
- Runs as redundant cluster - each node connected to different aggregation switch

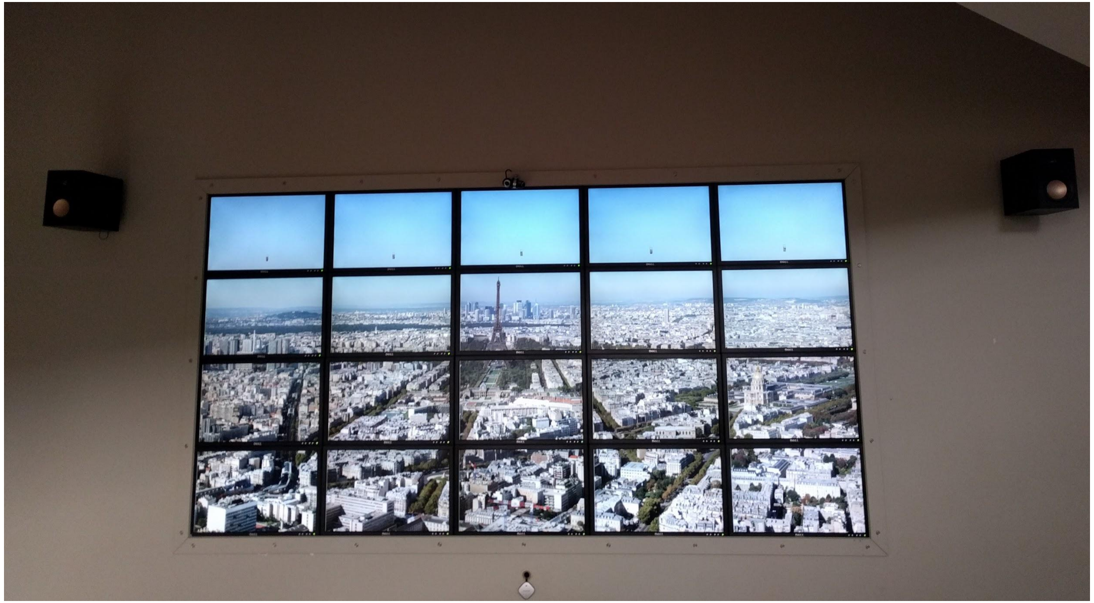

pi-builder-2.redcables.wand.nz
pi-builder-1.redcables.wand.nz

https://github.com/wandsdn/redcables-ansible/commit/f6011c3de7f87dd2df484a3fa2e630ec04bd7c71

What does a redcables user look like?

Display Wall

- 35 Megapixels
- In a public place on campus for marketing/art installations



https://github.com/wandsdn/redcables-ansible/commit/b9f4a679d1b0d8609bc0924d0989c37ef83cbf7d

What does a redcables user look like?

GovHack 2017

- Public data government hackathon
- 48 hours over a single weekend

https://github.com/wandsdn/redcables-ansible/commit/20b731fbef018e93b1473e662d4dae33ff1a14ae

Other deployments

- NZNOG 2017
  - 150 network engineers using our OpenFlow WiFi
  - Similar architecture to redcables
  - 1G WAN
  - Success!
  - Allowed us to collect a lot of data that we used to scale FAUCET
  - Open source: https://github.com/wandsdn/conference-sdn-nfv-network
- NZNOG 2018
  - 10G WAN
  - Watch this space

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

Questions?