Workshop 2012

LuaFlow, an open source Openflow Controller

Raphael Amorim

raphael@atlantico.com.br

raphael.leite@hp.com

Renato Aguiar

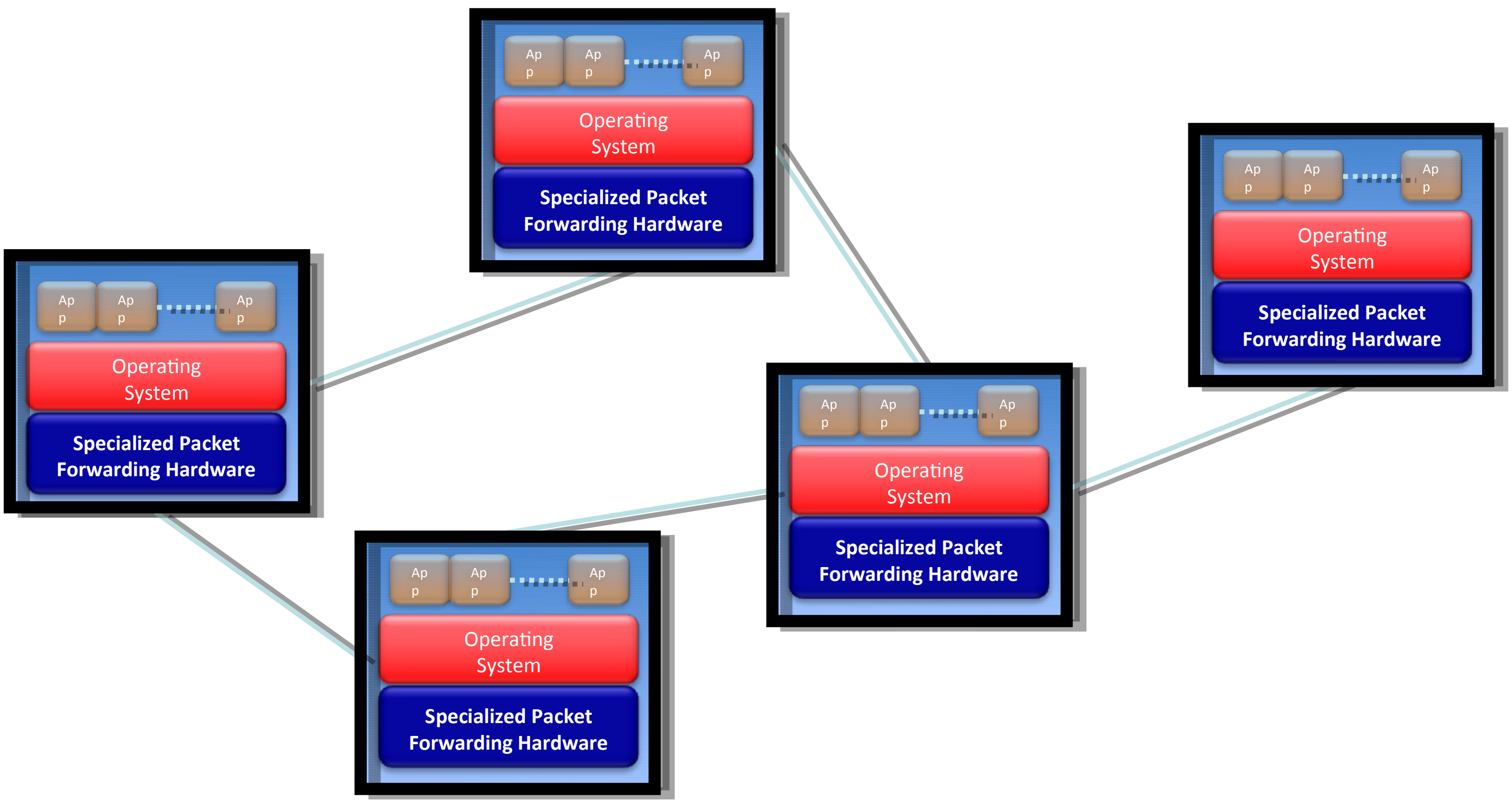
aguiar_renato@atlantico.com.br

Talk Overview

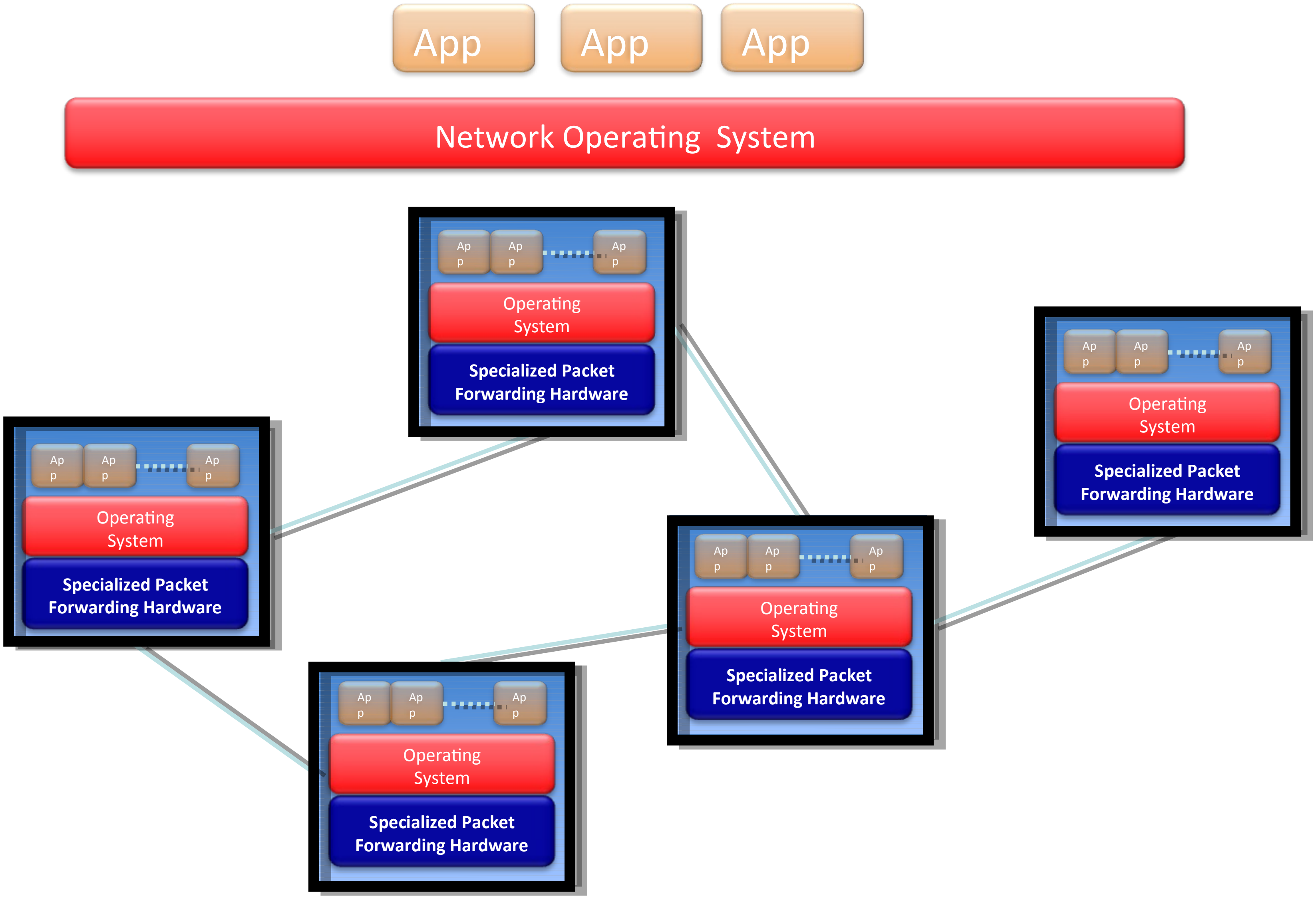
- What is OpenFlow?
- How OpenFlow Works
- Lua Flow approach
- Demo
- Next steps

Current Internet

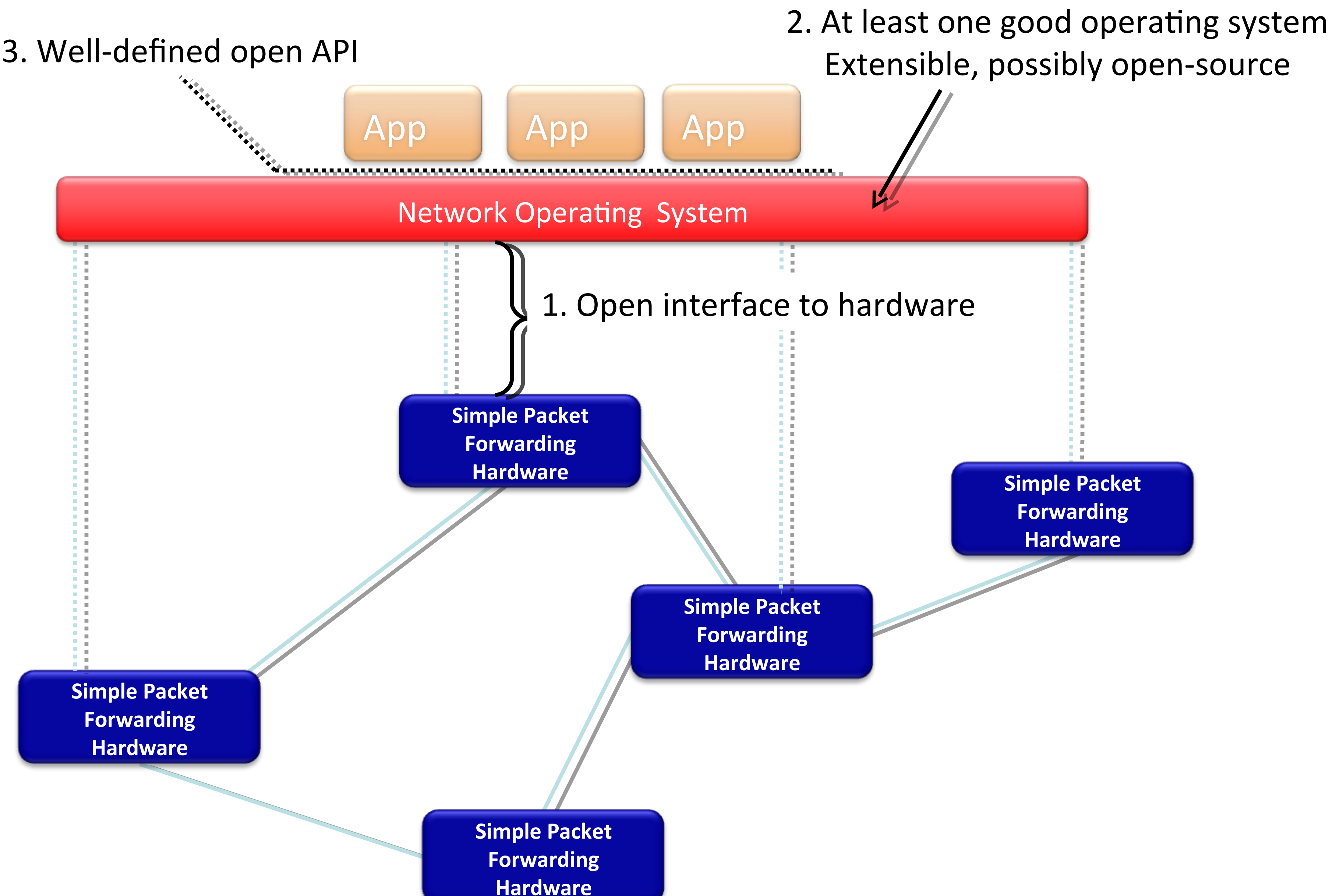
Closed to Innovations in the Infrastructure



“Software Defined Networking” approach to open it



The “Software-defined Network”

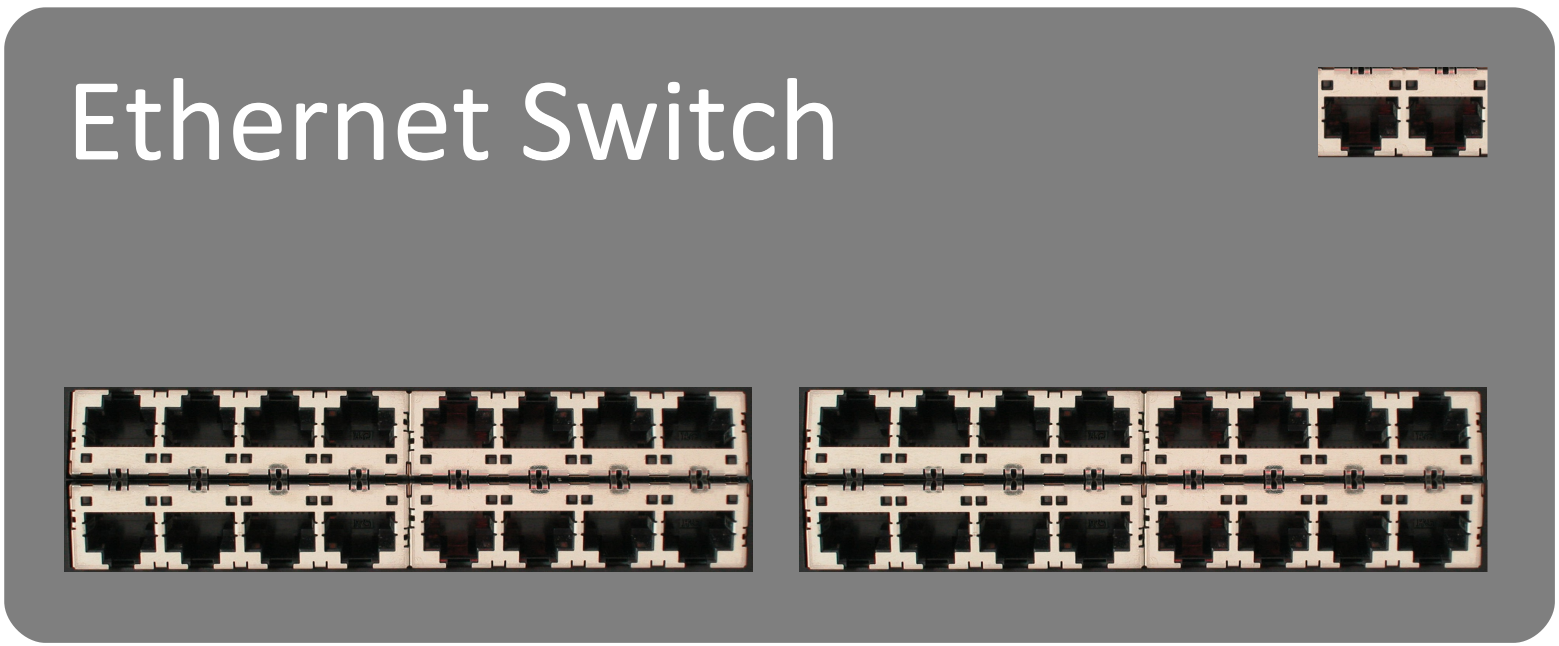


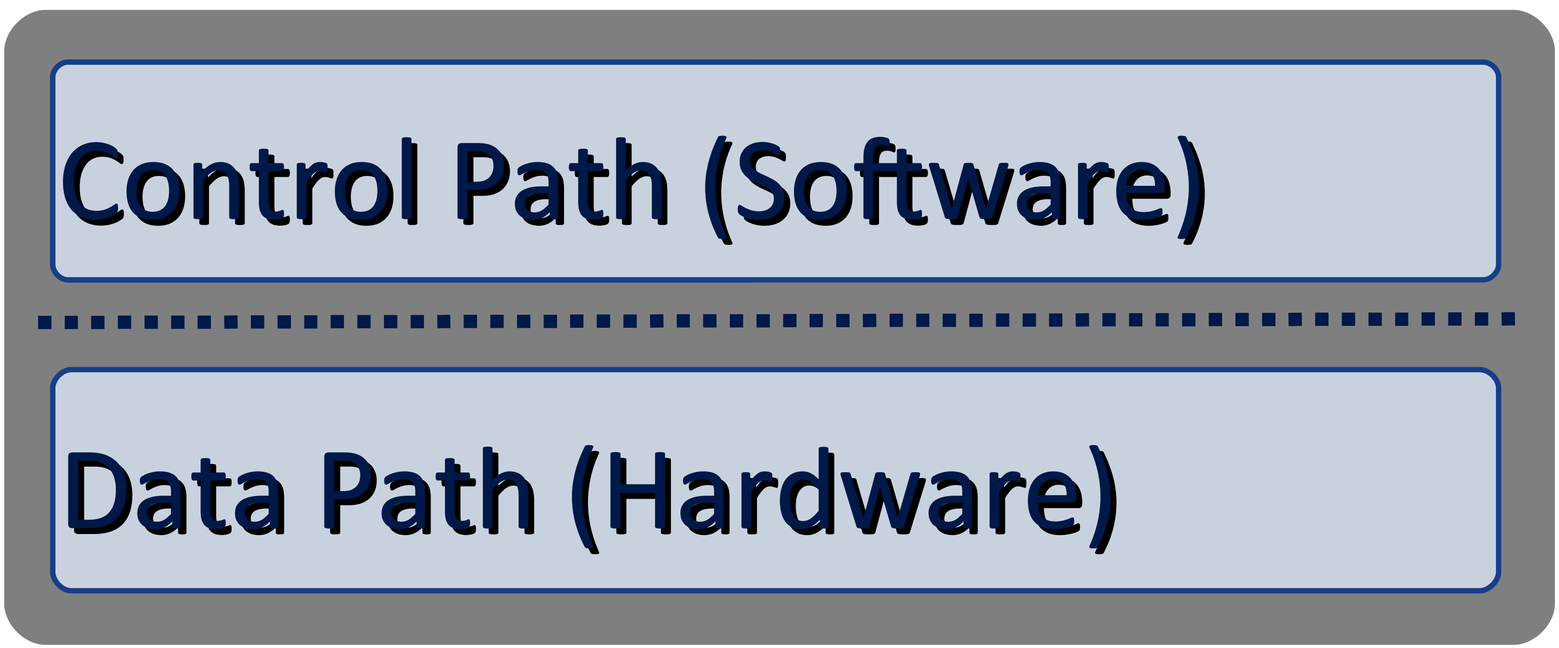
What is OpenFlow?

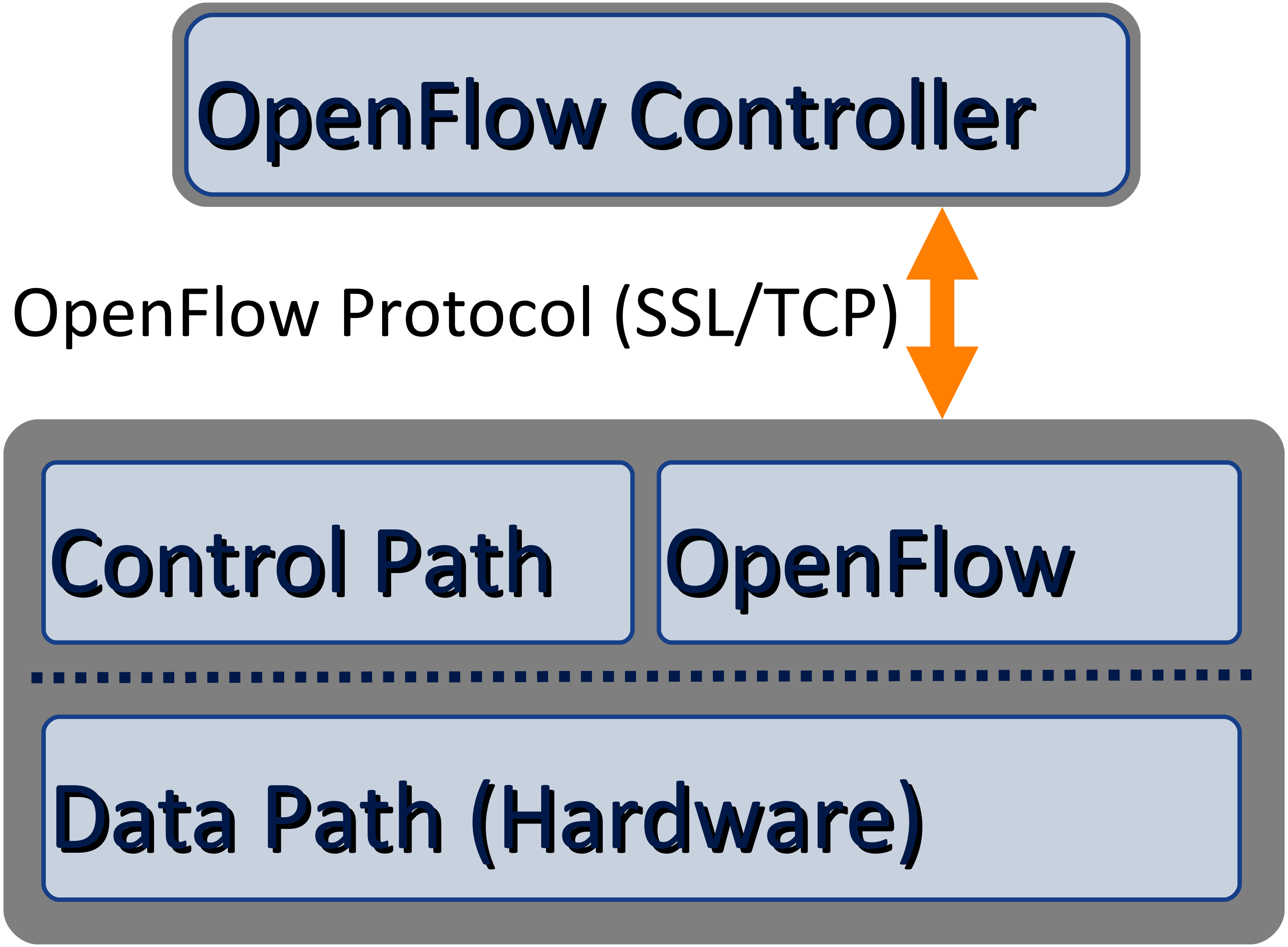
Short Story: OpenFlow is an API

- Control how packets are forwarded
- Make deployed networks programmable
  - not just configurable
- Makes innovation easier
- Goal (experimenter's perspective):
  - No more special purpose test-beds
  - Validate your experiments on deployed hardware with real traffic at full line speed

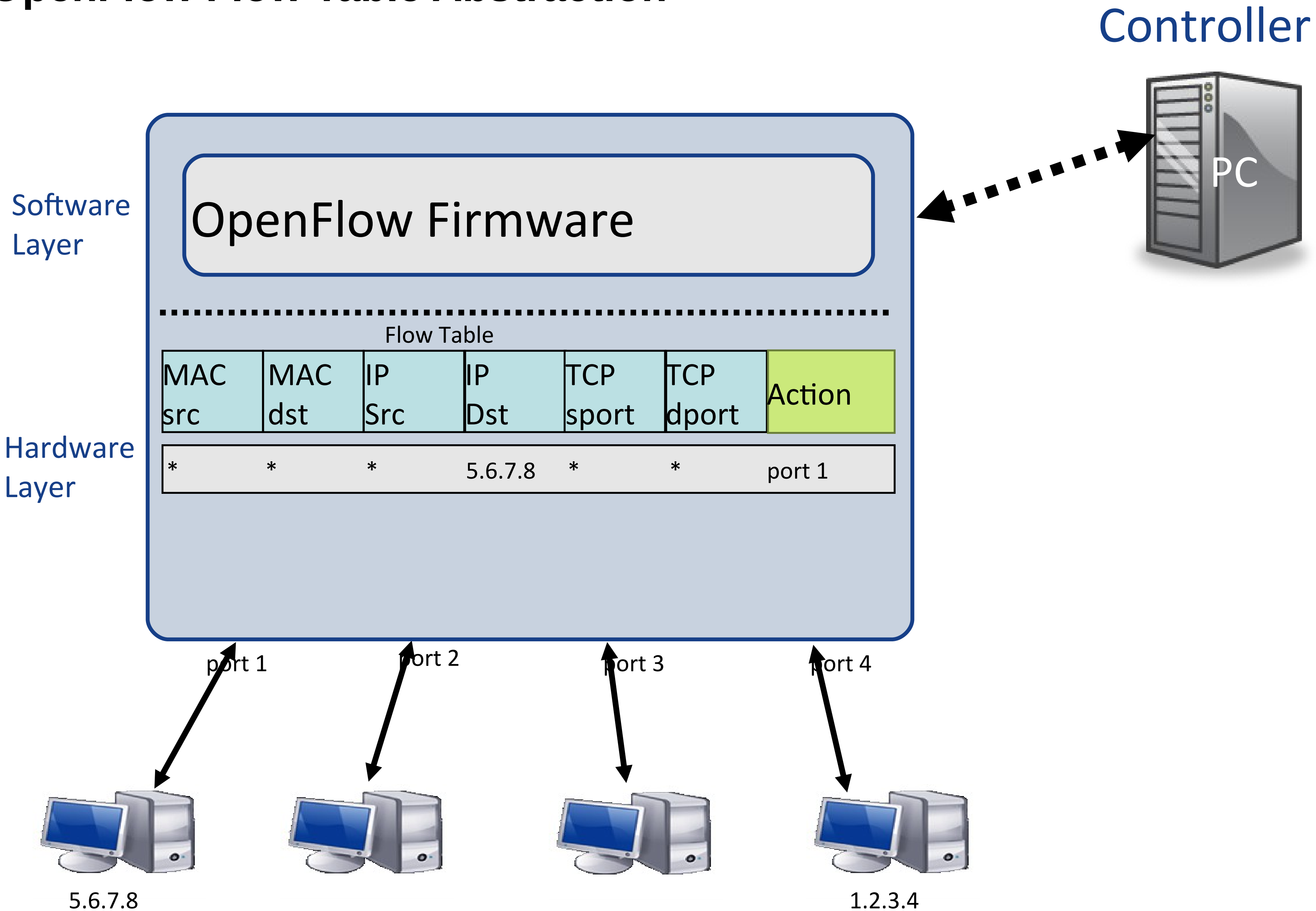
How Does OpenFlow Work?


Ethernet Switch


Control Path (Software)
Data Path (Hardware)


OpenFlow Controller
OpenFlow Protocol (SSL/TCP)
Control Path
OpenFlow
Data Path (Hardware)

OpenFlow Flow Table Abstraction



OpenFlow Basics

Flow Table Entries

Rule	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

+ mask what fields to match

Examples

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

Examples

Routing

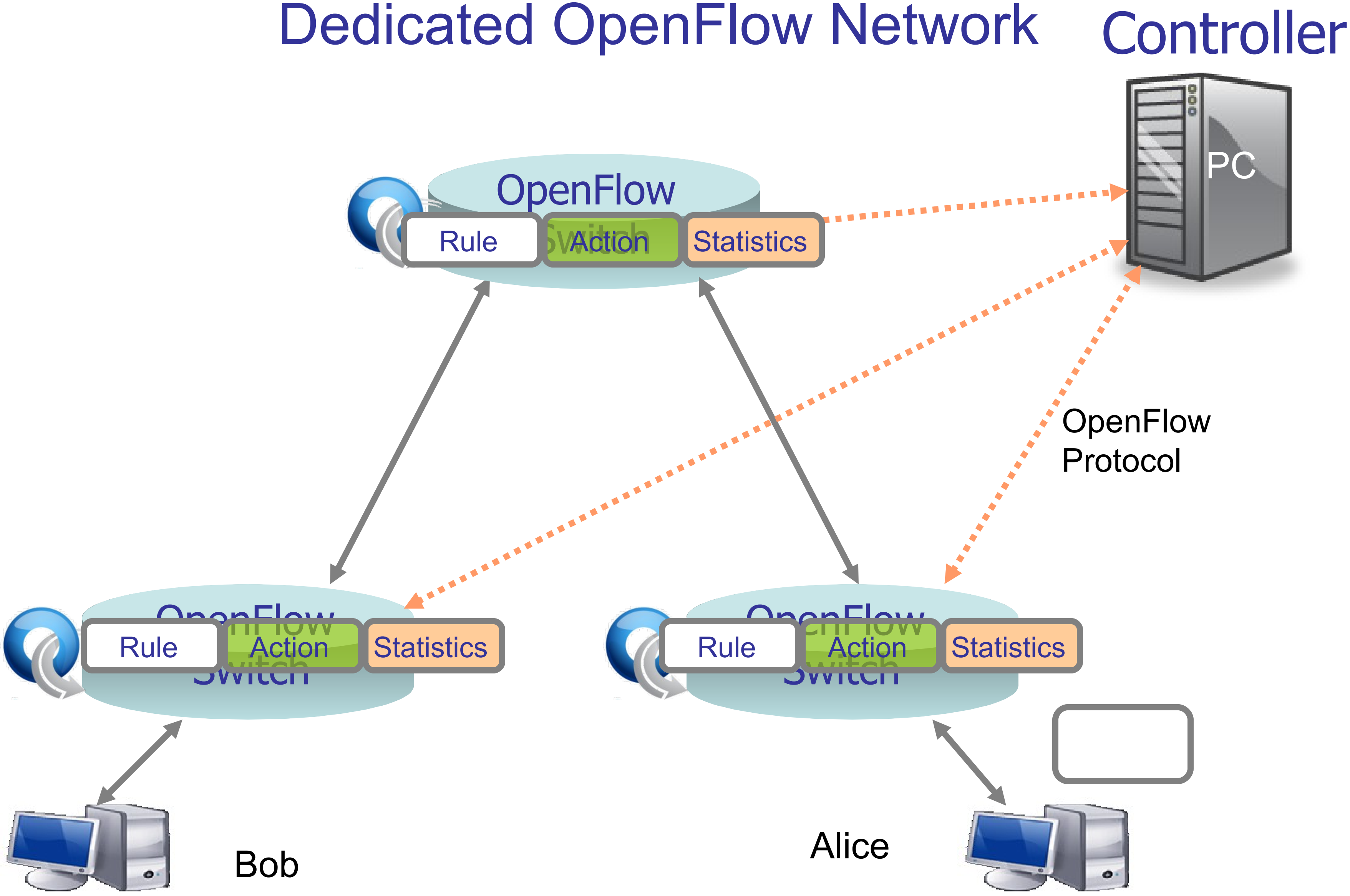
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

VLAN Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

OpenFlow Usage

Dedicated OpenFlow Network

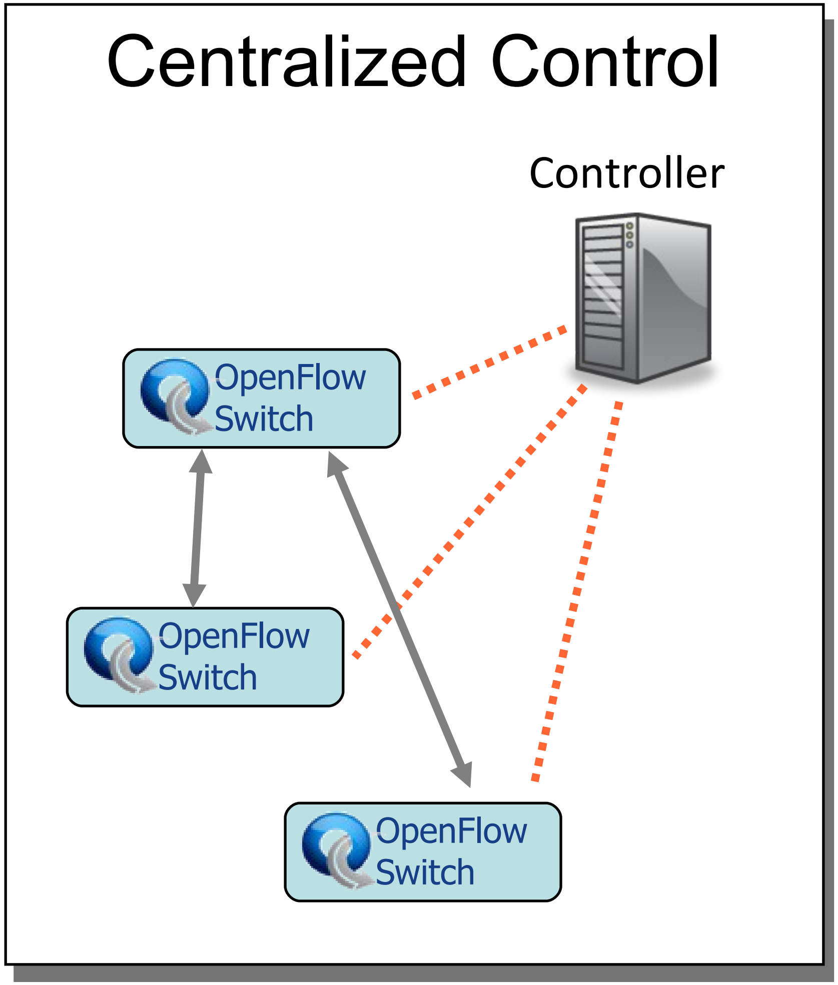


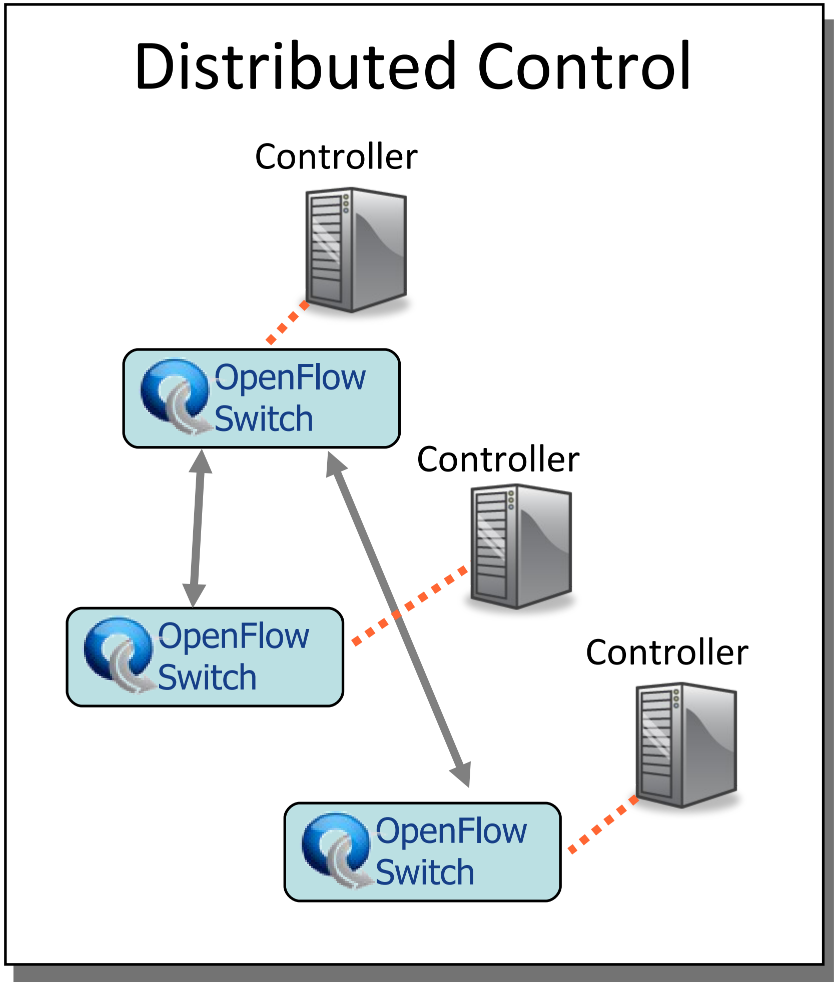
Experiment Design Decisions

- Forwarding logic (of course)
- Centralized vs. distributed control
- Fine vs. coarse grained rules
- Reactive vs. Proactive rule creation

- Likely more: open research area

Centralized vs Distributed Control





Flow Routing vs. Aggregation

Both models are possible with OpenFlow

Flow-Based

- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control, e.g. campus networks

Aggregated

- One flow entry covers large groups of flows
- Wildcard flow entries
- Flow table contains one entry per category of flows
- Good for large number of flows, e.g. backbone

Reactive vs. Proactive

Both models are possible with OpenFlow

Reactive

- First packet of flow triggers controller to insert flow entries
- Efficient use of flow table
- Every flow incurs small additional flow setup time
- If control connection lost, switch has limited utility

Proactive

- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggregated (wildcard) rules

Examples of OpenFlow in Action

- VM migration across subnets
- Identity-Based QoS
- Energy-efficient data center network
- Network slicing
- Load balancing (DNS for instance)

Industry Embracing SDN

Largest Network Providers/Operators

BT
Google
verizon
INTERNET2
NLR
Level(3)
docomo
Microsoft
facebook
amazon.com

More...

Vendors and start-ups

BROADCOM
JUNIPER NETWORKS
ERICSSON
hp invent
NEC
ciena
DELL
NETGEAR Connect with Innovation
extreme networks
ARISTA
nicira
FUJITSU
HUAWEI
cisco
MARVELL
big switch networks
IBM

More...

Slide Credits

- Guido Appenzeller
- Nick McKeown
- Guru Parulkar
- Brandon Heller
- Rob Sherwood
- Lots of others
  - (this slide was also stolen)

LuaFlow's approach

Official Open Source controllers

- NOX (Python/C)
  - Mixed approach
- Beacon (Java)
  - Focus in production environments
  - Java “enterprise” code
- Trema (Ruby)
  - Focus on prototyping testing

Write it short

There's a strong correlation between the length of code (number of tokens) and programmers' productivity

e.g. Arc Programming Language [Paul Graham]

With smaller code:

- less time to write consistent code
- less chances for bugs

LuaFlow is specialized for programmers' productivity,
But not compromising efficiency

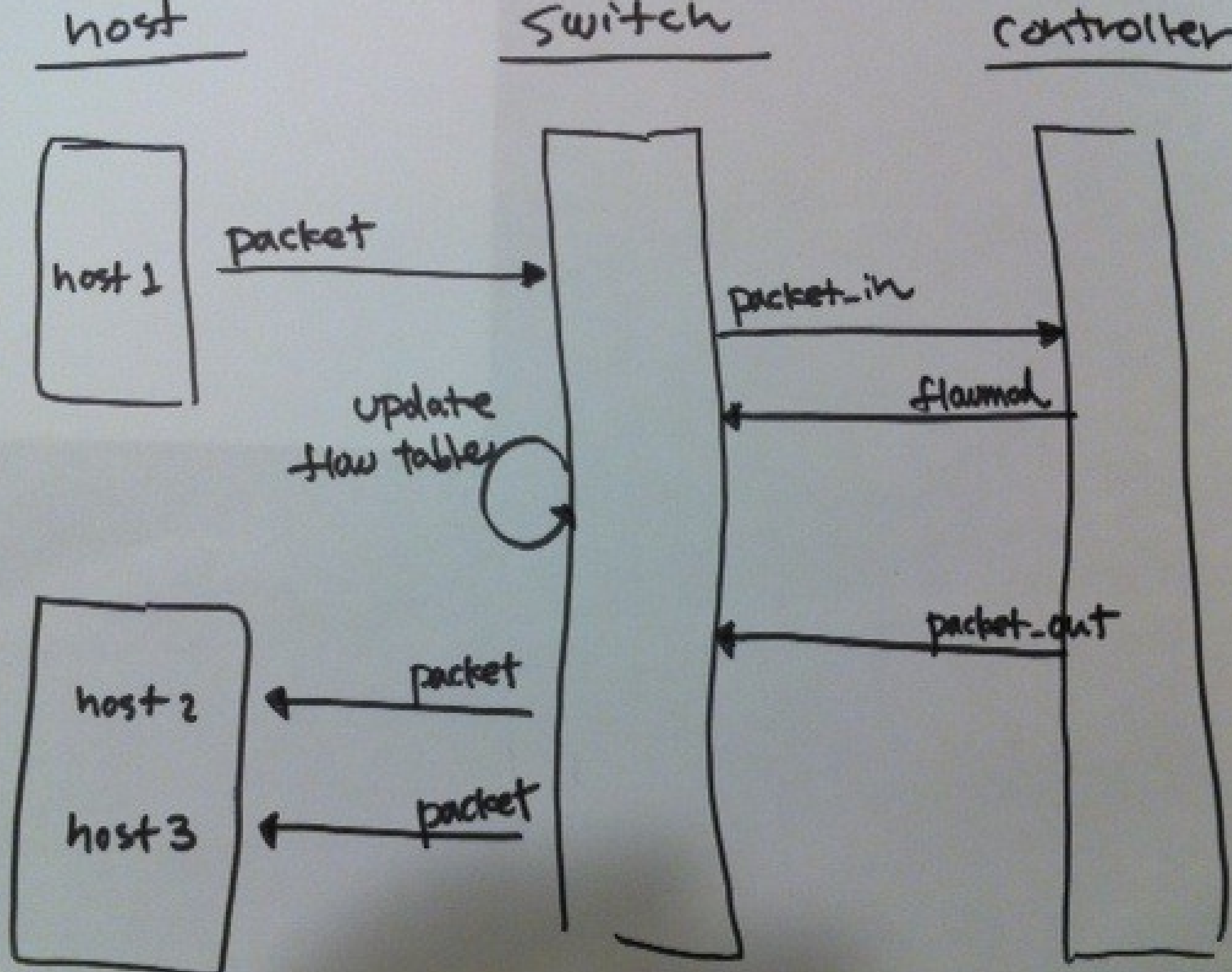
Why LuaFlow

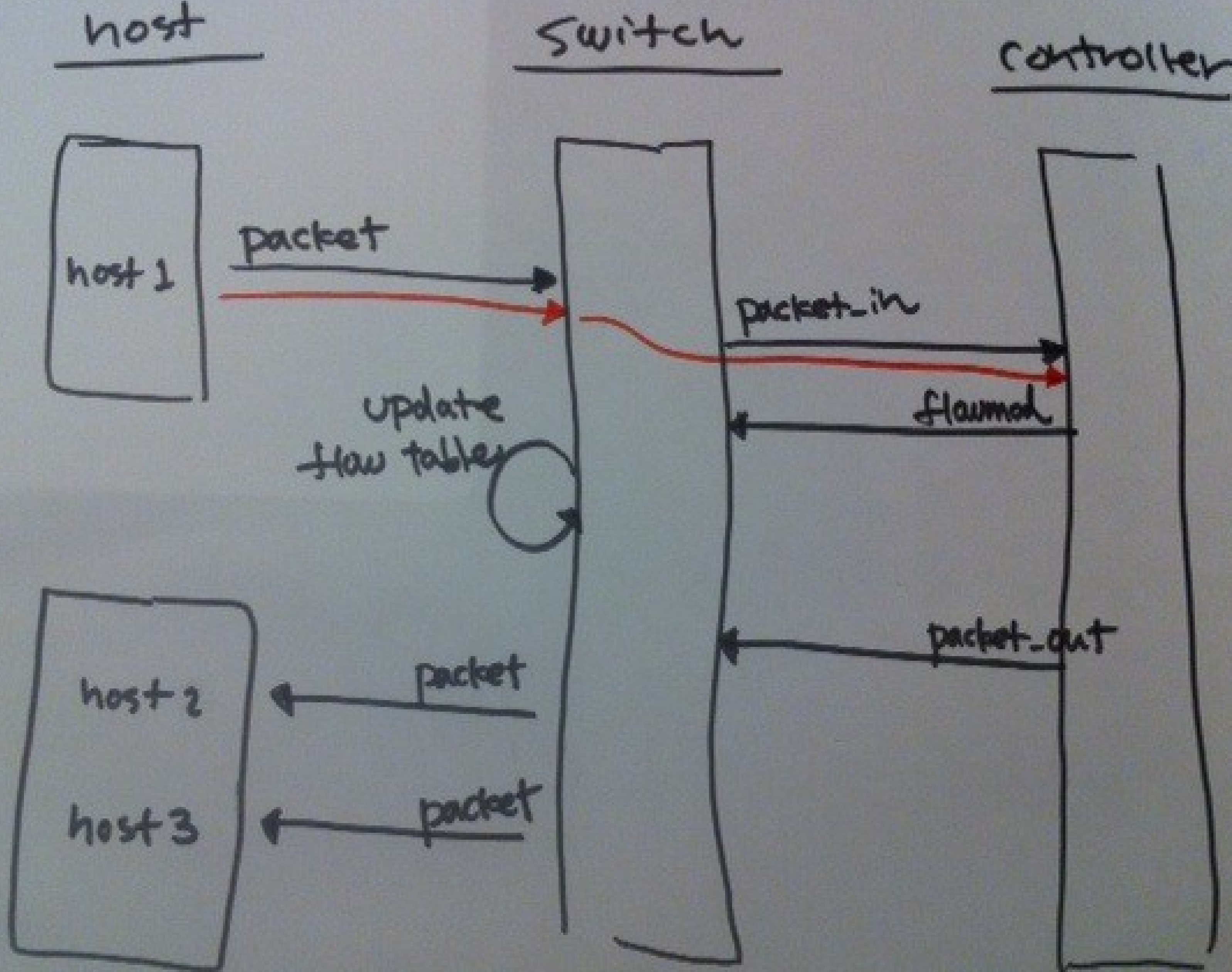
... because we write it in C and Lua
(NOX written in C++ and Python, Beacon written in Java)

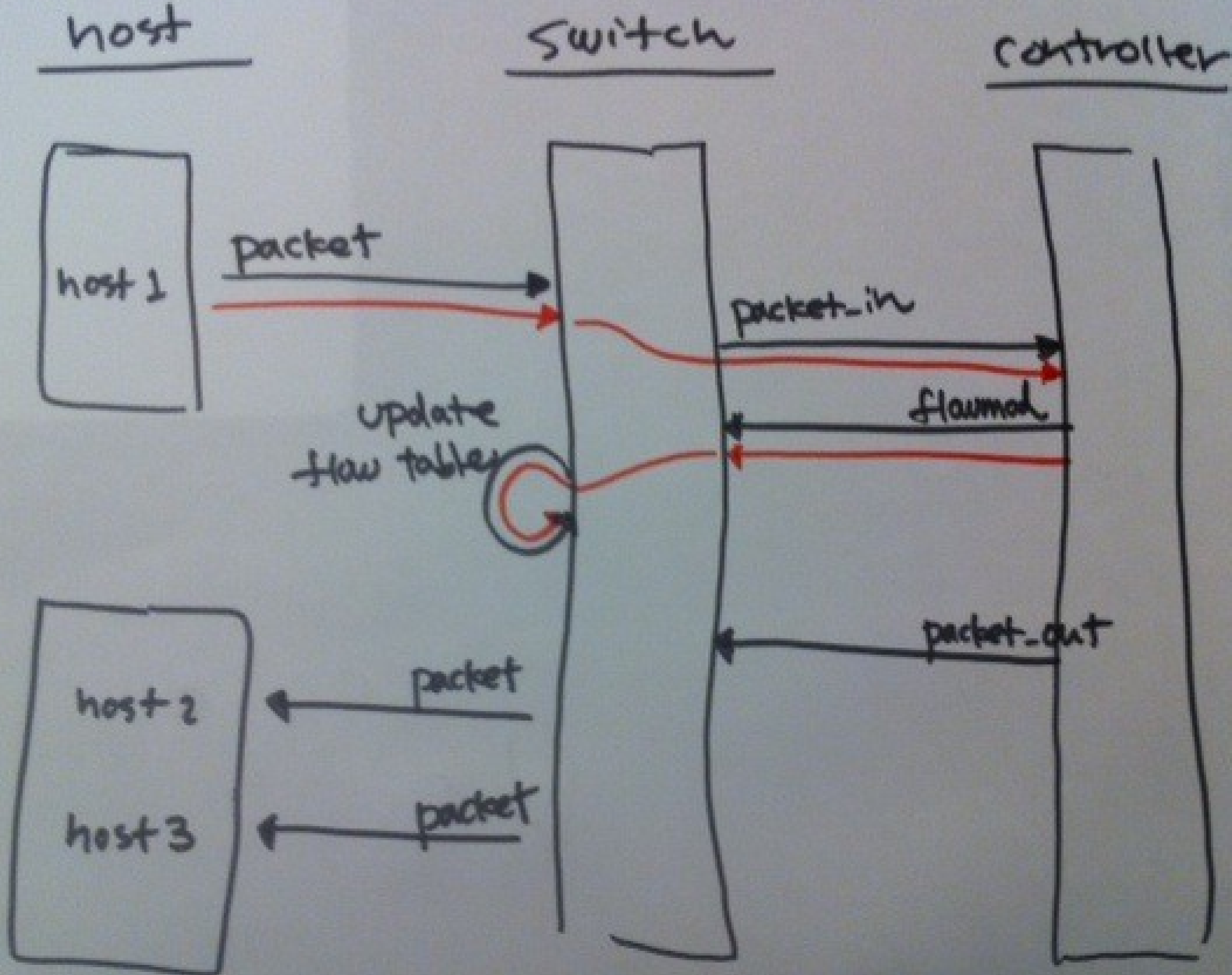
This is the main reason!

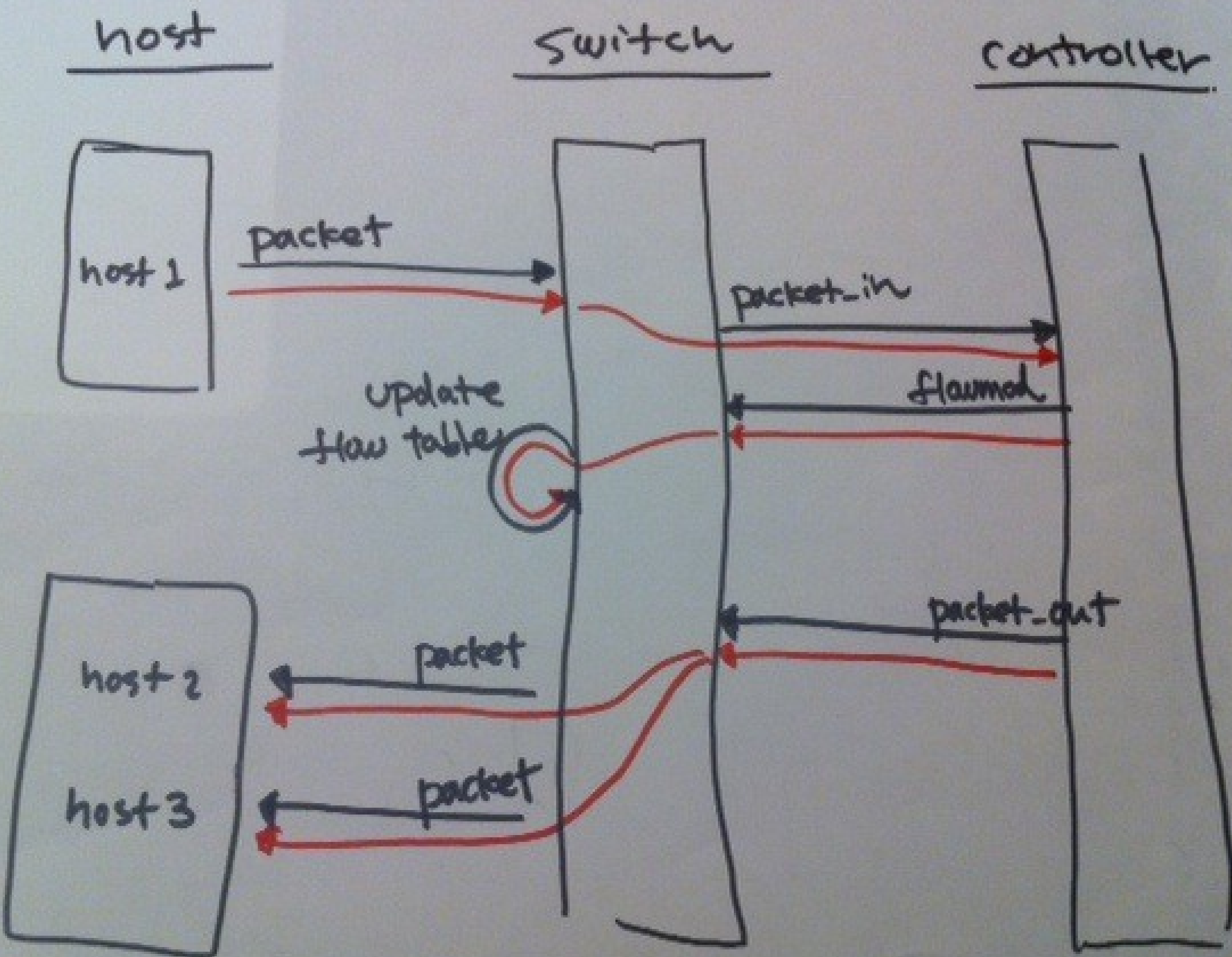
Network configuration file

```
switches{
switch1 = {datapath_id = "00:00:00:00:00:00:00:01"},
switch2 = {datapath_id = "00:00:00:00:00:00:00:02"},
}

hosts{
host1 = {mac = "00:00:00:00:00:03"},
host2 = {mac = "00:00:00:00:00:04"},
}

-- Connections: Connection.switch[port#] = {switch=port#} or
--              Connection.switch[port#] = {host} or
--              Connection.host = {switch=port#}

Connection.host1       = { switch1 = 2}
Connection.host2       = { switch2 = 2}
Connection.switch1[1]  = { switch2 = 1}
```


host
switch
controller
host 1
packet
packet_in
flowmod
update
flow table
packet_out
host 2
packet
host 3
packet


host
switch
controller
host 1
packet
packet_in
flowmod
update
flow table
packet_out
host 2
packet
host 3
packet


host
switch
controller
host 1
packet
packet_in
flowmod
update
flow table
packet_out
packet
host 2
packet
host 3


host
switch
controller
host 1
packet
packet_in
flowmod
update
flow table
packet_out
host 2
packet
host 3
packet

 # Luaflow

```
add_simple_flow(dpid,
                flow,
                buffer_id,
                out_port,
                cache_timeout)
```

VS

 # NOX Python

```
inst.install_datapath_flow(
  dpid,
  extract_flow(packet),
  CACHE_TIMEOUT,
  openflow.OFP_FLOW_PERMANENT,
  [[openflow.OFPAT_OUTPUT, [0, prt[0]]]],
  bufid,
  openflow.OFP_DEFAULT_PRIORITY,
  inport,
  buf
)
```

Catching network events

```
function switch_ready(dpid, features)
  print(">> New switch connected: " .. dpid)
  for k,v in pairs(features) do
    if k == "ports" then
      for i,p in ipairs(v) do
        print("Port " .. i)
        for k1,v1 in pairs(p) do
          print(k1, v1)
        end
      end
    else
      print(k, v)
    end
  end
end
```

Catching network events

```
function packet_in(dpid, buffer_id, flow)
  print(">> New packet (" .. buffer_id .. ") received from " .. dpid)
  local idle_timeout = 10
  local out_port = "all"
  add_simple_flow(dpid, flow, buffer_id, out_port, idle_timeout)
end
```

Base classes

- base_config.lua
- custom_topology_config.lua
- Topology.lua
- Port.lua
- Host.lua
- Switch.lua
- Link.lua
- Dijkstra.lua
- Controller.lua
- Flow.lua

Base classes

```
require "Topology"

myTopology = Topology:new{name = "mininet"}
myTopology:load_config("custom_topology_config.lua")

function switch_ready(dpid, features)
    print(">> New switch connected: " .. dpid)
    --TODO
    --Insert switch features into switch objects
end

function packet_in(dpid, buffer_id, flow)

  print(">> New packet received from " .. dpid)
  route = myTopology:getRoute(flow.dl_src, flow.dl_dst)
  ...
end
```

Demo

Next steps

- Pure lua controller using ffi/luajit
- More real-world scenarios
- Serious evaluation
- Open WRT Openflow wireless devices
- Community pull-requests
  - Both ideas & Code

Thank you all

Questions?