

Challenges of supporting SDN in production

A.J. Ragusa - GlobalNOC @ Indiana University



OESS / FSFW / AL2S

- What we have in production today
- OESS - Open Exchange Software Stack
 - Point-to-Point / Multi-Point VLAN provisioning Service 100% openflow
- FSFW - FlowSpace Firewall
 - OpenFlow Network Slicer - Network hypervisor allowing multiple controllers to talk to a single set of switches without letting any one controller effect a change that will affect another.

The screenshot shows the OS³E (Open Science, Scholarship & Services Exchange) network management interface. The top navigation bar includes "OS³E" and "The Open Science, Scholarship & Services Exchange". Below this, there are tabs for "Active VLANs", "Network Status", "Available Resources", "Users", and "Actions". The main content area is divided into three sections: a map of the United States showing network links, a "Link Status" table, and a "Circuit Status" table. The "Link Status" table lists links like "I2-CLEV-STAR-100GE-07736" with a status of "up". The "Circuit Status" table lists circuits like "I2-NEWY32A0A-WASH-VLAN-09178" with a status of "primary".

Link	Status
I2-CLEV-STAR-100GE-07736	up
I2-LOSA-SUNN-100GE-07755	up
I2-DENV-KANS-100GE-07746	up
I2-HOUH-TULS-100GE-07751	up

name	Status
I2-NEWY32A0A-WASH-VLAN-09178	primary
I2-NEWY32A0A-CLEV-VLAN-09177	primary
I2-ATLA-WASH-VLAN-09179	primary
I2-CHIC-CLEV-VLAN-09180	primary
I2-ATLA-CHIC-VLAN-09181	primary
I2-CHIC-KANS-VLAN-09183	primary
I2-KANS-SALT-VLAN-09185	primary
I2-LOSA-SALT-VLAN-09188	primary
I2-SALT-SEAT-VLAN-09186	primary
I2-LOSA-SEAT-VLAN-09187	primary
I2-ATLA-HOUS-VLAN-09182	primary
I2-HOUS-KANS-VLAN-09184	primary
I2-LOSA-HOUS-VLAN-09189	primary
I2-CLEV-WASH-VLAN-09178	primary

Writing custom controllers

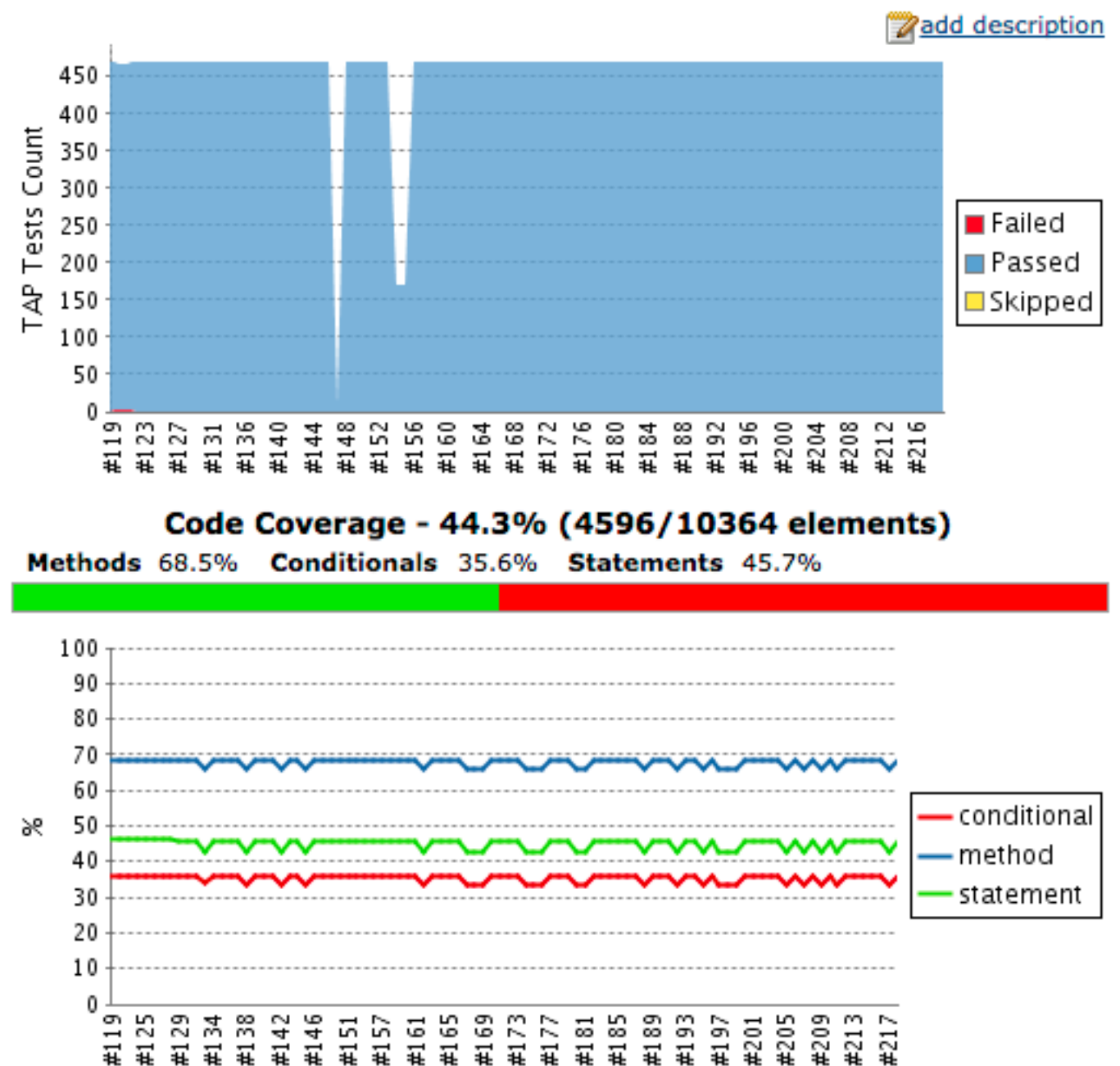
○ You are now writing Routing Engine Code

- Automated Unit testing
- Automated Systems testing
- Automated Build Process
- Scale testing

○ Code Coverage Analysis

○ Strict release workflow

○ Code Review (not by the developers)



Running custom controllers

◎ Operational workflow changes

- software developers, systems engineers, and network engineers need to work closely together to troubleshoot issues
- Determining the source of the problem can be difficult:
 - Controller / Application
 - Device Hardware
 - Device Software

◎ Postmortems after incidents

- Bugs/Problems are going to happen
- Review changes in policies to reduce down time
- Add policies to prevent similar issues from occurring

Running custom controllers

◎ Network Operator is now the System Integrator

- Easy to underestimate the amount of time to test vendor code changes
- People (Software Developers) who are not used to dealing with devices must learn to work with them

◎ Build Once and deploy is un-realistic

- every vendor update requires re-testing against the current released version
- changing any part of the stack requires re-testing the entire stack

◎ Need additional troubleshooting tools

- TCPDump of the control channel (your vendors are going to expect it)
- additional vendor commands to see flows as programmed in hardware

Process Improvements

◎ Every time there is an issue ask these questions

- What was the issue
- What was affected
- How did it get past testing
- How did the issue get introduced
- What could we do to prevent this in future

◎ Implement any changes that would have prevented it in the future

- Eg. Instead of adding unit tests for a missed feature, add policy for unit tests for ALL new features

Troubleshooting Outages

- © New untrusted technology always gets the blame (rightfully so)
 - controllers need to provide a list of flows they expect on the system
 - training of Network Engineers / System Engineers / Software Engineers for troubleshooting is very important
 - Must have experience running the entire stack in test before in production
 - provides experience troubleshooting issues
 - enhances troubleshooting capabilities
 - will drive future development to aid operations
 - Its not always the new technology!
 - it is possible to spend lots of time troubleshooting an issue that is thought to be in the SDN stack but the problem is in a traditional network

Troubleshooting Outages

◎ Roles get blurred

- Software Engineers need access to run commands on devices to see what the device is doing
- Network Engineers need access to logs from the controller to determine what it is attempting to do
- Improved communications between groups is needed
 - must speak the same language
 - must listen to each other
- Over time these roles might become more converged

◎ Need more troubleshooting tools from vendors (not hidden commands)

Controller/App Vendors

© Lots of people are writing controllers/apps... but

- Where is the long term maintenance release
- Where/How do you get support
- Where is the documentation?
- not many are thinking about operations (packaging, logging, troubleshooting)
- Mininet != a sufficient testing platform for production controllers
 - does not account for subtle differences between vendors implementations
 - DPIDs don't look the same
 - Port IDs are different

© Where is the RedHat/CentOS/Ubuntu of controllers



Controller/App Vendor

- ⦿ Added change management process to control risk
 - Testing
 - Mininet + device testing
 - Testing in a stage environment with an almost exact copy of the network
 - Testing takes 2-3 weeks to complete
 - Time to deploy a new controller is slow
- ⦿ Constrains nimbleness (us and vendor)
- ⦿ Constrains pace of innovation (us and vendor)
- ⦿ This is not as good as we hoped with the initial vision of OpenFlow

Complexity and Vagueness

- ◎ OpenFlow is not the universal language (or at least not treated that way)
 - In many cases the Spec is vague leaving it up to vendors on how to implement
 - People are adding layers of abstraction to manage this
 - Making the controller the commonality instead of the protocol
- ◎ We are not seeing OF 1.3 fix this
 - Vendors are not implementing (at least not quickly)
 - Still missing many of the required features
 - not even thinking about optional features

Security with SDN

- ◎ Proper software / systems architecture will protect against many attack possibilities
 - Separation of privileges
 - Eg. public web-service should be different process than what talks OpenFlow
- ◎ Control plane inaccessible from public internet
- ◎ It took years to develop trust with the core routing today
 - what is the right way to gain the same assurance for SDN
- ◎ FSFW - resource protection difficult to perform
 - Valid but different messages can trip up vendor hardware
 - volumetric attacks are interesting and difficult to protect against

Security with SDN

© Plenty of opportunities to apply SDN towards Security problems

- SciPass - OpenFlow load balancer and ScienceDMZ
- Remote Triggered black hole
- Dynamic Honeypot