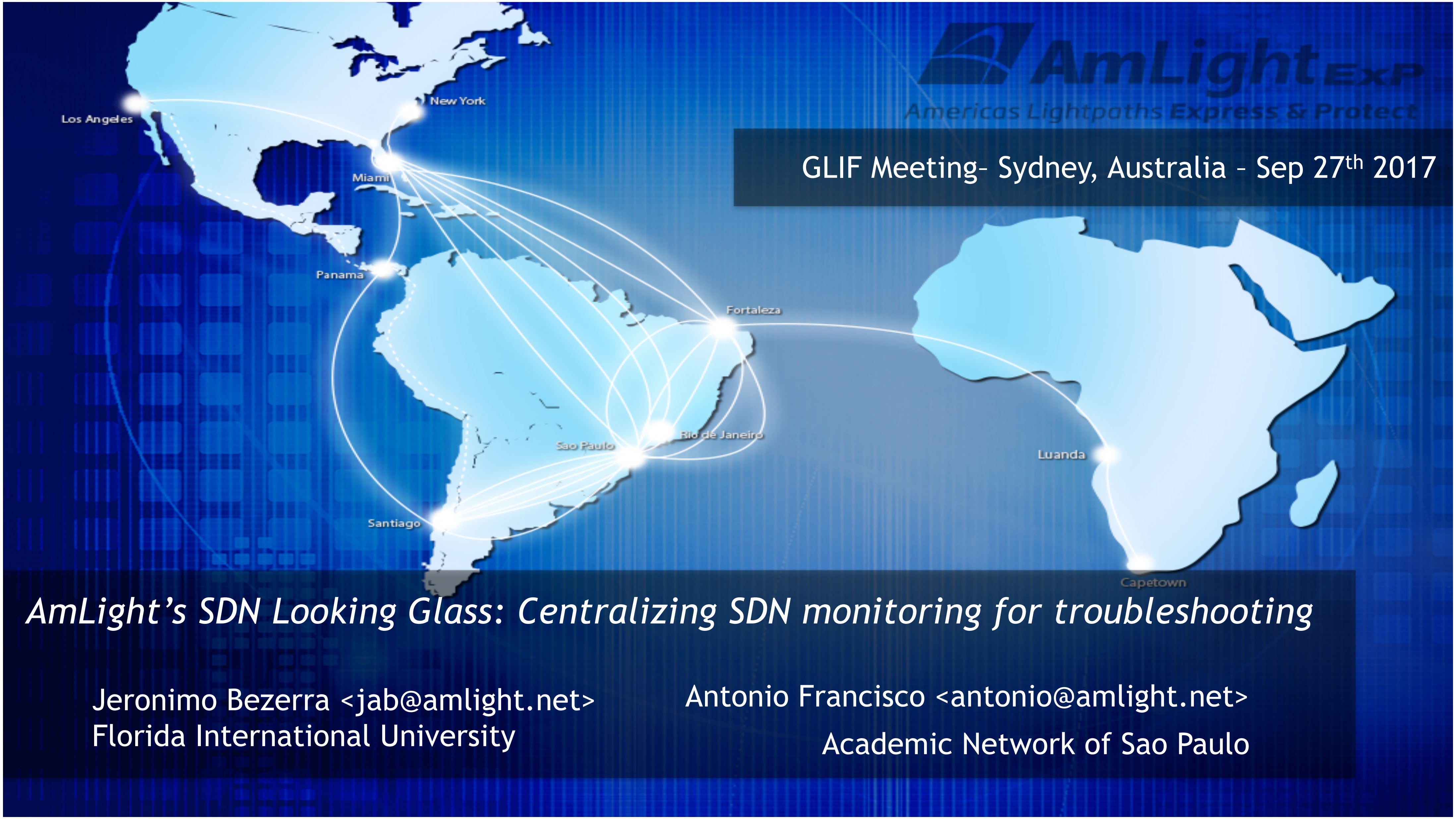
AmLight ExP
Americas Lightpaths Express & Protect

GLIF Meeting– Sydney, Australia – Sep 27th 2017



AmLight's SDN Looking Glass: Centralizing SDN monitoring for troubleshooting

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

Antonio Francisco <antonio@amlight.net>
Academic Network of Sao Paulo

Outline

- Why a tool just for troubleshooting?
  - Shouldn’t the SDN controller take care of it?

AmLight: a Distributed Academic Exchange Point

- Production SDN Infrastructure since Aug-2014
- Collaboration: FIU, NSF, ANSP, RNP, Clara, REUNA and AURA
- Includes two GLIF GOLEs: AMPATH (Miami) and SouthernLight (Brazil)
  - 4 x NAPs: Brazil(2), Chile and Panama
  - Multiple 10G and 100G links
  - 2000+ institutions connected
- Carries Academic and Commercial traffic
- Control Plane: OpenFlow 1.0 (with an OF1.3 overlay)
- Network Programmability/Slicing
  - OESS/NOX, ONOS, Kytos and Ryu
- NSI-enabled
- Currently, operating with more than a 1000 flow entries

SDN vs. Troubleshooting

Why troubleshooting a SDN network is so complex?

- OpenFlow has minimum support for troubleshooting
  - For instance, there are no special/reserved flow cookies
- Vendors assume that their job is done once OpenFlow agents are (partially) implemented
  - No passive OpenFlow connection supported by some vendors
  - No sFlow/Netflow supported for ”OpenFlow” entries in some vendors
  - Not all flow entries have reliable counters
  - Lack of visibility of what is happening inside the datapath’s OpenFlow agent
- Current SDN applications only consider network provisioning
  - Need for troubleshooting features only appears once things start falling apart

SDN vs. Troubleshooting (2)

- Most current SDN applications are developed only by software developers
  - Network Engineers could help with the monitoring/troubleshooting specification

- 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
  - Most solutions consider using Table 0 without addressing the table shift with the ”main” SDN app

- 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

SDN vs. Troubleshooting vs. Production Networks

- Troubleshooting production networks has different requirements
  - Has to be agile, least disruptive as possible and needs historical data
  - Tools have to be handy

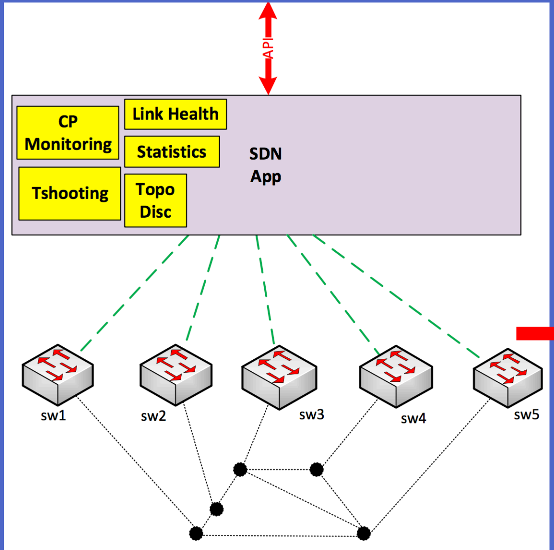
- More than ever, deep knowledge of the hardware and software platforms are required:
  - Use of ”hidden” commands and application logs become part of your routine

- A ”premium” support contract with hardware vendor is desired
  - Going through the Level 2 TAC team every time will lower your will to live and increase the network recovery time

Our Vision

- A single side-application for troubleshooting 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
  - Cons:
    - Parallel applications is still a challenge
      - Not OpenFlow Equal/Equal support by some vendors and OpenFlow controllers
      - Some apps delete flows they don't recognize (!)
    - No East-West protocol standardized
      - Each SDN app will have to be customized to gather status and counters from a remote app
    - Another application to maintain

Currently


CP Monitoring

Link Health

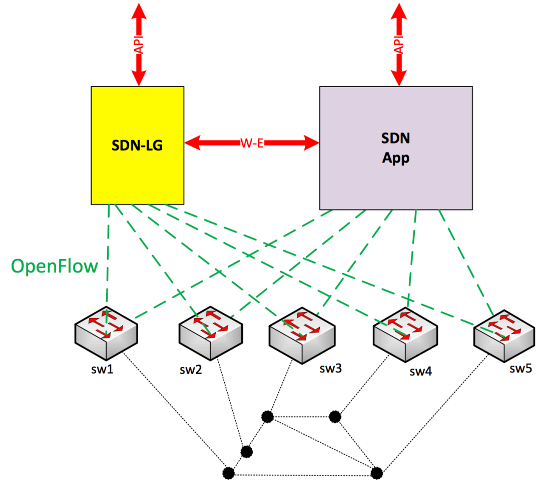
Statistics

Tshooting

Topo Disc

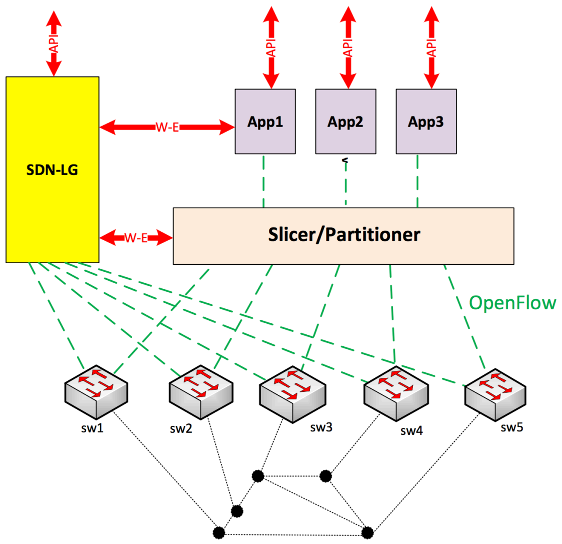
SDN App

Next Phase


SDN-LG

SDN App

OpenFlow


SDN-LG

App1

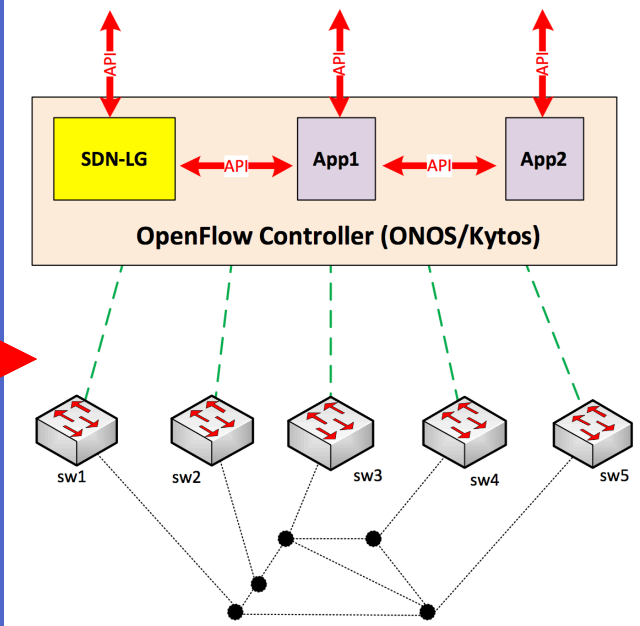
App2

App3

Slicer/Partitioner

OpenFlow

Goal


SDN-LG

App1

App2

OpenFlow Controller (ONOS/Kytos)

AmLight SDN Looking Glass

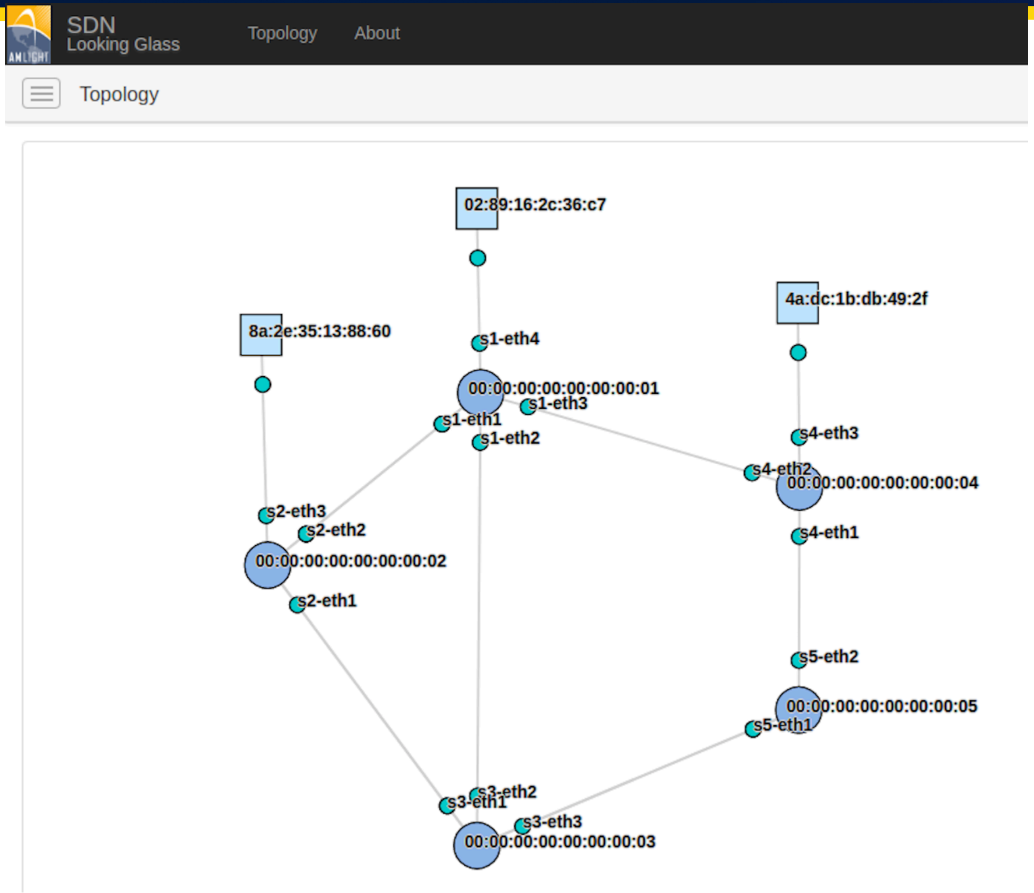
- Central point for SDN troubleshooting:
  - Centralizes all monitoring and troubleshooting information being slice/app-independent
  - Stores all statistical data (flow, ports, etc.) and OpenFlow messages into a persistent backend
  - Tracks real time OpenFlow control plane messages
  - Tracks non-OpenFlow information (for instance, CPU utilization)
  - Runs trace paths (”traceroute”), including inter-domain
  - Sends alerts via e-mail and Slack
  - Takes network snapshots: save the network state for future troubleshooting and capacity planning
  - Provide REST to be used by external SDN apps, auditing tools and external NMS
  - Supports active and passive topology discovery (LLDP or input file)
- Development team: FIU and ANSP
- Collaboration with State University of Sao Paulo / Kytos developers
- Launch date: Internet2 Technology Exchange 2017 (October 2017) version 0.1

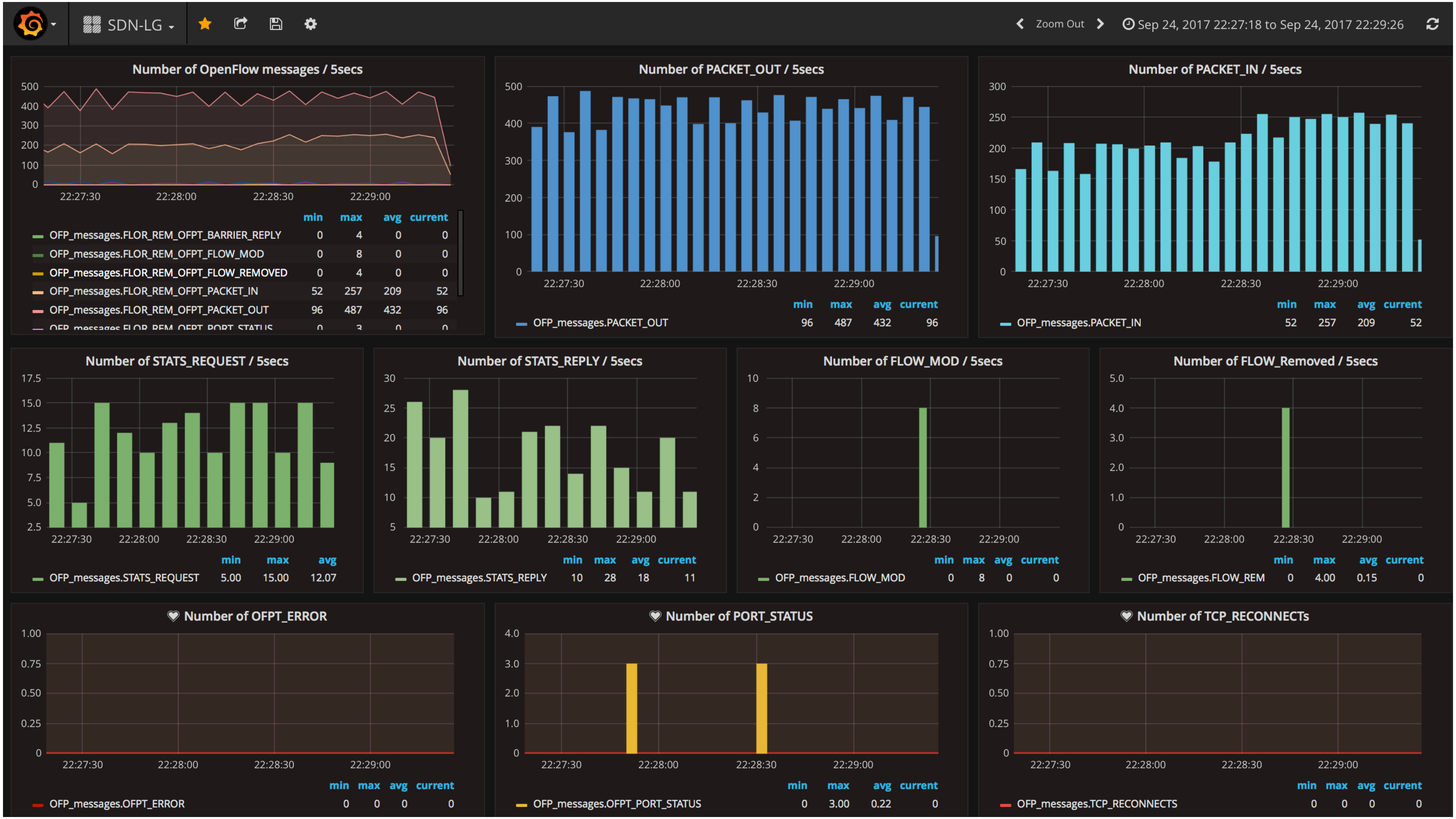
kytos

AmLight SDN Looking Glass [2]

- Developed in Python 3.6
- Leverages the *python-openflow* library
- Built as a Napp on top of Kytos SDN framework
- Uses Influxdb, Mongodb and MySQL for persistence
- Uses Grafana and JavaScript for visualization
- Supports both OpenFlow 1.0, OpenFlow 1.3 and SNMP
- Saves all control plane messages in 100MB files
- Works with OESS's Forwarding Verification module
- Inter-domain trace using our own protocol (soon with NSI)
- Open Source/GPL

Topology Discovery





Sep 24, 2017 22:27:18 to Sep 24, 2017 22:29:26

Number of OpenFlow messages / 5secs

	min	max	avg	current
▬ OFP_messages.FLOR_REM_OFPT_BARRIER_REPLY	0	4	0	0
▬ OFP_messages.FLOR_REM_OFPT_FLOW_MOD	0	8	0	0
▬ OFP_messages.FLOR_REM_OFPT_FLOW_REMOVED	0	4	0	0
▬ OFP_messages.FLOR_REM_OFPT_PACKET_IN	52	257	209	52
▬ OFP_messages.FLOR_REM_OFPT_PACKET_OUT	96	487	432	96
▬ OFP_messages.FLOR_REM_OFPT_PORT_STATUS	0	3	0	0

Number of PACKET_OUT / 5secs

	min	max	avg	current
▬ OFP_messages.PACKET_OUT	96	487	432	96

Number of PACKET_IN / 5secs

	min	max	avg	current
▬ OFP_messages.PACKET_IN	52	257	209	52

Number of STATS_REQUEST / 5secs

	min	max	avg
▬ OFP_messages.STATS_REQUEST	5.00	15.00	12.07

Number of STATS_REPLY / 5secs

	min	max	avg	current
▬ OFP_messages.STATS_REPLY	10	28	18	11

Number of FLOW_MOD / 5secs

	min	max	avg	current
▬ OFP_messages.FLOW_MOD	0	8	0	0

Number of FLOW_Removed / 5secs

	min	max	avg	current
▬ OFP_messages.FLOW_REM	0	4.00	0.15	0

♥ Number of OFPT_ERROR

	min	max	avg	current
▬ OFP_messages.OFPT_ERROR	0	0	0	0

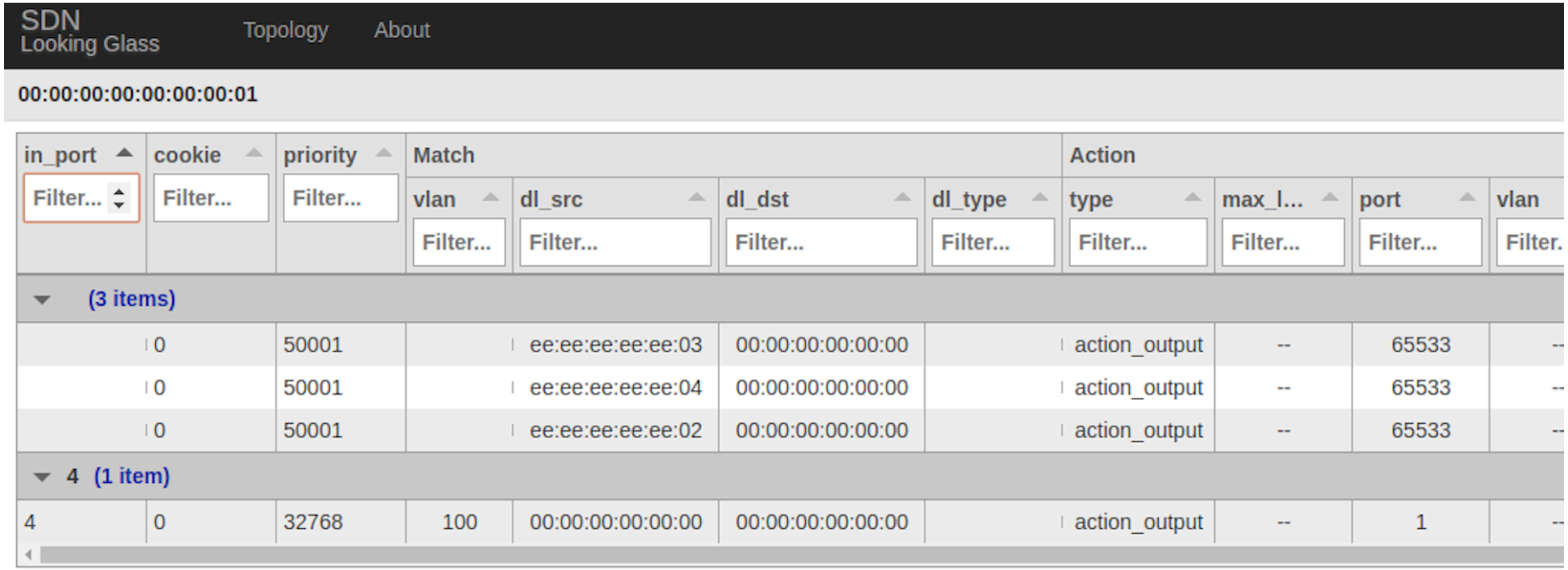
♥ Number of PORT_STATUS

	min	max	avg	current
▬ OFP_messages.OFPT_PORT_STATUS	0	3.00	0.22	0

♥ Number of TCP_RECONNECTs

	min	max	avg	current
▬ OFP_messages.TCP_RECONNECTS	0	0	0	0

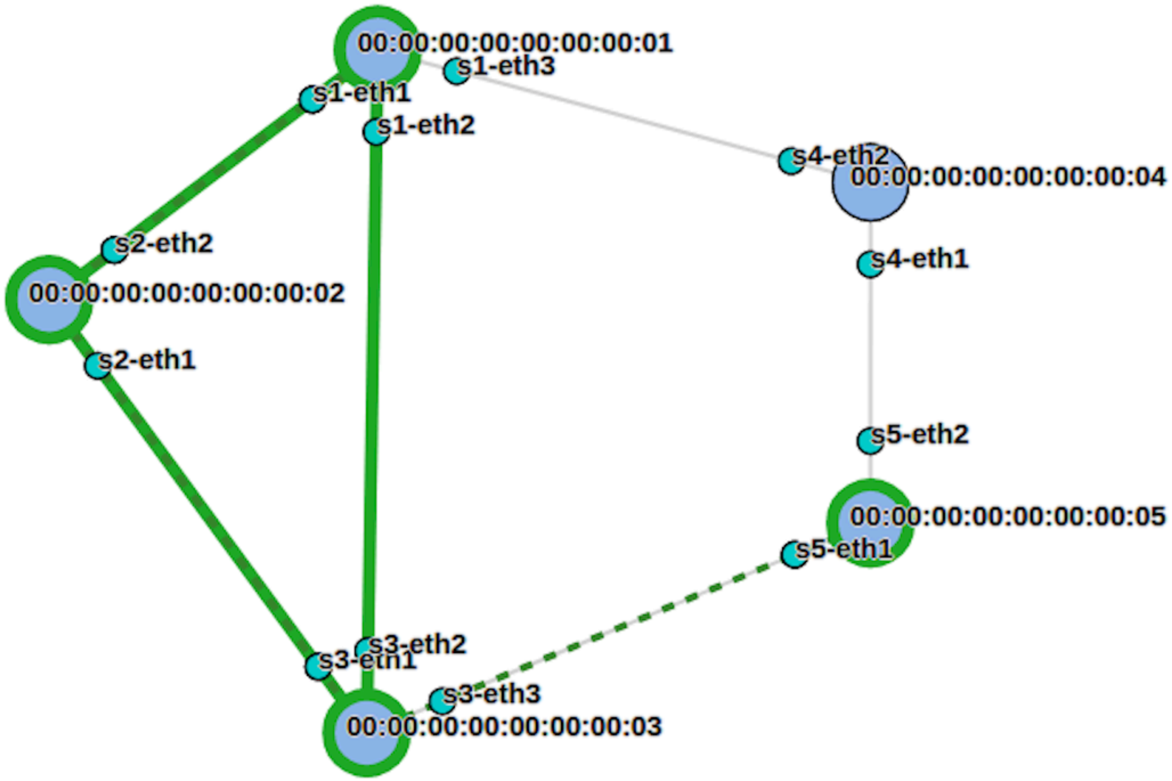
List of Flows

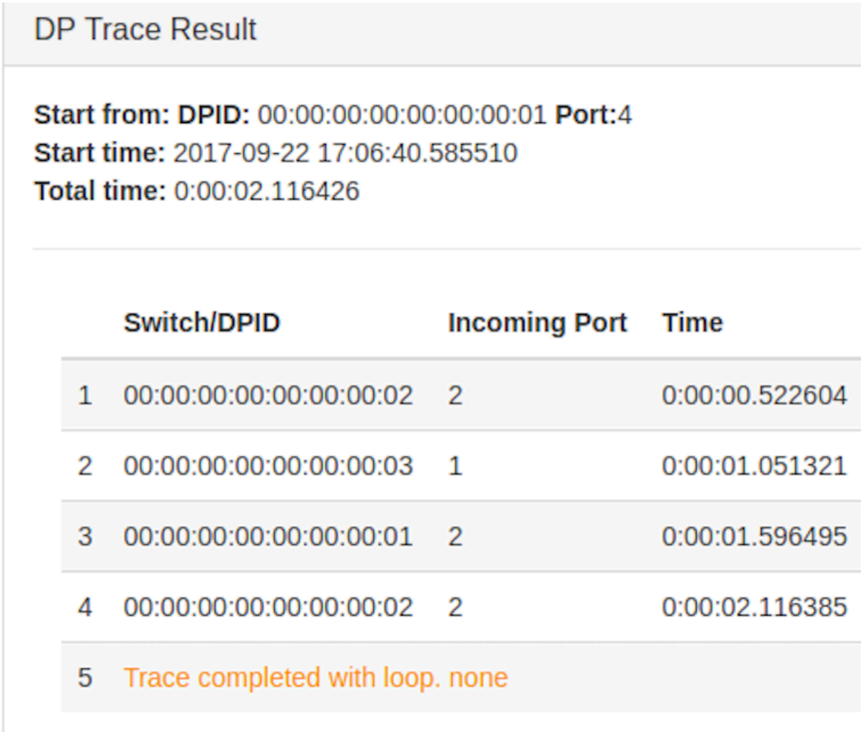


00:00:00:00:00:00:00:01

in_port	cookie	priority	Match				Action			
			vlan	dl_src	dl_dst	dl_type	type	max_l...	port	vlan
Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter.
(3 items)										
	0	50001		ee:ee:ee:ee:ee:03	00:00:00:00:00:00		action_output	--	65533	--
	0	50001		ee:ee:ee:ee:ee:04	00:00:00:00:00:00		action_output	--	65533	--
	0	50001		ee:ee:ee:ee:ee:02	00:00:00:00:00:00		action_output	--	65533	--
4 (1 item)										
4	0	32768	100	00:00:00:00:00:00	00:00:00:00:00:00		action_output	--	1	--

Trace Path (with loop)



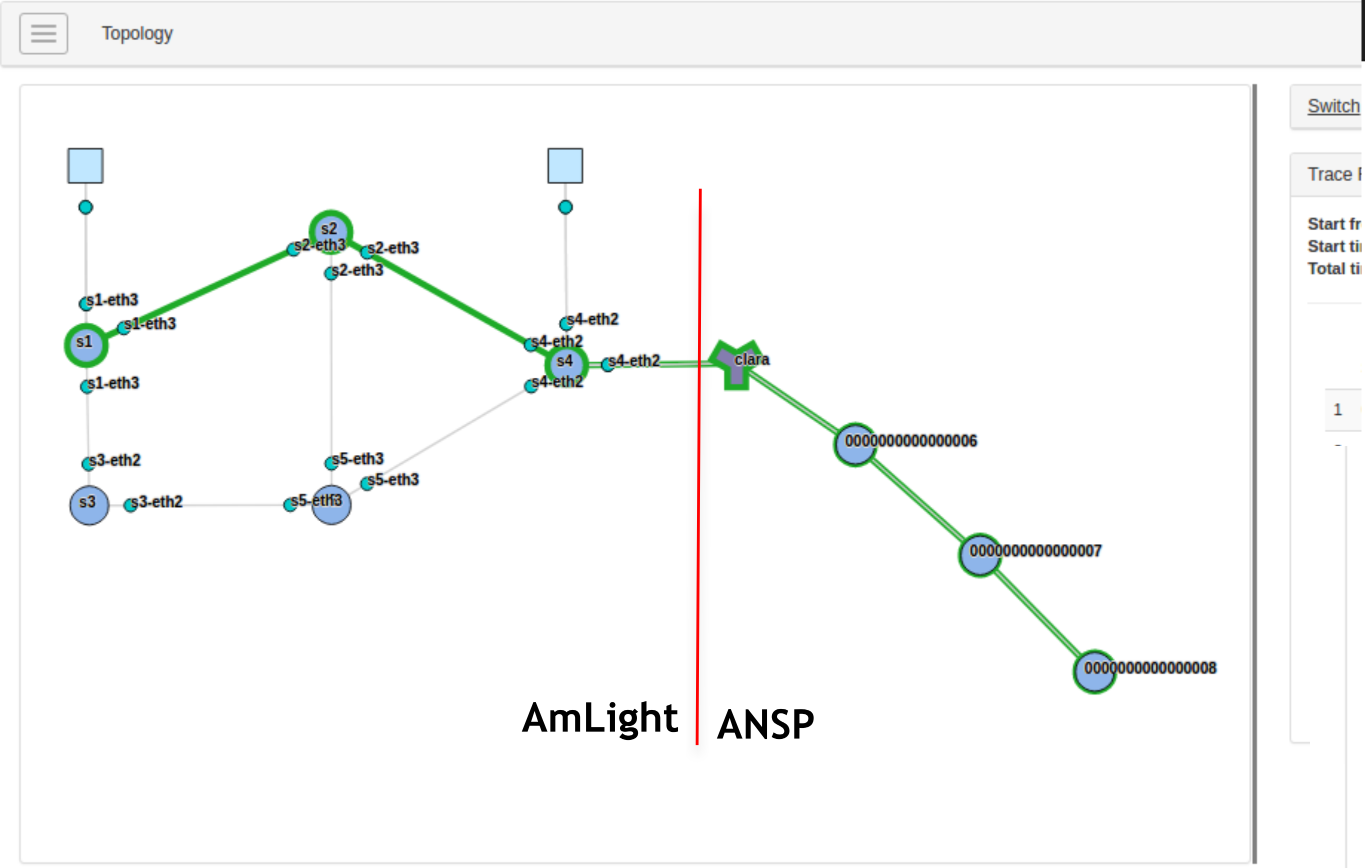


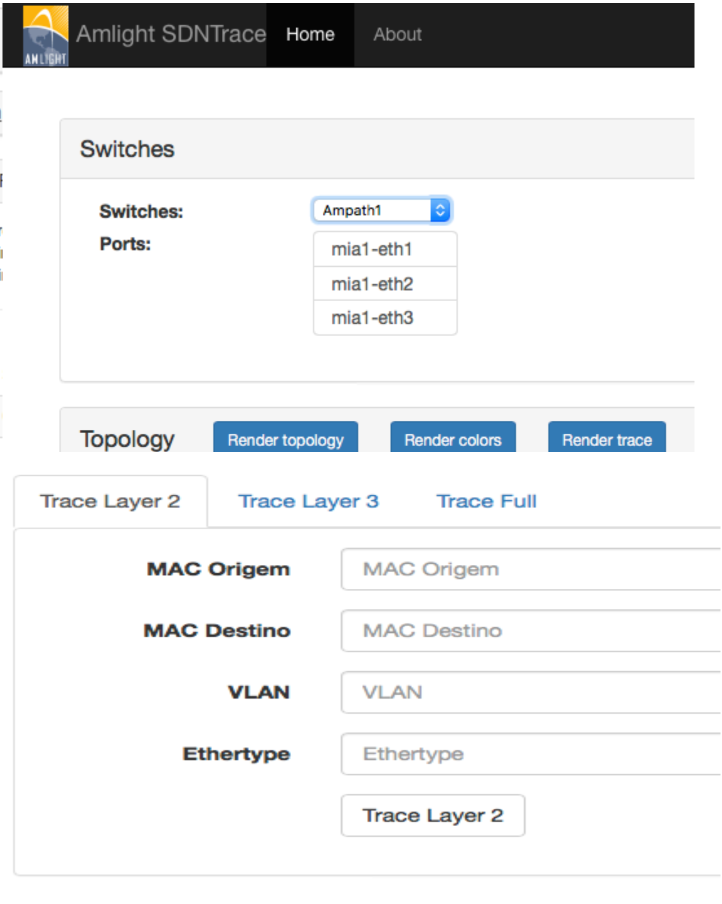
DP Trace Result

Start from: DPID: 00:00:00:00:00:00:00:01 **Port:**4
Start time: 2017-09-22 17:06:40.585510
Total time: 0:00:02.116426

	Switch/DPID	Incoming Port	Time
1	00:00:00:00:00:00:00:02	2	0:00:00.522604
2	00:00:00:00:00:00:00:03	1	0:00:01.051321
3	00:00:00:00:00:00:00:01	2	0:00:01.596495
4	00:00:00:00:00:00:00:02	2	0:00:02.116385
5	Trace completed with loop. none		

Inter-domain Trace Path





Switches

Switches: Ampath1

Ports:

- mia1-eth1
- mia1-eth2
- mia1-eth3

Topology

MAC Origem	MAC Origem
MAC Destino	MAC Destino
VLAN	VLAN
Ethertype	Ethertype

THANK YOU!

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

Antonio Francisco <antonio@amlight.net>
Academic Network of Sao Paulo