RouteFlow

Virtualized IP Routing Services in OpenFlow networks

Agenda

- Background: OpenFlow, Logical/Virtual Routers, Network Virtualization
- Project Overview
- Motivation
- Architecture
  - Controller
  - Server
  - Slave
  - Protocol
- Evaluation
- Work ahead
- Demo and hands-on Tutorial

RouteFlow

The Project

RouteFlow

RouteFlow is an open-source project to provide IP routing & forwarding services in OpenFlow networks

CPqD

Marcelo Nascimento

Christian E. Rothenberg

Marcos Salvador

Eder Leao Fernandes

Rodrigo Denicol

Alisson Soares

UniRio

Carlos Corrêa

Sidney Lucena

Unicamp

Mauricio Magalhães

Indiana University

Stanford University

UFSCAR

UFPA

...

CPqD · GIGA REDE EXPERIMENTAL DE ALTA VELOCIDADE · FINEP · FUNTTEL

About CPqD

- Major telecom R&D center in LATAM with expertise in various areas:
  - Optical (WDM, PON), Wireless (WiMax, LTE), IP (IMS/NGN, OpenFlow), OSS/BSS, Digital TV…
  - Today with ~1200 highly-skilled employees
- Created in 1976 as R&D branch of Telebras - Brazilian telecom monopoly
- Private foundation since 1998 after Telebras was privatized
- Purpose to foster innovation to help (mainly) Brazilian companies and society
  - Focus on technology R&D
  - Bridge the gap between universities and the industry
- Near highly-ranked universities in Brazil
  - History of collaborations

RouteFlow

About the GIGA Project Testbed

FINEP
Ciência e Tecnologia
Ministério da Ciência e Tecnologia

FUNTTEL
Comunicações
Ministério das Comunicações

- 800km total fiber span over 7 cities in 2 states (SP, RJ)
- 66 labs from 26 institutions connected (fiber to the lab) at 1 and 10 Gbps
- Manually provisioned (VLAN) circuits for stable traffic
- e2e dynamic (VLAN) multidomain protected circuits for L2 and above on demand experiments
- Manually provisioned wavelengths for L1 and above experimentation
- Focus on technology R&D and the industry

RouteFlow Project Timeline

Jan / 2010 — Aug / 2010 — Dez / 2010 — May / 2011 — Oct / 2011 — Nov / 2011 — ... / 2012

- Start Msc. Thesis work by Marcelo N.
- First Prototype
- First Short-Paper @ WPEIF
- Evalaluation on NetFPGA testbed
- QuagFlow Poster @ SIGCOMM
- Open-Source Release
- Demos @ ONS11
- Tutorial & Demo @ OFELIA/CHANGE SS
- Indiana University - Pronto OF switches + BGP peering with Juniper MX
- Demo @ SuperComputing 11
- Nation-wide field trials (Brazil, Internet2) @ ONS12
- RouteFlow + OpenStack (Cloud RouteFlow)

RouteFlow

… building a community

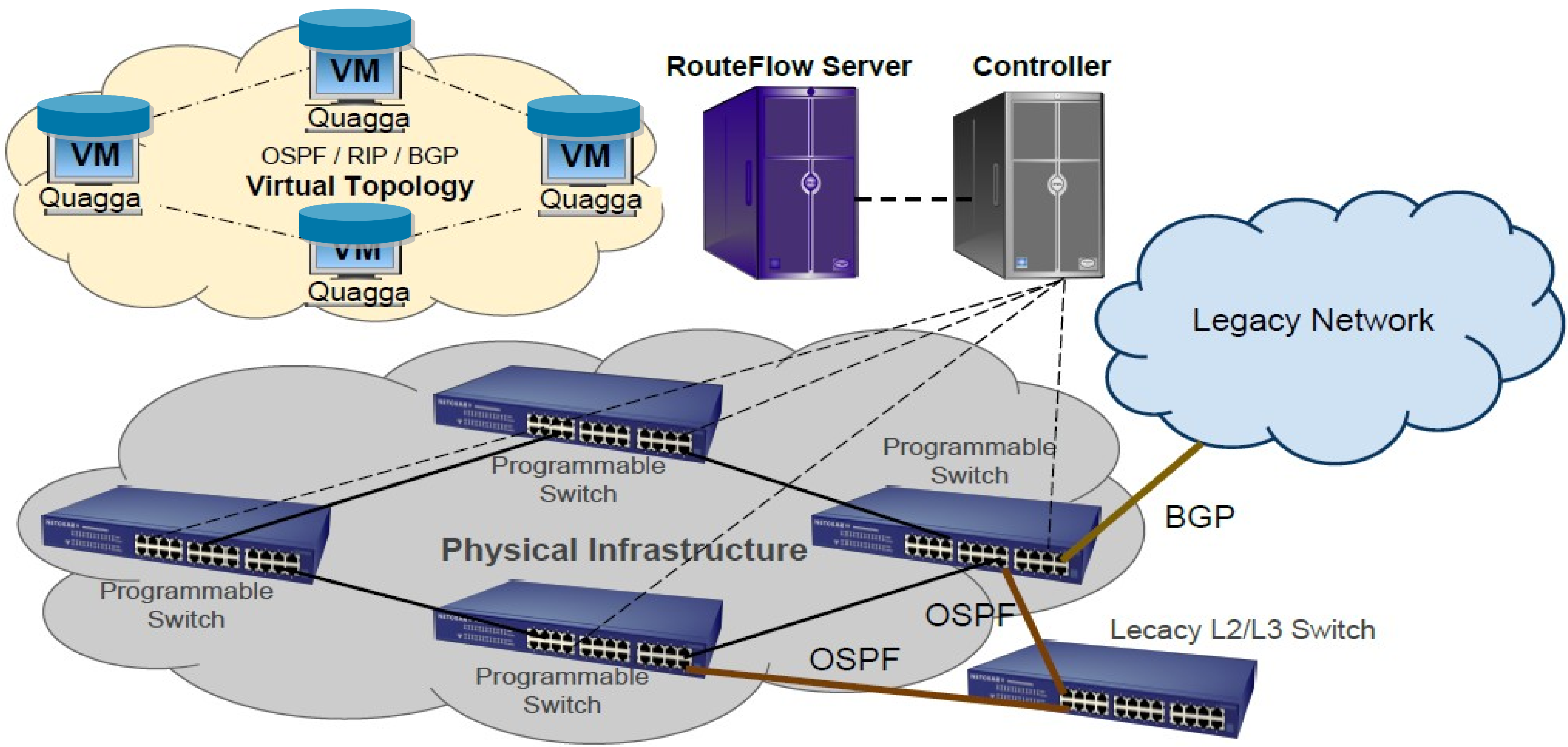
1,390 visits came from 333 cities

189
days since
Project Launch

RouteFlow

http://go.cpqd.com.br/routeflow/
Apache-licensed code @ github

Overview



VM
Quagga
OSPF / RIP / BGP
Virtual Topology
RouteFlow Server
Controller
Legacy Network
Programmable Switch
Physical Infrastructure
BGP
OSPF
OSPF
Lecacy L2/L3 Switch

Motivation v1

Original motivation around RouteFlow (formerly QuagFlow)

(Seeded in experience building a Broadcom-based L2/L3 switch prototype)

- Current “mainframe” model of networking equipment:
  - Costly systems based on proprietary HW and closed SW;
  - Lack of programmability limits cutomization and in-house innovation;
  - Ossified architectures.

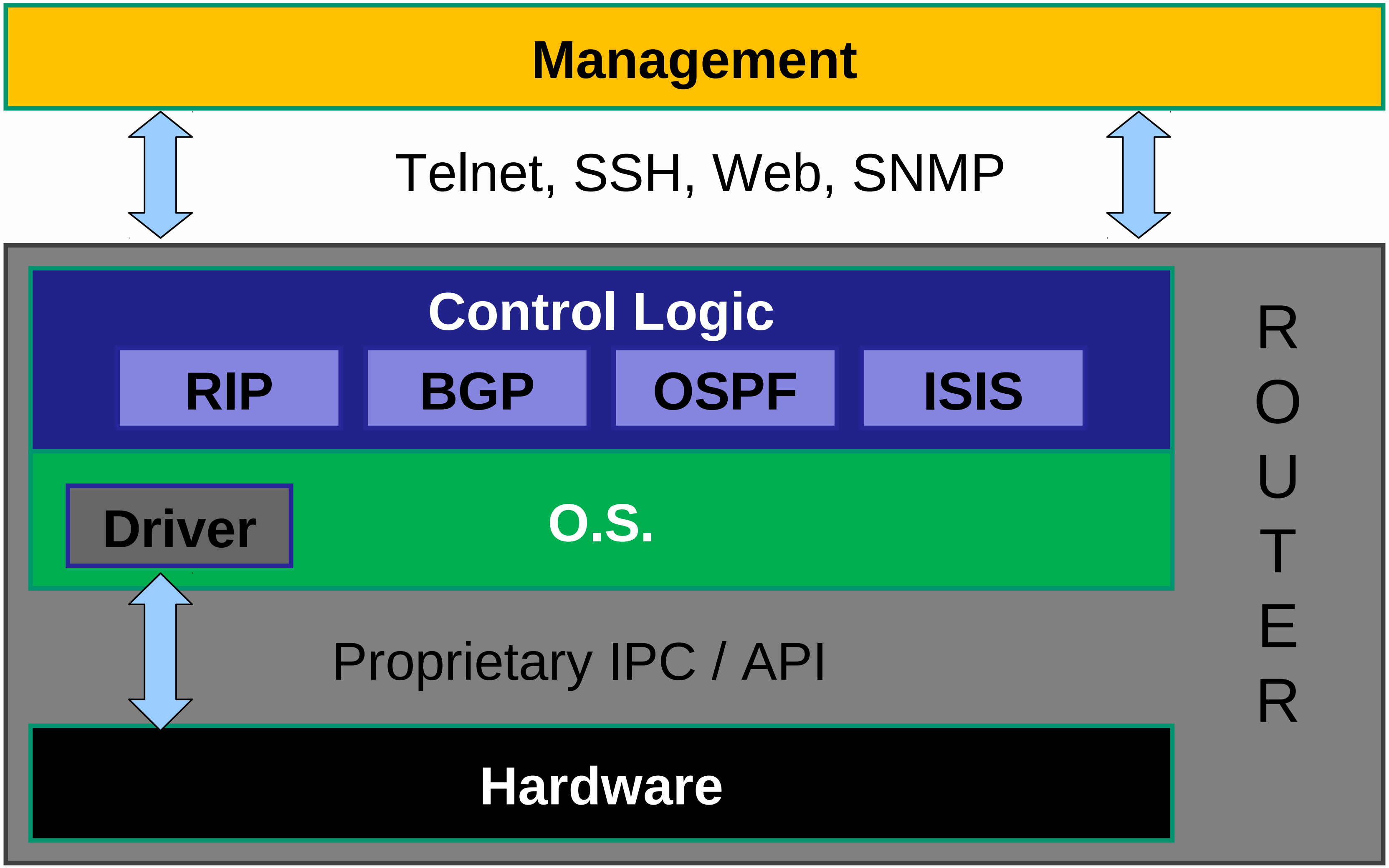
- Goal: Open commodity routing solutions:

  + open-source routing protocol stacks (e.g. Quagga)

  + commercial networking HW with open API (i.e. OpenFlow)

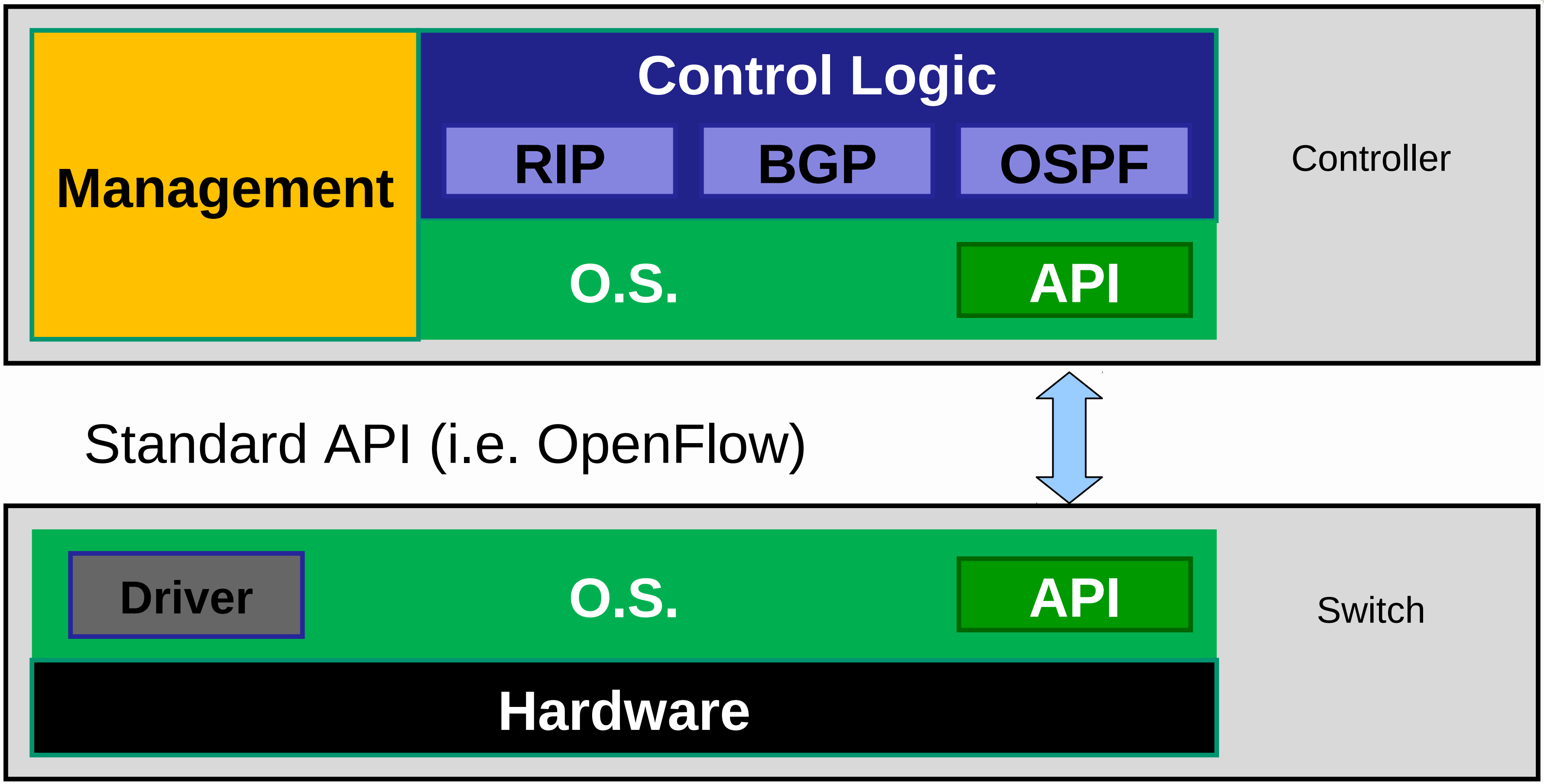
  = line-rate performace, cost-efficiency, and flexibility!

Current router architectures



RouteFlow

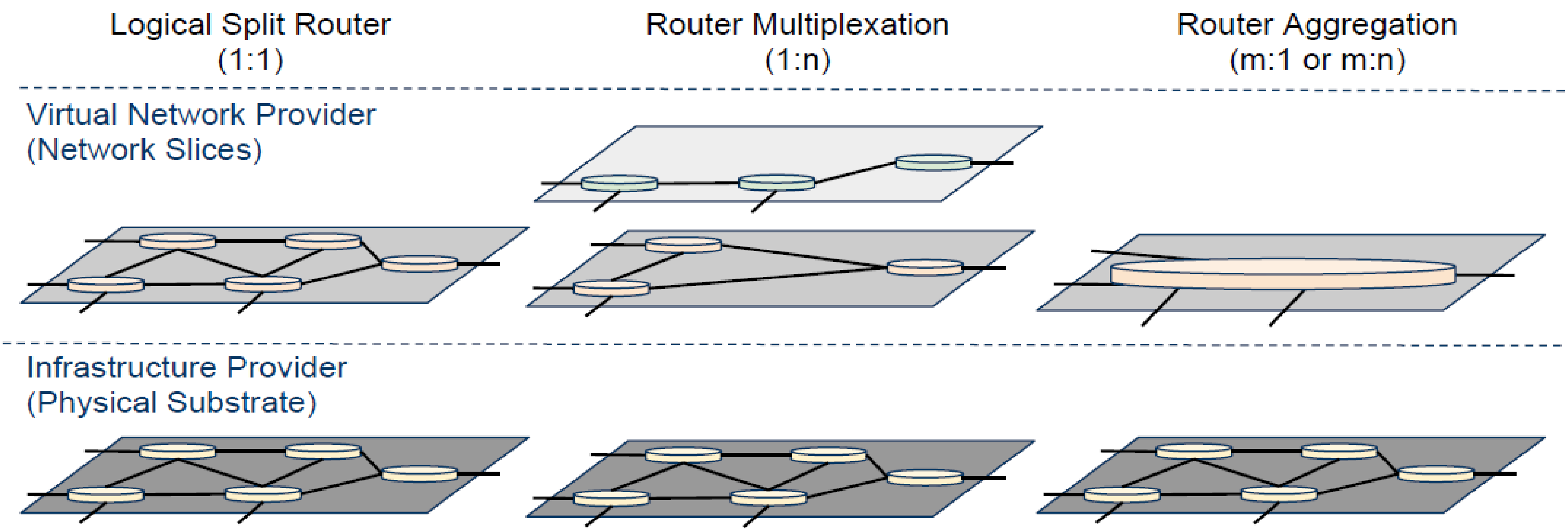
OpenFlow model



Motivation v2

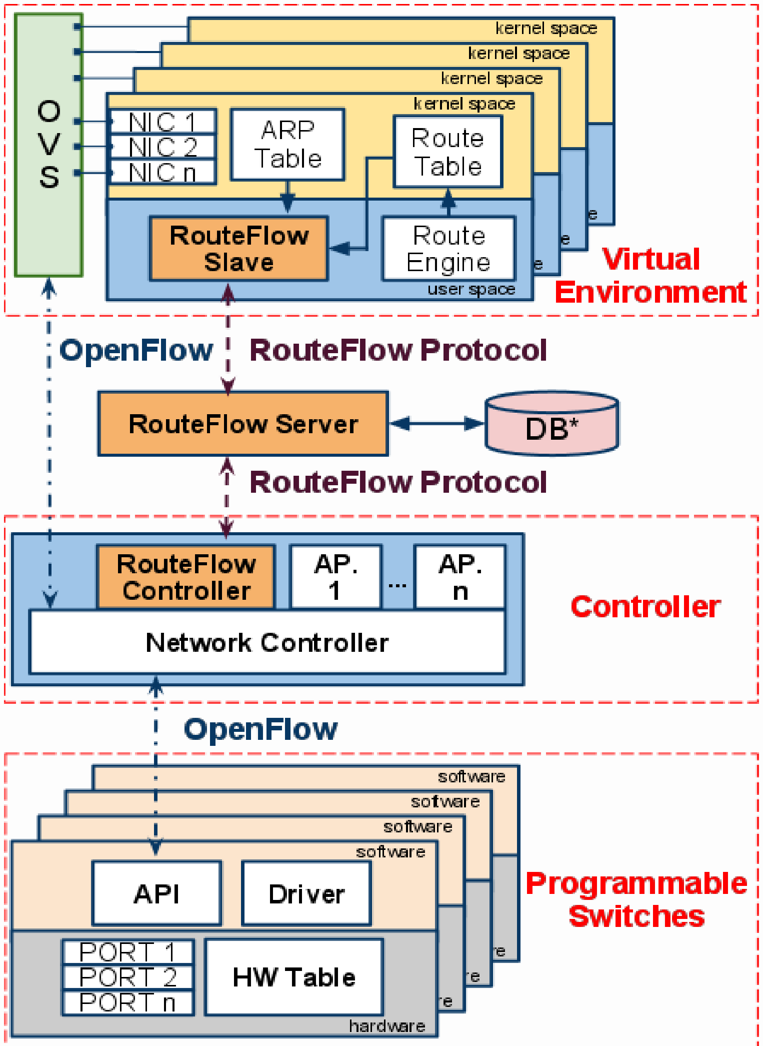
- A transition path, incrementally deployable:
  from current IP networks to SDN
  - Hybrid modes of operations: traditional IP control planes along SDN
- Innovation around IP control planes
  - Simplified network mgm, protocol optimization, shadow networks
- Advancing IP Network Virtualization
  - From flexible Virtual Routers to IP Network-as-a-Service

Use Cases



RouteFlow

Architecture



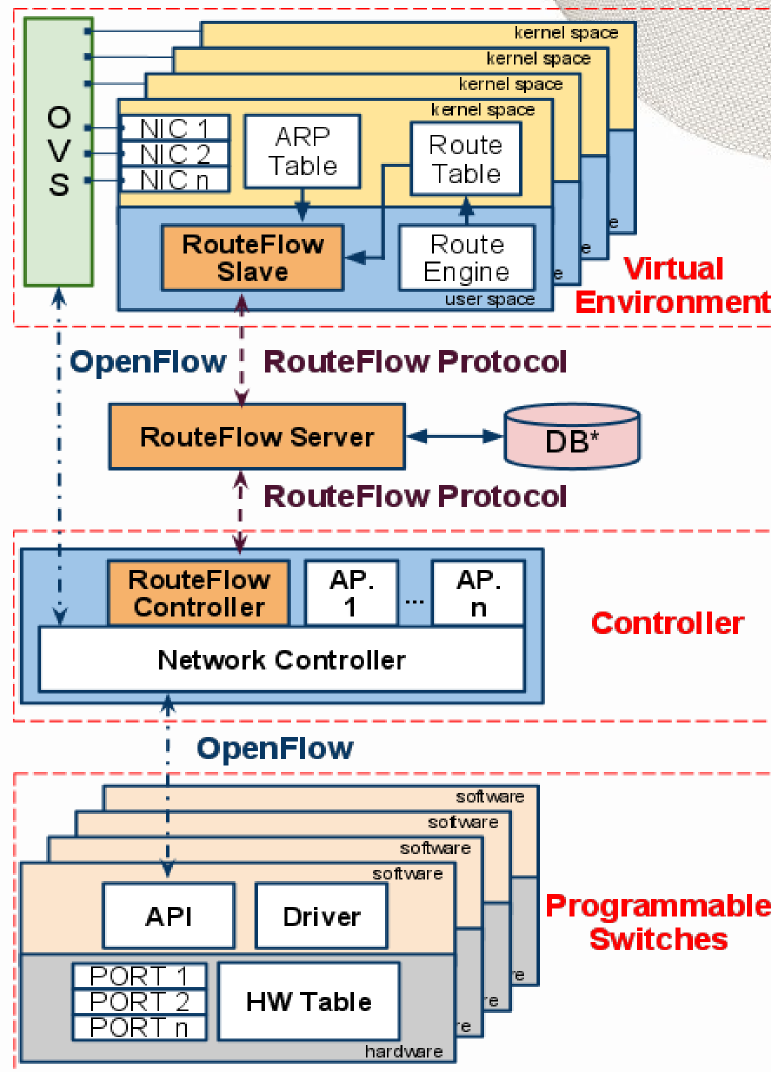
Key Features

- Separation of data and control planes;
- Loosely coupled architecture:
  - Three RF components:
    1. Controller, 2. Server, 3. Slave(s)
- Unmodified routing protocol stacks;
  - Routing protocol messages can be sent 'down' or kept in the virtual environment;
- Portable to multiple controllers:
  - RF-Controller acts as a “proxy” app.
- Multi-virtualization technologies
- Multi-vendor data plane hardware

RouteFlow

RF-Controller application

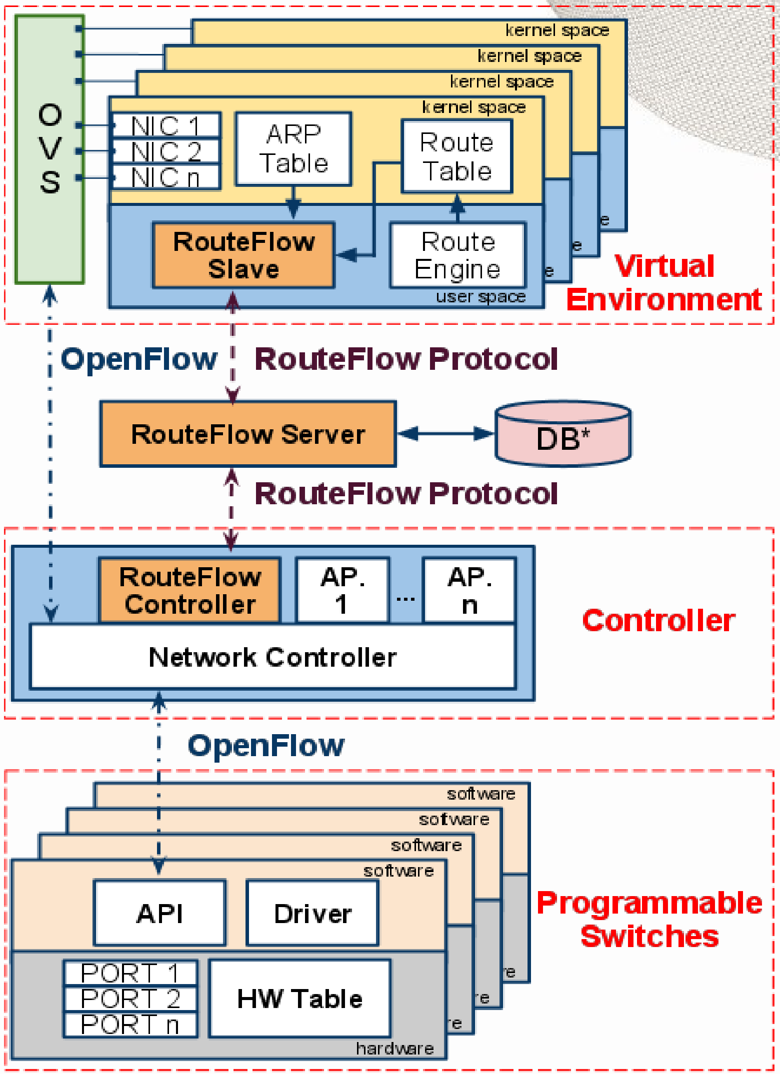
- Shim application on an OpenFlow controller
- Mainly acts like a proxy for the OpenFlow API
- Interacts with the OpenFlow datapaths
- Filters relevant events to the RF-Server
- Receives flow mod commands
- Delivers traffic to/from VM interfaces via OVS



RouteFlow

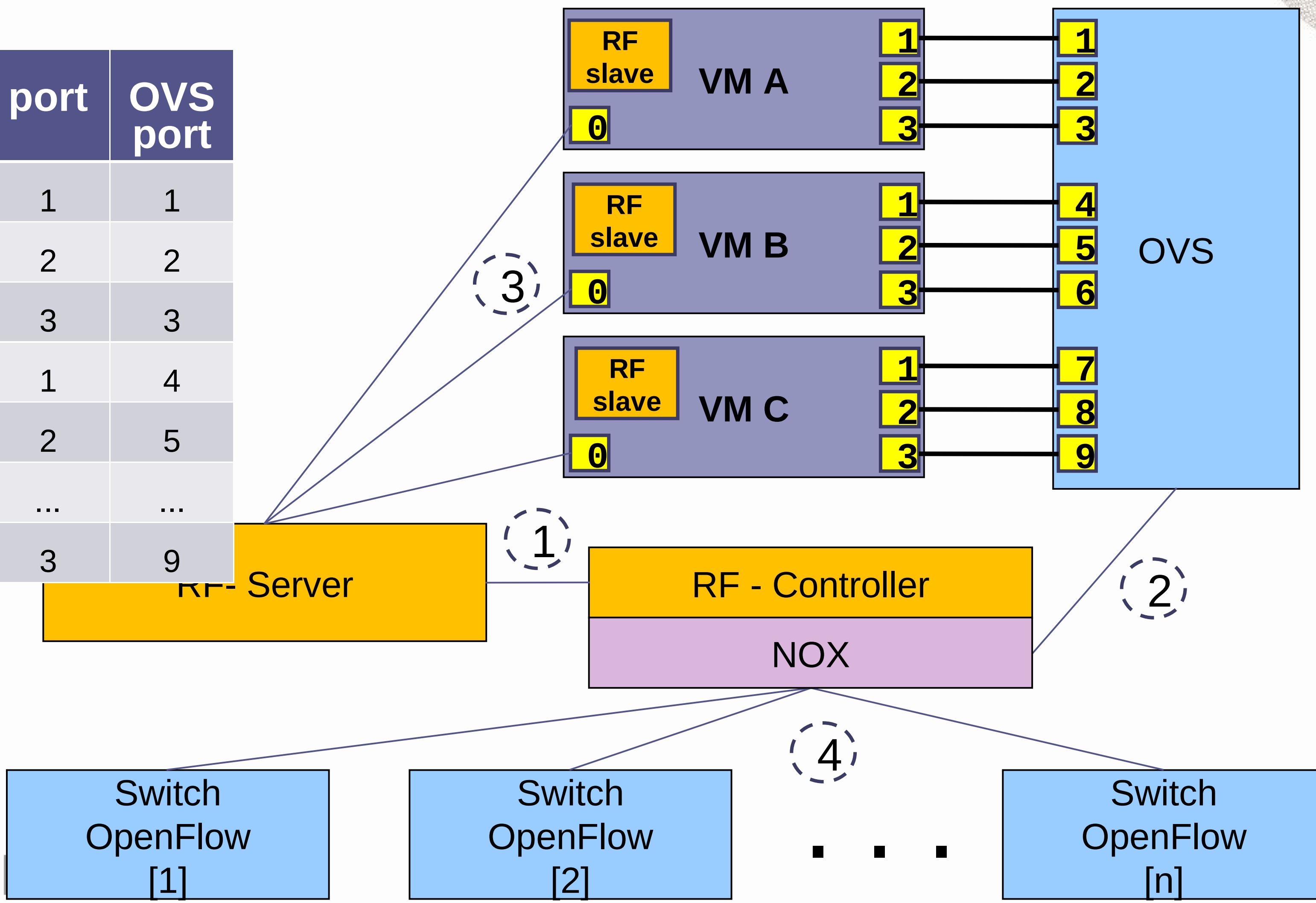
RouteFlow Server

- The “brain” of RouteFlow;
- Manages available virtual machines (VM);
- Configures the virtual environment
- Receives events from the RF- controller
  - Switch join/leave, packet-in;
- Associates VMs and OpenFlow switches;
- Determines packet delivery from/to VMs
- Requests flow installation / modification in OpenFlow switches.



RF-Server: Association of VMs and DPs

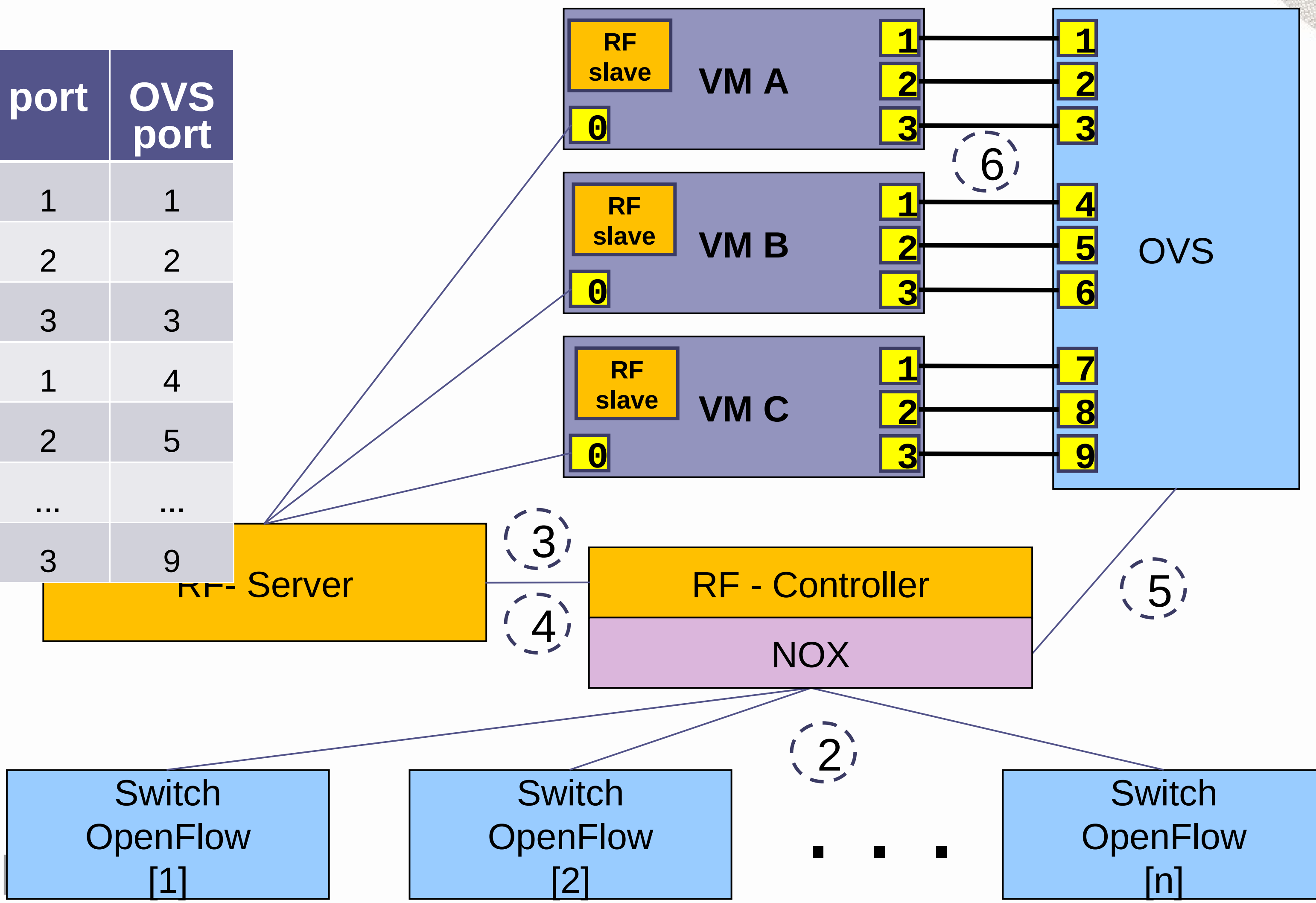
VM	DP	port	OVS port
A	1	1	1
A	1	2	2
A	1	3	3
B	2	1	4
B	2	2	5
...	...	...	...
C 5	3	3	9



RouteFl

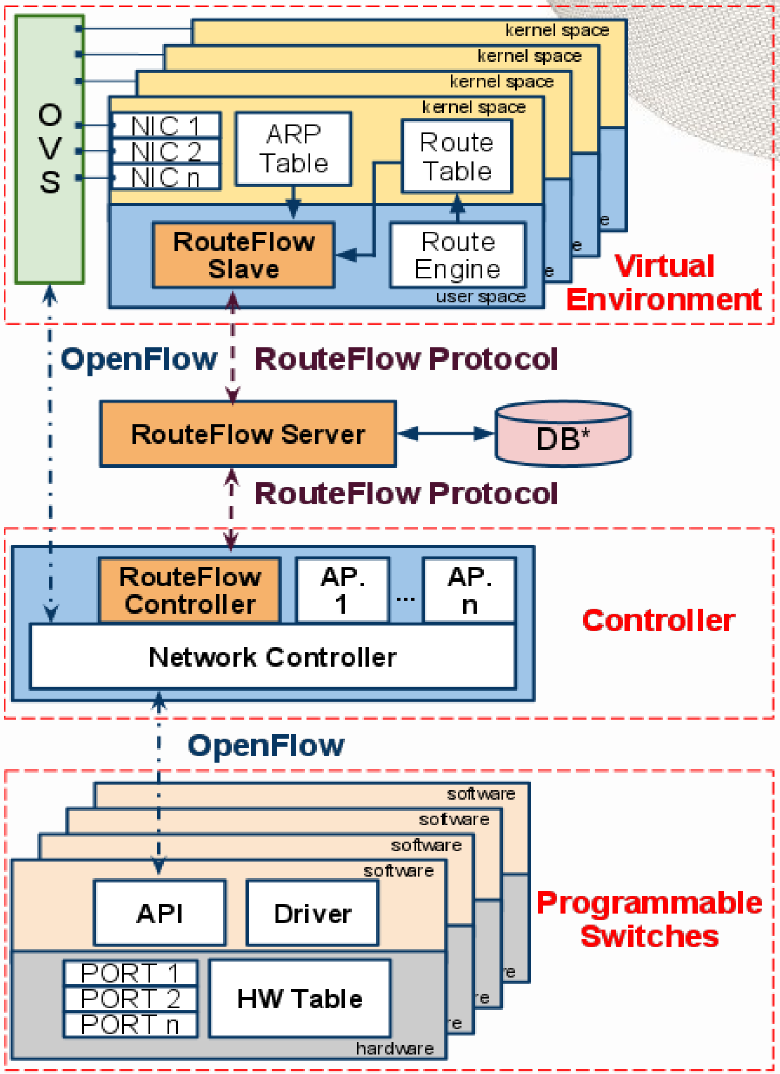
RF-Server: Flow of Routing Control Packets

VM	DP	port	OVS port
A	1	1	1
A	1	2	2
A	1	3	3
B	2	1	4
B	2	2	5
...	...	...	...
C	3	3	9

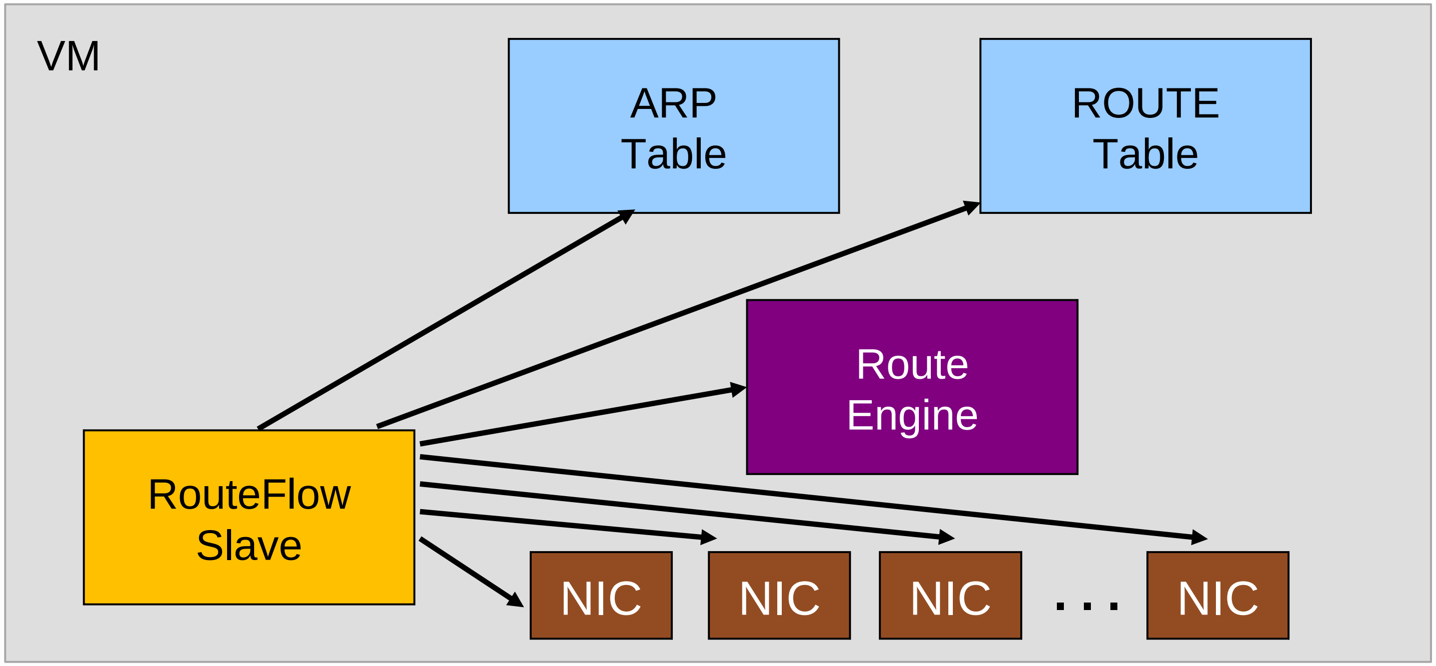


RouteFlow-Slave

- Runs as a daemon in Linux-based VM
- Registers the VM with the RF-Server
- Configures the VM (e.g., interfaces)
- Listens to ARP and IP table updates via Linux Netlink events
  - Linux Routing stack independent (Quagga, XORP)
- Translates routing updates into flow rules;
  - Match: DST_MAC + DST_IP + MASK
  - Actions: Re-write MACs + port-out
- Translates ARP entries into flow rules
  - Match: DST_MAC + DST_IP
  - Actions: Re-write MACs + port-out
- Sends flow update commands to RF-Server
- Runs VM-OVS attachment discovery protocol



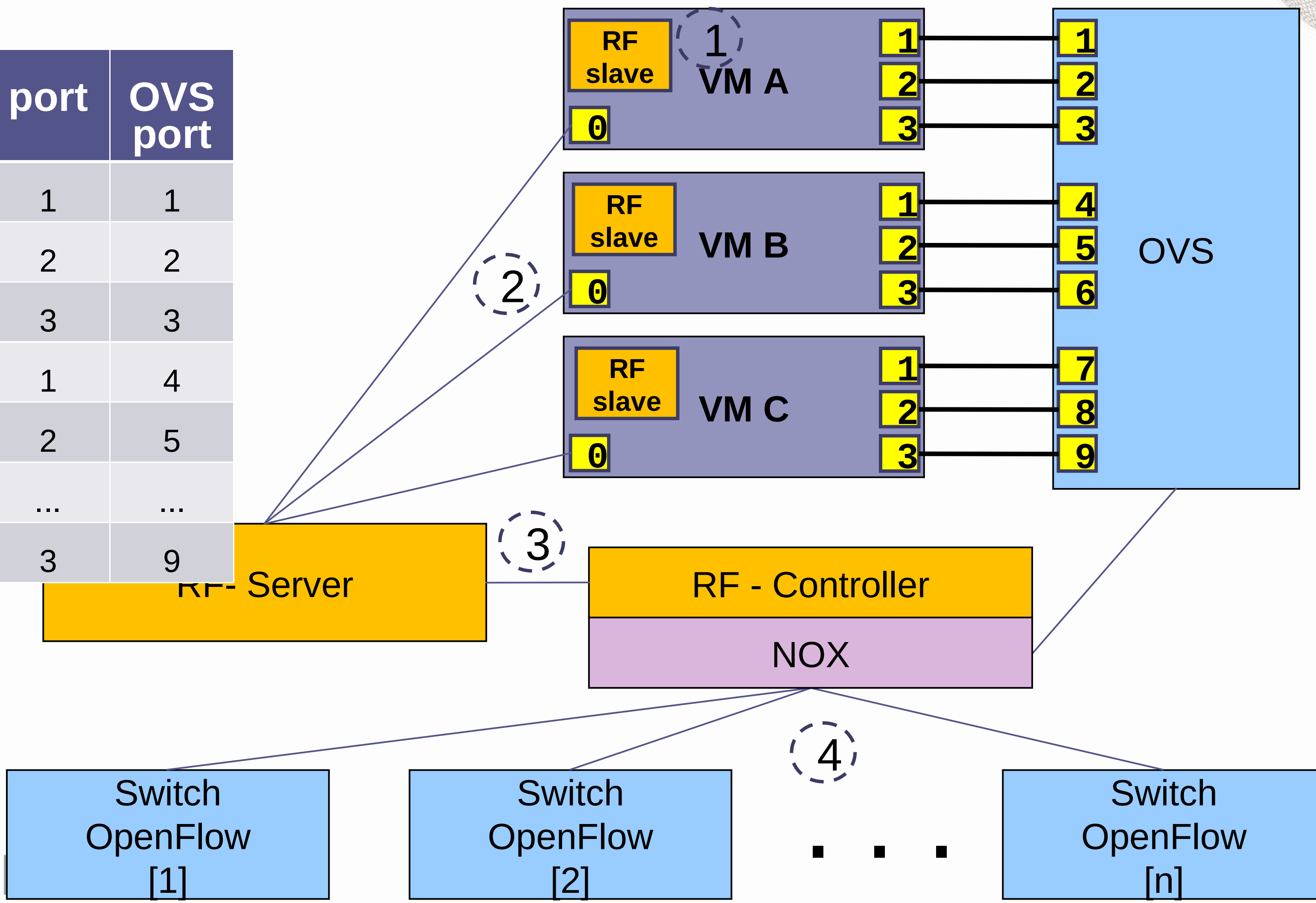
RF-Slave: VM configuration



- Cofigure the amount of interfaces (enable/disable);
- Start/Stop Routing Engine;
- Clean interface configuration and ARP/ROUTE tables

RF Add/Remove Routes

VM	DP	port	OVS port
A	1	1	1
A	1	2	2
A	1	3	3
B	2	1	4
B	2	2	5
...	...	...	...
C	3	3	9



RouteFlow

IP Forwarding Rules in OpenFlow

RF-Slave info from the Linux network stack

- Route =IP + MASK [Rede]+IP[Gateway]+Interface
- ARP= IP[Host]+MAC[Host]+Interface

OpenFlow 1.0 entry:

- Match: DST_MAC + DST_IP + SUBNET_MASK
- Actions:
  - Re-Write [SRC_MAC (Interface)], Re-Write [DST_MAC (Nexthop)]
  - Forward [Port-out(Interface)]

Longest Prefix Match (LPM)

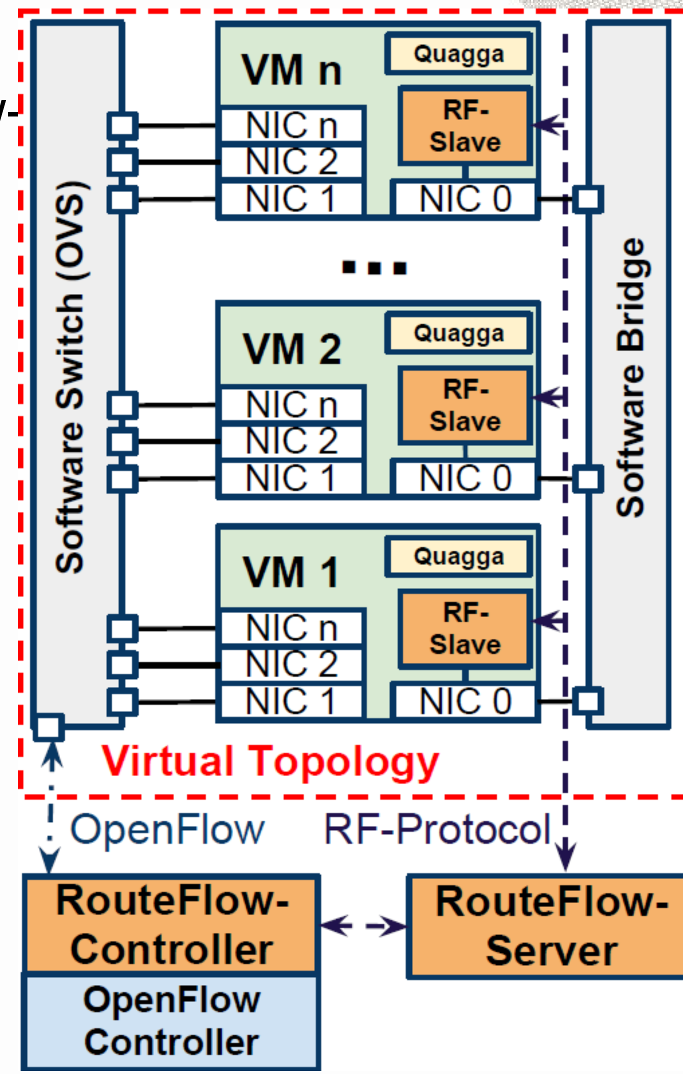
- Add priority to flow entry based on the length of the subnet mask

In OpenFlow 1.1:

- Addictional actions: TTL decrement, checksum update
- Multiple-Table: Table[0] Matches DST_MAC, Table[1] Matches DST_IP

Virtual Environment

- V1 used TUN/TAP devices and payload encapsulation in the RF-Protocol
- V2 manages VM connectivity through an OpenFlow-capable soft-switch
- Routing engines (e.g. Quagga) exchange routing protocol packets
  - Two modes of operation for VM packet exchange:
  - UP: Directly through the OVS (requires Topology Disc)
  - DOWN: Through the physical switches
- Centralized but logically distributed
  - Can be physically distributed
- Support of different virtualization technologies
  - From QEMU to LXC
- VM-OVS Attachment Discovery Protocol



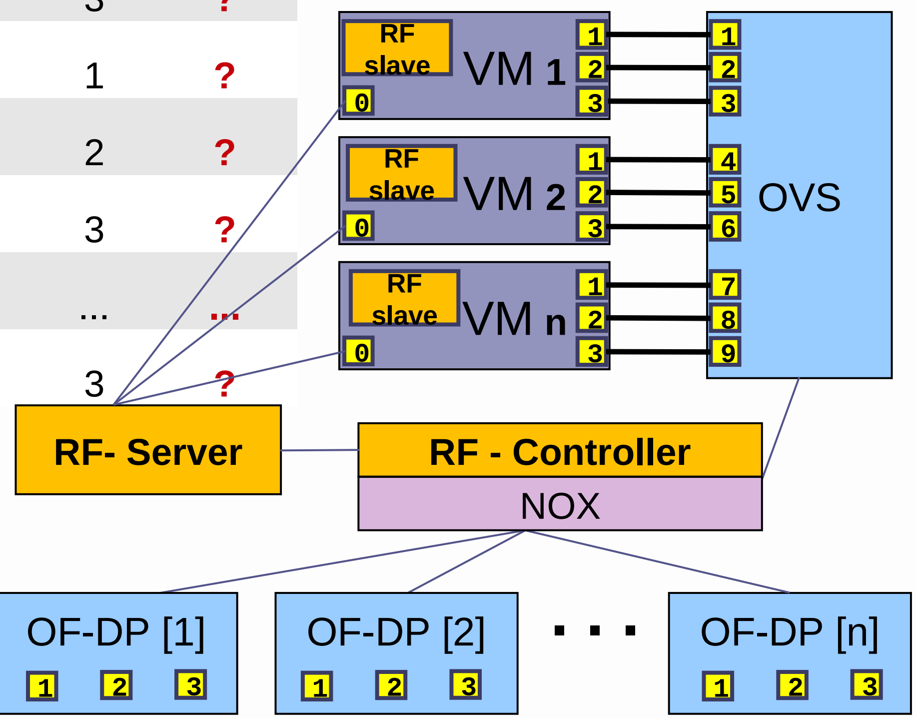
RouteFlow

RF-Slave: Interface Attachment discovery (1)

- Discovery of VM interfaces attachment to OVS.
- Virtual interfaces are dinamically attached to the OVS
  - No guarantee of order
  - VMs may have an arbitrary number of interfaces
- When VM registers to the RF-Server the OVS ports in use are unknown.

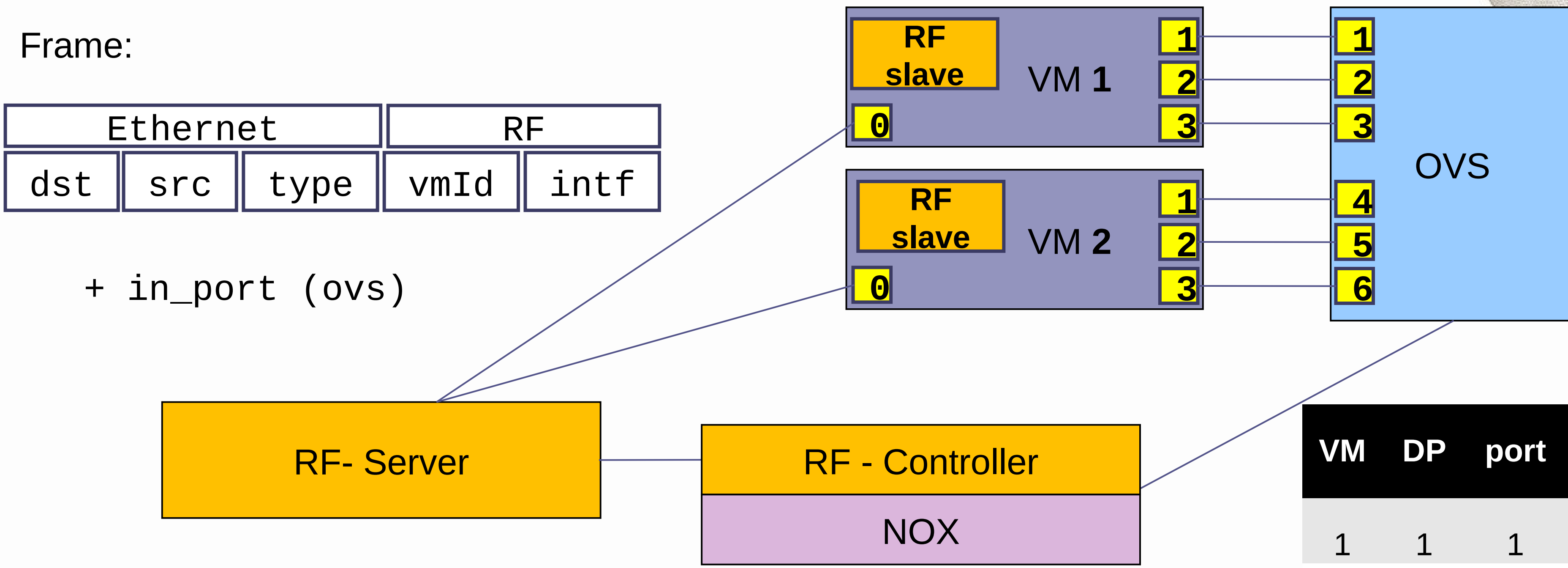
VM	DP	port	OVS port
1	1	1	?
1	1	2	?
1	1	3	?
2	2	1	?
2	2	2	?
2	2	3	?
...	...	...	...
n	n	3	?

Mapping Table



RouteFlow

RF-Slave: Interface Attachment discovery (2)



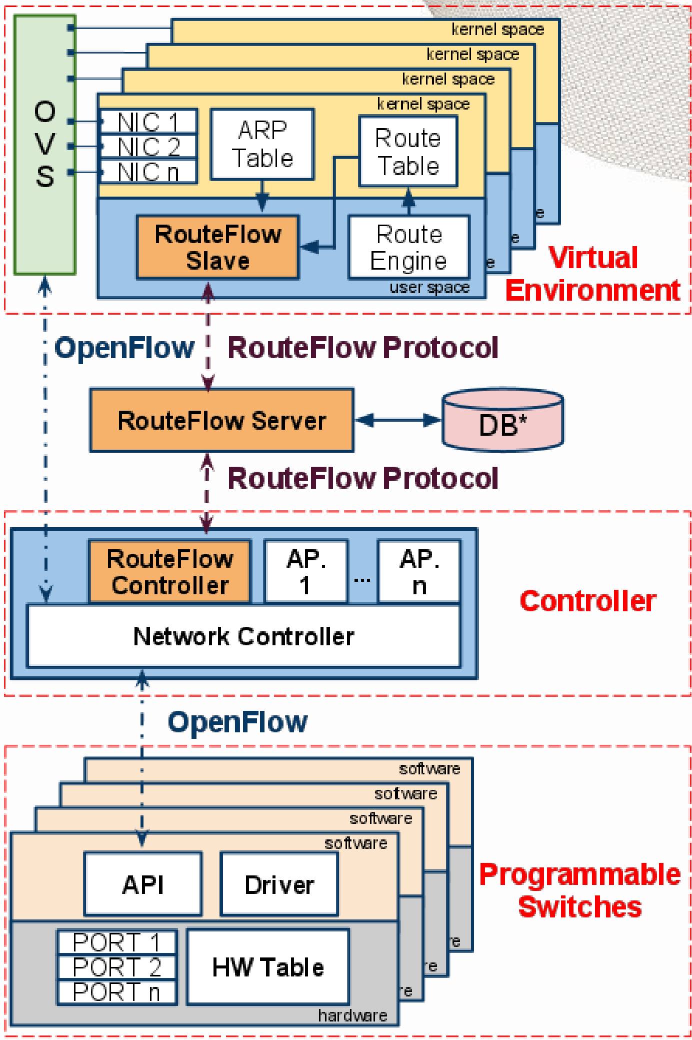
VM	DP	port	OVS port
1	1	1	1
1	1	2	2
1	1	3	3
2	2	1	4
2	2	2	5
2	2	3	6

- Discover the VM interfaces (**ETHX**)
- RF-Slaves sends discovery frames to all ifaces except ETH0;
- OVS forwards the packet-in to RF-Controller along the OVS port-in information.
- RF-Server sets the mapping of VM-DP-Port-OVS_port.

The RouteFlow protocol

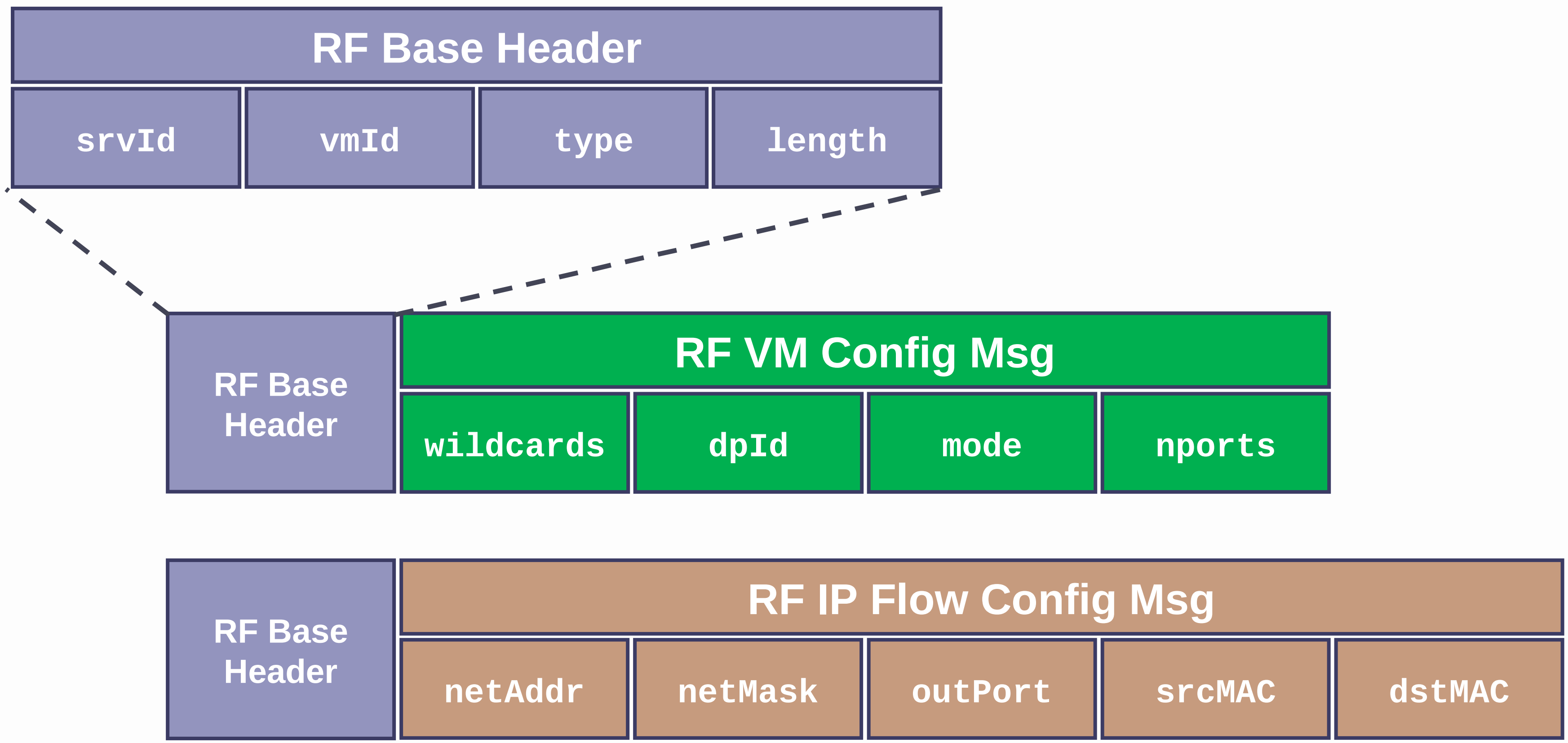
- Allowing a loosely couple architecture with two simple interfaces:
- Protocol between RF-Server and RF-Slave
  - VM registration and configuration,
  - Generate OpenFlow rules:
    - Translate changes in IP and ARP tables into OF modification messages.
- Protocol between RF-Server and RF-Controller
  - Basically, an API to controller OpenFlow stack
    - Subset of OpenFlow commands and events
    - Plus VM-OVS attachment discovery event
- In short, an IPC/RPC mechanism
  - Application-level on top of TCP, Client-Server, Assynchronous, Without Confirmation

Evolving to Apache Thrift & REST + JSON

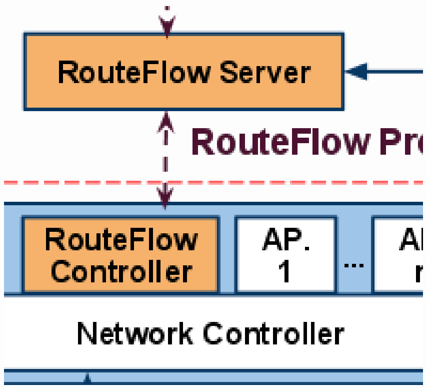


RouteFlow

RF-Protocol: Frame

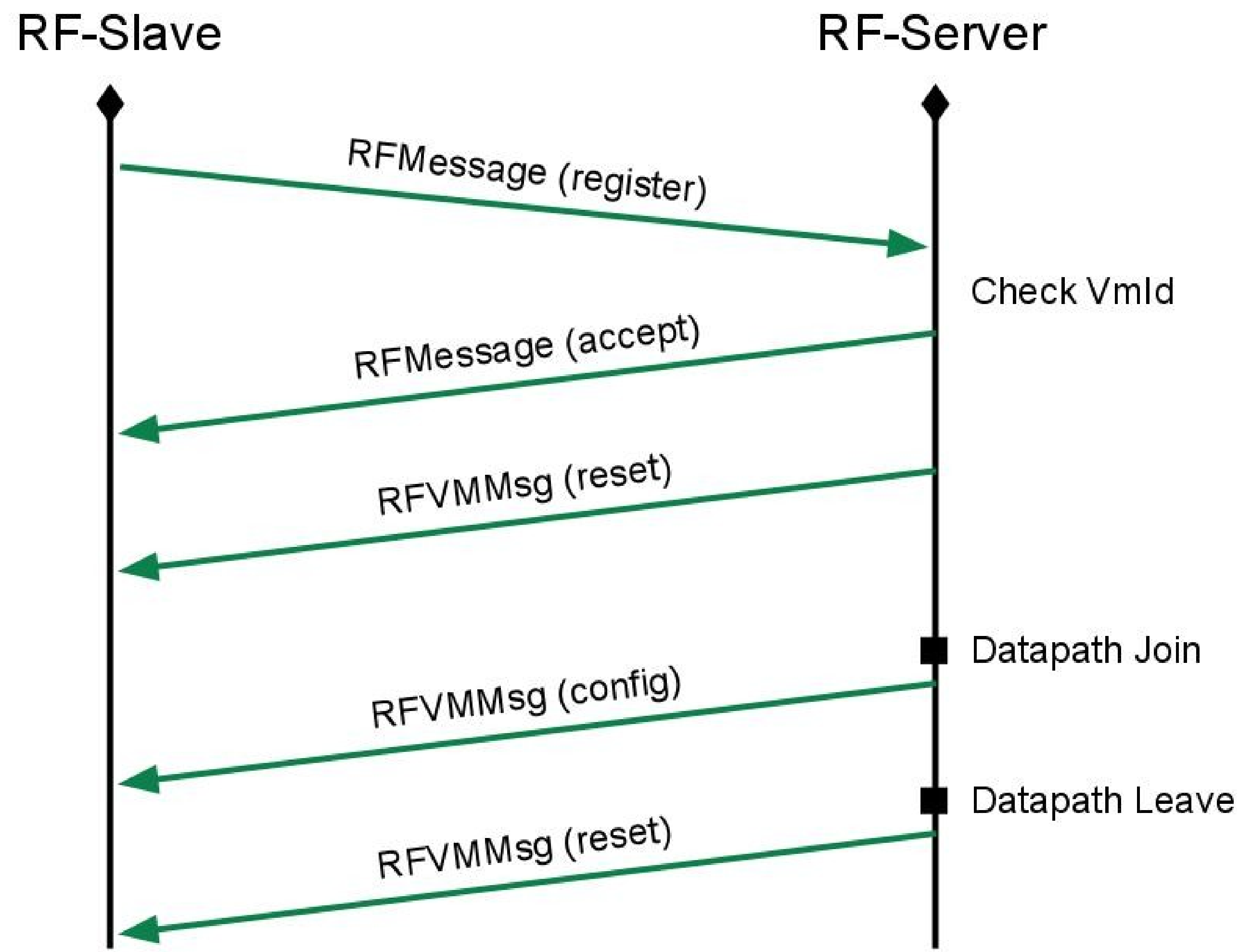


API between RF-Controller and RF-Server



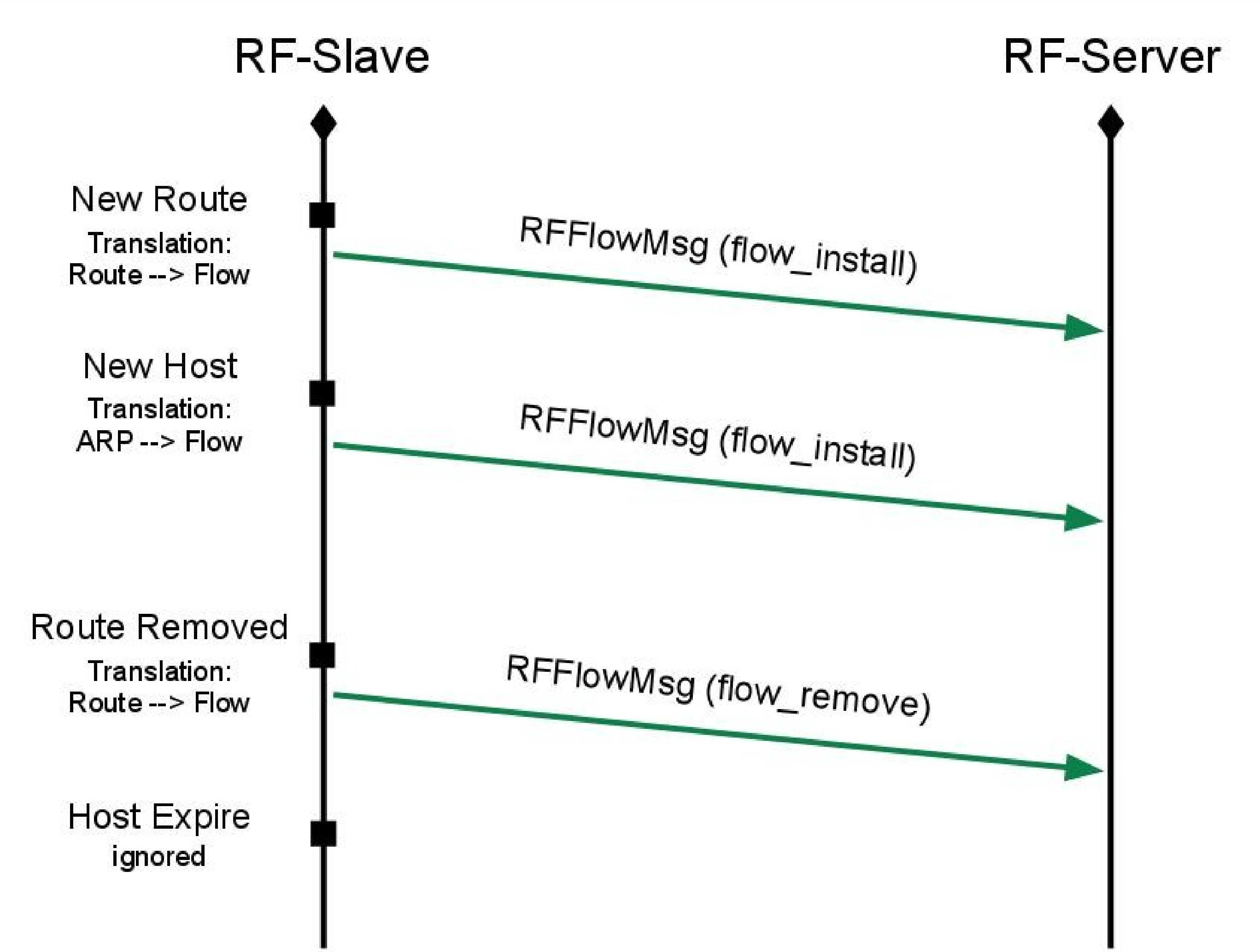
Group	Type	Payload
EVENT	packet_in	datapath_id (8 bytes) port_in (2 bytes) pkt_id (8 bytes) type (4 bytes)
EVENT	datapath_leave	datapath_id (8 bytes)
EVENT	datapath_join	datapath_id (8 bytes) no_ports (4 bytes) hw_desc (100 bytes)
EVENT	link_event	reason (1 byte) dp1 (8 bytes) port_1 (2 bytes) dp2 (8 bytes) port_2 (2 bytes)
EVENT	map_event	VmId (8 bytes) VmPort (2 bytes) OvsPort (2 bytes)
COMMAND	flow	datapath_id (8 bytes) flow_mod (2036 bytes)
COMMAND	send_packet	datapath_id (8 bytes) port_out (2 bytes) pkt_id (8 bytes)

VM Registration and Configuration



RouteFlow

Flow Modification messages

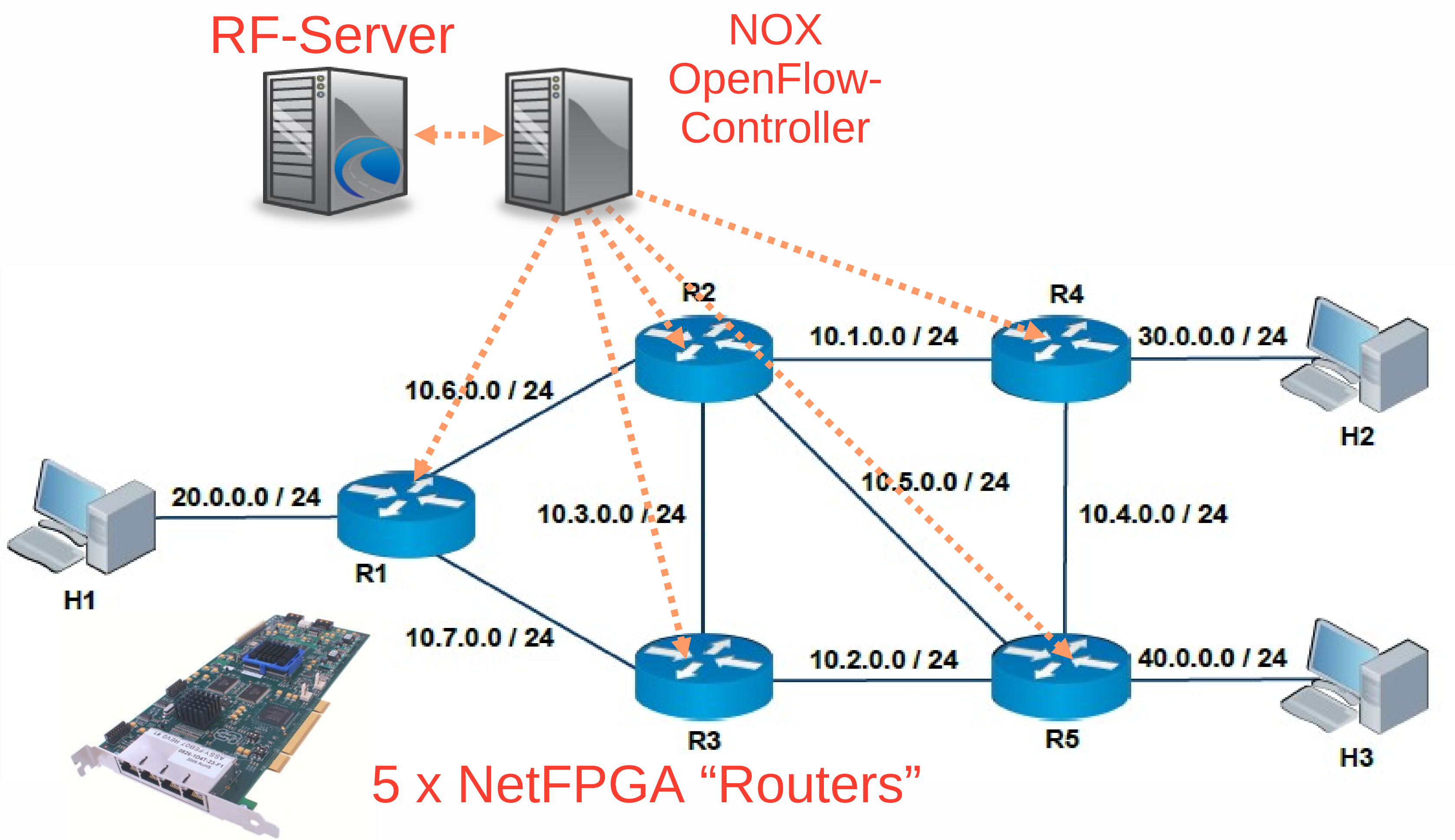


RouteFlow

Agenda

- Background: OpenFlow, Logical/Virtual Routers, Network Virtualization
- Project Overview
- Motivation
- Architecture
  - Controller
  - Server
  - Slave
  - Protocol
- Evaluation
- Work ahead
- Demo and hands-on Tutorial

NetFPGA-based testbed evaluation

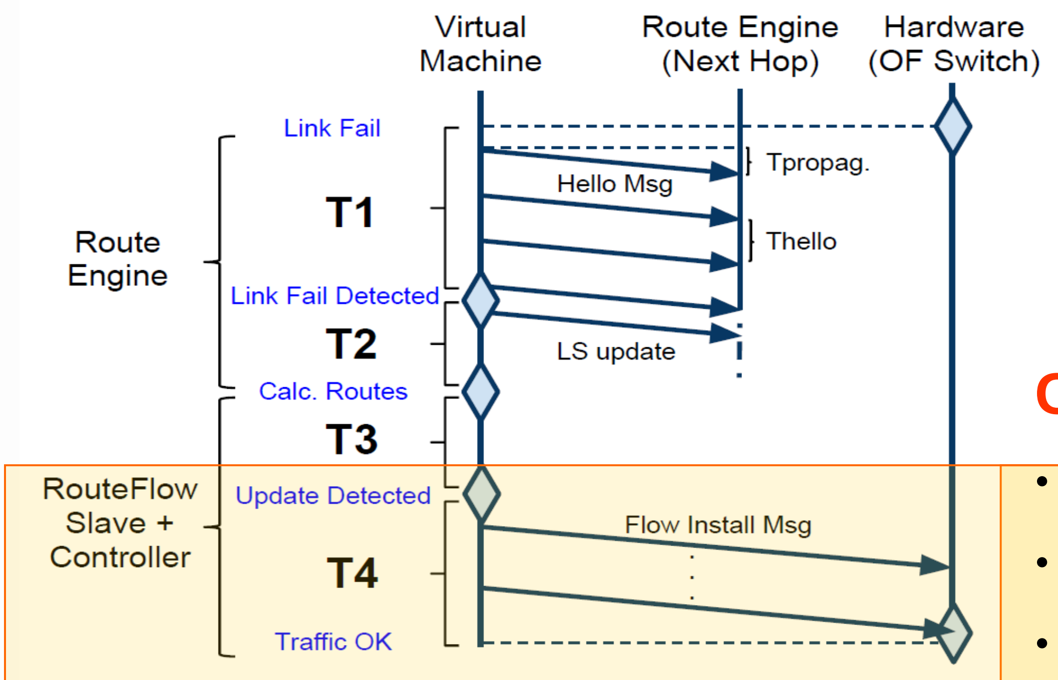


RouteFlow

Prototype evaluation

- Setup
  - NOX controller
  - Quagga routing engine
  - 5 x NetFPGAs
- Results
  - Interoperability with traditional networking gear
  - Route convergence time is dominated by the protocol time-out configuration (e.g., 4 x HELLO in OSPF) not by slow-path operations
  - Larger latency only for those packets that need to go to the slow-path:
    - Lack FIB entry, need processing by the OS networking / routing stack e.g., ARP, PING, routing protocol messages.

Experimental results: Route Convergence



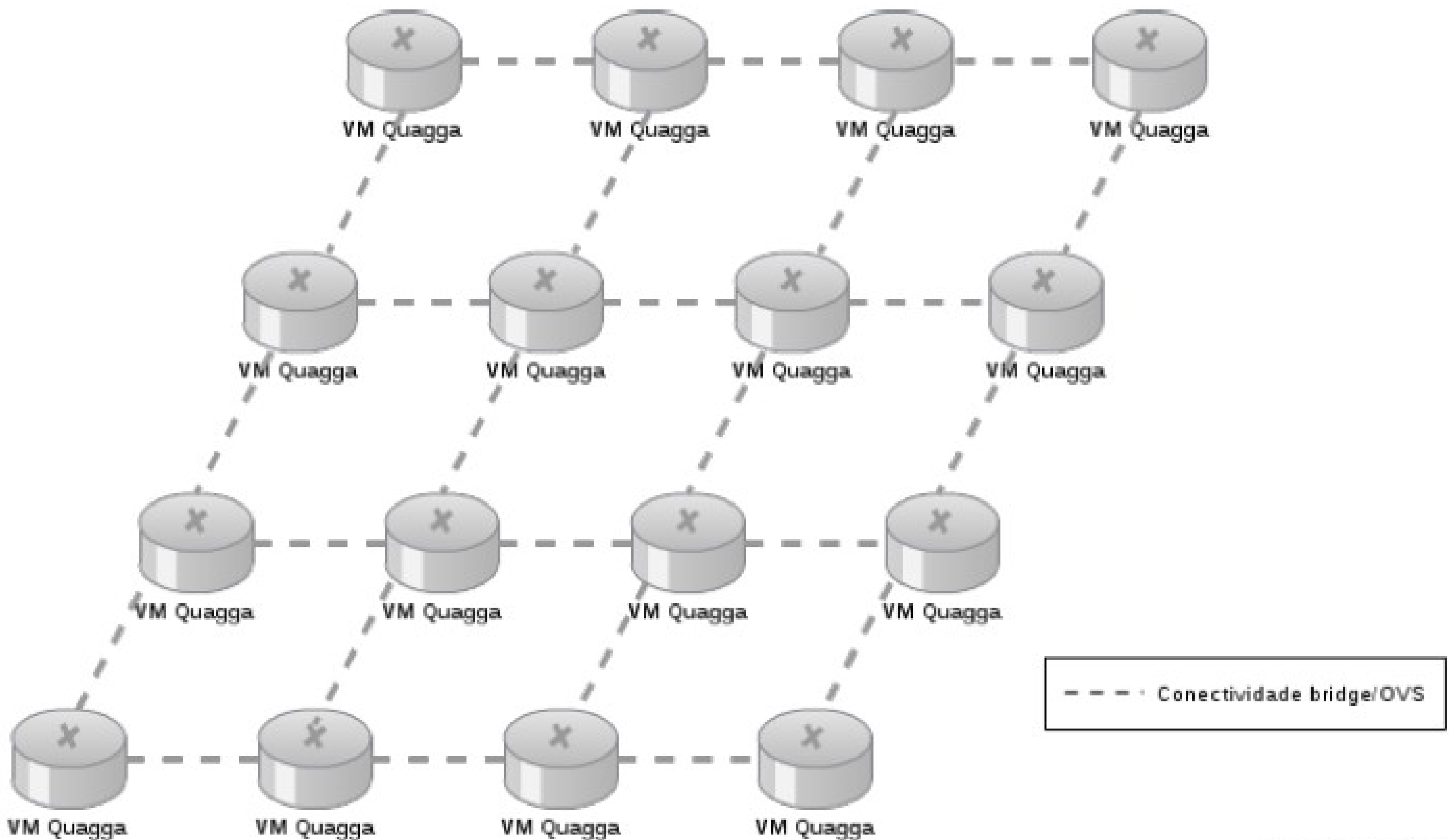
Caveat: Lab-scale conditions!

- **Low-latency links to RF-Controller**
- **No cross-traffic**
- **No CPU competition in OF switches**
- **Small FIBs, few topology changes**

Hello Time	T_1 [s]		T_2+T_3 [s]		T_4 [s]		T_{total} [s]	
OSPF	$T_{med.}$	$T_{90\%}$	$T_{med.}$	$T_{90\%}$	$T_{med.}$	$T_{90\%}$	$T_{med.}$	$T_{90\%}$
1 sec.	3.249	3.923	0.360	0.398	0.070	0.123	3.700	4.373
5 sec.	16.713	18.937	0.320	0.389	0.057	0.099	17.135	19.308
10 sec.	36.406	37.846	0.358	0.497	0.042	0.106	36.807	38.266

RouteFlow

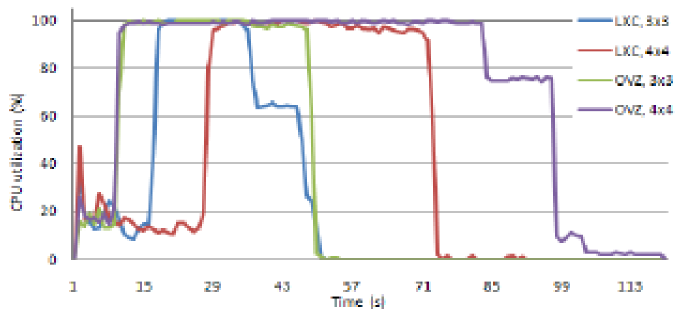
Scaling the Virtual Environment



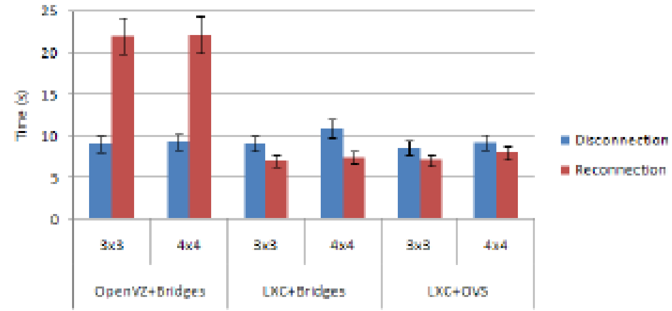
UNIRIO
Universidade Federal do
Estado do Rio de Janeiro

RouteFlow

Evaluation of the Virtualization Environment

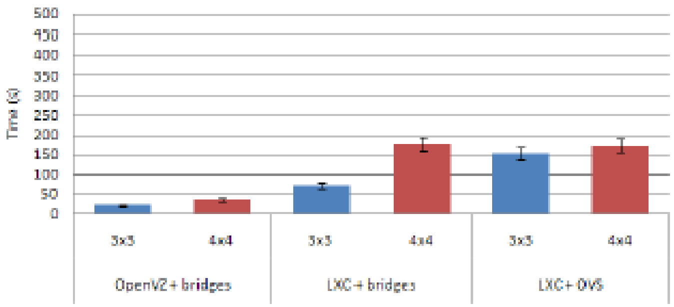


(a) CPU utilization

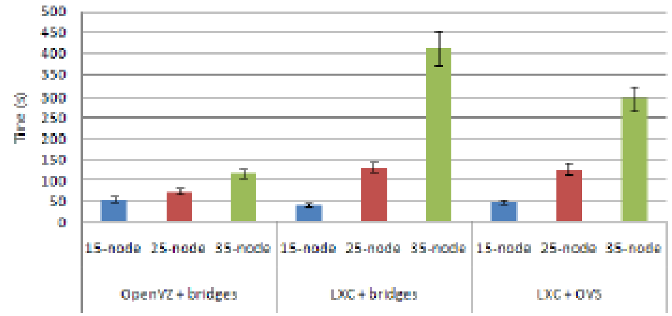


(b) Convergence after connectivity modification events

Fig. 3. Grid topologies' CPU use and link events convergence details

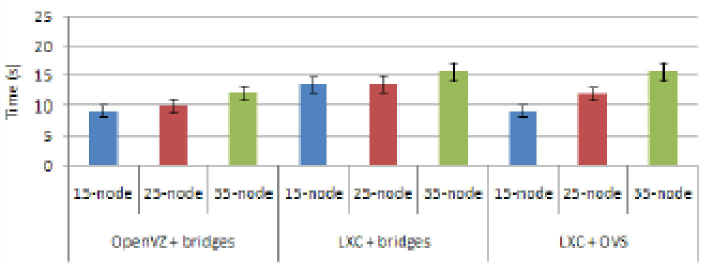


(a) Initial convergence on grid topologies

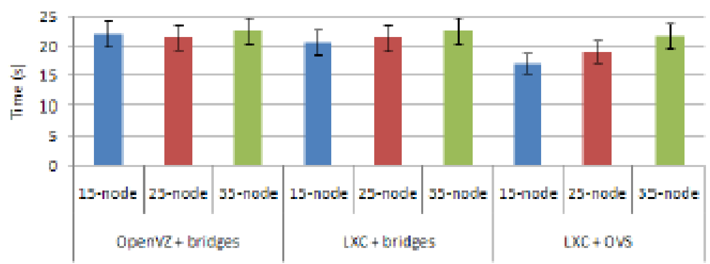


(b) Initial convergence on full-meshed topologies

Fig. 4. Initial OSPF convergence for both connectivity layouts evaluated



(a) Convergence after disconnection of a node



(b) Convergence after reconnection of node

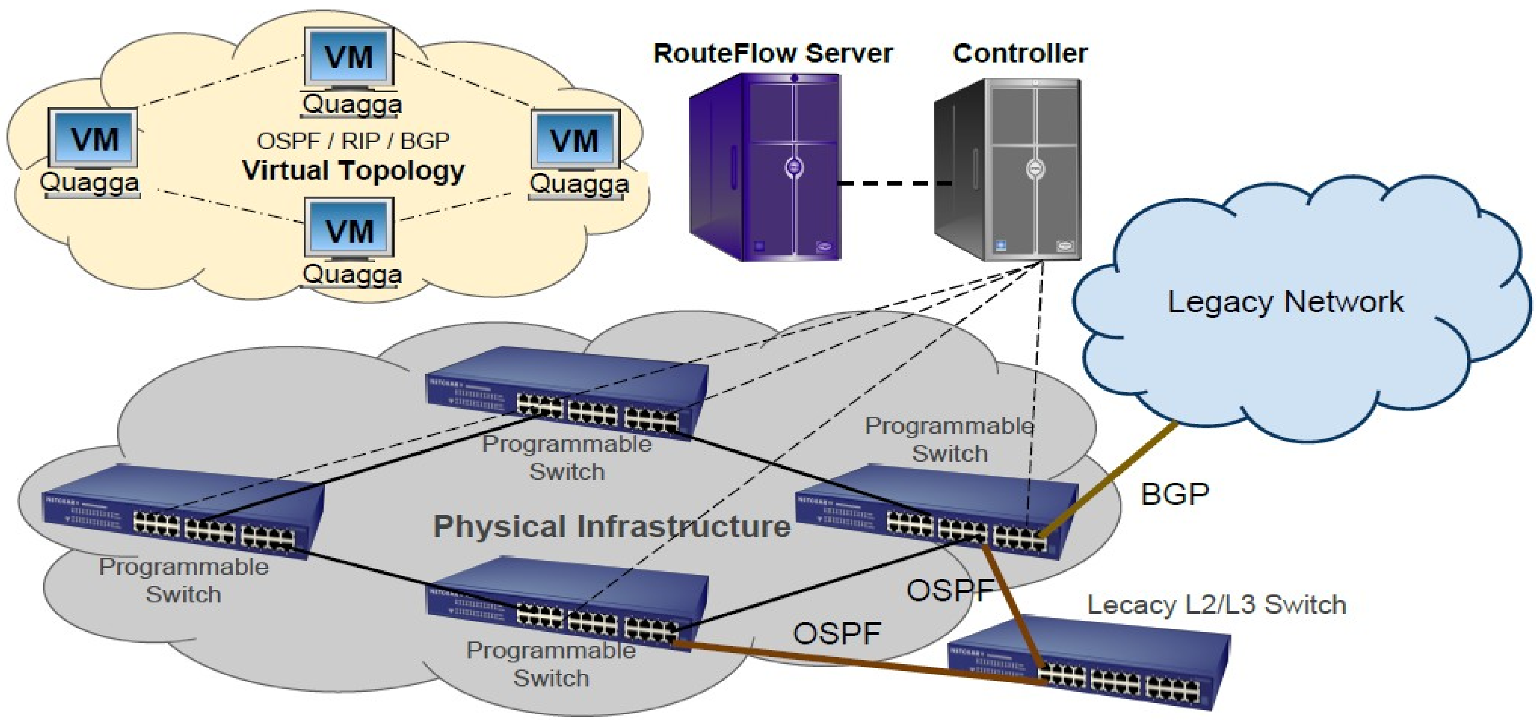
The Path Ahead

- OpenFlow 1.1
- Controller API: Rest-API JSON & Apache Thrift
- Advancing the IP Network Virtualization
  - Protocol Optimization, Modes of operation, Router Migration
- Scalability and Resiliency
- System Limits and Stress testing
- Live Trials
  - Reality-Checks at Scale
- Embrace related work (past & ongoing)
  - SoftRouter, VROOM, DROP, FIBIUM, ONIX, etc.
- Build a community!
  - Student Projects corner (https://sites.google.com/site/routeflow/projects)

RouteFlow

Protocol Optimization

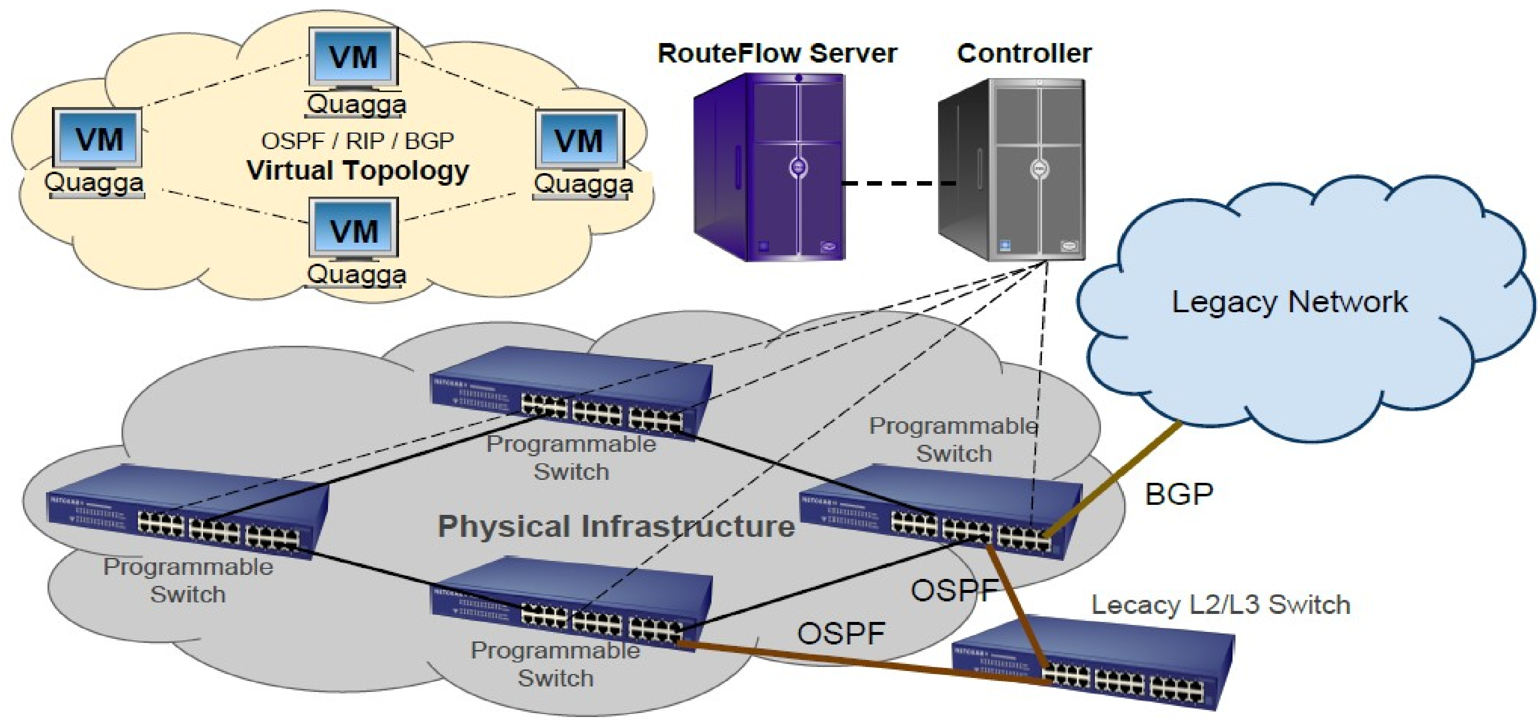
- Separation of concerns between topology maintenance and routing state distribution
  - E.g. HELLOs sent “down” while LSA are kept “up”
  - E.g. BFD-like fault detection substitute HELLOs



RouteFlow

Resiliency and Scalability

- Distributed Virtual environment with distributed OVS for load balancing, replication, and advanced VM management (e.g., migration)
- NoSQL-like distributed database for core RouteFlow state
- Multi-controller environments
- Fault-tolerance: Master / Slave, Master / Master, ...?



RouteFlow

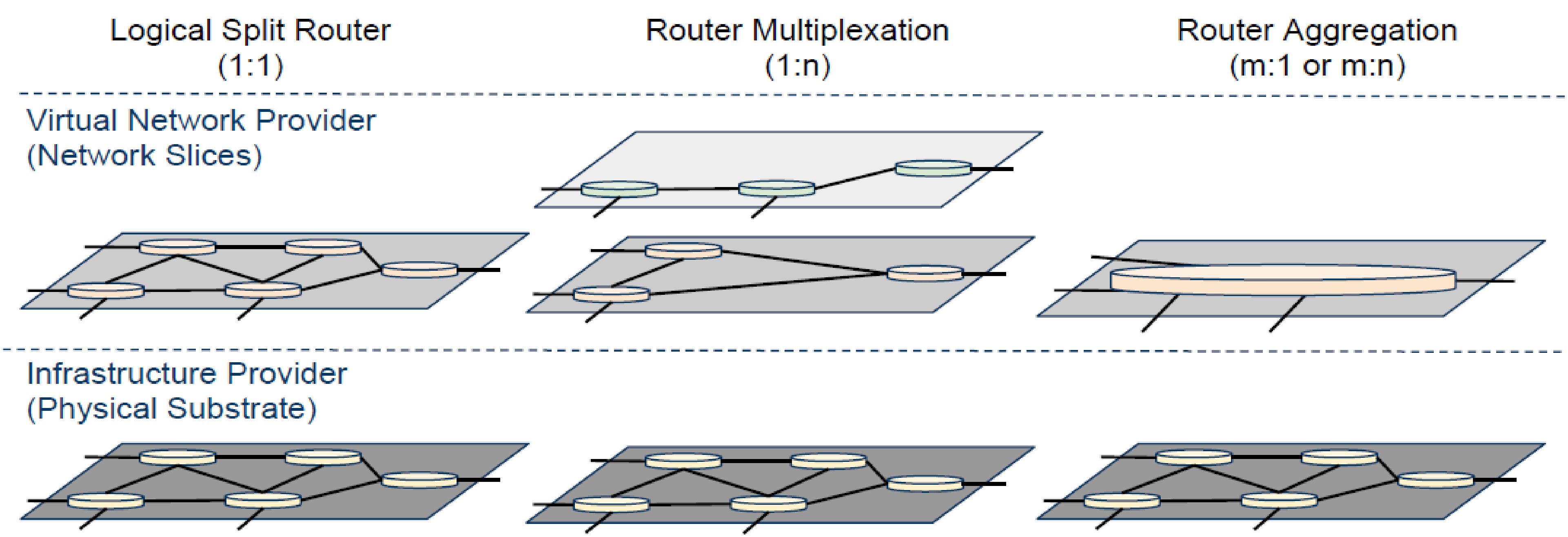
System limits and Stress testing

- Increase network size
  - Increase flowmod/sec
    - Variable OpenFlow control packet handling / processing:
      Impact on Routing Protocol?
      Impact on topology maintainance protocol, e.g., LLDP-based?

- Scale limitation (Flow table size) of logical / large routing tables
  - Smart shared multiple table lookup in OF.1.1
  - Smart caching, hybrid software-hardware flow state
  - Related Work (e.g., ViAggre)
  - etc.

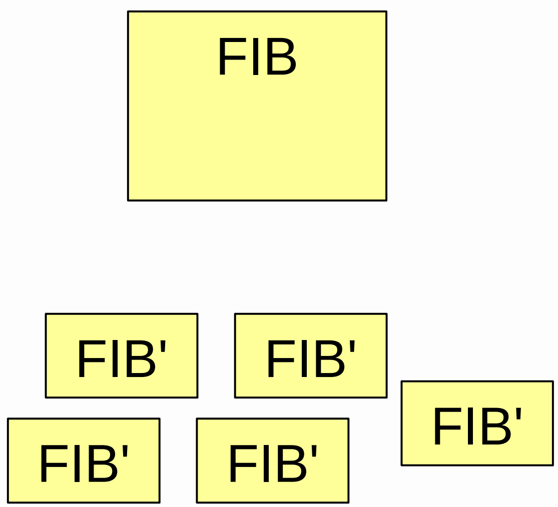
Advancing the Use Cases and Modes of Operation

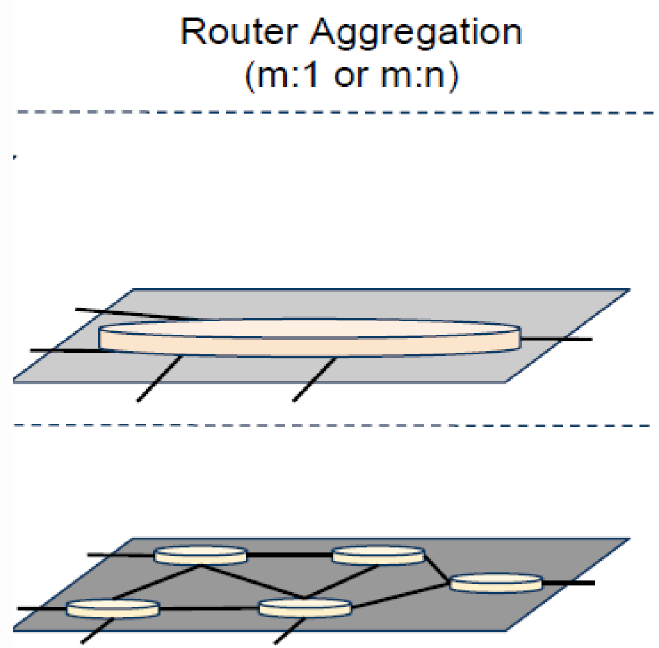
- From logical routers to flexible virtual networks



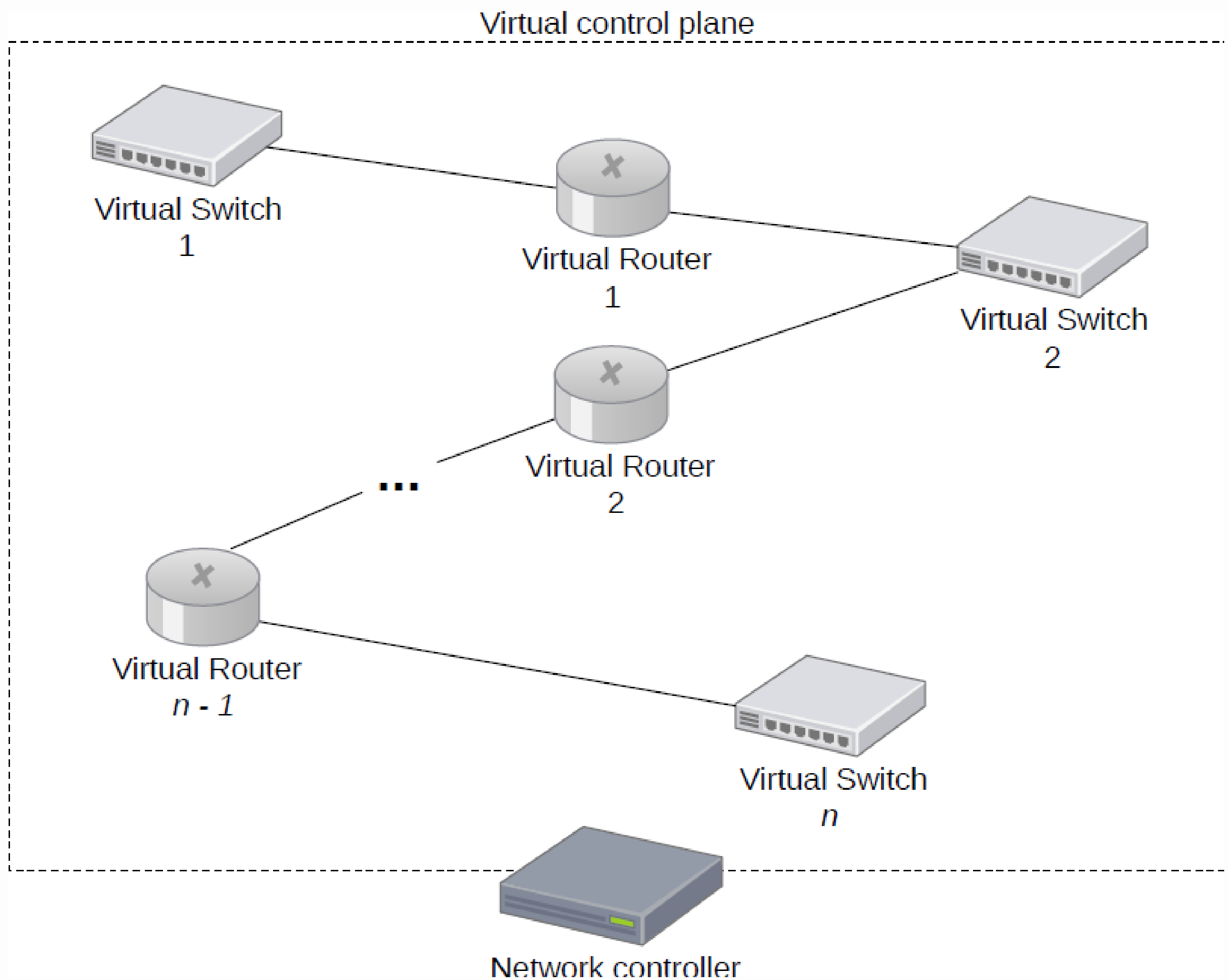
Aggregated Router

- Scenarios:
  - a single BGP router aggregating a number of OpenFlow switches
  - L3 services in data center distributed single virtual switch
- Distributed lookup?
  - E.g., Smart FIB generation and distribution
- Intra-router switching strategy?





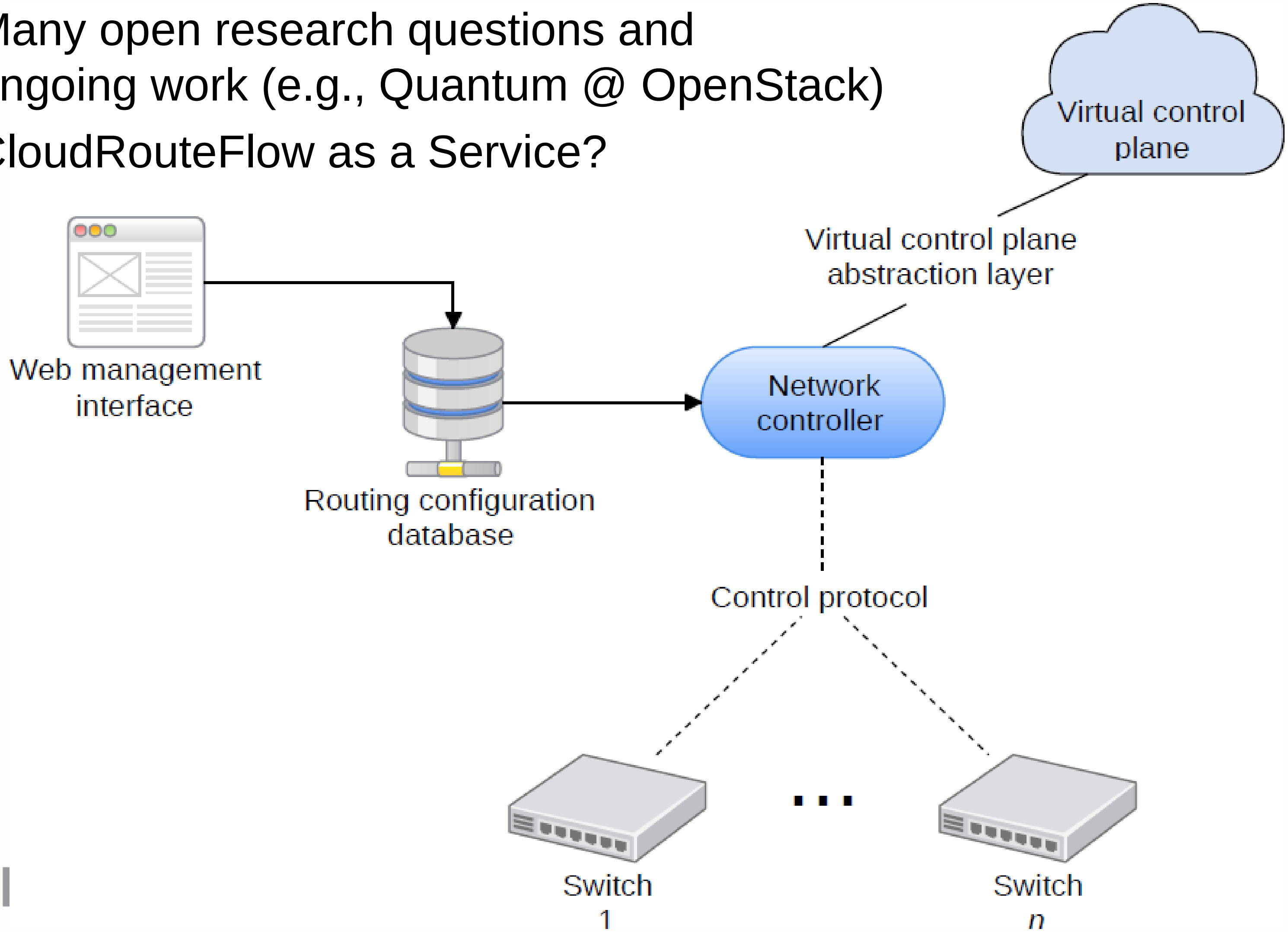
NaaS - Network-as-a-Service



RouteFlow

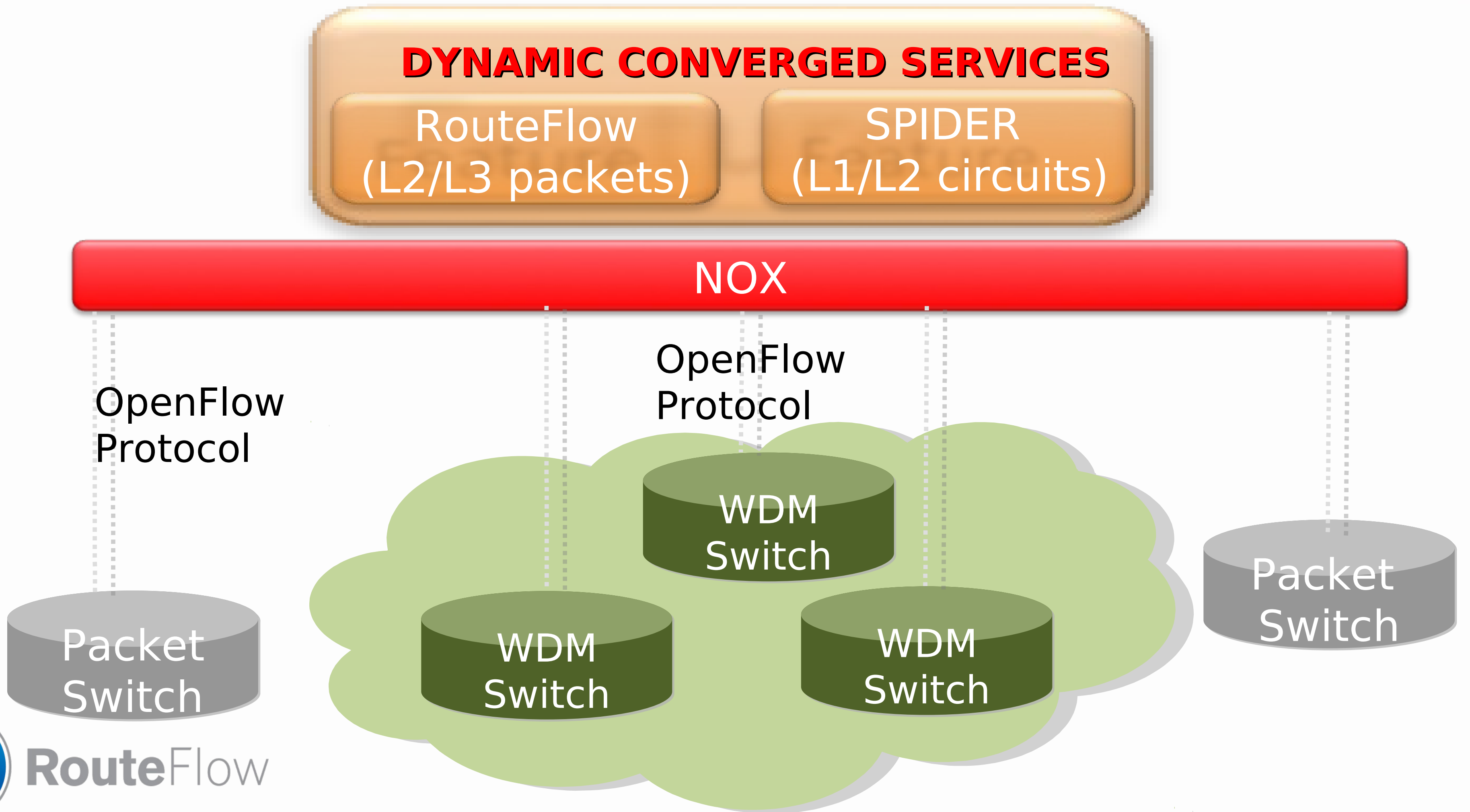
NaaS - Network-as-a-Service

- Enabling Virtual networks as a Service
- Many open research questions and ongoing work (e.g., Quantum @ OpenStack)
- CloudRouteFlow as a Service?



CPqD Dynamic Converged (Packet and Circuits) Services

Goal: Common control plane for Layers 1 to 3 networks aiming at NaaS, RaaS, VNO
Approach: OpenFlow + RouteFlow + SPIDER (virtualization comes in a subsequent phase)

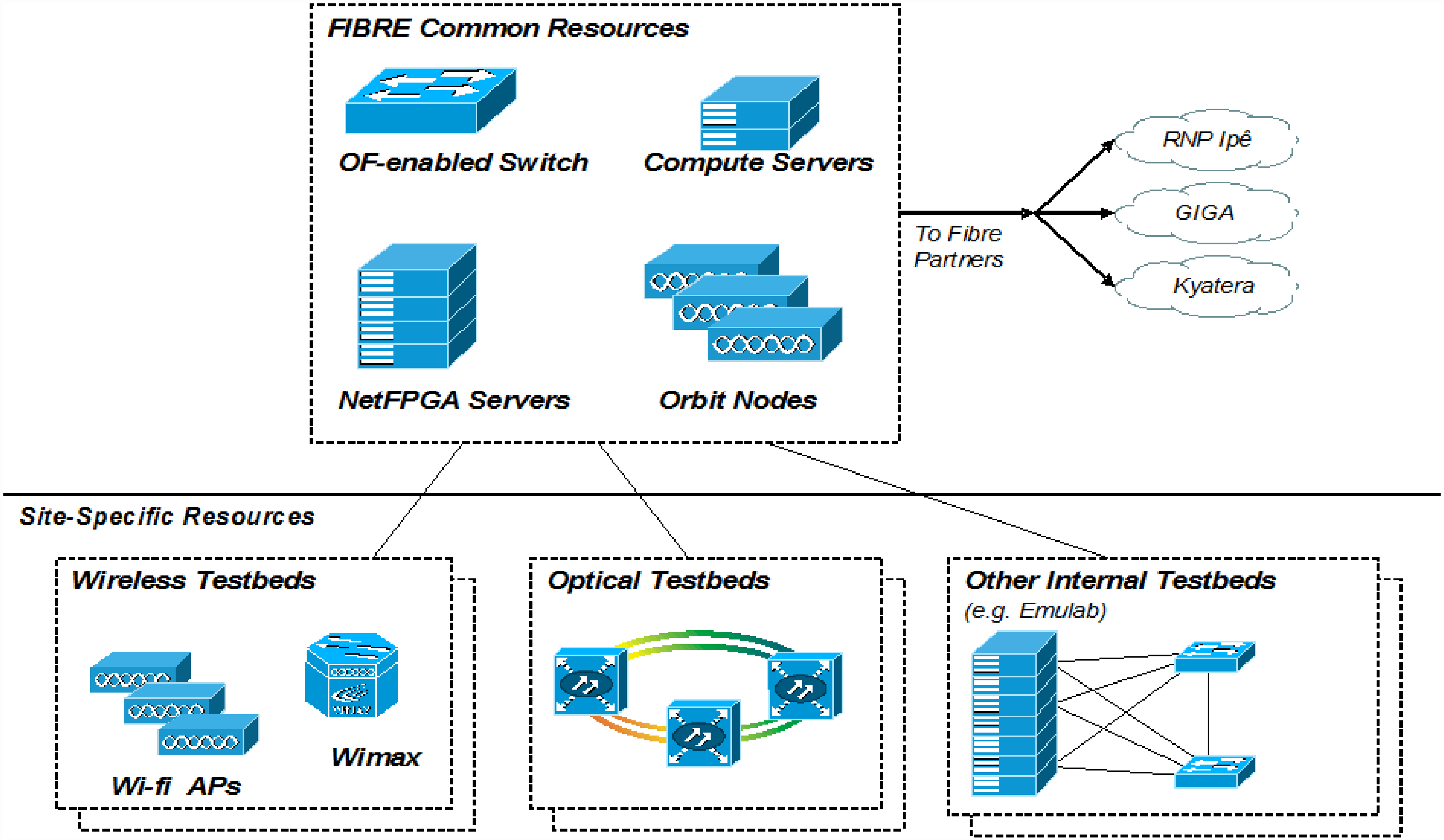


FIBRE: FI testbeds between BRazil and Europe

- Joint EU-Brazil project between 9 partners from Brazil (6 from GIGA), 5 from Europe (4 from Ofelia and OneLab) and 1 from Australia (from OneLab)
  - Design, implementation and validation of a shared **Future Internet sliceable/programmable research facility**, supporting the joint experimentation of European and Brazilian researchers.
- The objectives include:
  - the development and operation of a new experimental facility in Brazil
  - the development and operation of a FI facility in Europe based on enhancements and the federation of the existing OFELIA and OneLab infrastructures
  - The federation of the Brazilian and European experimental facilities, to support the provisioning of slices using resources from both testbeds
- Officially started on Oct 1 2011
- Duration: 36 months

RouteFlow

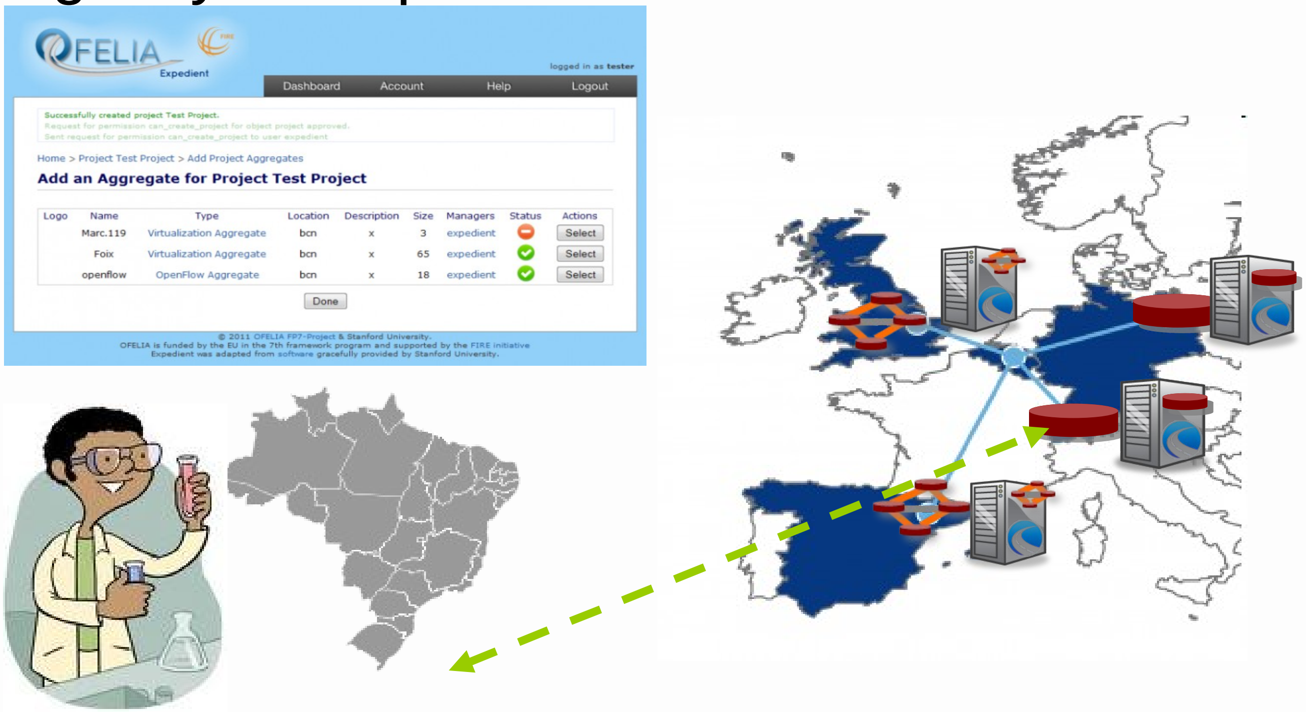
FIBRE site in Brazil



RouteFlow

OFELIA-enabled Experiments

- One RouteFlow platform running in each OpenFlow island controlling only the OpenFlow switches in the same facility.

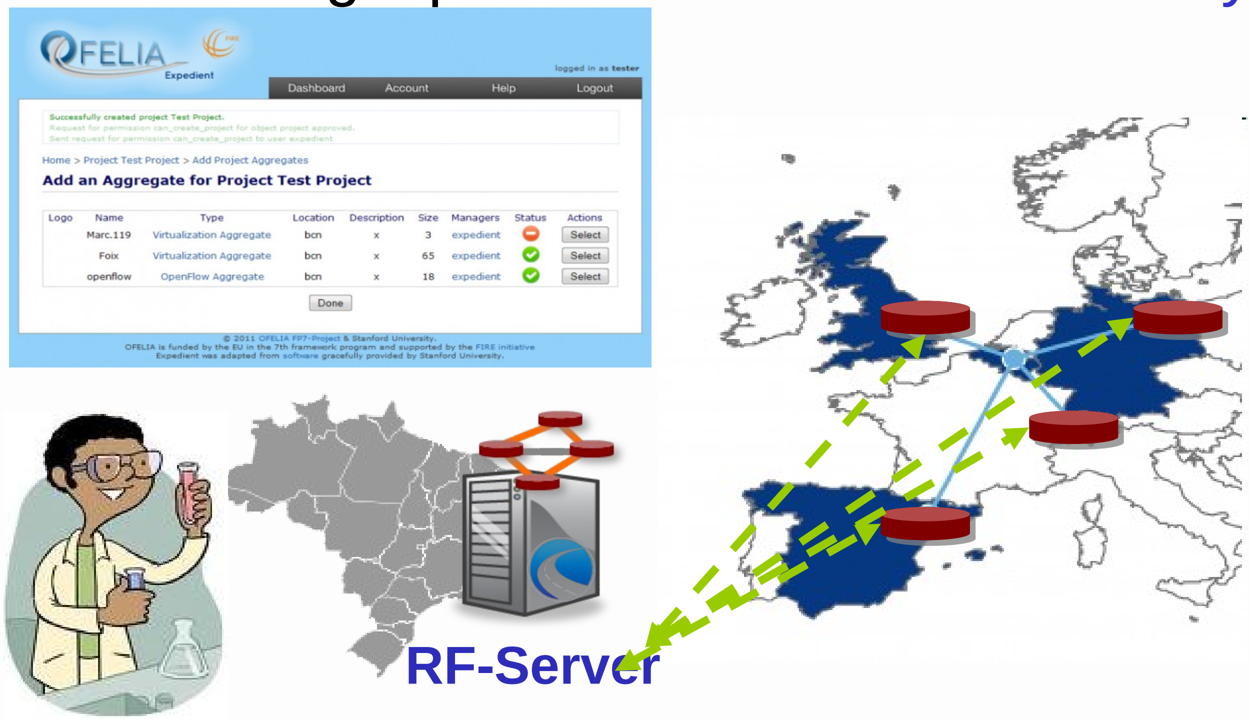


- Experiment outputs: Platform behaviour in geo-distributed setup, route convergence times, interoperability tests

RouteFlow

OFELIA-enabled Experiments

- Only one RouteFlow platform running in a single facility at a time and controlling OpenFlow switches from every facility.



- Experiment outputs: Protocol behaviour under remote operation, route convergence times, slow-path performance

RouteFlow

Reality check at Euro-scale

Experimental work	Current	Ofelia-enabled
Scale	5 to 10 x 4-port NetFPGAs	10s of OpenFlow switches
Equipment	Software-based switches, NetFPGAs	Multi-vendor commercial OpenFlow switches
Realism	Few, small topologies Synthetic traffic (control + user) and failures	Geo-distributed topologies Real traffic (control & data) ? and failure scenarios
Performance Fidelity	Low latency LAN	Variable network conditions

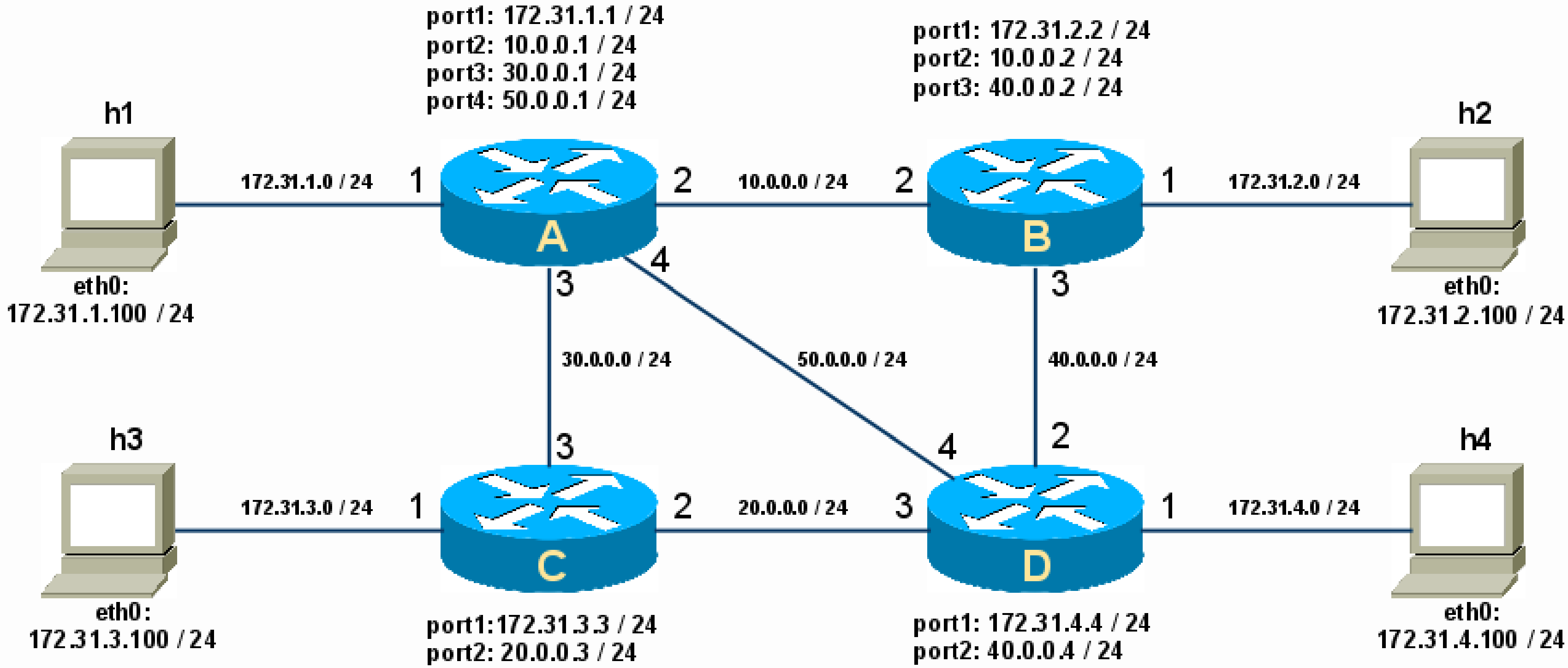
Demo @ ONS 2011

Indiana University

Pronto 3240/3290

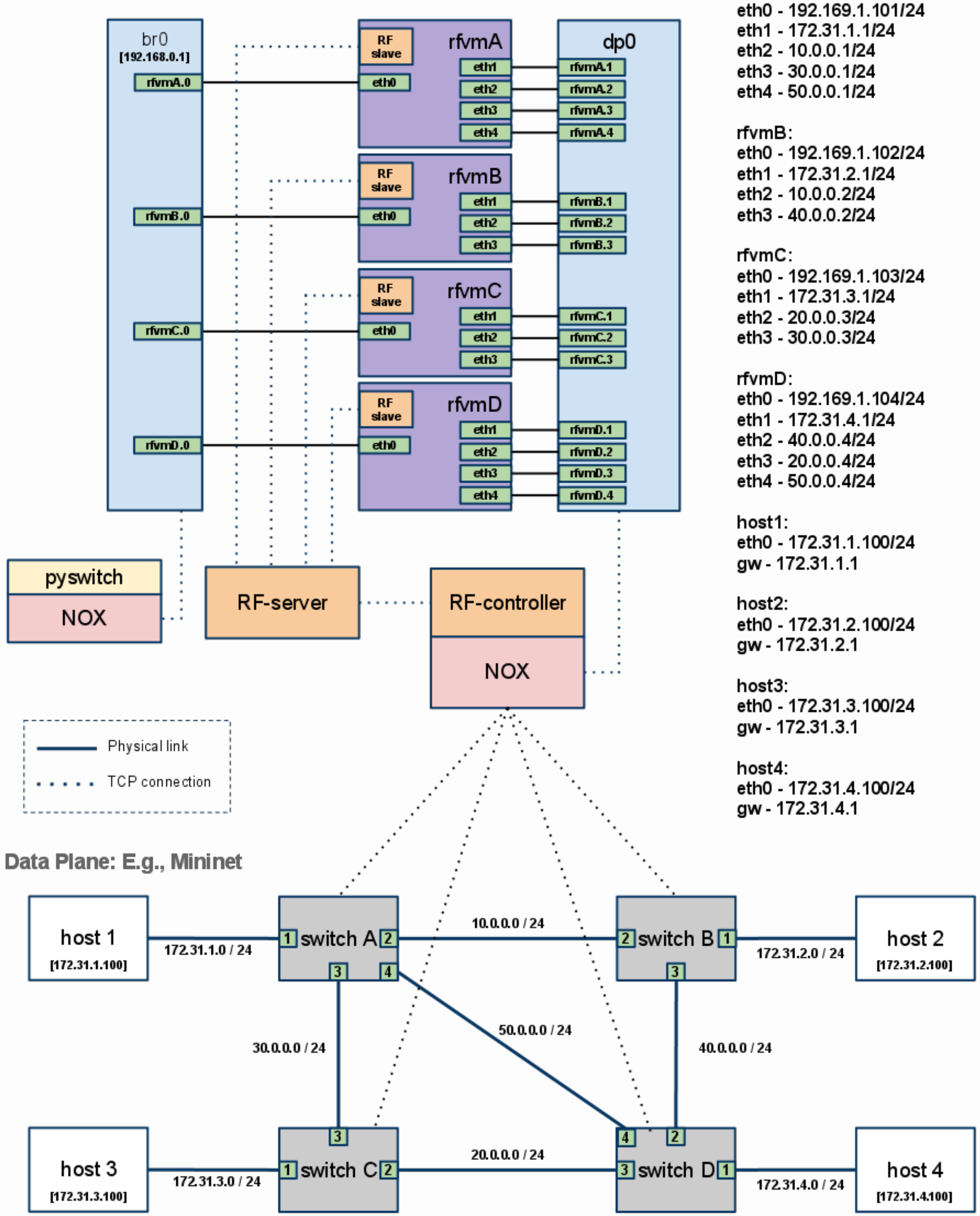
Tutorial 2

Traditional Scenario



Tutorial 2

RouteFlow Scenario



rfvmA:
eth0 - 192.169.1.101/24
eth1 - 172.31.1.1/24
eth2 - 10.0.0.1/24
eth3 - 30.0.0.1/24
eth4 - 50.0.0.1/24

rfvmB:
eth0 - 192.169.1.102/24
eth1 - 172.31.2.1/24
eth2 - 10.0.0.2/24
eth3 - 40.0.0.2/24

rfvmC:
eth0 - 192.169.1.103/24
eth1 - 172.31.3.1/24
eth2 - 20.0.0.3/24
eth3 - 30.0.0.3/24

rfvmD:
eth0 - 192.169.1.104/24
eth1 - 172.31.4.1/24
eth2 - 40.0.0.4/24
eth3 - 20.0.0.4/24
eth4 - 50.0.0.4/24

host1:
eth0 - 172.31.1.100/24
gw - 172.31.1.1

host2:
eth0 - 172.31.2.100/24
gw - 172.31.2.1

host3:
eth0 - 172.31.3.100/24
gw - 172.31.3.1

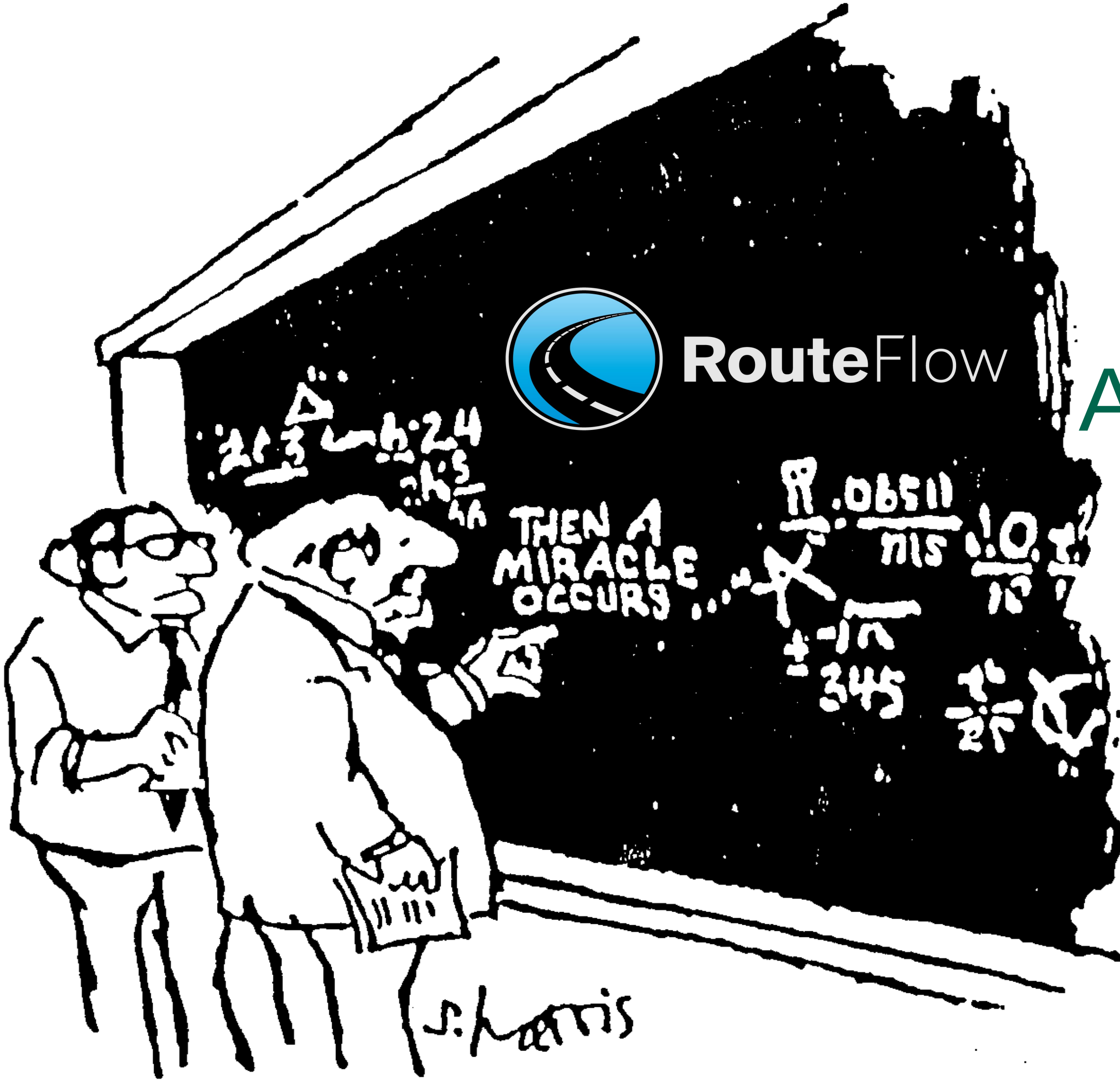
host4:
eth0 - 172.31.4.100/24
gw - 172.31.4.1

RouteFlow

Conclusions

- RouteFlow proposes a commodity routing architecture that combines the line-rate performance of commercial hardware with the flexibility of open-source routing stacks (remotely) running on PCs;
- Allows for a flexible resource association between IP routing protocols and a programmable physical substrate:
  - Multiple use cases around virtualized IP routing services.
  - IP routing protocol optimization
  - Migration path from traditional IP deployments to software-defined networks

Questions?



Thank you!

Ask and contribute!

routeflow-discuss@googlegroups.com

Learn more!

https://go.cpqd.com.br/routeflow

Get the Code!

https://github.com/CPqD/RouteFlow

RouteFlow

Christian Esteve Rothenberg
esteve@cpqd.com.br
+55 19 3705-4479

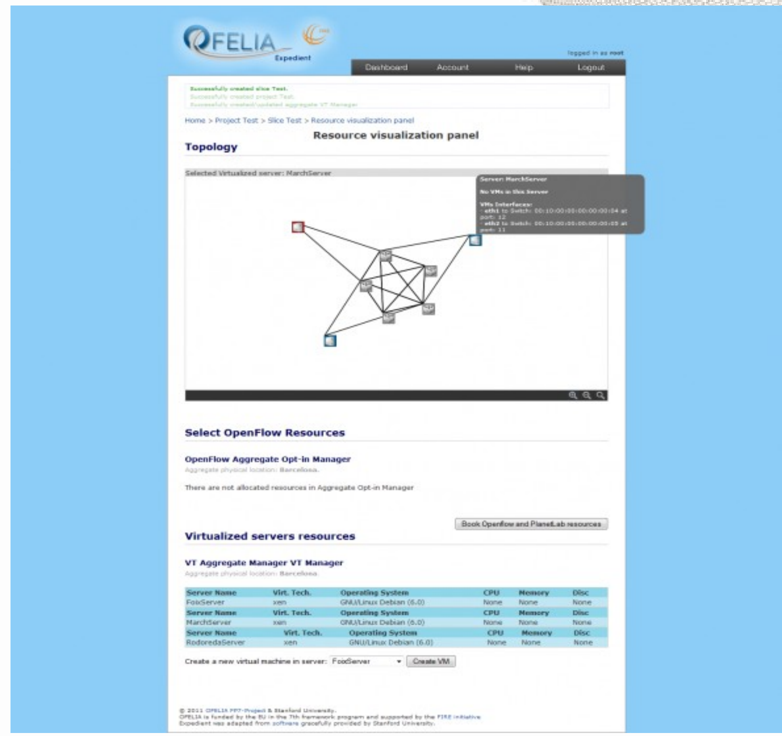
BACKUP

Proposed experiments

Mainly two types of experiments:

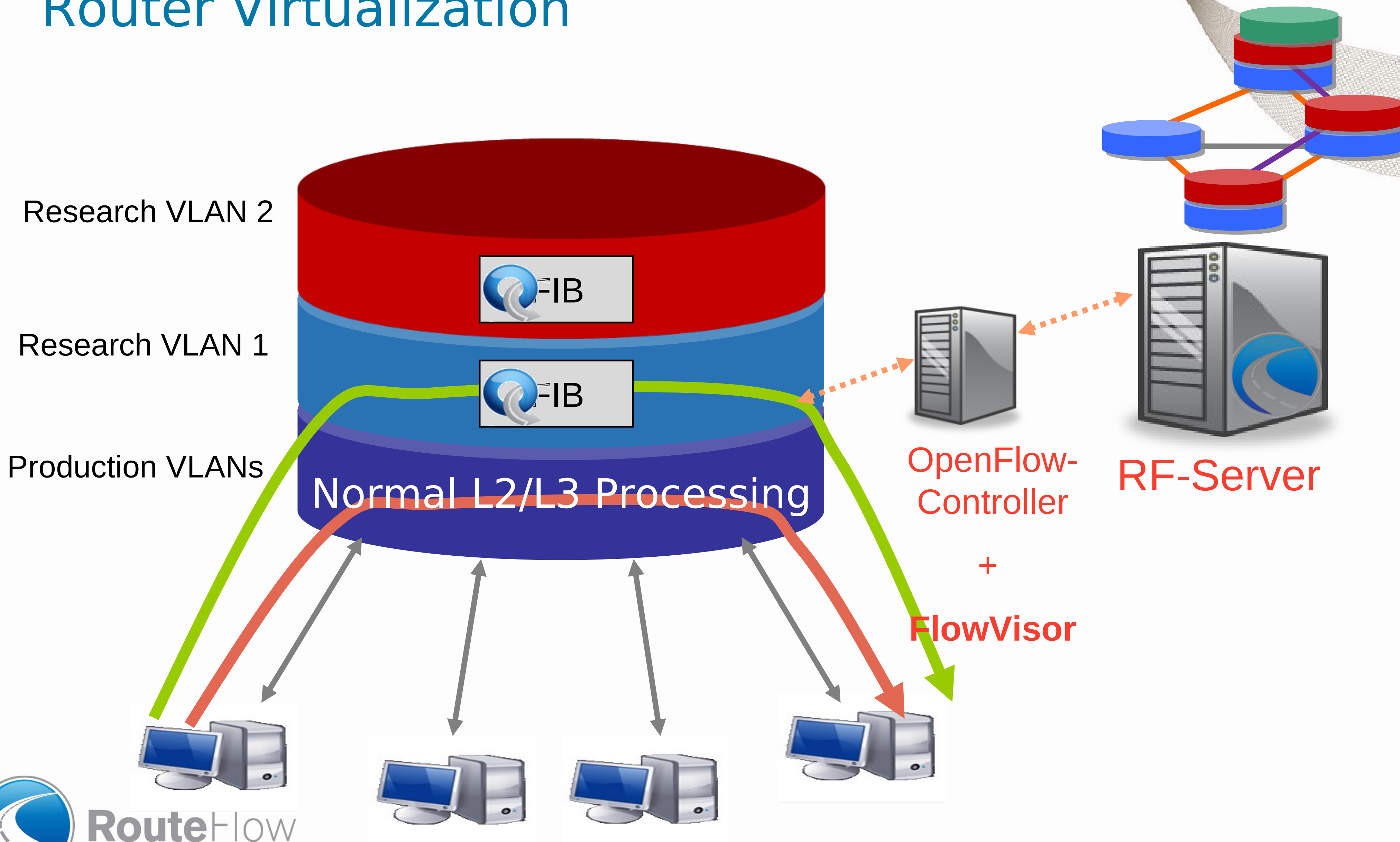
1. One RouteFlow platform running in each OpenFlow island controlling only the OpenFlow switches in the same facility.

2. Only one RouteFlow platform running in a single facility at a time and controlling OpenFlow switches from every facility.



The resulting combination of scenarios will allow to validate the scalability and performance limits of the remote operation of the IP routing stacks provided by RouteFlow.

Interop Experiments & Realistic Router Virtualization



RouteFlow

Expected results

- **Technical viability:** Exploring the scalability and performance limits
  - convergence times not penalized by remote routing protocol stacks.
  - suitable distribution of control plane entities and the physical counterparts.
  - real-world networking conditions (e.g., latencies, failures, traffic)
- **Interoperability and generality of RouteFlow**
  - different open-source routing protocol stacks (XORP and Quagga),
  - different virtualization technologies (e.g., LXC and QEMU)
  - different OpenFlow controllers (e.g., NOX and Beacon),
  - all inter-working with commercial OpenFlow-enabled switches and legacy networking equipment.
- **Assessment of the OFELIA testbed facilities**
  - E.g., Capabilities of the Expedient CMF, effective resource sharing, controller application deployment, etc.