RYU OpenFlow Controller

Dean Pemberton – NSRC

Andy Linton – NSRC

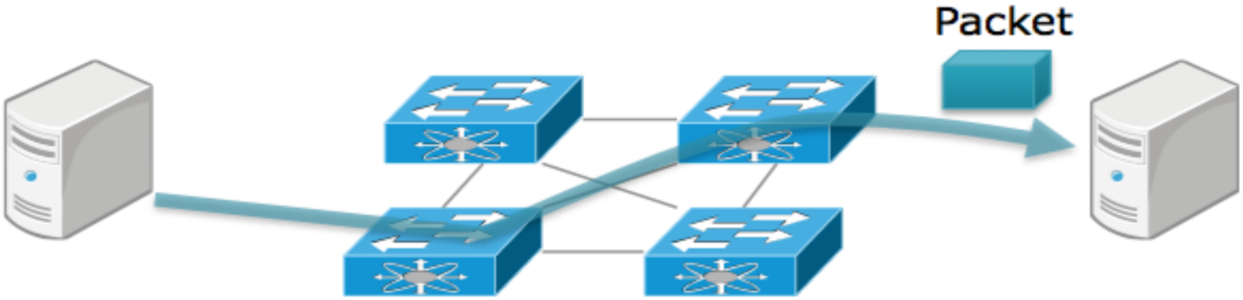
Sam Russell - REANNZ

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

What is Ryu?

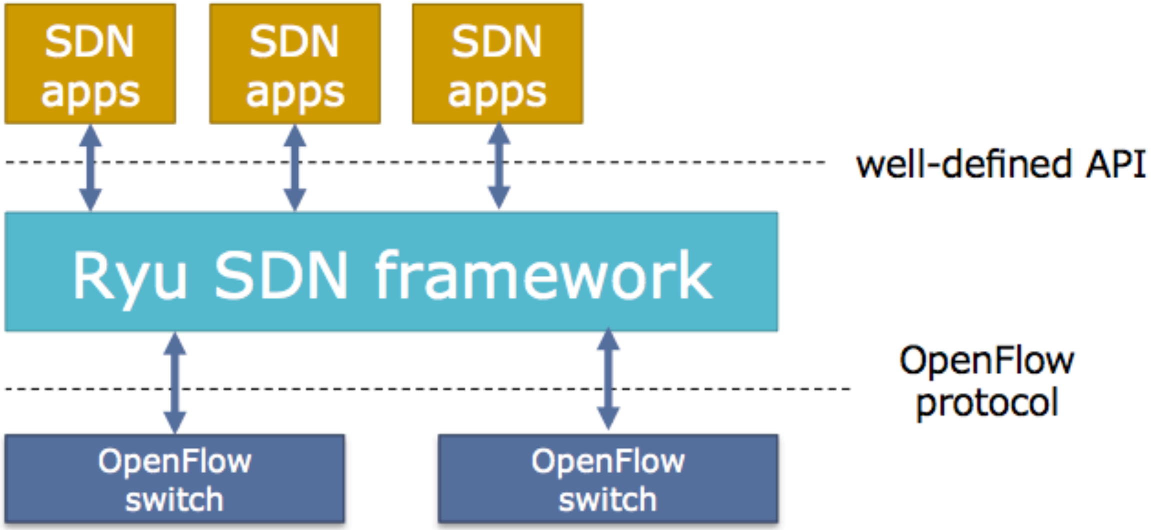
- Name comes from a Japanese word meaning “flow”
- Ryu manages “flow” control to enable intelligent networking



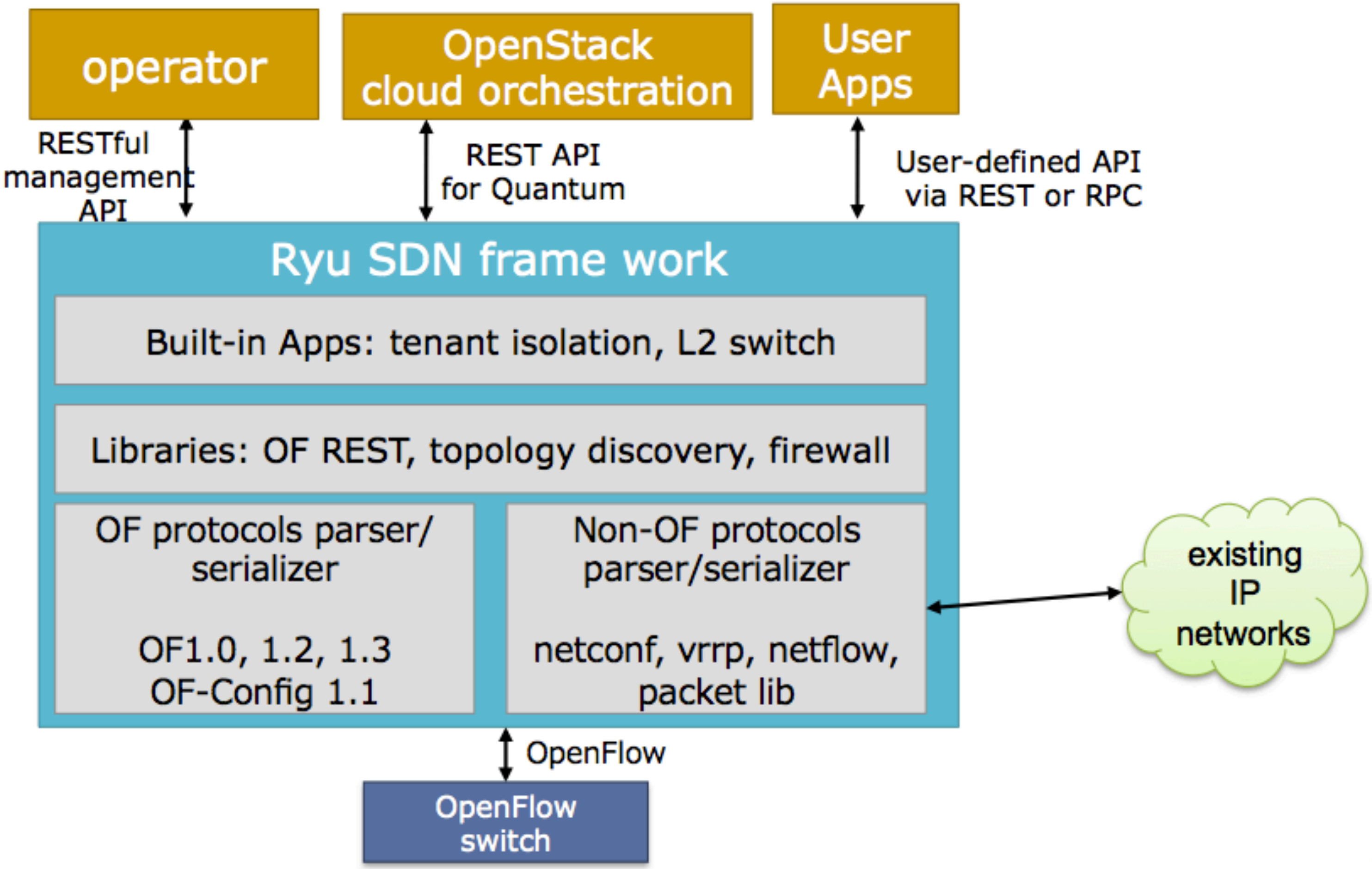
Philosophy

- Agile
  - Framework for SDN application development instead of all-purpose big monolithic ‘controller’.
- Flexible
  - Vendor-defined “Northbound” APIs are not enough to differentiate.

Where does Ryu sit?



Architecture



operator

OpenStack cloud orchestration

User Apps

RESTful management API

REST API for Quantum

User-defined API via REST or RPC

Ryu SDN frame work

Built-in Apps: tenant isolation, L2 switch

Libraries: OF REST, topology discovery, firewall

OF protocols parser/ serializer

OF1.0, 1.2, 1.3
OF-Config 1.1

Non-OF protocols parser/serializer

netconf, vrrp, netflow, packet lib

existing IP networks

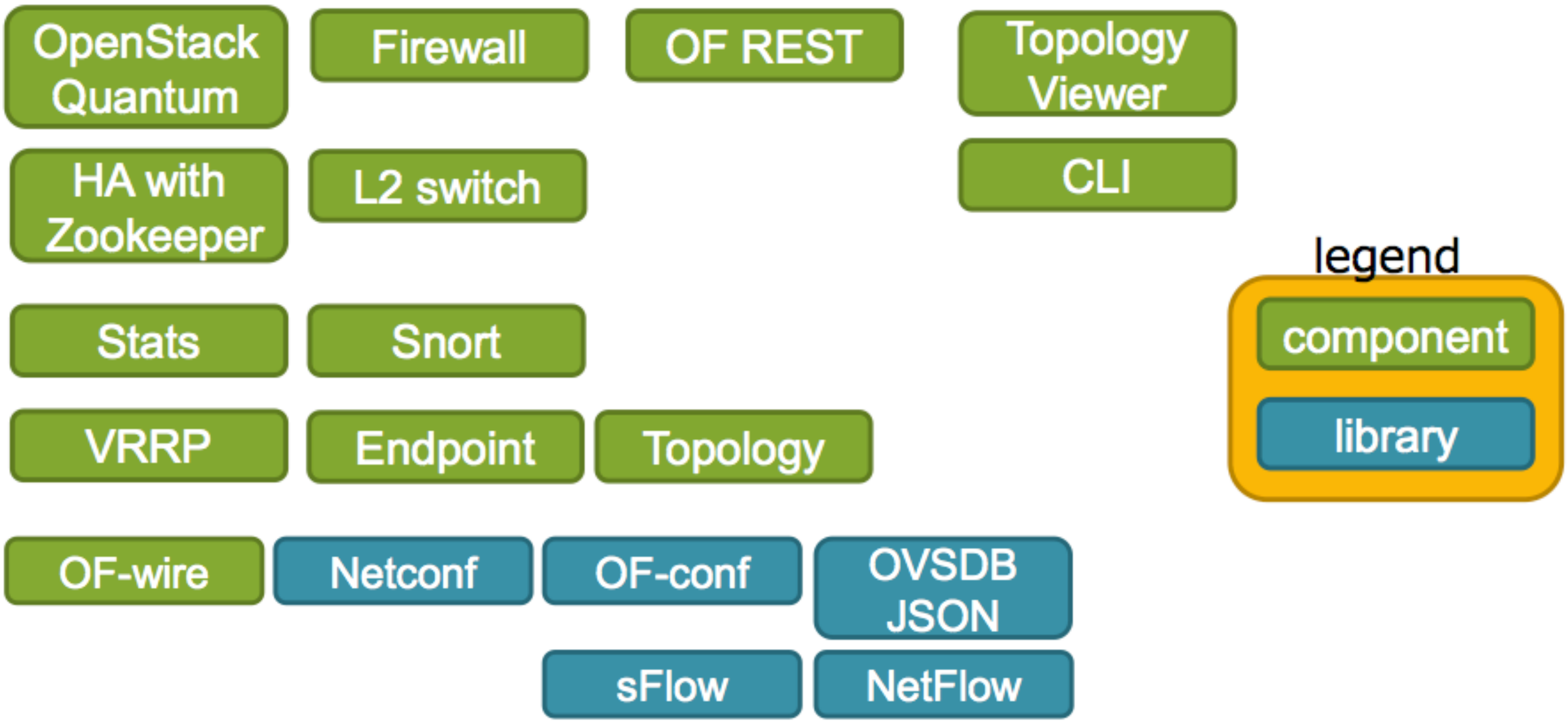
OpenFlow

OpenFlow switch

Ryu: Component-based framework

- Your application consists of component(s)
- Ryu provides a bunch of components useful for SDN applications.
- You can modify the existing components and implement your new components.
- Combines the components to build your application.

Components and libraries included in Ryu



OpenStack Quantum

Firewall

OF REST

Topology Viewer

HA with Zookeeper

L2 switch

CLI

Stats

Snort

VRRP

Endpoint

Topology

OF-wire

Netconf

OF-conf

OVSDB JSON

sFlow

NetFlow

legend

component

library

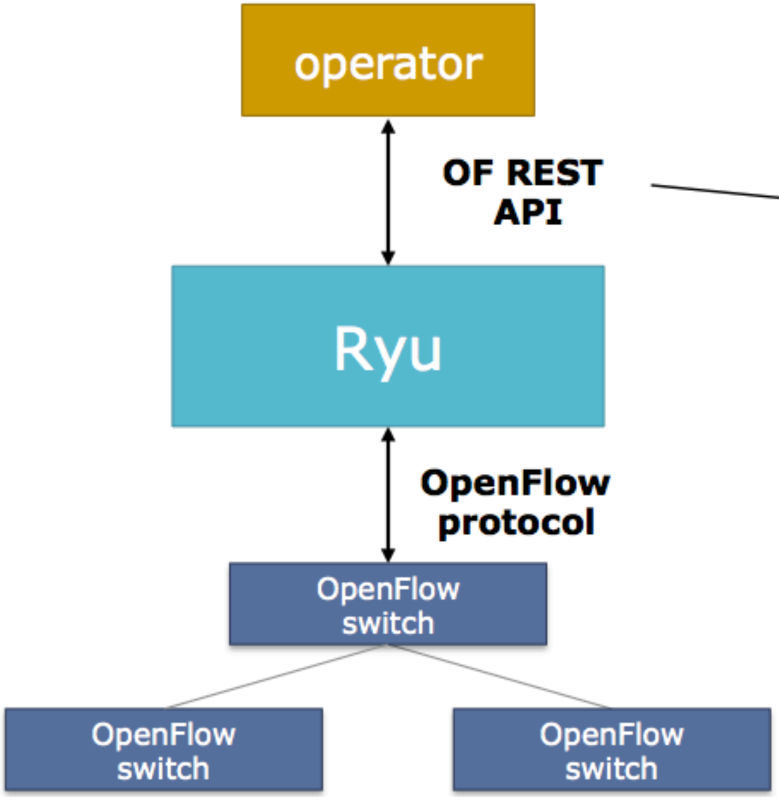
Current Status…

- OpenFlow protocol
  - OF1.0 + nicira extensions, OF1.2, OF1.3, OF-Config 1.1
- Other protocols
  - netconf, vrrp, xFlow, snmp, ovsdb
- Ryu applications/libraries Topology viewer
  - OF REST
  - Firewall
  - Some sample apps are in the ryu/app directory

…Current Status

- Switch Interoperability
  - Referenced by some switch vendors
  - Open vSwitch
    - Integration testing with Open vSwitch (OF1.0, OF1.2) nicira extensions, OVSDB
- Integration with other components
  - HA with Zookeeper
  - IDS (Intrusion Detection System)
  - OpenStack Quantum

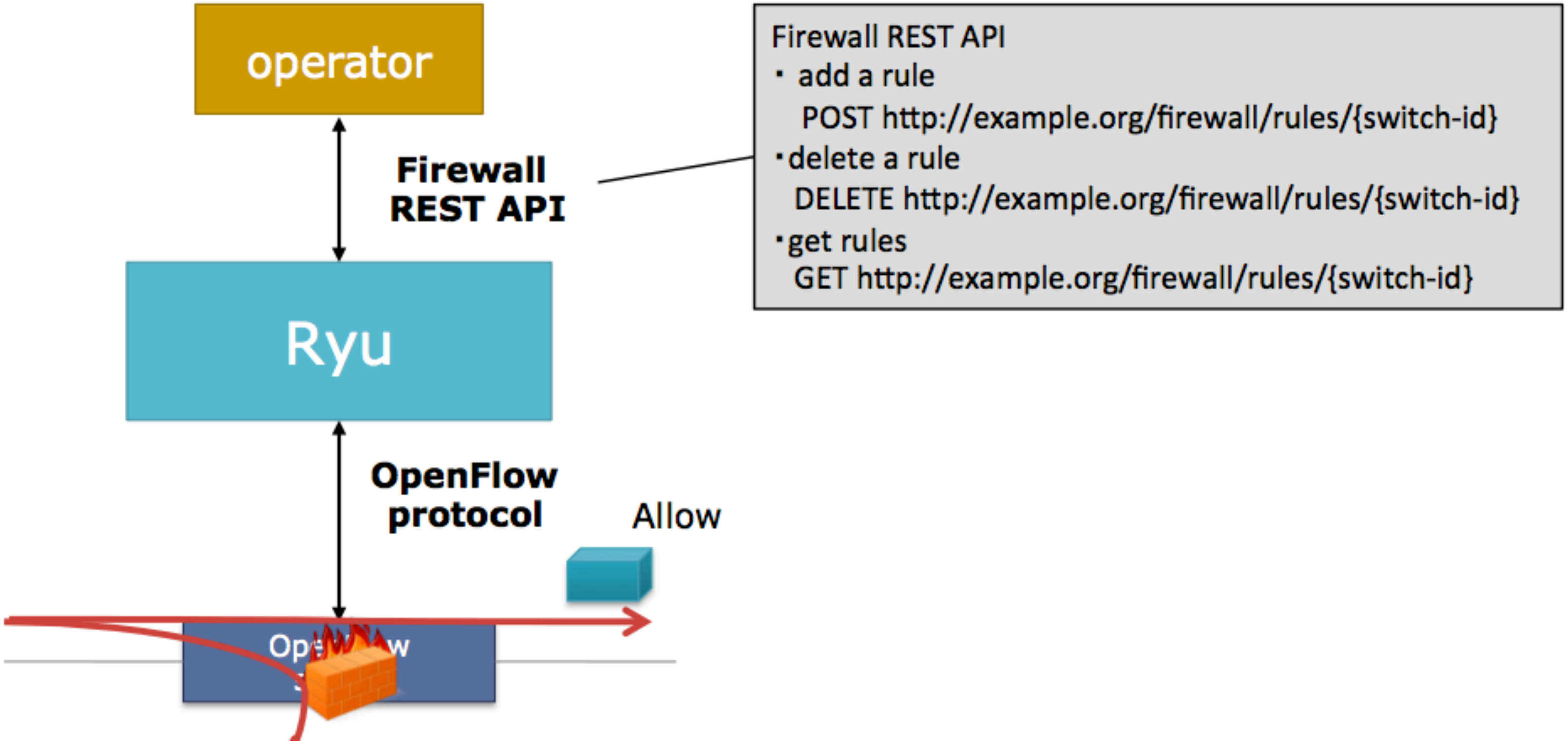
Restful interface available



OF REST API

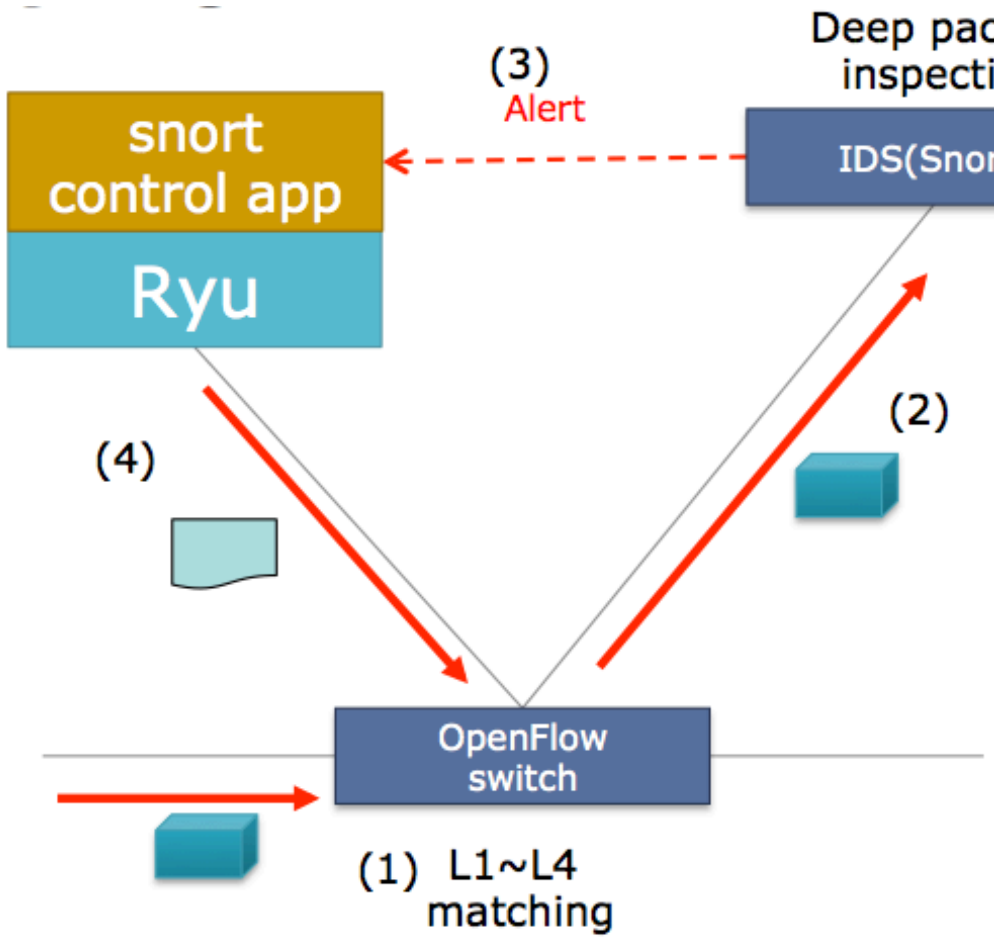
- add a flow entry
  POST http://example.org/stats/flownetry/add
- delete flow entries
  DELETE http://example.org/stats/flowentry/delete
- get flow stats
  GET http://example.org/stats/flow/{dpid}

Firewall

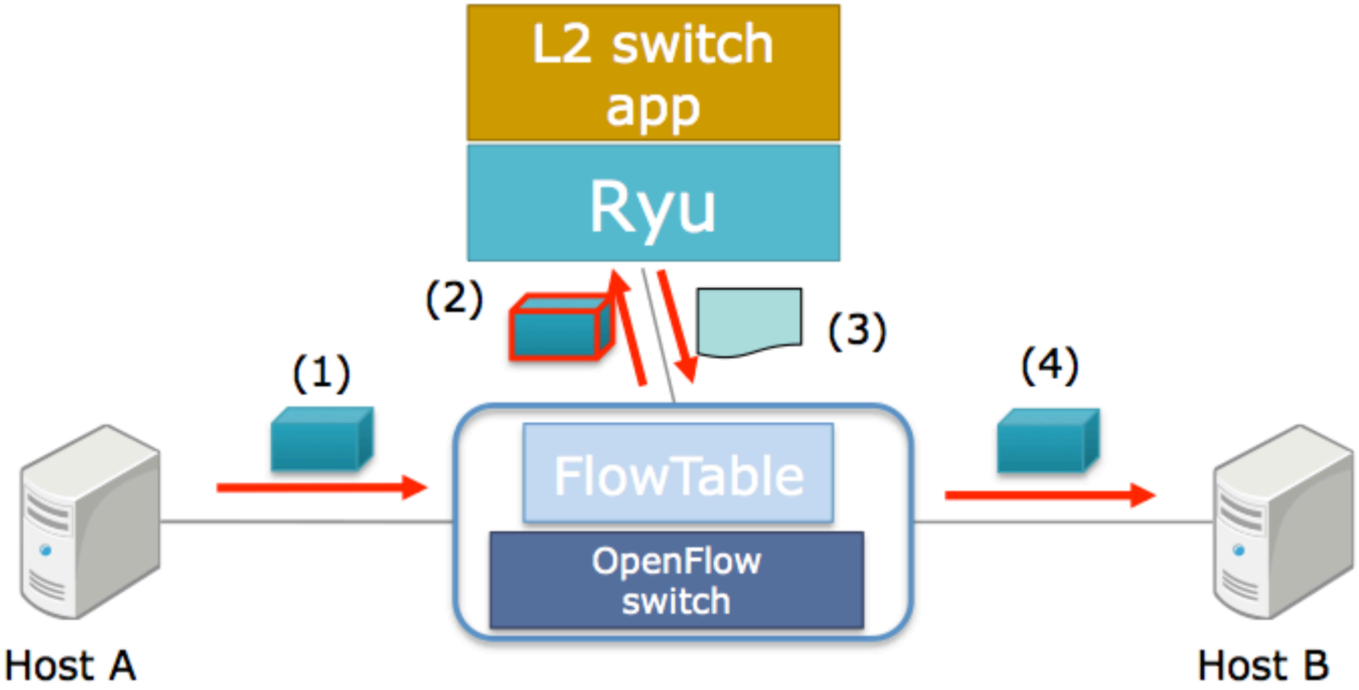


Firewall REST API
- add a rule
  POST http://example.org/firewall/rules/{switch-id}
- delete a rule
  DELETE http://example.org/firewall/rules/{switch-id}
- get rules
  GET http://example.org/firewall/rules/{switch-id}

Intrusion Detection System



L2 switch


L2 switch app
Ryu
(2)
(3)
(1)
(4)
FlowTable
OpenFlow switch
Host A
Host B

Installation

- Using pip command is the easiest option:

```
% pip install ryu
```

- If you prefer to install from the source code:

```
% git clone git://github.com/osrg/ryu.git
% cd ryu; python ./setup.py install
```

What does the code look like?

```
class L2Switch(app_manager.RyuApp):
    def __init__(self, *args, **kwargs):
        super(L2Switch, self).__init__(*args, **kwargs)

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def packet_in_handler(self, ev):
        msg = ev.msg
        dp = msg.datapath
        ofp = dp.ofproto
        ofp_parser = dp.ofproto_parser
        in_port = msg.match['in_port’]
```

…What does the code look like?

```
actions = [ofp_parser.OFPActionOutput(ofp.OFPP_FLOOD)]
out = ofp_parser.OFPPacketOut(
    datapath=dp, buffer_id=msg.buffer_id, in_port=in_port,
    actions=actions)
dp.send_msg(out)
```

- So is this a hub or a switch?
- Should you use OFPPacketOut a lot?

So what’s missing?

- Mac address table
- Port up/down events
- VLANs
- LLDP
- ???

Python Performance?

- You need scalability probably
  - – Language runtime efficiency can’t solve scalability problem
  - – Scalability about the whole system architecture.
- Still need to improve runtime efficiency
  - – Pypy: another python runtime using JIT.
  - – Using C for such components.

Future work

- Make SDN development more agile
  - – Adds more components (protocols, IaaS, stats, security, etc).
  - – Introducing network abstraction model (hide southbound difference, etc).
  - – Improves distributed deployment component (cluster support).
  - – New testing methods (Ryu has more than 15,000 lines test code).

Ryu is an ongoing project

- Ryu project needs more developers
  - NTT team wants to make Ryu usable for many organizations.
  - The development is truly open and Ryu already has some code from non NTT developers.
  - NTT team would like to help you to use Ryu in production.

Links

- http://osrg.github.io/ryu/