AmLight’s SDN Looking Glass - An SDN centralized monitoring system

Jeronimo Bezerra  
<jab@amlight.net>  
Florida International University

Antonio Francisco  
<antonio@amlight.net>  
Academic Network at São Paulo
AmLight: a Distributed Academic Exchange Point

- Production SDN Infrastructure since Aug-2014
- NAPs: Miami, Brazil(2), Chile, and Panama
- Multiple 10G and 100G links
- Carries Academic and Commercial traffic
- Control Plane: OpenFlow 1.0 and 1.3
- Network Programmability/Slicing
- Inter-domain Provisioning with NSI
- 2000+ R&E institutions connected

AmLight’s SDN Looking Glass - An SDN centralized monitoring system - TNC 2018
Why troubleshooting a SDN network is so complex?

- OpenFlow has minimum support for troubleshooting
  - For instance, there are no special/reserved flow cookies

- Current SDN applications only consider network provisioning
  - Need for troubleshooting features only appears once things start falling apart

- Many academic papers suggesting solutions that do not fit in production
  - Highly dependent on the controller for actions
  - Heuristic and Machine Learning per unknown packet do not scale
• SDN concept itself makes things harder sometimes
  – Because datapaths have no intelligence at all, controllers always have to be involved
    • Creating scalability and timing issues
    • Making controllers more complex to operate and maintain

• Lack of support from vendors for useful OpenFlow features
  - Counters are not trustworthy, action OFPP_Table is not supported, …
Our 2016/2017 Vision

• **A single side-application for troubleshooting made sense:**
  – **Pros:**
    • Frees the provisioning developers to focus on provisioning
    • Avoids duplicated data when multiple SDN applications are running in production
    • Eases auditing
    • Centralizes all troubleshooting data, making it easier to correlate events
      – OpenFlow agent, NMS, SDN app, slicer and sniffer’s data are processed by just one entity
  – **Cons:**
    • Running OpenFlow applications in parallel is still a challenge:
      – ONOS deletes unrecognized flows, which makes design with other third parties controllers inviable.
      – OpenFlow roles and multiple Masters: multiple masters ack as standalone controllers and just a few controllers actually support all role modes (SLAVE, MASTER, NONE)
    • No East-West protocol standardized
      – Each SDN app will have to be customized to gather status and counters from a remote app
Our 2017/2018 Vision

A set of single micro-applications over the same framework makes more sense:

• Pros:
  – *Frees the provisioning developers to focus on provisioning*
  – *Avoids duplicated data when multiple SDN applications are running in production*
  – *Eases auditing*
  – *Centralizes all troubleshooting data, making it easier to correlate events*
    • *OpenFlow agent, NMS, SDN app, slicer and sniffer’s data are processed by just one entity*
  - No conflicts when running OpenFlow controllers in parallel
  - Easy integration with production applications in the same framework (Kytos, ONOS, ODL, etc.)

• Cons:
  - OpenFlow controller-dependent solution:
    • Kytos Napps only work with Kytos SDN controller, and ONOS apps only work with ONOS SDN controller
AmLight SDN Looking Glass

- **Central point for SDN monitoring:**
  - Centralizes all monitoring and troubleshooting information being slice/app-independent
  - Stores all statistical data (flow, ports, etc.) and OpenFlow messages into a persistent backend
  - Tracks real time OpenFlow control plane messages
  - Takes network snapshots: save the network state for future troubleshooting and capacity planning

- **Central point for SDN troubleshooting:**
  - Runs trace paths (”traceroute”) in both Data Plane and Control Plane
  - Sends alerts via e-mail and Slack
  - Provide REST to be used by external SDN apps, auditing tools, and external NMS
The AmLight SDN Looking Glass is composed of the following Napps:

- **Stats**: Gets flow stats from switches and store them.
- **Sniffer**: Monitors the Control Plane
- **SDNTrace CP**: Traces a path using flows acquired by the Stats app
- **SDNTrace DP**: Data Plane path tracing
- **Notifications**: Sends notifications when problems occur
- **Topology**: Discovers topology
- **Web UI for network visualization**
AmLight SDN Looking Glass

- Developed in Python 3.6
- Built as a series of Napps on top of Kytos SDN framework
- Uses Influxdb and Kytos/StoreHouse for persistence
- Uses Grafana and JavaScript D3 for visualization
- Supports both OpenFlow 1.0 and OpenFlow 1.3
- Integrates with the AmLight OpenFlow Sniffer
- Open Source/GPL
AmLight SDN Looking Glass Software Stacks

Controller: kytos
Web: Grafana
DB: influxdb
Messaging: slack

{RESTful API}
Topology Discovery

AmLight’s SDN Looking Glass - An SDN centralized monitoring system - TNC 2018
Switch info
# Flow Table

AmLight’s SDN Looking Glass - An SDN centralized monitoring system - TNC 2018

![Flow Table Image](image_url)
Tracing Paths

- Control Plane: uses flow entries collected through OFP_STATS_REQUEST
- Data Plane: sends Ethernet probes to match user flow entries (via OFPP_TABLE action)
- Manual tests are supported:
  - The SDN LG always compares the Control Plane results ("correct" path) with the Data Plane results ("actual" path detected).
  - In case CP and DP results are different, a notification is send to Admins
- Scheduled routine run CP path traces at every interval and randomly sample some circuits to test in the Data Plane:
  - Reason is to avoid overloading the switches' CPU
Trace Path Form
Trace Path

AmLight’s SDN Looking Glass - An SDN centralized monitoring system - TNC 2018
**SDN Looking Glass Compatibility**

**Evaluated OpenFlow Switches:**

<table>
<thead>
<tr>
<th>Switch Platform</th>
<th>Firmware Version</th>
<th>OpenFlow version</th>
<th>Control Plane Tracing</th>
<th>Data Plane Tracing</th>
<th>Topology Discovery</th>
<th>Flow Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td>OvS</td>
<td>2.9.0</td>
<td>1.3</td>
<td>SUPPORTED</td>
<td>SUPPORTED</td>
<td>SUPPORTED</td>
<td>SUPPORTED</td>
</tr>
<tr>
<td>Dell Z9100-ON</td>
<td>9.11(0.0P6)</td>
<td>1.3</td>
<td>SUPPORTED</td>
<td>NOT SUPPORTED</td>
<td>SUPPORTED</td>
<td>SUPPORTED</td>
</tr>
<tr>
<td>Corsa DP 2400</td>
<td>3.0.2 build 12</td>
<td>1.3</td>
<td>SUPPORTED</td>
<td>NOT SUPPORTED</td>
<td>SUPPORTED</td>
<td>SUPPORTED</td>
</tr>
<tr>
<td>Brocade MLXe</td>
<td>6.2.0b T177</td>
<td>1.3</td>
<td>SUPPORTED</td>
<td>NOT SUPPORTED</td>
<td>SUPPORTED</td>
<td>SUPPORTED</td>
</tr>
</tbody>
</table>
Two installation methods:
- Single server with all components (complex how-to)
- Docker containers:
  - Kytos
  - InfluxDB
  - Grafana
  - ofp_sniffer

GitHub: [https://github.com/amlight/sdn-lg](https://github.com/amlight/sdn-lg)
• **Next Steps:**
  - Monitoring OpenFlow Meters
  - Monitor Interface Queue utilization
  - Use SNMP to monitor CPU and memory utilization
  - Network configuration snapshots
    - Save the network state (flows, CPU/memory, topology, link status, last OF messages exchanged, etc.) for future troubleshooting
  - Small functionalities:
    - Work in fully passive mode (for parallel OF controller scenarios)
    - "Discover" topology through other approaches (file, API, etc.)
THANK YOU!

Jeronimo Bezerra
<jab@amlight.net>
Florida International University

Vinicius Arcanjo
<vinicius@amlight.net>
Rede Nacional de Ensino e Pesquisa

Antonio Francisco
<antonio@amlight.net>
Academic Network at São Paulo

Rogerio Motitsuki
<rogerio@amlight.net>
Academic Network at São Paulo

AmLight’s SDN Looking Glass - Central monitoring system for SDN
TNC 2018