Disaggregating the SDN Control Plane

David Bainbridge
Ciena Corporation

Open Networking Summit – Europe
September 25 - 27, 2018
For millions of happy users all over the world, the iPhone is fantastic just as it is. It's beautiful, elegant and easy to use, and there are thousands upon thousands of apps and oodles of content for them to choose on the App Store.

And then there are the people who aren't so happy. People who want to break free of the restrictions they believe Apple has forced upon us all - from the default apps that come with iOS to the fact that its underlying structure cannot be customized by individual programmers, third-party developers or even users themselves.”

From macworld.co.uk article by Rob Mead-Green, April 13, 2017
Jail Breaking SDN

Process/Language/Scale/Failure Boundary

SDN Application

Intent Management

Prediction & Resolution (ML)

Inventory

Flow Management

Telemetry

Protocol Management

Message Distribution

SDN Application

Intent Management

Prediction & Resolution (ML)

Inventory

Flow Management

Telemetry

Protocol Management

Copyright © Ciena Corporation 2018. All rights reserved. Confidential & Proprietary.
Control Plane Disaggregation

- **OFTeer**
  - Proof of concept, not the end destination

- **Control Plane Re-envisioned**
  - A set of cooperating services that provide complete control plane
    - Protocol connection maintenance
    - Inter-component communication (messaging)
      - Inventory
      - Flow management
      - Intents
      - Etc.

- **SDN Applications**
  - Peers with other services

Diagram:
- SDN Application
- Intent Management
- Prediction & Resolution (ML)
- Inventory
- Flow Management
- Telemetry
- Protocol Management
- Message Distribution
- Minimal Viable Controller Components
Scale – From the Simple

- **Replicate Individual Components**
  - Performance
  - Resiliency

- **Stateful v Stateless Components**
  - Components should be stateless
    - (recoverable state OK)
  - Connection based protocol handlers are stateful
  - Could leverage capabilities from projects such as VOLTHA
Scale – To the Complex

- Everything from simple model … plus
- Information sharding / distribution
  - Single/Multiple geography based
- Centralized Control v. Distributed Execution
  - Handle decisions closes to source / target
  - Set policy globally
- The complex model should also work for simple deployments
- Use Internal, Commercial, Existing, and/or Innovative Tools to Manage
Failure Boundaries

- **Fast fail-over**
  - Kill and respawn

- **Redundancy**
  - Load balance between stateless instances

- **External and redundant state**
  - Component may cache
  - Multiple fail-over schemes possible with state

- **Protocol connections may be stateful**
  - Fail-over possible, but more complicated

- **Health monitoring**
  - Component report health to external management system
I want to write an SDN application once and use it across different SDN controllers.
Writing a Multi-Controller SDN Application Today

1. **Write and test it in one controller**
   - Learn the controller’s application environment
   - Learn the controller’s internal model
   - Learn the controller’s internal packet manipulation library
   - Learn the controller’s internal packet emitting library
   - Learn the controller’s selected language
   - All this and it scales / fails as part of the controller process

2. **Repeat 1 for each desired controller**

OR, you could **write an event externalization solution for each controller**
   - Still requires per controller modifications
OFtee, An Experiment in Control Plane Disaggregation

What if you could write an SDN application completely independent of any given controller

- Use 3rd party, open source packet libraries
- Select a language that best fits the requirements
- Scale & Fail the SDN application independently of the controller and other SDN applications

Don’t address all disaggregation issues

- Keep scope limited
- More a feasibility study than a production solution
The Inspiration

**Linux tee**

- **INPUT**
- **STDOUT**
- **FILE**
Adapting to Open Flow

Open Flow tee

OF DEVICE → SDN APPLICATION → CONTROLLER
Adapting to Open Flow

Open Flow

OF DEVICE

CONTROLLER

SDN APPLICATION

SDN APPLICATION

SDN APPLICATION
### What it does

**To the controller, OFtee is the device; To the device, OFtee is the controller**

#### Intercepts the Open Flow communications
- Processes *Features Reply* messages to sniff DPIDs
- Processes *Packet In* messages to forward to external processes
- Injects *Packet Out* messages to forward to the OF devices
- Everything else is pass through

#### Essentially a match / action soft switch whose only supported action is “packet out” to an external application
- Supports ethernet type matches [today]
- Support HTTP "packet out" [today]
- Other transports being considered: TCP socket, Kafka, GRPC
What it doesn’t

- **Packet In flow provisioning**
  - The switch still requires flows to be pushed that “packet in” the desired packets to the controller
  - **Umbrella: A Unified Software Defined Development Framework**

---

**Umbrella: A Unified Software Defined Network Programming Framework**

- **Motivation**
  - Increase portability of SDN applications and services across heterogeneous SDN controllers, making it easy to compare results and application performance on various controllers.

- **Main Design Goals:**
  - Provide a new set of abstractions for SDN applications, keeping the abstractions independent of the NB APIs that specific SDN controllers offer.
  - Create a framework that offers increased scalability by following a hybrid approach that incorporates a reactive paradigm for writing applications that manage SDN networks as well as the traditional proactive paradigm.
  - Reduce programming complexity by providing a software defined network programming framework that, allows a programmer to write SDN applications without requiring a programmer to master low-level details of specific SDN controllers, and avoids locking an application to a specific controller.

- **More info:**
  - [http://umbrella-project.org](http://umbrella-project.org)
  - CS Systems Research Group, Purdue University, [https://systems.cs.purdue.edu](https://systems.cs.purdue.edu)
Complexities

Inserting Packet Out messages into communications from device to controller
• OFtee must process Open Flow connection as messages

Context (DPID) and Port information for Packet In Messages
• Open Flow Packet In doesn’t container DPID information
• OFtee adds context (DPID + port) as it forwards to external application

Barrier Messages
• Ignored
• Need to better understand what is acceptable behavior

TLS Connections
• Not currently supported

Performance
• Not yet validated
Demonstration

- All components invoked as Docker containers under a single node Docker Swarm instance
- Inject OVS port into client container using ovs-docker
- Use controller specific REST interface to add flows to device for Packet In relevant packets (DHCP and EAPOL)*
- EAPOL – written in Go language using 3rd party Open Flow and packet libraries
- DHCP Relay – written in Python using Scapy
- Source: https://github.com/dbainbri-ciena/oftee_workspace

* More on this later
Demonstration
Well and Good, but …

A disaggregated control plane still needs to be built

And we don’t want

https://xkcd.com/927/
Links

OF Tee on GitHub: https://github.com/ciena/oftee

OF Tee workspace on GitHub: https://github.com/dbainbri-ciena/oftee_workspace

Demonstration Video: https://youtu.be/QzDDe59MCdw
Mèsi Anpil