Using an Open Source SDN Controller to Deploy a Multi-Terabit/s Production Network

Brad Cowie
University of Waikato
Introduction to faucet

- Lightweight Open Source SDN controller
- OpenFlow v1.3
- Supports Layer 2 and Layer 3
- Policy driven approach to extensibility
Why use OpenFlow in 2019?

- OpenFlow is Software-defined Networking
- NETCONF/YANG/etc is software-deployed Networking
- Decent support by many hardware vendors
Why use faucet?

- Low system resource requirements
- Uses multi-table OpenFlow pipeline
- Multi-vendor without drivers
- Well tested, production quality
- Automated device qualification
- High Availability through idempotency
- Controller is not in the forwarding path
Faucet configuration

- YAML configuration file
- Represents topology & features of network
- Faucet is idempotent
  - Give 2 controllers same configuration and they will configure the network the same

```yaml
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"
        native_vlan: office
      2:
        description: "host2"
        native_vlan: office
```
SC18

- Annual SuperCompute Conference in the US
- 30th anniversary was November 2018 in Dallas, Texas
- “SCinet” network built each year to power conference
  - One of the world’s fastest temporary networks
- Faucet was deployed on SCinet for SC18
SCinet 2018

- 4.02 terabit/s connectivity
- $52 million in hardware
- 4.25 tons of equipment
- 225 volunteers
Let’s take a step back

- How did we get involved?
- Approached conference committee April 2018
- Proposed a mutually beneficial faucet deployment
- Our proposal was accepted and work begun...
Phase 1: Planning

- Months of video conferences from NZ to USA/Europe
- Faucet SCinet team distributed between
  - Hamilton, New Zealand
  - Wellington, New Zealand
  - Berkeley, California, USA
  - Champaign, Illinois, USA
  - Dallas, Texas, USA
Scope for faucet deployment

- Faucet will run in parallel with regular network
- Peer with regular SCinet core and advertise/receive routes
- Will provide percentage of booths on show floor Internet access
- Each booth has separate VLAN and subnet
- Booth can be connected at 1GbE, 10GbE or 100GbE
- Customer information comes from Django web app
Initial design

- Faucet needs physical presence in
  - Core layer (NOC)
  - Access layer (DNOC)
Phase 2: Source hardware

● Approach hardware vendors and ask nicely for hardware
● Thanks Allied-Telesis, Cisco and NoviFlow!
  ○ 2 devices each
● Coordinate shipping to test lab in Berkeley, California
Wait

In transit

TO

BERKELEY, CA US
Iterate on network design

Port counts:
- 75x 100GbE
- 146x 10GbE
- 48x 1GbE

9.008 Terabit per second
Network functions

Faucet will provide:
- VLANs
- IPv6
- Router Advertisements
- Inter-VLAN Routing
- Network Security Policy

NFV Services:
- DHCP
- BGP
Brief introduction to P4

- P4 program
- P4 architecture model
- P4 compiler
- Control plane
  - Runtime API
  - P4 Runtime
  - Control signals
- Data plane
  - Tables
  - Memory
- Load API
- Load dataplane runtime
Combining P4 and OpenFlow

1. Write OpenFlow as P4 application
2. Compile application with Barefoot’s P4 compiler to run on Tofino ASIC
3. Faucet uses OpenFlow as runtime API (instead of P4 Runtime) to push rules to P4 ASIC

Feature written by NoviFlow and available in their NoviWare Network OS
Development

- Make sure required features work by adding integration tests
- Built a scale model back in New Zealand
- Started work on automation
Automation

- Define a set of tasks once, can then repeat tasks at no cost
- Applies perfectly for network service definitions
- Faucet SCinet deployment heavily used ansible
## Automation

<table>
<thead>
<tr>
<th>SCinet Customer DB</th>
<th>Python</th>
<th>Ansible</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| Source of truth for network                            | Use django models to dump all customer connections into YAML | Take YAML and use jinja2 templates to output configuration | • Faucet  
• DHCP  
• BGP  
• Linux Interfaces |

<table>
<thead>
<tr>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>290 lines of YAML</td>
</tr>
<tr>
<td>180 lines of ansible</td>
</tr>
<tr>
<td>260 lines of jinja2 templates</td>
</tr>
<tr>
<td>1200 lines of config</td>
</tr>
</tbody>
</table>
Available on GitHub

● Ansible code
  ○ https://github.com/wandsdn/sc18-ansible

● Generated faucet configuration
  ○ https://github.com/wandsdn/sc18-faucet-configs
Delivered
Friday 8/10/2018 at 9:02 am

TO
BERKELEY, CA US
Phase 3: Testing

- Use ansible to deploy configurations
- Throw a lot of traffic at it
- Network namespaces are great!
  - Can easily spin up thousands of network namespaces on a single machine
  - Use a network namespace to represent fake booth
- Write some bash scripts to manage
  - Test reachability (`onall.sh pingall.sh`)
  - Test throughput (`randomiperf.sh`
Ship network off to conference venue

Delivered
Friday 10/19/2018 at 10:02 am

TO
DALLAS, TX US
Phase 4: Staging

- SCinet equipment is staged away from show floor
- Network is turned on for the first time
- Interop testing and diagnosing problems
Faucet staging team photo after network booted up without issue
Phase 5: Show Setup

- Customer database is finalised
- Trigger automation to deploy configuration
- Work with booths as they are built to ensure access works
DNOC

Faucet switch

Regular switch

Optical transport
Load test

- Borrow some linux boxes full of 100G interfaces to load test network
Phase 6: Show Time

- Look for issues
  - Watch tickets queue
  - Watch monitoring

- Fix issues
  - Deploy light meters
  - Swap optics
  - Rolling fibres
Phase 6: Cleanup

- 7 months of planning
- 3 weeks of testing
- 3 weeks setup
- 1 day of tear down
What did we learn?

- Automating the network services that 90% of customers use helped operators have more time to spend working on the interesting 10% of customers.
- Faucet allowed us to effectively ignore the underlying network equipment.
- Proved 10,000 lines of python can easily handle many terabit/s.
Questions?

https://faucet.nz
@faucetsdn

brad@waikato.ac.nz
@nzgizmoguy