Relazione sulle attività svolte

Open Source Hybrid IP/SDN (OSHI) Networking

Candidato

Pier Luigi Ventre

(pl.ventre@gmail.com)

Tutor

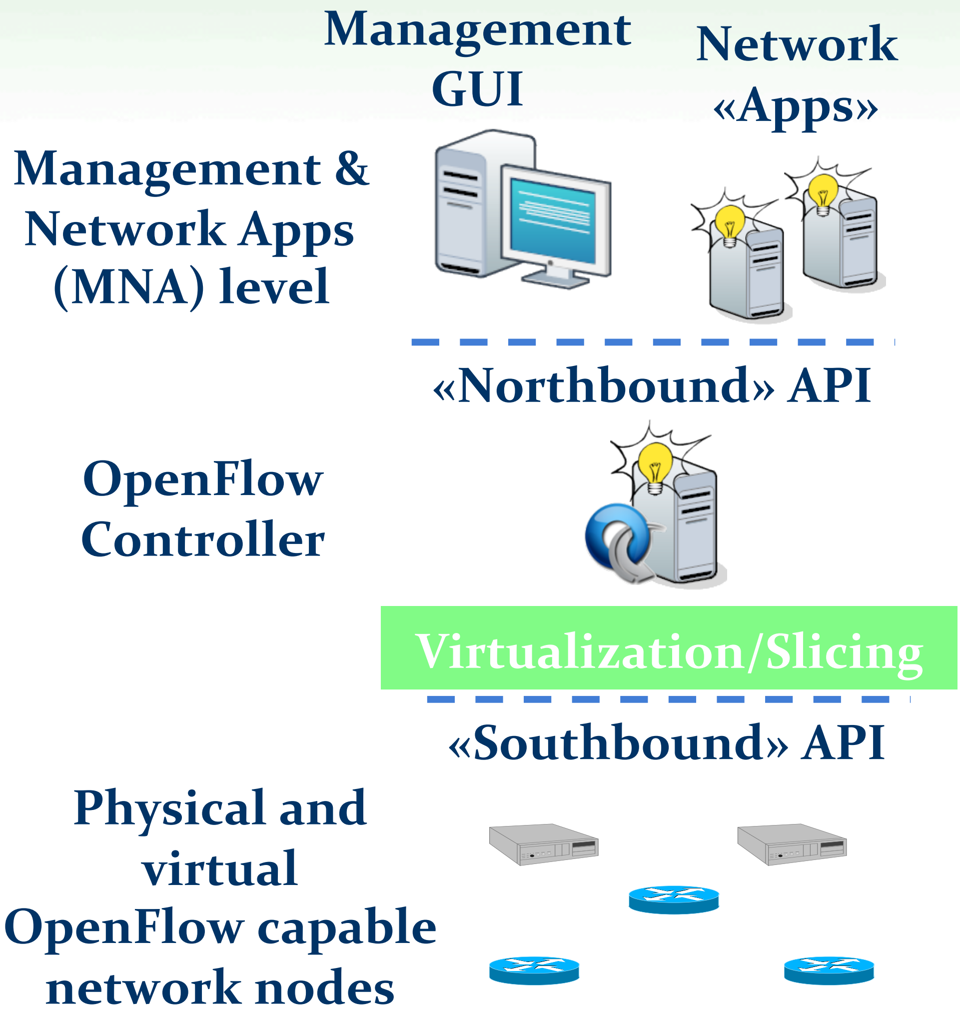
Stefano Salsano

(stefano.salsano@uniroma2.it)

UNIVERSITA' degli STUDI di ROMA TOR VERGATA

***5° Borsisti Day** – 13/05/2014*

Software-Defined Networking: The New Norm for Networks [3]



- Enables innovation in the infrastructure;
- Decouples control plane from data plane;
- Enables researchers to run experimental protocols;
- OpenFlow is the first open Southbound API, a single “piece” in the SDN puzzle;

DREAMER Project

Distributed REsilient sdn Architecture MEeting carrier grade Requirements

- **Partners:**

CNIT

CREATE-NET

Consortium GARR

- **DREAMER goal:**
  - Investigate how a network based on an OpenFlow/SDN control plane can provide the same functionalities of an IP/MPLS control plane;

Objectives

- **Introduce the SDN paradigm in the IP backbones;**
  - Development of a flexible and scalable solution - fully open source;
- **Provide the functionalities of an IP/MPLS net;**
  - Replication and improvement of services provided by current nets;
- **Emulation on Mininet and OFELIA testbed;**
  - Development of emulation tools for “local” and distributed testbed
- **Performance Testing;**
  - Evaluation of packet processing overhead ;

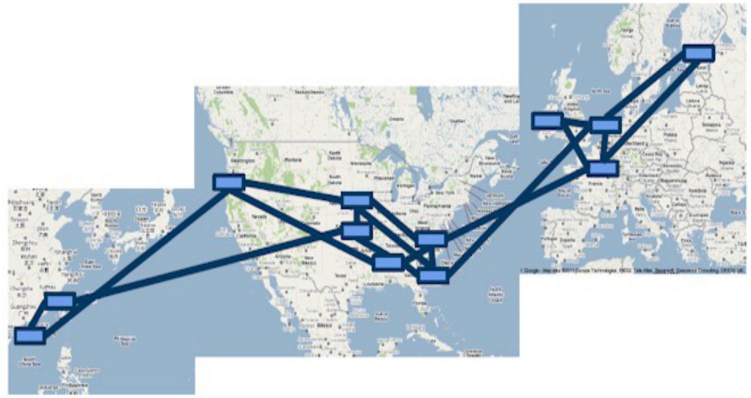
Outline

1. Related work & novelty;
2. OSHI networking;
3. Ethernet Virtual Leased Line
4. OSHI emulation tools;
5. Performance evaluation;
6. Second year plan;

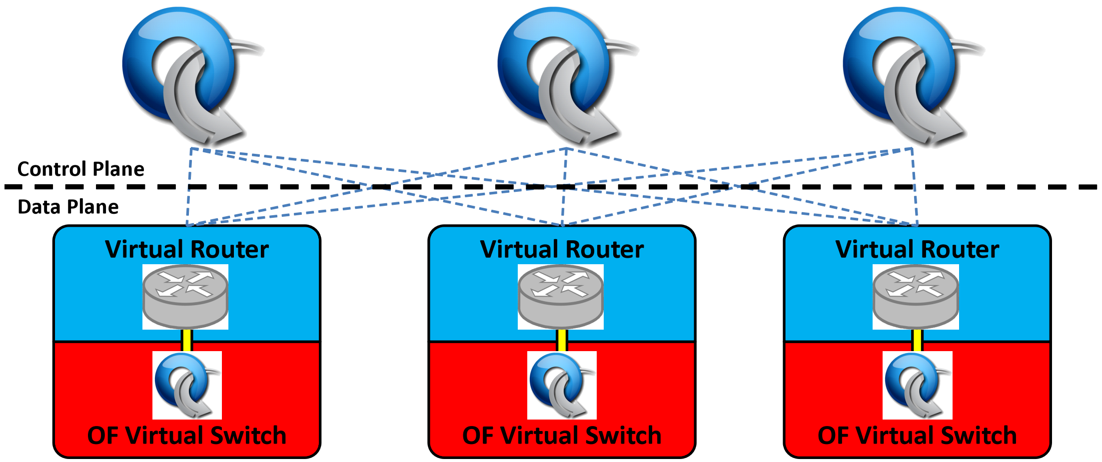
Open Source Hybrid IP/SDN (OSHI)

- **OSHI high level design choices:**
  – Coexistence of IP routing/forwarding and SDN based forwarding;
  – OSHI node acts as plain IP router, as well as SDN nodes;
- **Related work design choices:**
  – Extraction of IP routing logic and execution in the controller (**RF** and **B4**);
  – Combination of SDN data plane and IP control plane in order to realize a label switch router (**OpenLSR**)
  – Addition of BGP speaker in the Control Plane (**SDN IP-Peering**);
- **Industry believes in hybrid approach but…** $

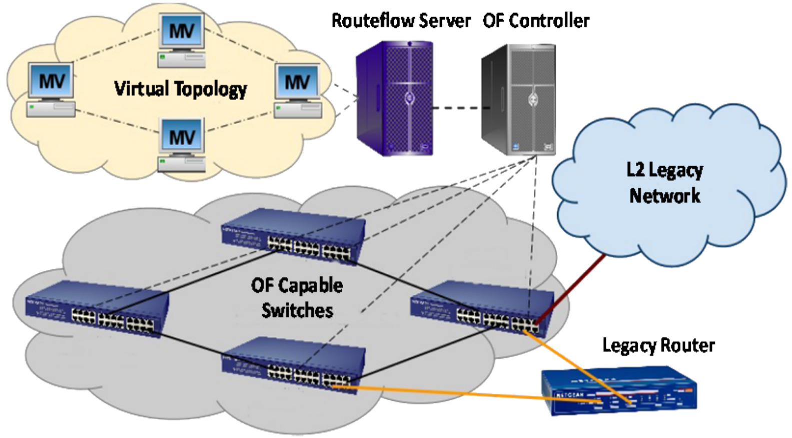
Google's B4 WAN



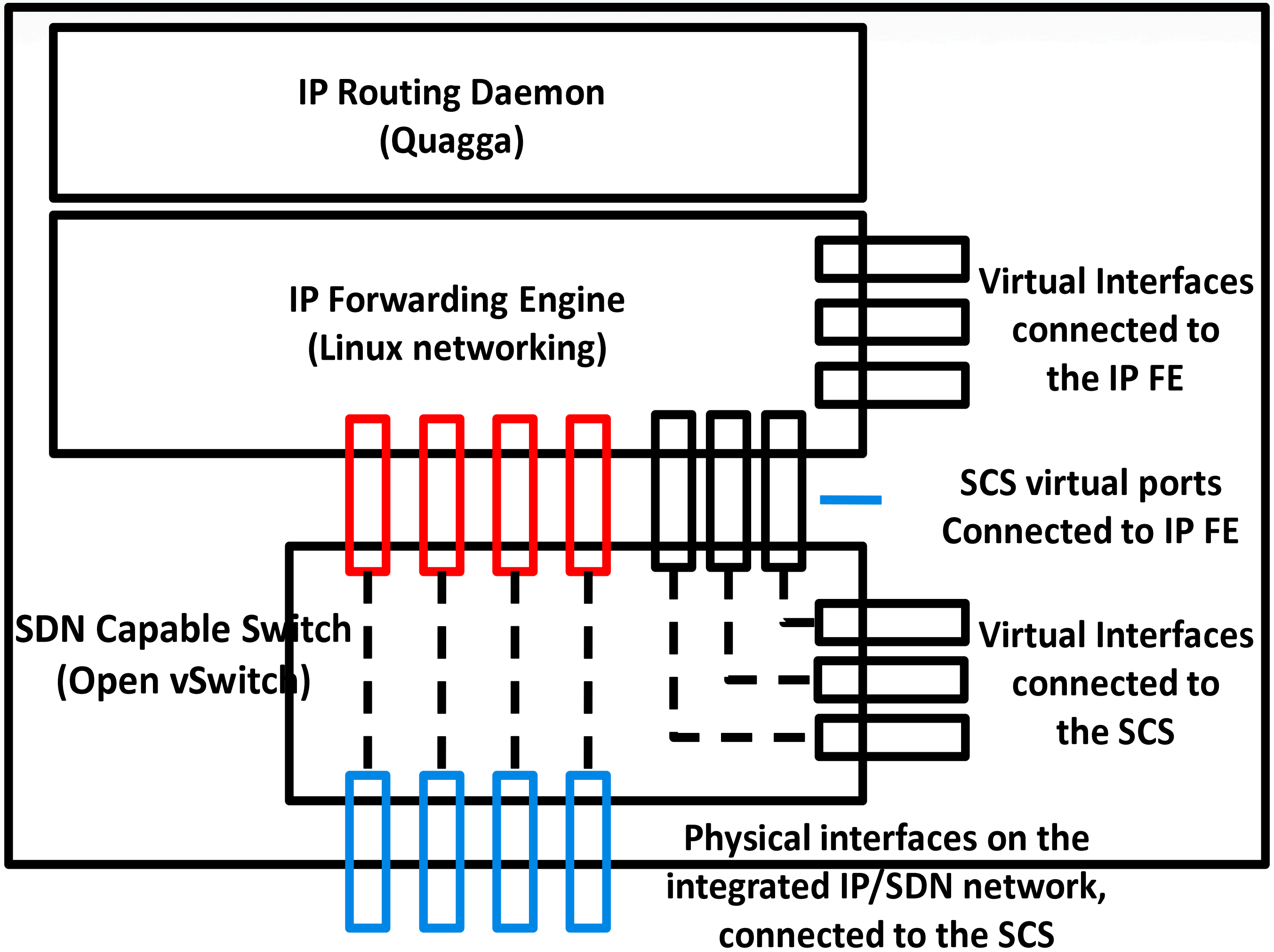
DREAMER Approach



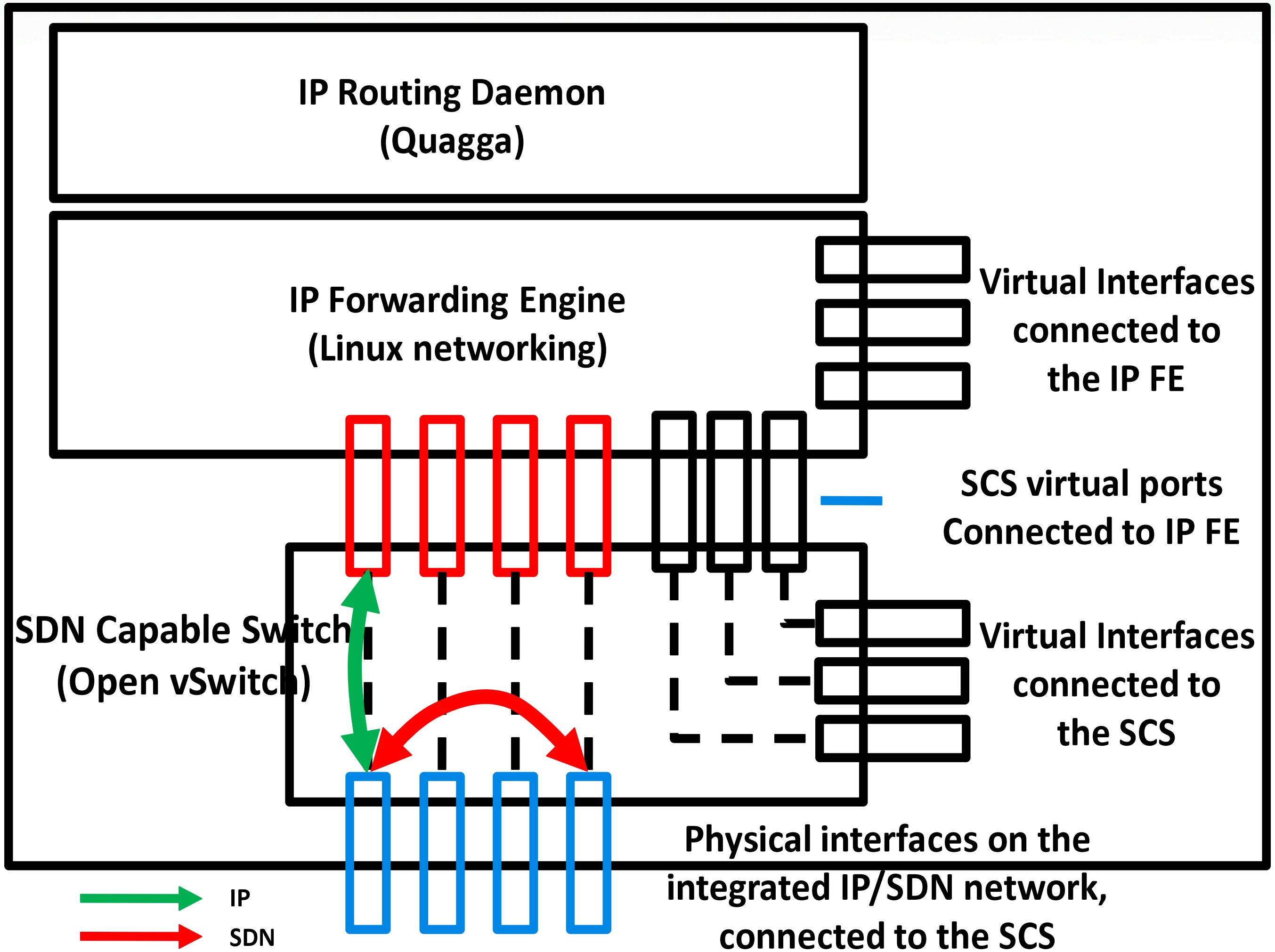
Route Flow Approach



OSHI Node architecture



OSHI Node architecture



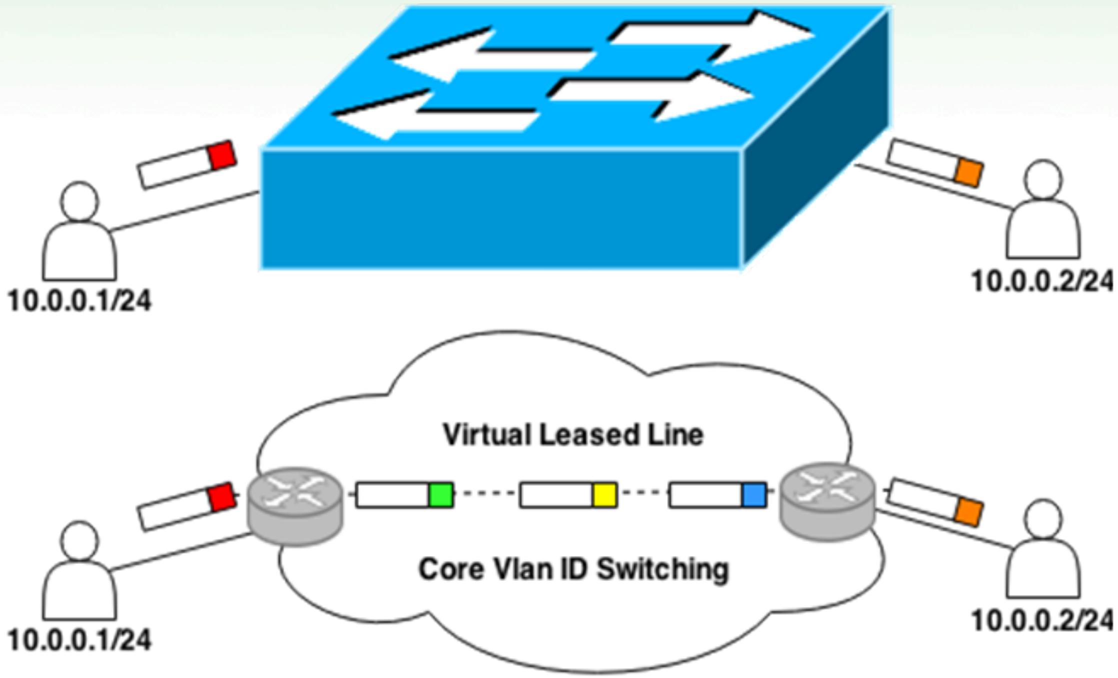
OSHI Node architecture (2)

- **OSHI fundamental blocks:**
  - –Coexistence mechanisms;
  - –Ingress classification functions;
  - –Tunneling mechanisms;

↓

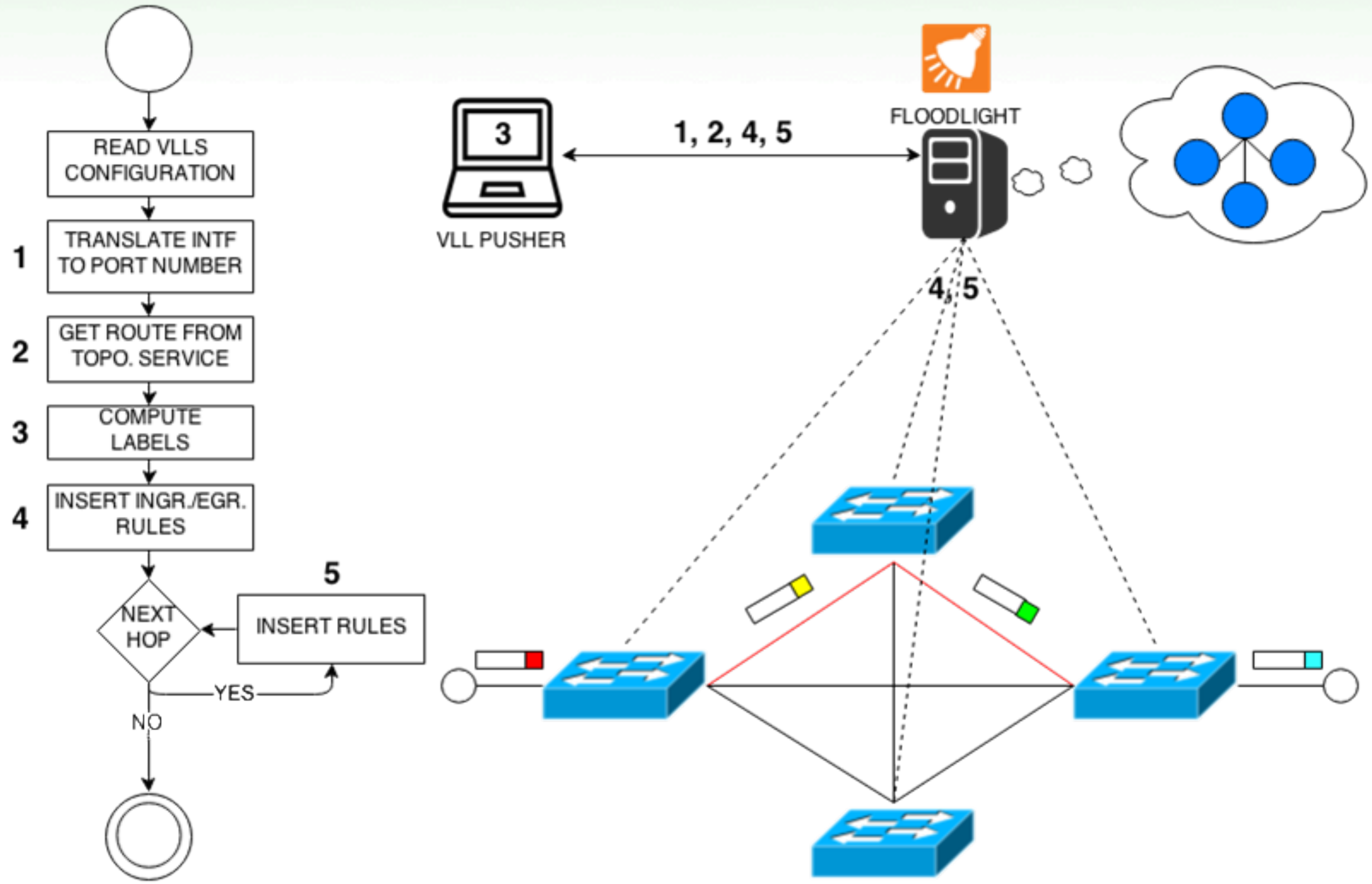
VLAN Tags and Ports (Currently)

Ethernet Virtual Leased Line

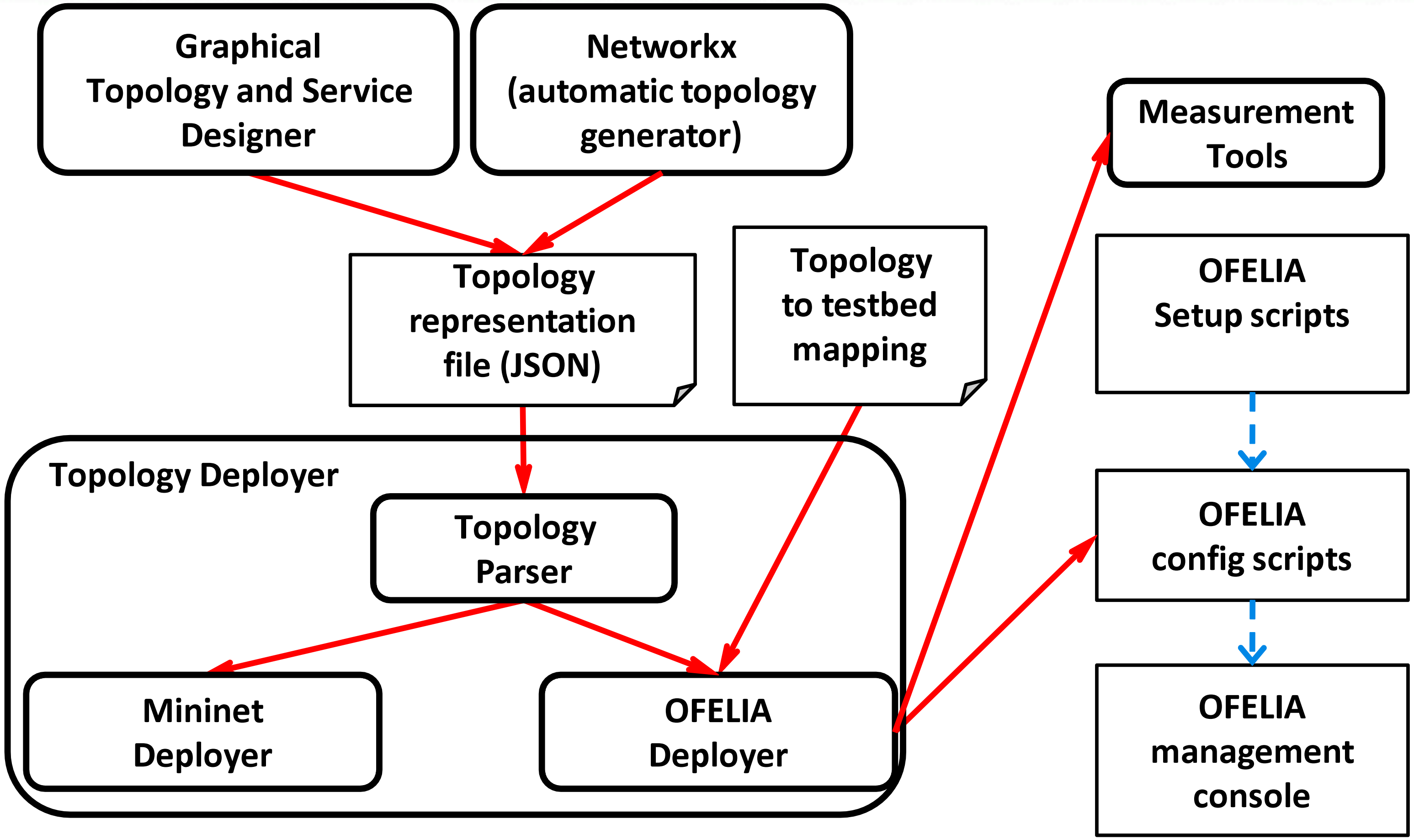


- Guarantees to the served end-points to be directly interconnected as if they were in the same Ethernet LAN;
- VLL is provided in OSHI network through a SDN Based Path (SBP) using VLAN tags switching;

Virtual Leased Line Pusher

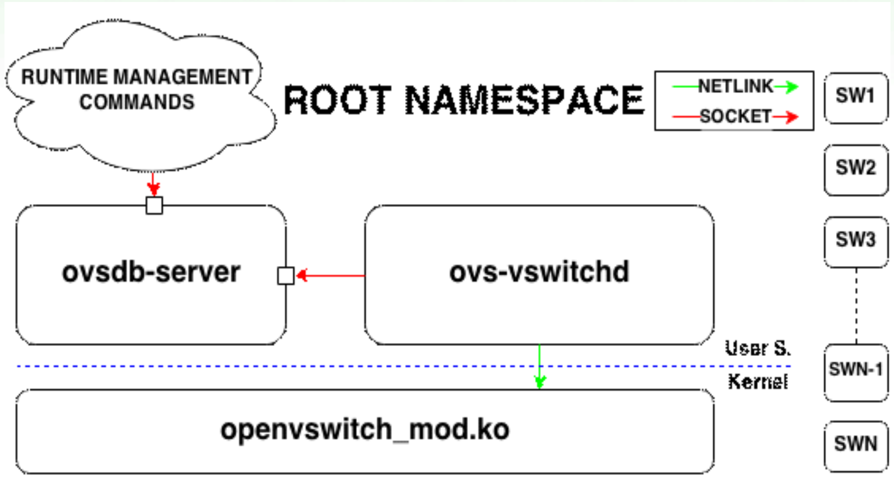


OSHI emulation workflow

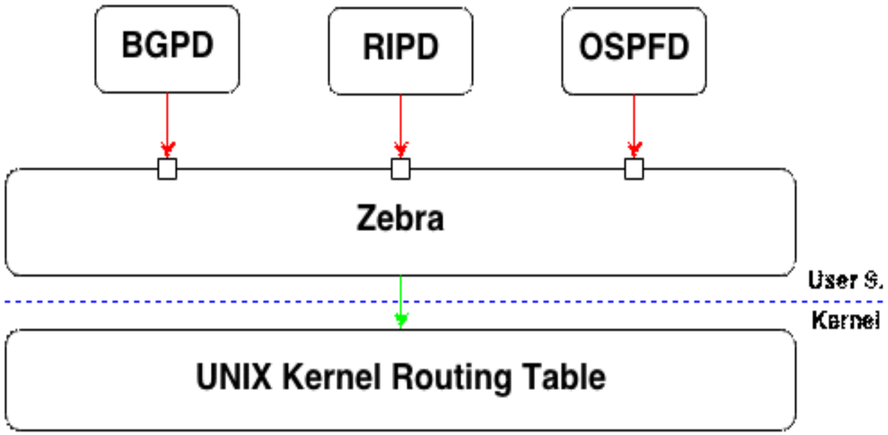


Emulation on Mininet

Open vSwitch architecture

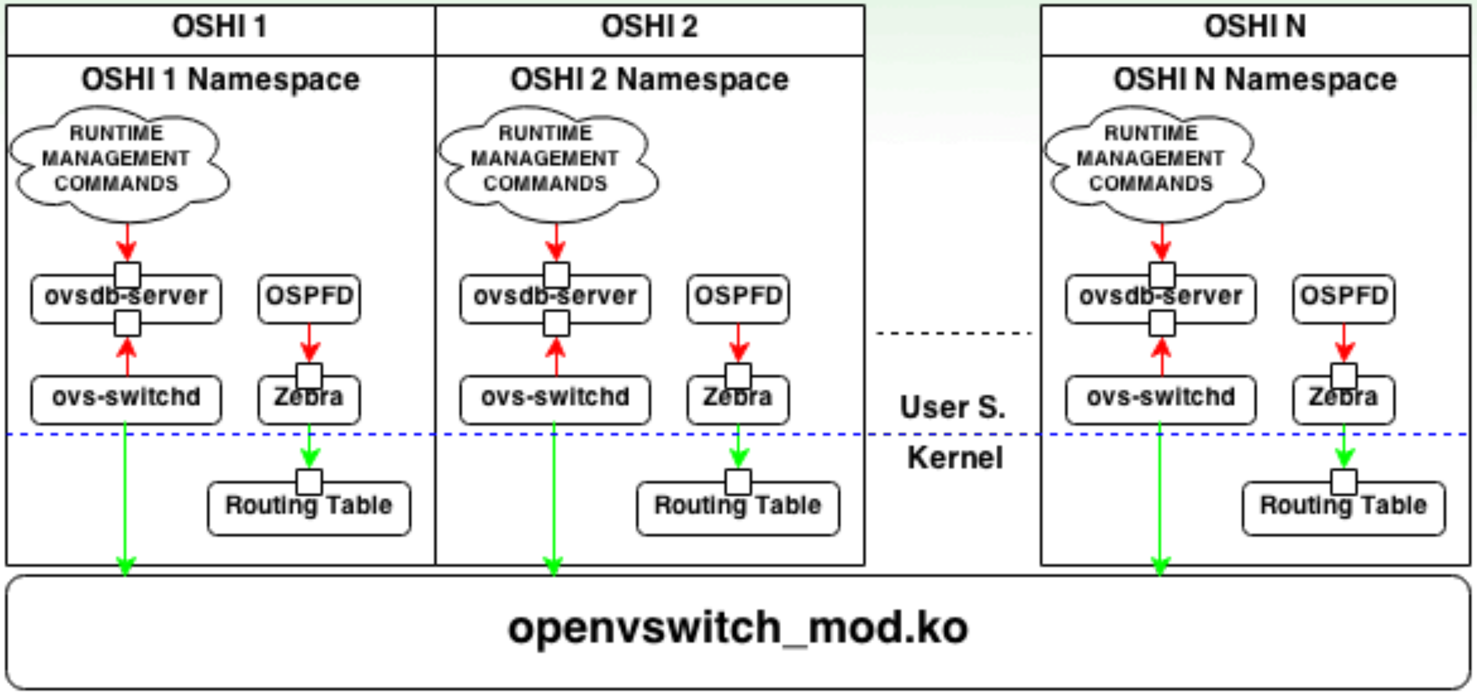


Quagga architecture



- **Objective:**
  - Extend Mininet functionalities in order to allow emulation of OSHI networking ;
- **Issues:**
  - Each Node in Mininet is a process, but all share the same network namespace;
  - OVS cannot work with private namespace;
  - The same processes run simultaneously in the same machine;
  - Manual configuration is tedious and error prone;
  - Mininet uses shared file system approach;
- **Solutions:**
  - Each Node has private namespace;
  - Each Node forks OVS and Quagga daemons;
  - Automate as possible the configuration;
  - Each Node has a private "file system";

Mininet Deployer



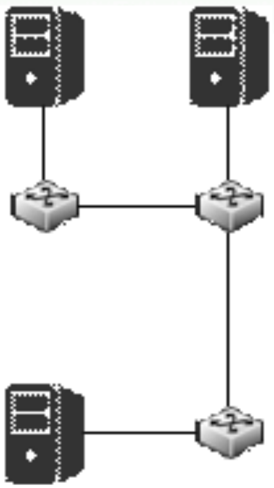
- Able to start in a few seconds whatever experimental topology in Mininet;
- Nodes ready to start the experiments;
- One click experiments;
- Easy to extend with new features;

Emulation on OFELIA testbed

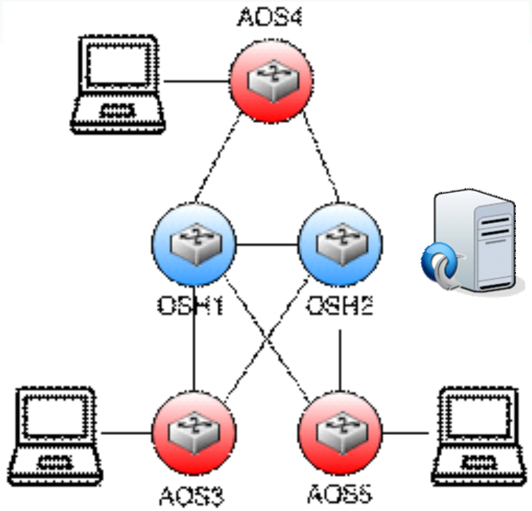
Physical OFELIA testbed

VM server

Physical switch



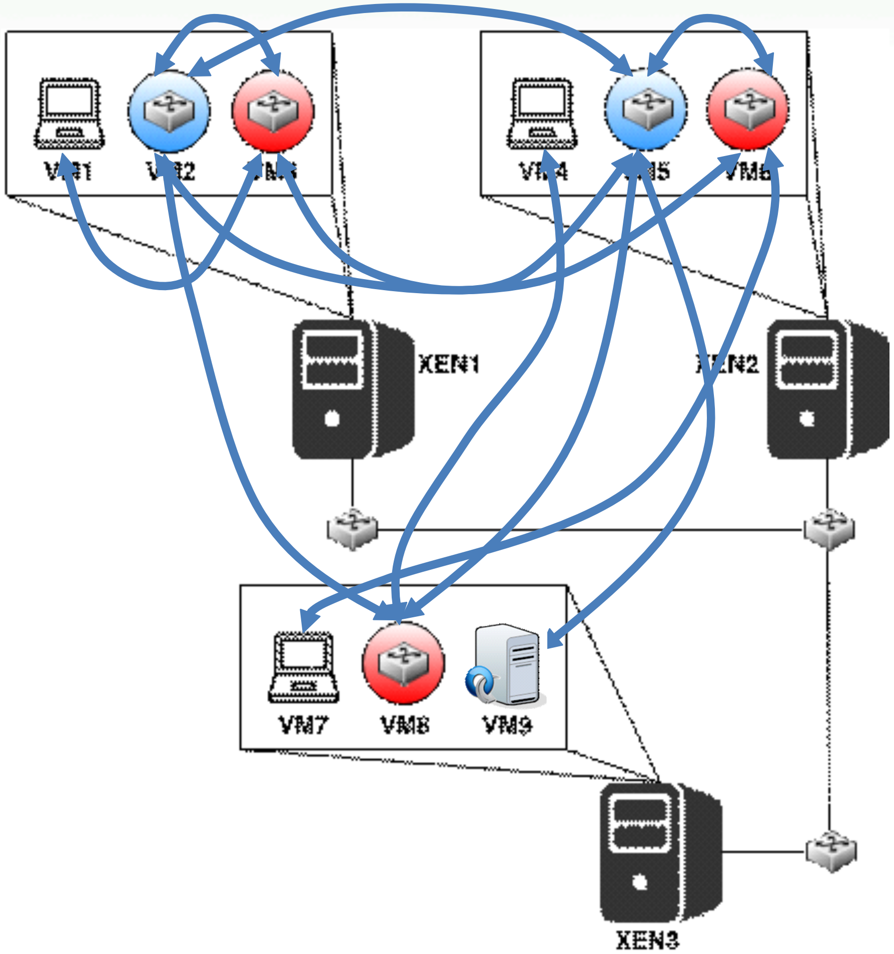
Overlay Experiment Topology



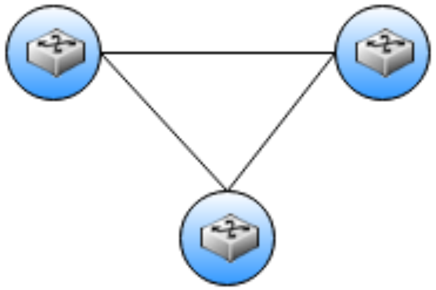
- **Objective**:
  - Emulate arbitrary overlay topology on a set of VM servers and physical links;
- **Issues**:
  - A large number of overlay topology links needs to be configured;
  - Manual configuration is tedious and error prone;

Emulation on OFELIA testbed (2)

- **Solutions**:
  - Deploy a number of nodes (OSHI, CTRL, HOST) over a corresponding number of VMs;
  - Create an overlay network topology of Ethernet over UDP tunnels among the VMs;
  - Automate as possible the configuration;



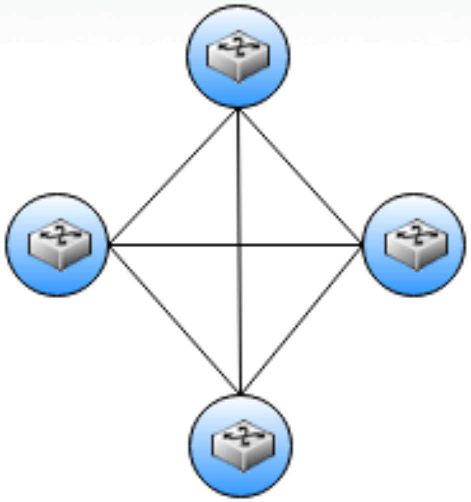
Testbed Deployer Library



```
//Deploy of Full Mesh Topology
. . . // Initialization section of temporary variable
testbed = TestbedOFELIA("ofelia.map") //We create Testbed object passing as
parameter the mapping file
for i in range (0, size): // the size parameter is the dimension of the mesh topology
  oshi = testbed.addOshi(name) // Add a new OSHI to the experiment. Return an
Oshi object
   for lhs in oshis:
  testbed.addPPLink(lhs_name, oshi_name) //Add an overlay link among lhs
and oshi. Return a composite object, that represents in our case an overlay link.
        oshis.append(oshi)
ctrl = testbed.addController(name, OF_tcp_port) // Add a controller to the
experiment. Return a Controller object
testbed.addPPLink(oshi, ctrl) //Connect the Controller to the overlay network
testbed.configure() //Generate the configuration file for the overlay topology to
deploy
```

- Able to start in a few minutes through config. script whatever experimental topology;
- Nodes ready to start the experiments;
- Easy to extend with new features;

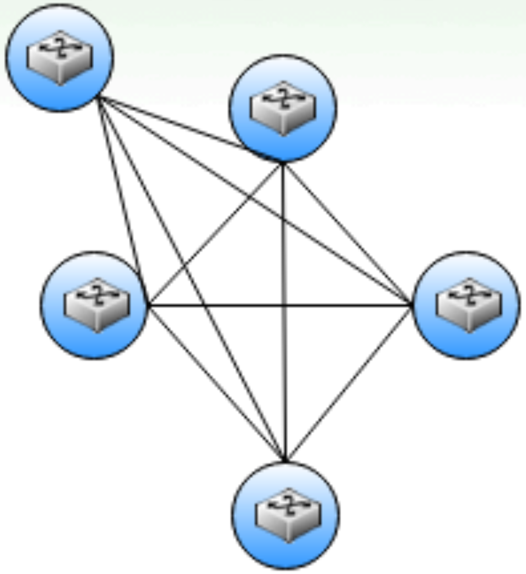
Testbed Deployer Library



```
//Deploy of Full Mesh Topology
. . . // Initialization section of temporary variable
testbed = TestbedOFELIA("ofelia.map") //We create Testbed object passing as
parameter the mapping file
for i in range (0, size): // the size parameter is the dimension of the mesh topology
   oshi = testbed.addOshi(name) // Add a new OSHI to the experiment. Return an
Oshi object
    for lhs in oshis:
   testbed.addPPLink(lhs_name, oshi_name) //Add an overlay link among lhs
and oshi. Return a composite object, that represents in our case an overlay link.
          oshis.append(oshi)
ctrl = testbed.addController(name, OF_tcp_port) // Add a controller to the
experiment. Return a Controller object
testbed.addPPLink(oshi, ctrl) //Connect the Controller to the overlay network
testbed.configure() //Generate the configuration file for the overlay topology to
deploy
```

- Able to start in a few minutes through config. script whatever experimental topology;
- Nodes ready to start the experiments;
- Easy to extend with new features;

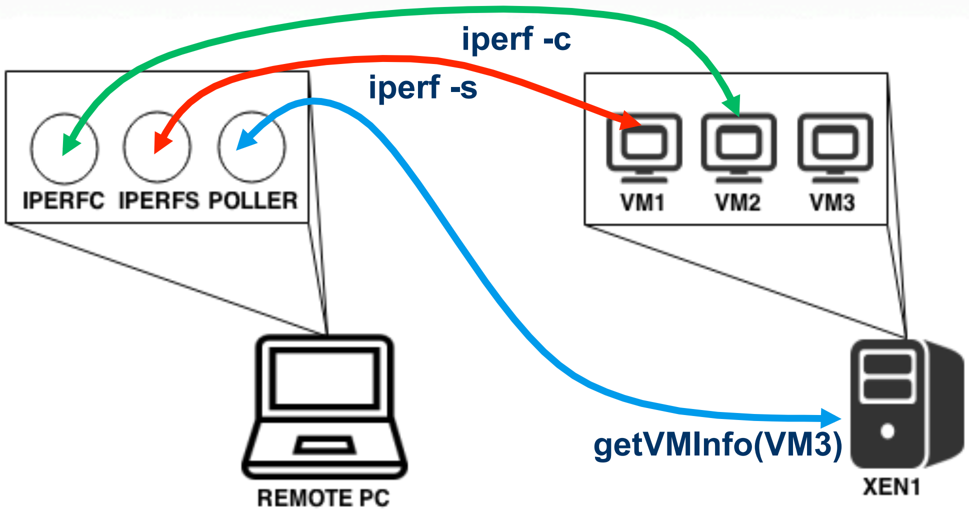
Testbed Deployer Library



```
//Deploy of Full Mesh Topology
. . . // Initialization section of temporary variable
testbed = TestbedOFELIA("ofelia.map") //We create Testbed object passing as
parameter the mapping file
for i in range (0, size): // the size parameter is the dimension of the mesh topology
   oshi = testbed.addOshi(name) // Add a new OSHI to the experiment. Return an
Oshi object
    for lhs in oshis:
   testbed.addPPLink(lhs_name, oshi_name) //Add an overlay link among lhs
and oshi. Return a composite object, that represents in our case an overlay link.
             oshis.append(oshi)
ctrl = testbed.addController(name, OF_tcp_port) // Add a controller to the
experiment. Return a Controller object
testbed.addPPLink(oshi, ctrl) //Connect the Controller to the overlay network
testbed.configure() //Generate the configuration file for the overlay topology to
deploy
```

- Able to start in a few minutes through config. script whatever experimental topology;
- Nodes ready to start the experiments;
- Easy to extend with new features;

Measurement Tools

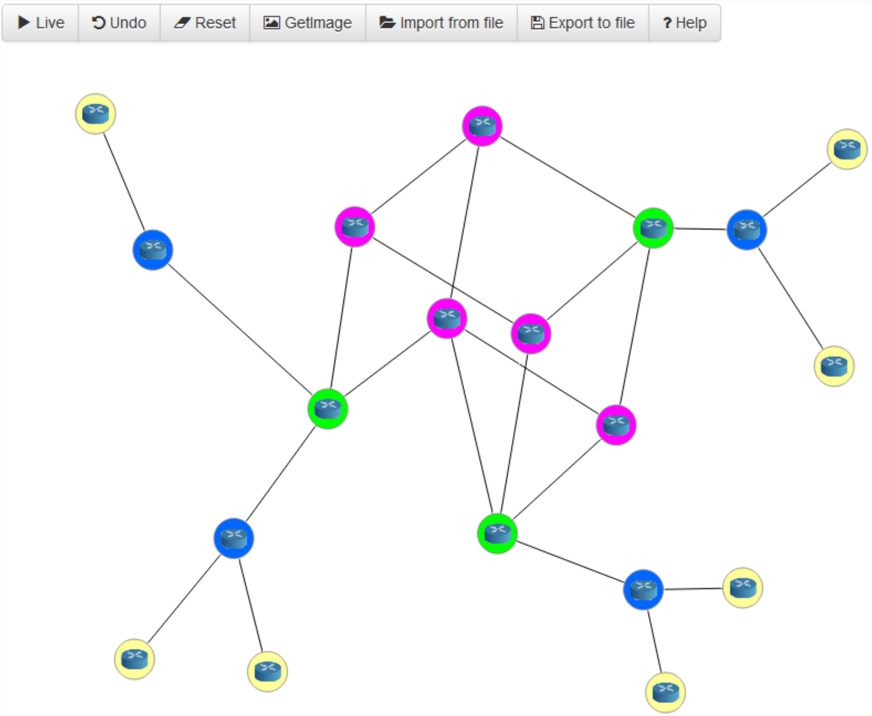



```
... // Load login info;
c1 = iperfClient([ipClient1, username, password]); // Create
iperfClient;
s1 = iperfServer([ipServer1, username, password]); //Create
iperfServer;
s1.start() ; // Start the server;
ex1 = Experiment(5, c1, s1, [ipSOv, "5","4"], {ipXEN:["Node4",]);
// Create an experiment . Params : n ° run, client, server, IP
Server in Overlay  network, iperf rate, IP Xen, VM to monitor;
ex1.start()
ex2 = Experiment(5, c1, s1, [ipSOv, "5",“8"], {ipXEN:[“EUH1",]);
ex2.start()
... // Close experiment, Client and Server;
```

- **Objectives:**
  - Gather load information about VMs;
  - Send traffic probing;
- **Issues:**
  - top tool from within the VMs is not reliable;

- **Measurement Tools is the solution; a OO multithreading python library:**
  - Gathers load information using xentop command;
  - Runs via ssh iperf commands;
  - Allows to run concurrently multiple experiments;

Performance evaluation



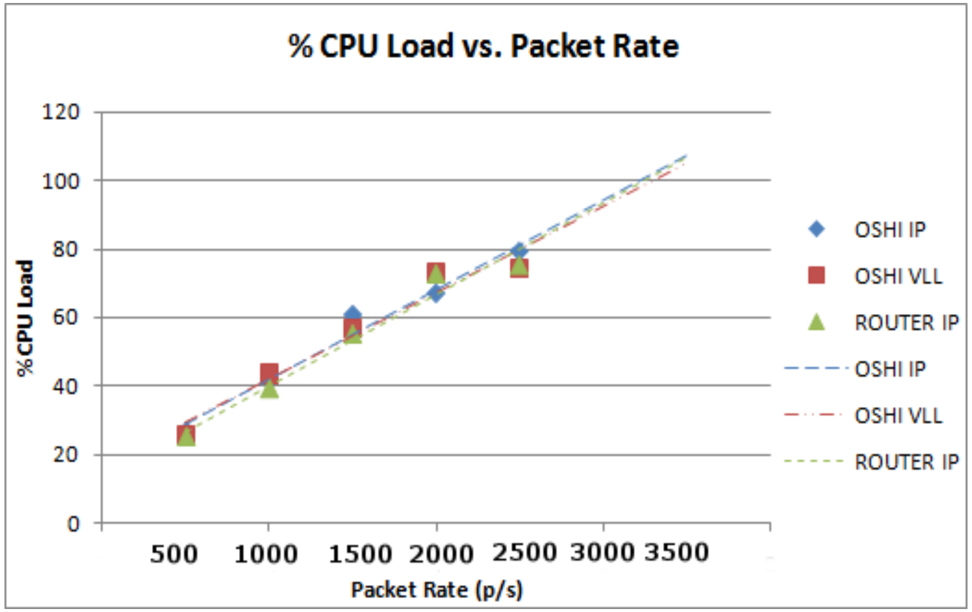
- Mininet Emulator;
- 7 EUHs, 4 Switches, 3 Access OSHIs and 5 Core OSHIs in a laptop;
- Evaluation of TCP throughput;
- Comparing IP vs. VLL solution;

#	VLL (Mb/s)	IP (Mb/s)	% GAIN
AVG	1555	1150	26,04 %
STD DEV	21,8	20	#

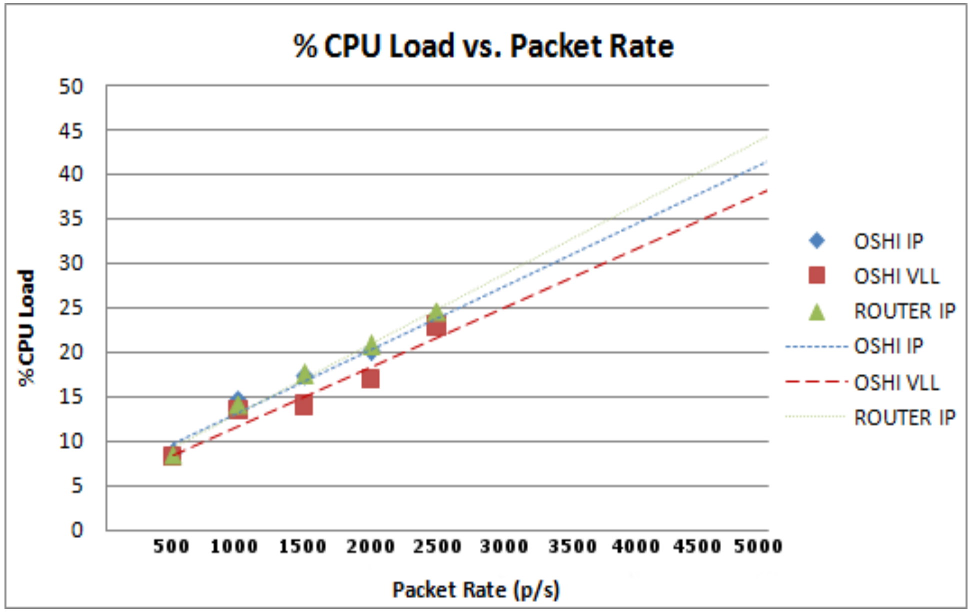
- TCP throughput is limited by sum of CPU load;
- Label switching is less expensive than IP forw.;

Performance evaluation (2)

OpenVPN tunnels

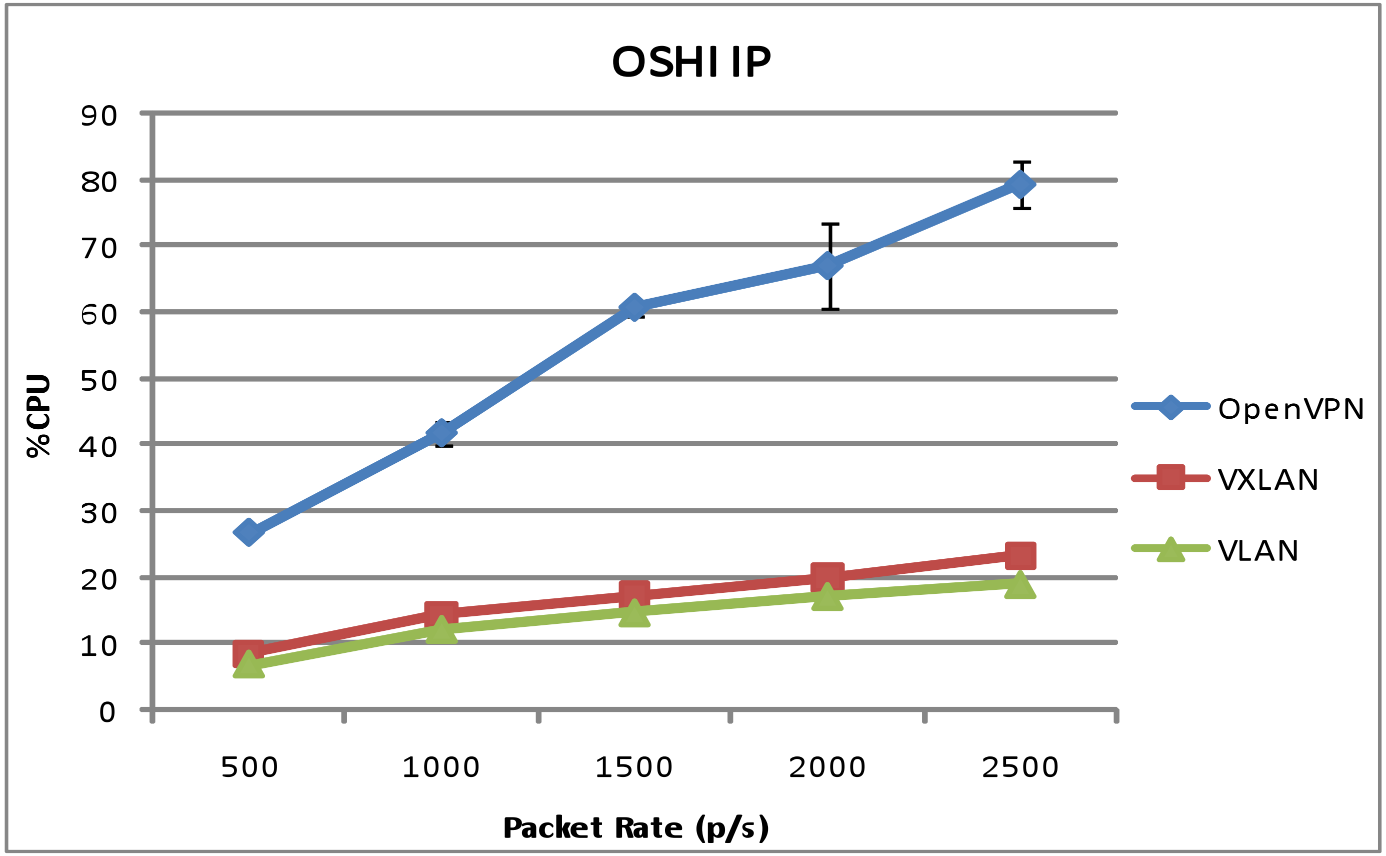


VXLAN tunnels



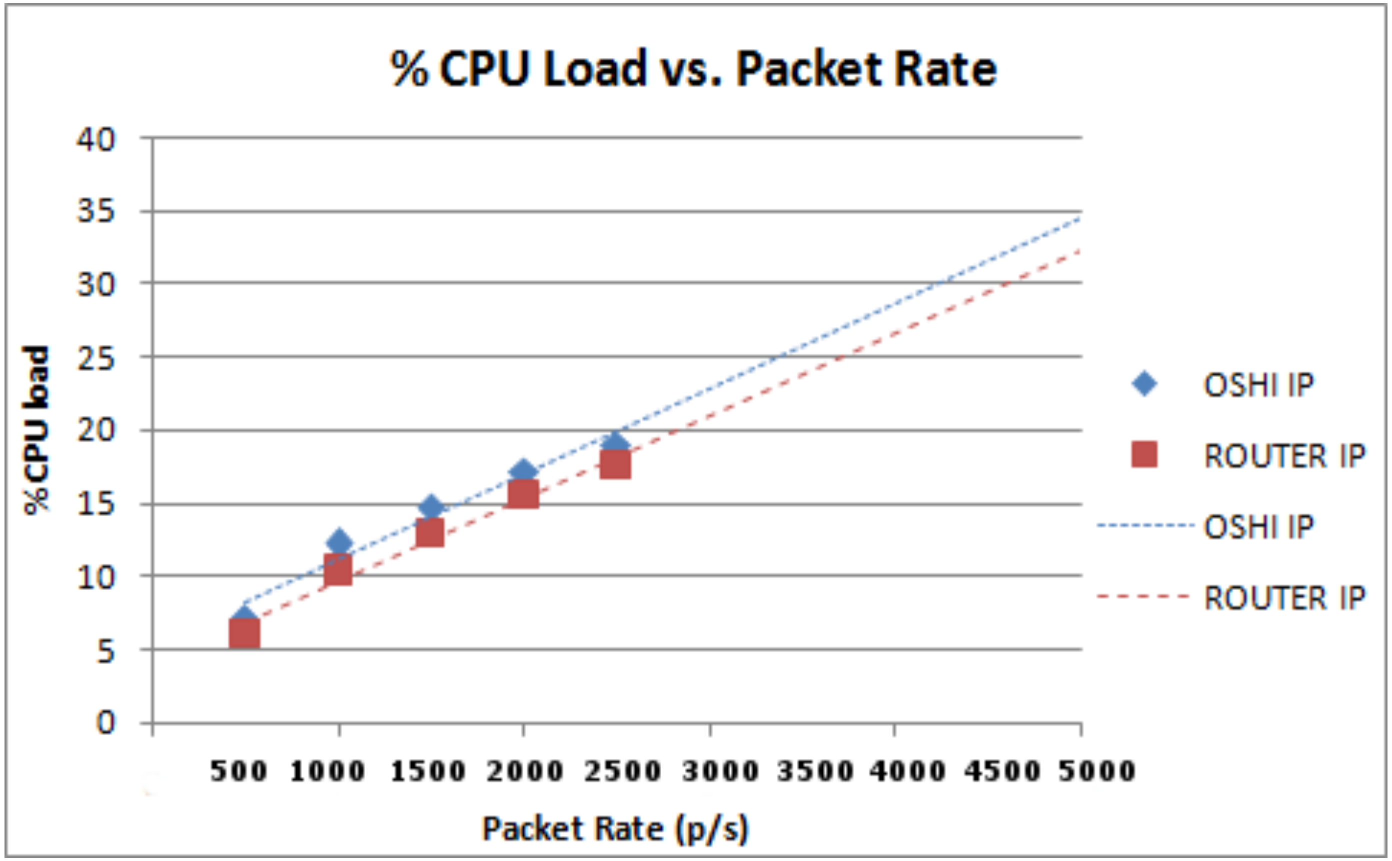
Performance evaluation (3)

OpenVPN vs. VXLAN vs. VLAN



Performance evaluation (4)

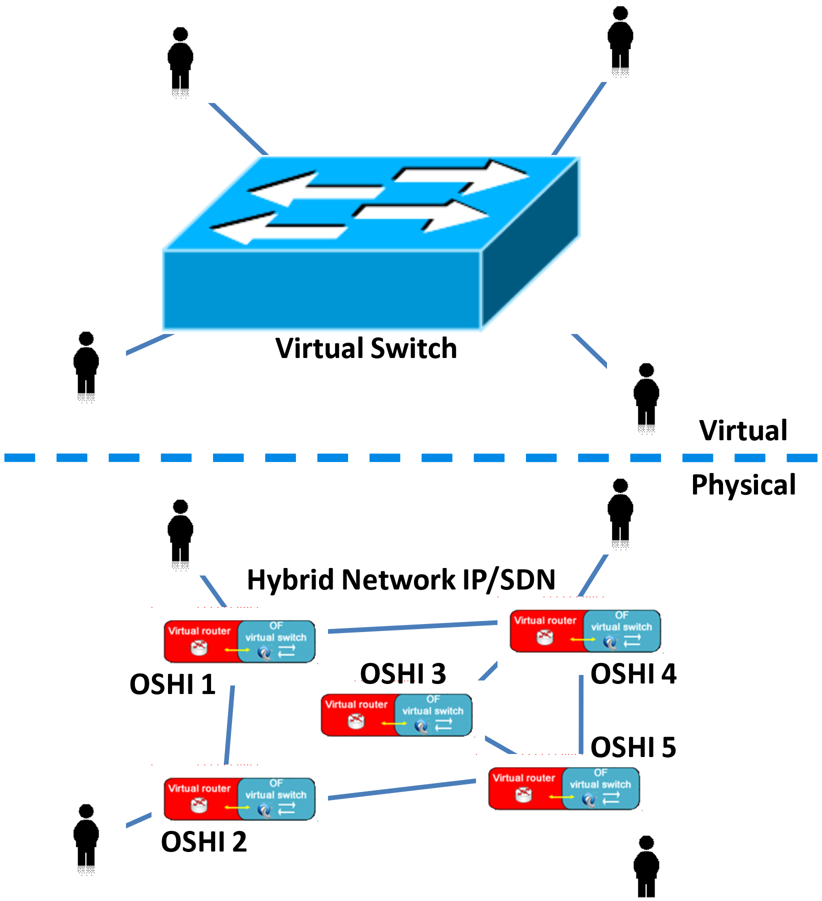
VLAN – No tunnels



Second year plan

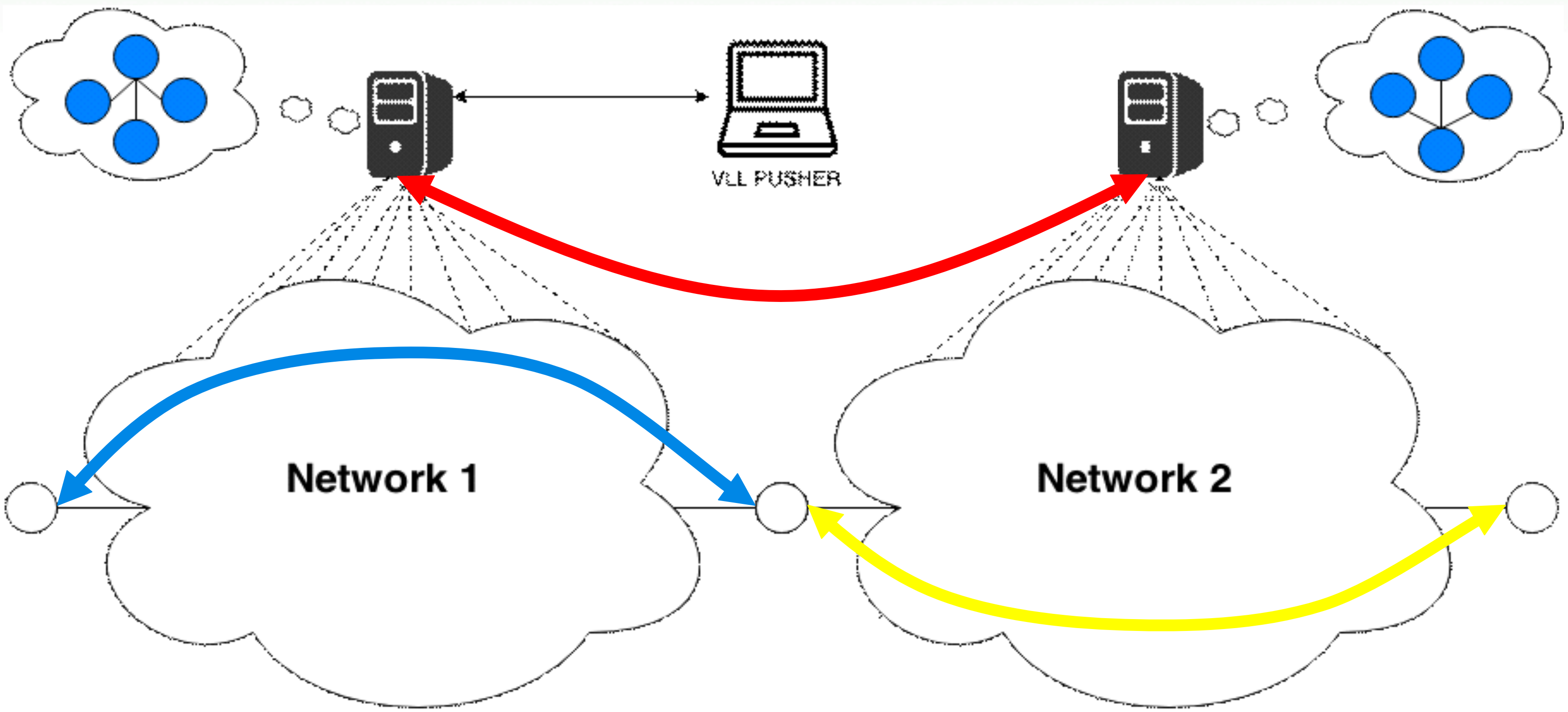
1. Deployment of new services;
2. Control plane resiliency in a WAN;
3. SDN exchange;

Ethernet Virtual Switch



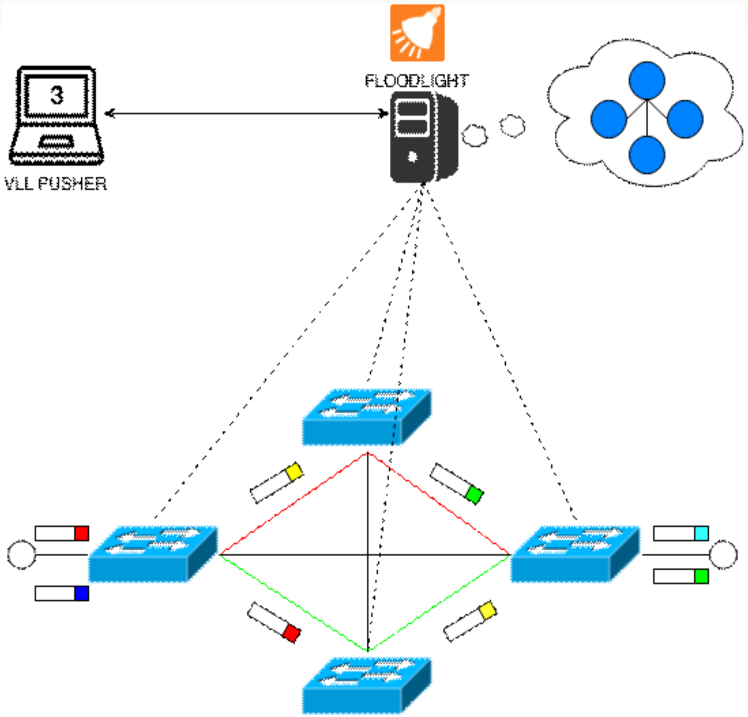
- OSHI network acts as an Ethernet Virtual Switch;
- More than one SBT using VLAN tags switching;
- SBTs “bridged” in a designed point in the network;

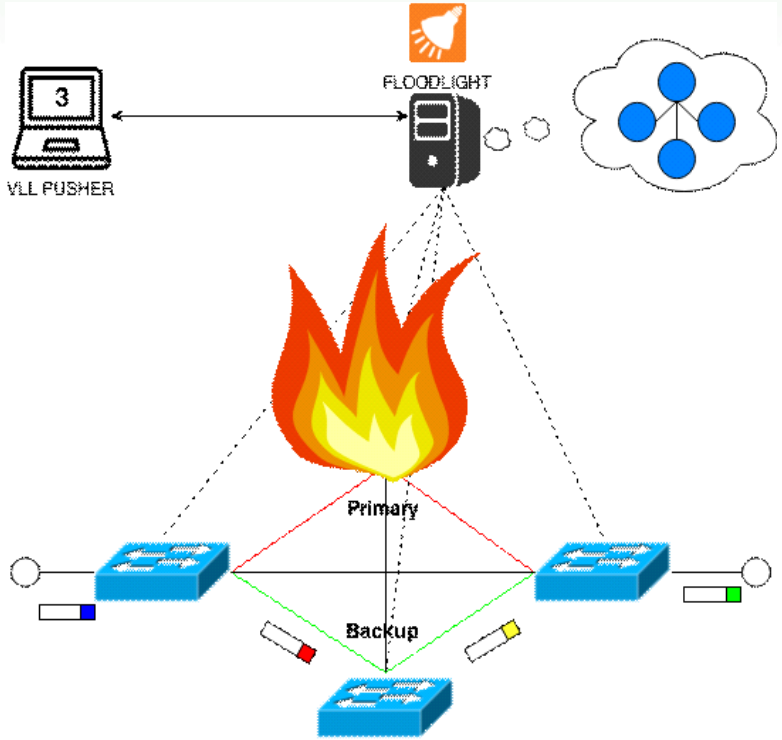
Multi – domain services



- **VLL service cannot work in a multi – domain scenario:**
  - Necessary a collaboration among "Controllers";

TE and Data Plane resiliency

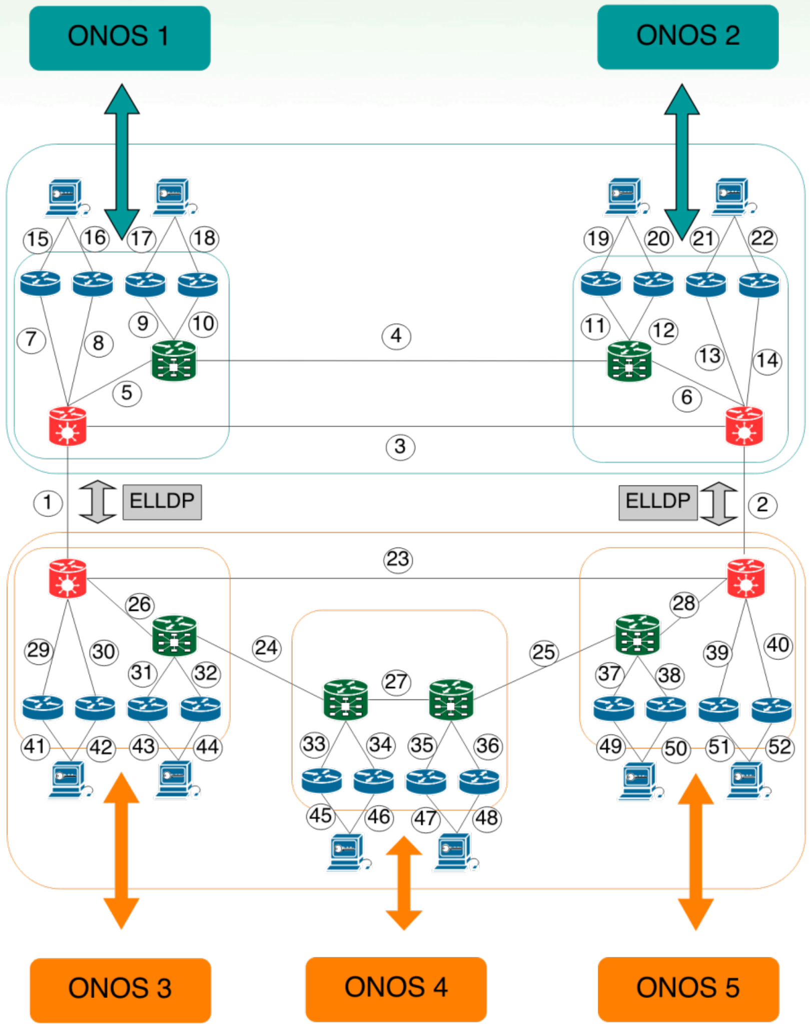




- **VLL service is "Routed" and a Single point of failure:**
  - Addition of SBP Traffic Engineering;
  - SBT backup to guarantee data plan resiliency;

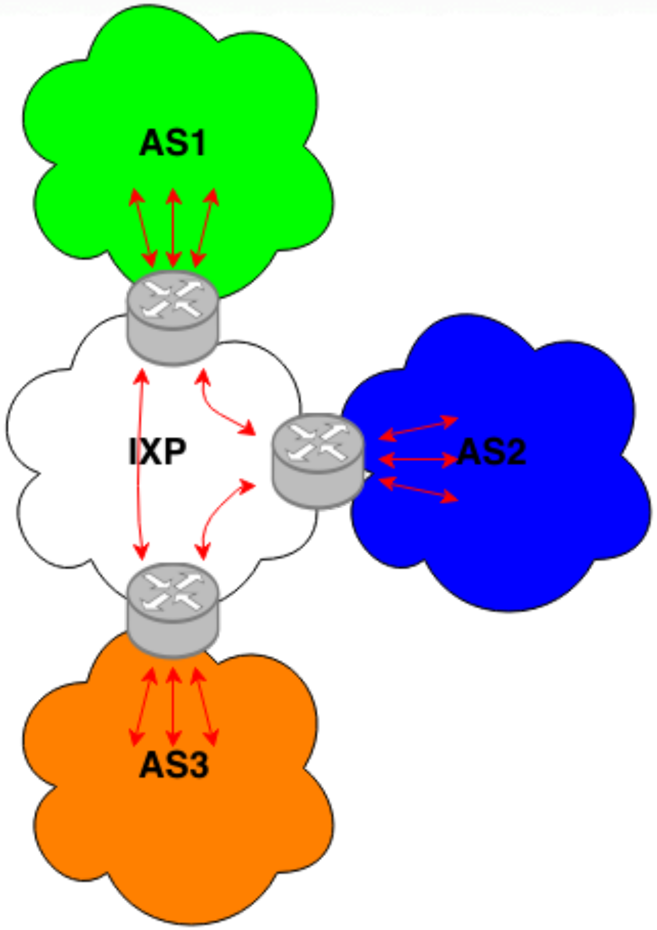
Solution for Control Plane resiliency

- Based on network operating system ONOS [17];
- ONOS provides high-available and scalable control plane;
- ONOS is not designed for a WAN scenario;
- Our approach combines multiple ONOS clusters (blue and orange);
- ISP network is logically partitioned and each cluster controls a single region;
- Clusters cooperate and share a partial view of the network through BGP;



IXP + SDN = SDN exchange

- Infrastructure through which ASs exchange traffic between them;
- BGP's domain;
- Aim of this work is to investigate how an IXP based on SDN control plane can:
  - Provide the same functionality of today's IXP;
  - Augment the capability of an IXP thanks to SDN;
- 4 steps :
  - Identify requirements of an IXP;
  - Define the state of art;
  - Deployment of a realistic use case and evaluation;
  - Extend functionality of selected project;



Potential impact on GARR

- Extending the results of DREAMER project;
- “Extension of OSHI services”: deals with services of interest for a NREN;
- “Control plane resiliency in a WAN scenario”: needed to extend SDN paradigm beyond the Data Center use case;
- “SDN exchange”: application of SDN infrastructure in a IXP scenario;

Questions ?

References

1. J. Rexford, "Enabling Innovation Inside The Network", ACM 2012;
2. S. Salsano, "Software Defined Networking And OpenFlow", Course TPIN 2013-2014;
3. ONF, "Software- Defined Networking: The New Norm For Networks", ONF White Paper 2012;
4. Floodlight's homepage – http://www.projectfloodlight.org/floodlight/;
5. Open vSwitch homepage – http://openvswitch.org/;
6. Mininet homepage – http://mininet.org/;
7. Quagga homepage – http://www.nongnu.org/quagga/
8. J.Petit and E. Lopez: "OpenStack: OVS Deep Dive" – November 2013;
9. Networkx homepage - http://networkx.github.io/
10. J. Kempf, et al "OpenFlow MPLS and the open source label switched router".2011
11. M. R. Nascimento, et al."Virtual routers as a service: the Routeflow approach leveraging software-defined networks". 6th Conference on Future Internet Technologies;
12. Pingping Lin et al. "Seamless Interworking of SDN and IP" Demo at SIGCOMM 2013
13. OpenFlow Switch Specification Version 1.3.1 (Wire Protocol 0x05) September 6, 2012
14. S. Jain, et al, "B4: Experience with a Globally-Deployed Software Defined WAN" in SIGCOMM, 2013
15. OSHI homepage – http://netgroup.uniroma2.it/OSHI
16. VXLAN draft – http://tools.ietf.org/pdf/draft-mahalingam-dutt-dcops-vxlan-01.pdf
17. ONOS – http://onlab.us/tools.html#os
18. Laurent Vanbever – "Novel Applications for a SDN-enabled Internet Exchange Point"
19. Feamster et al. – "SDX: A Software Defined Internet Exchange"
20. Gupta, Shahbaz, Vanbever et al. – SDX: A Software Defined Internet Exchange
21. Dean Pemberton – Project Cardigan: a SDN Controlled Exchange Fabric
22. Stringer, Fu et al. – Cardigan: Deploying a Distributed Routing Fabric
23. SDX homepage – https://github.com/sdn-ixp/