A Hypervisor for the Ryu NOS

Bachelor Thesis - Final presentation

Felix Breidenstein

FB 20 - Informatik
Technische Universität Darmstadt
E-Mail: mail@felixbreidenstein.de

Supervisor
Dipl.-Wirtsch.-Inf. Jeremias Blendin

Tuesday, 18 October 2016

Outline

- Motivation
- Background & Problem Definition
- System Design
- Implementation
- Evaluation
- Conclusion & Future Work

1. Motivation

- The controller is a critical part in a SDN network
  - Big impact if an app crashes the controller

- Malicious apps could (unintentionally)
  - Crash the controller
  - Jam the whole network

- Current state of other SDN Controller [0,1]:
  - OpenDaylight has two Plugins for app virtualisation [2]
    - No App-to-App communication, different API
  - ONOS has multi controller support
  - Rosemary has Resource Monitoring and app Isolation [3]
    - No code, just an idea
  - HyperFlex implements rate-limiting
    - with a complex setup [4]

  - Currently no controller monitors the switch ressources

1. Motivation - Goals

- Accelerate research progress in shared SDN testbeds
- Use case: Multiple apps work together: e.g. Segment Routing +SDM
- Ryu is one of the most common SDN controllers in research [5,6]
- Goal: Make app isolation possible with Ryu
- Impact:
  - Protect the controller, the network, and make SDN development easier
  - Build the foundation for a hypervisor with switch resource monitoring

2. Background & Problem Definition

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- No access control for apps
  - Should this app get all Events?
  - Is this app allowed to send FlowMod/PacketOut/… ?
- No sanity checks of the events
  - Valid matcher fields used?
  - Enough free space on the switch?
- Thread scheduling not enforced (non-preemtive)
  - ➢ An app can take 100% processing power forever
- No rate-limiting
  - ➢ An app can take 100% of the switch/controller ressources with event flooding

3. System Design

Approach:

- Put every app into a container
  - Can be distributed over the network
  - Not a full controller but enables app isolation
  - Malicious apps can now only crash their own container and not the controller

- Insert another layer in between to apply event filter rules
  - Only forward specific event types
  - Manipulate fields of the event message
  - "Virtual memory" concept for e.g.
    - Priorities and Flowtables

- This way, multiple researchers can work on their own projects on the same controller without disturbing each other

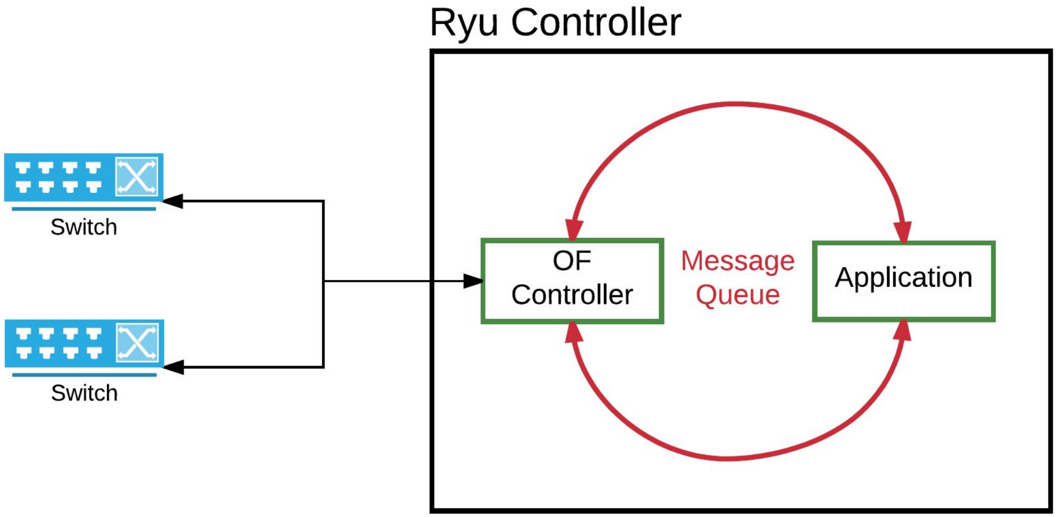
3. System Design

Design goals

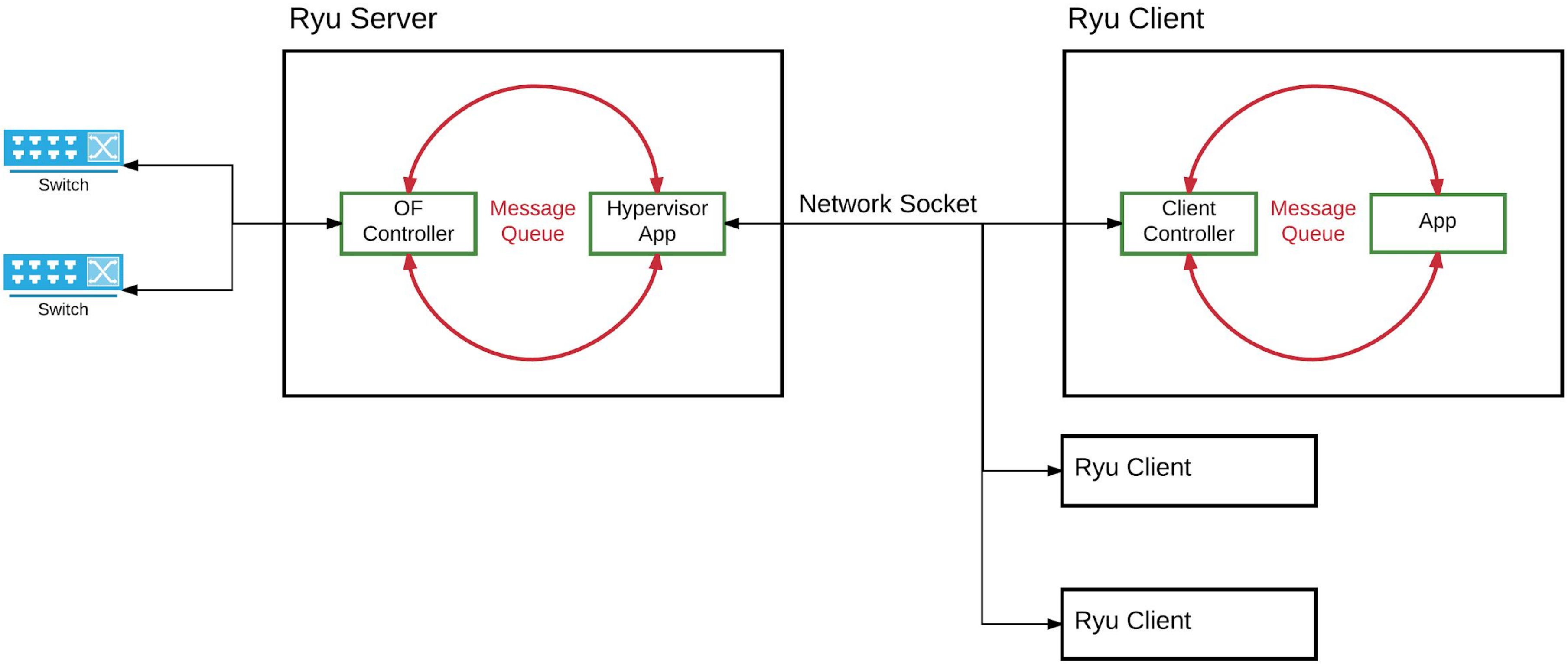
- No modification of the application code needed
  - App will not know if it's run inside the hypervisor
  - Possibility to use existing code
- Easy to setup
  - Just like installing a normal Ryu controller
  - No extra packages, programs or server needed
- Acceptable performance loss due the network communication
  - More in the section ‘Evaluation’
- Easy API for researchers to manage the hypervisor
- Basis for a hypervisor with switch resource protection

3. System Design

- Current Ryu architecture
  - Every app runs in a non-preemptive thread
  - Apps can register handlers to get events
  - Apps can generate events or directly send OF-Events to the switch
  - Ryu just takes events and forwards them

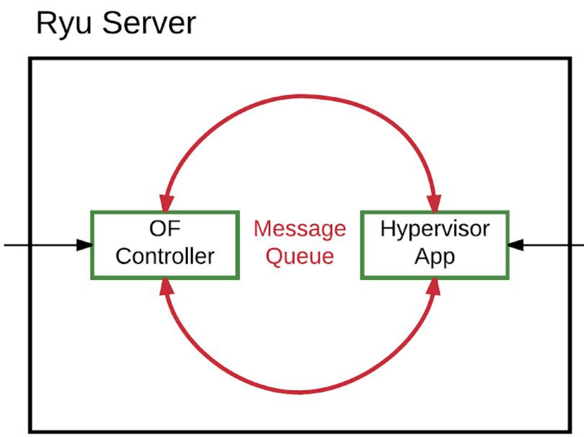


3. System Design

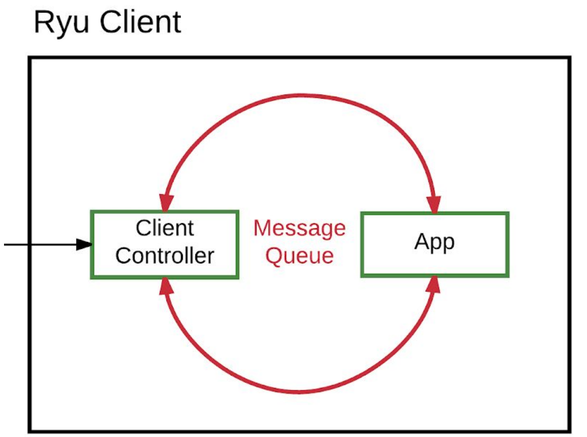


4. Implementation

- Server instance
  - “Hypervisor” implemented as an RyuApp
  - Register handler for all events
  - Handle the socket connection to the remote instances
  - Apply the filter rules on incoming & outgoing events



- Client instance
  - Connect to the master via a socket
  - Load a substitute controller instead of the OF-Controller
  - Create fake DataPath objects for the apps
  - Generate OF Events from the informations sent from the master



4. Implementation Decision

- Client/Server Setup
  - Best way to protect the controller from malicious apps
- Using NanoMsg for network communication
  - Lightweight
  - One-to-One and One-to-Many protocols
- cPickle for data serialization

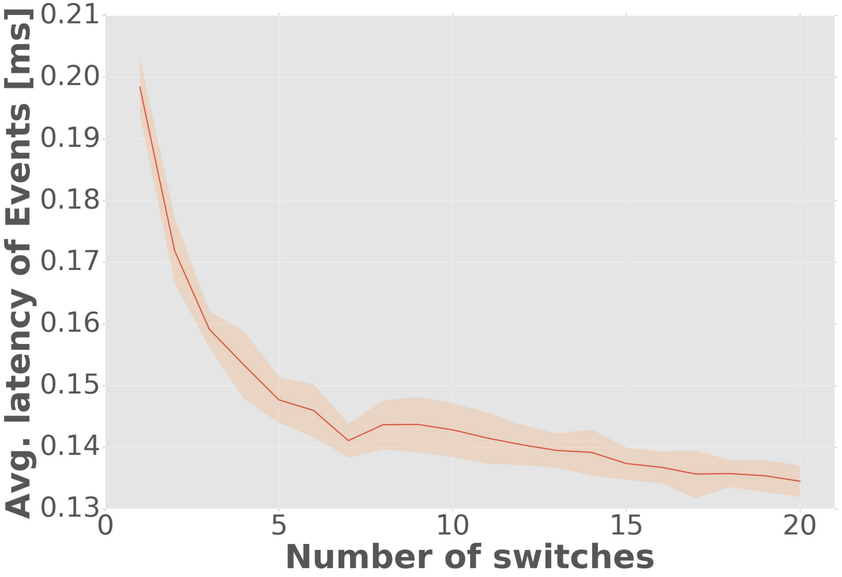
5. Evaluation

- Evaluation topics:
  - Performance
    - Plain Ryu vs. Hypervisor
    - Impact of multiple clients
  - Robustness
    - Impact of an malicious application
- Benchmarks were done with cbench
  - Simulate one to 20 switches
  - Repeat every test 100 times

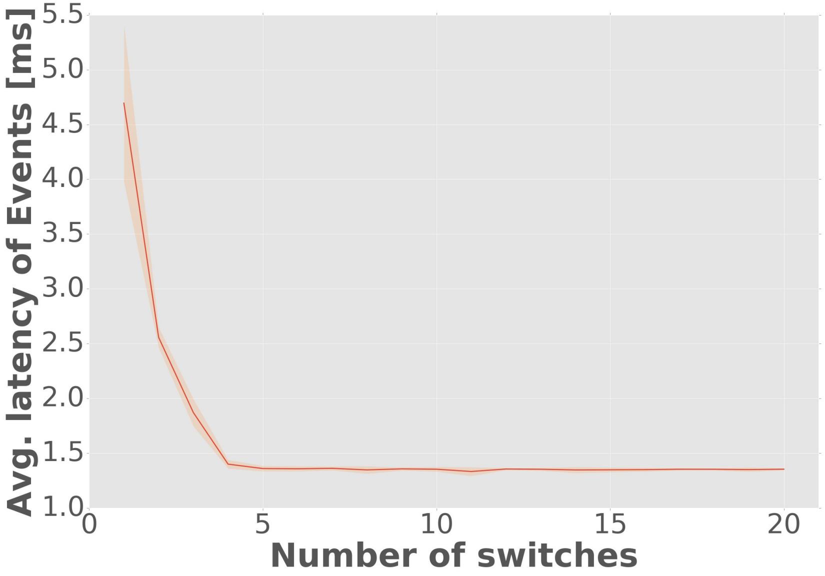
5. Evaluation - Performance

- Direct comparison or message latency
- Using the *ryu/app/cbench.py* application

Original Ryu Controller



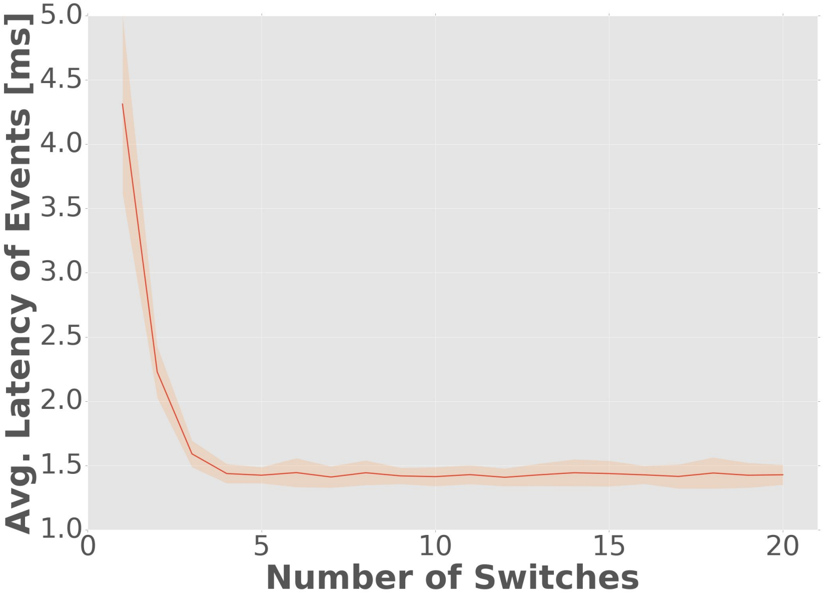
Hypervisor



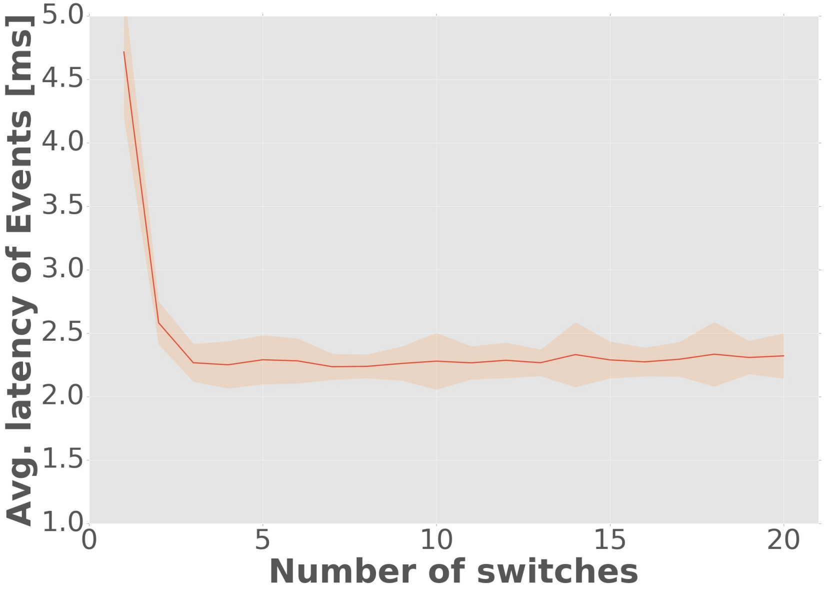
5. Evaluation - Performance

- Impact of multiple connected clients to the hypervisor

Hypervisor with five clients

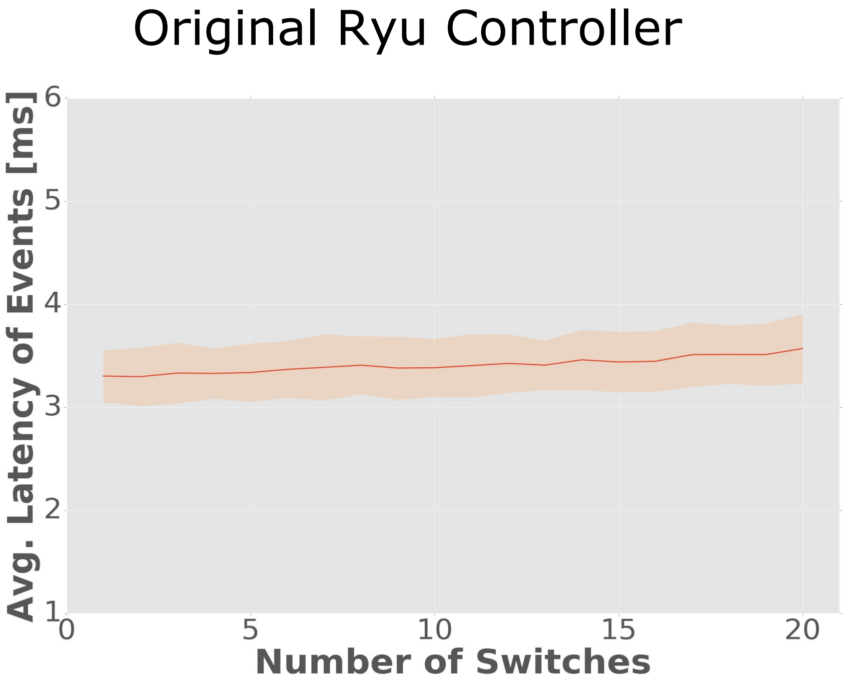


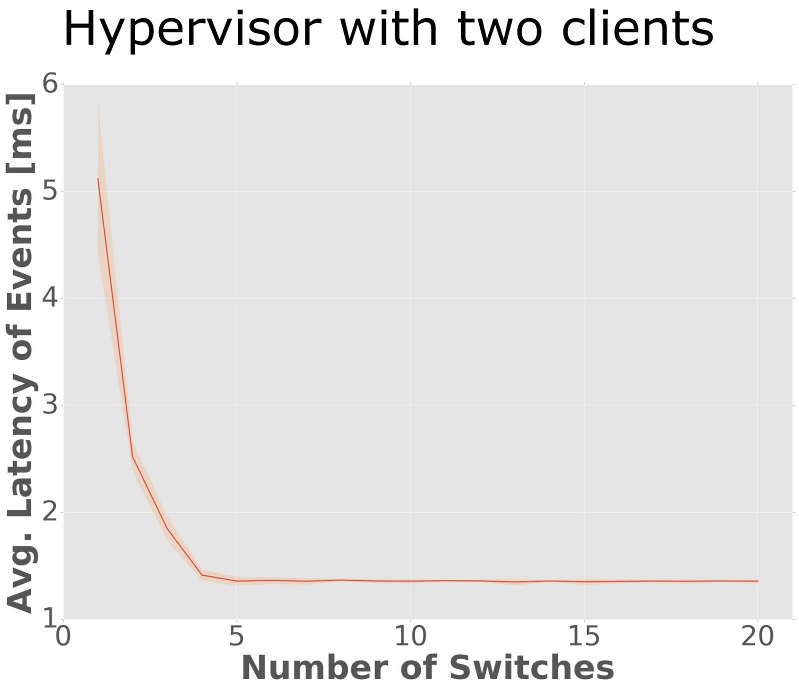
Hypervisor with ten clients



5. Evaluation - Robustness

- Impact of a malicious application
- Simulate computationally intensive behaviour with a *sleep()* call

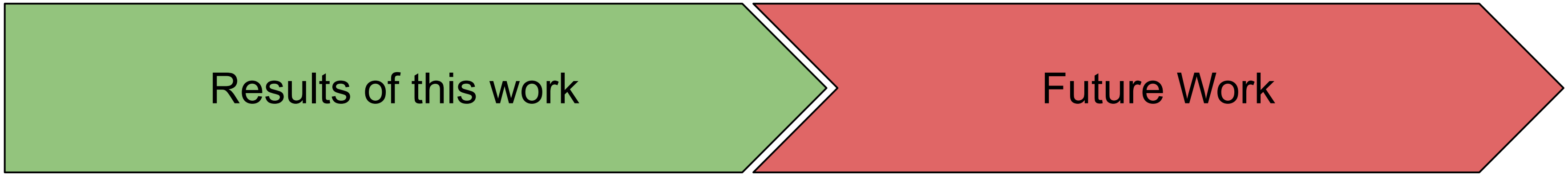




6. Conclusion & Future Work

- Convert Ryu into a Client/Server application
- Implement application isolation
- Message filtering

- Better socket handling to increase performance
- Define a filter language with more features
- Encryption & authentication



Results of this work

Future Work

Thank you for your attention!
Questions?

mail@felixbreidenstein.de

Links

[0] Sandra Scott-Hayward. Design and deployment of secure, robust, and resilient sdn controllers. In Network Softwarization (NetSoft), 2015 1st IEEE Conference on, pages 1–5. IEEE, 2015.

[1] Andreas Blenk, Arsany Basta, Martin Reisslein, and Wolfgang Kellerer. Survey on network virtualization hypervisors for software defined networking. 2015.

[2] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. Opendaylight: Towards a model-driven sdn controller architecture. In 2014 IEEE 15th International Symposium on, pages 1–6. IEEE, 2014.

[3] Seungwon Shin, Yongjoo Song, Taekyung Lee, Sangho Lee, Jaewoong Chung, Phillip Porras, Vinod Yegneswaran, Jiseong Noh, and Brent Byunghoon Kang. 2014. Rosemary: A Robust, Secure, and High-performance Network Operating System. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (CCS '14). ACM, New York, NY, USA, 78-89.

[4] Andreas Blenk, Arsany Basta, and Wolfgang Kellerer. Hyperflex: An sdn virtualization architecture with flexible hypervisor function allocation. In Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, pages 397–405. IEEE, 2015.

[5] A. Sgambelluri, A. Giorgetti, F. Cugini, G. Bruno, F. Lazzeri, and P. Castoldi, "First Demonstration of SDN-based Segment Routing in Multi-layer Networks," in Optical Fiber Communication Conference, OSA Technical Digest (online) (Optical Society of America, 2015), paper Th1A.5.

[6] F. Paolucci, A. Giorgetti, F. Cugini and P. Castoldi, "SDN and PCE implementations for segment routing," Networks and Optical Communications - (NOC), 2015 20th European Conference on, London, 2015, pp. 1-4., doi: 10.1109/NOC.2015.7238607