Software Defined Networking

Sarath Babu

Indian Institute of Space Science and Technology
Thiruvananthapuram, Kerala, India 695547

7th June, 2017

Outline

Traditional networks

- Routers with network protocols
- Distributed control
- Complex structure makes device management difficult
- Dependence on hardware vendors

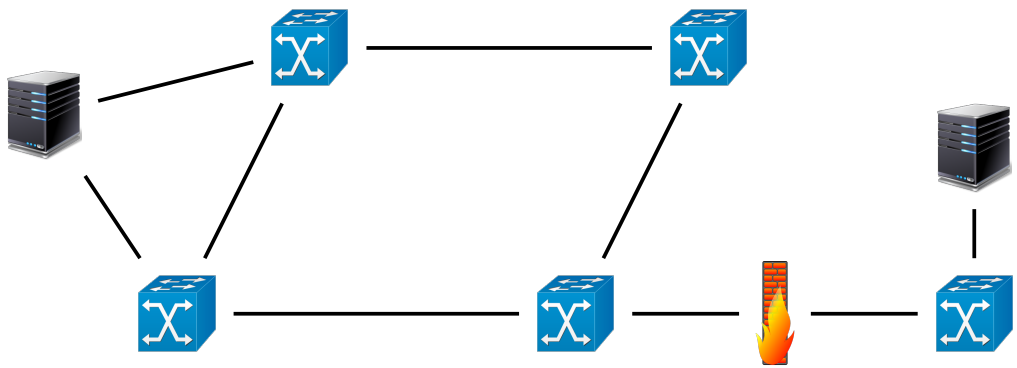


Figure 1: Traditional network structure

Network device

- Data Plane + Control Plane in same device
- **Control Plane**
  - Takes the control decisions
- **Data Plane**
  - Forwards the data as per the control decision from control plane

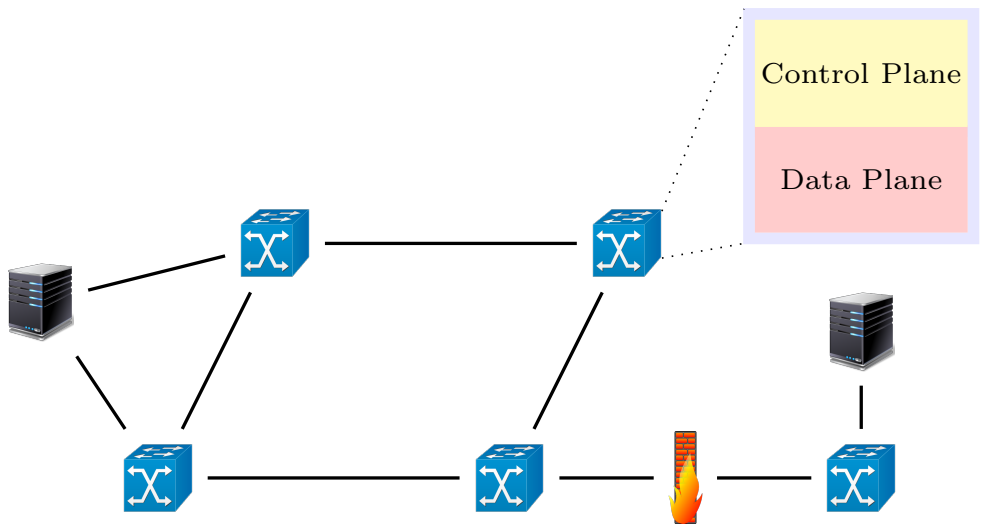


Figure 2: Network device

Cons of existing paradigm

- [1]Limitations to fulfill evolving business needs
- Protocols developed in isolation
- Vendor dependence
- Assumption of static nature of networks
- Difficulty in scalability
- Excessive effort in adopting new policies

[1]Open Networking Fundation. "Software-defined networking: The new norm for networks". In: *ONF White Paper* (2012).

Cons of existing paradigm

- [1]Limitations to fulfill evolving business needs
- Protocols developed in isolation
- Vendor dependence
- Assumption of static nature of networks
- Difficulty in scalability
- Excessive effort in adopting new policies

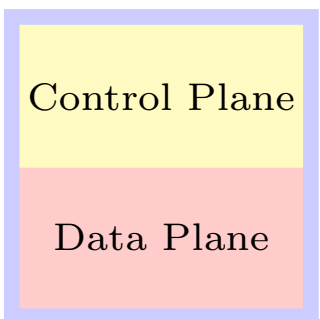
How to overcome?

[1]Open Networking Fundation. "Software-defined networking: The new norm for networks". In: *ONF White Paper* (2012).

Cons of existing paradigm

- [1]Limitations to fulfill evolving business needs
- Protocols developed in isolation
- Vendor dependence
- Assumption of static nature of networks
- Difficulty in scalability
- Excessive effort in adopting new policies

How to overcome?

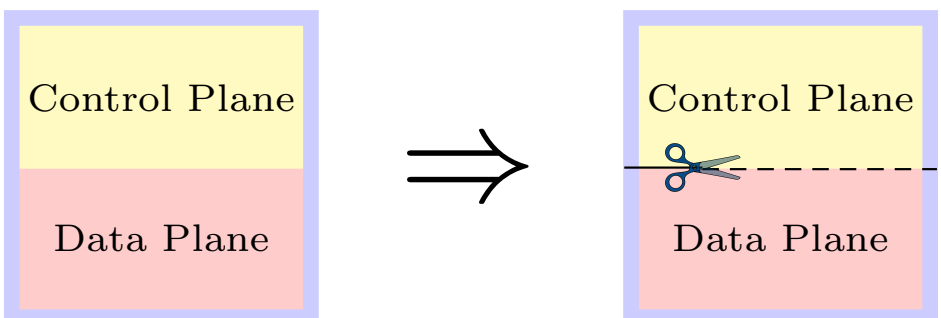


[1]Open Networking Fundation. "Software-defined networking: The new norm for networks". In: *ONF White Paper* (2012).

Cons of existing paradigm

- [1]Limitations to fulfill evolving business needs
- Protocols developed in isolation
- Vendor dependence
- Assumption of static nature of networks
- Difficulty in scalability
- Excessive effort in adopting new policies

How to overcome?

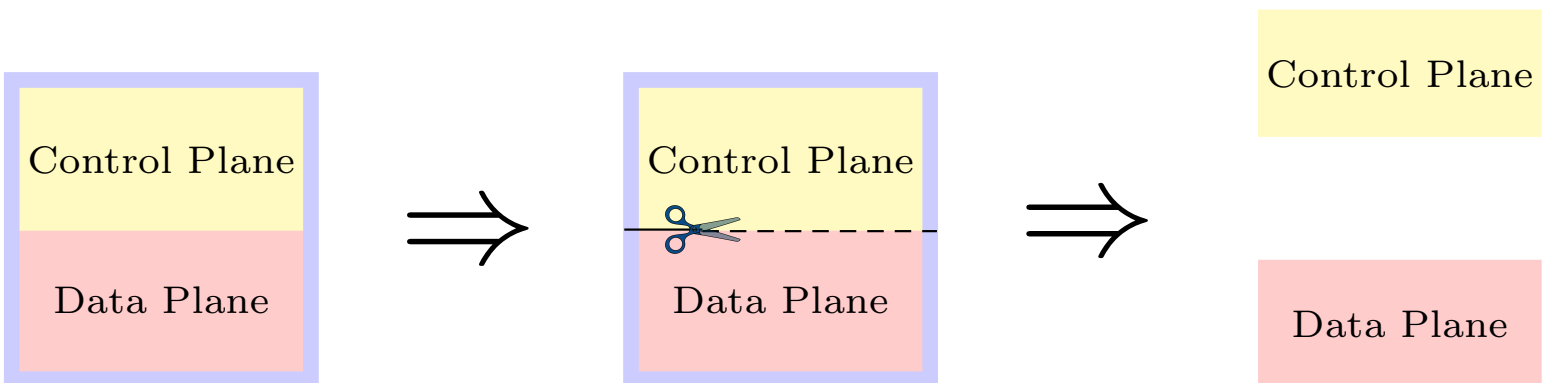


[1]Open Networking Fundation. "Software-defined networking: The new norm for networks". In: *ONF White Paper* (2012).

Cons of existing paradigm

- [1]Limitations to fulfill evolving business needs
- Protocols developed in isolation
- Vendor dependence
- Assumption of static nature of networks
- Difficulty in scalability
- Excessive effort in adopting new policies

How to overcome?



[1]Open Networking Fundation. "Software-defined networking: The new norm for networks". In: *ONF White Paper* (2012).

Software Defined Network (SDN)

Decouples Control Plane from Data Plane

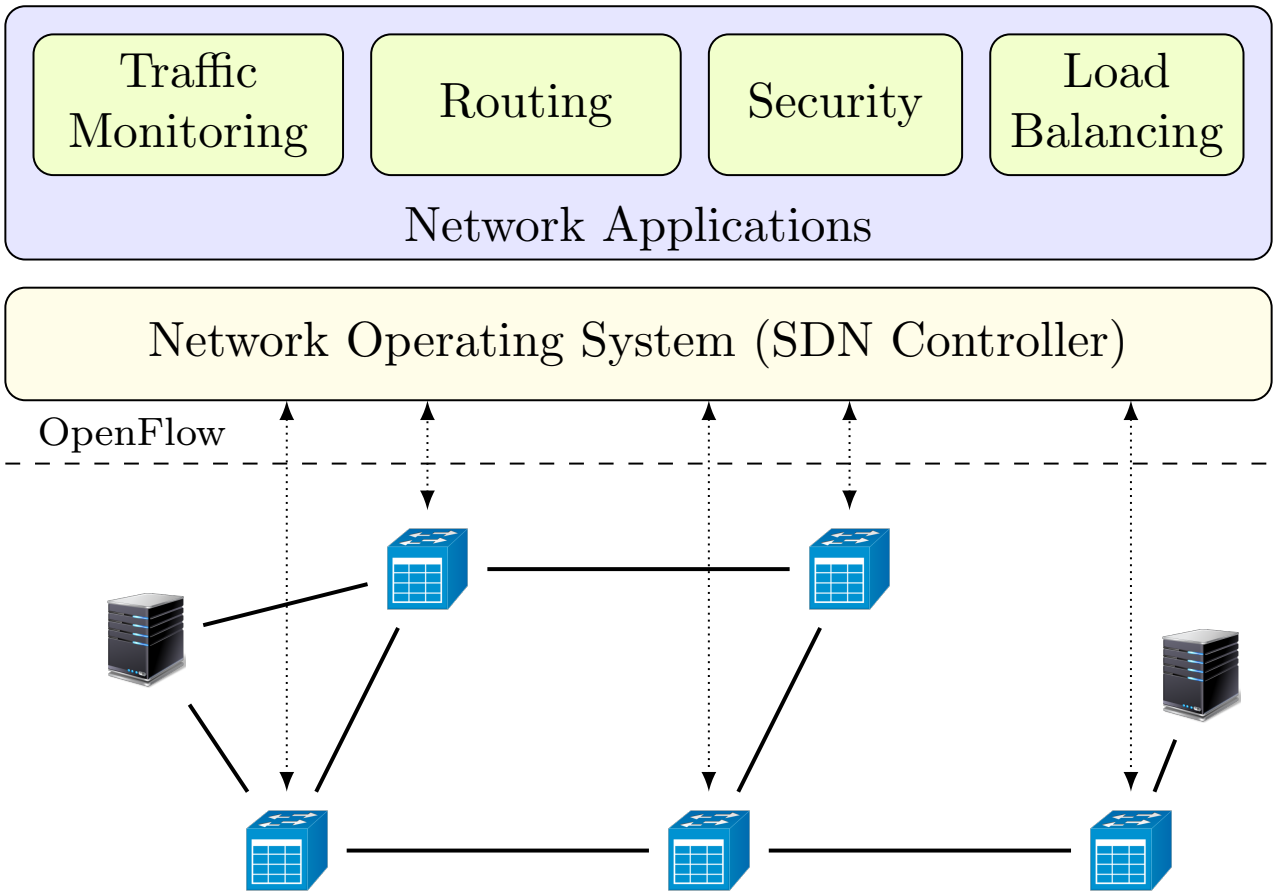


Figure 3: SDN architecture[2]

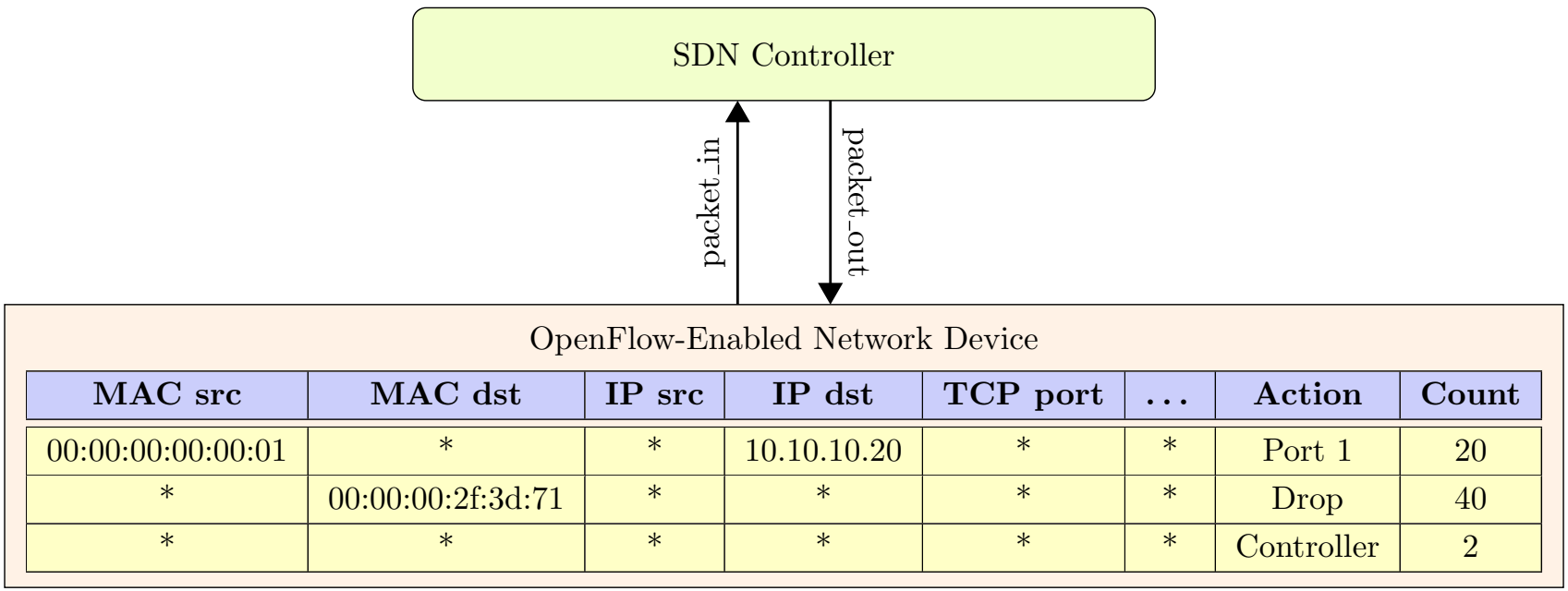
[2] Diego Kreutz et al. "Software-defined networking: A comprehensive survey". In: *Proceedings of the IEEE* 103.1 (2015), pp. 14–76

Data plane

- Primary function is to **forward** the data
- Consists of networking devices
  - Switches
  - Routers
  - Middleboxes
- Communicates to Network OS (SDN controller) through **OpenFlow** protocol
- Forwards the data using rules sent by the SDN controller

OpenFlow switch

- Software/hardware forwards data in SDN
- Components
  1. Flow table
  2. Secure channel to controller
  3. OpenFlow protocol



MAC src	MAC dst	IP src	IP dst	TCP port	...	Action	Count
00:00:00:00:00:01	*	*	10.10.10.20	*	*	Port 1	20
*	00:00:00:2f:3d:71	*	*	*	*	Drop	40
*	*	*	*	*	*	Controller	2

Figure 4: OpenFlow switch[3]

[3] Open Networking Fundation. "Software-defined networking: The new norm for networks". In: *ONF White Paper* (2012)

OpenFlow switch

Software switch

- Software which enables the SDN operation[4]
- Examples
  - **Open vSwitch** (Open community)
  - OpenFlow Reference (Stanford University)
  - XorPlus (Pica8)

Hardware switch

- Hardware dedicated for OpenFlow switch operation
- Examples
  - RackSwitch G8264 (IBM)
  - CX600 Series (Huawei)
  - Pica83920 (Pica8)
  - 8200zl and 5400zl (Hewlet-Packard)

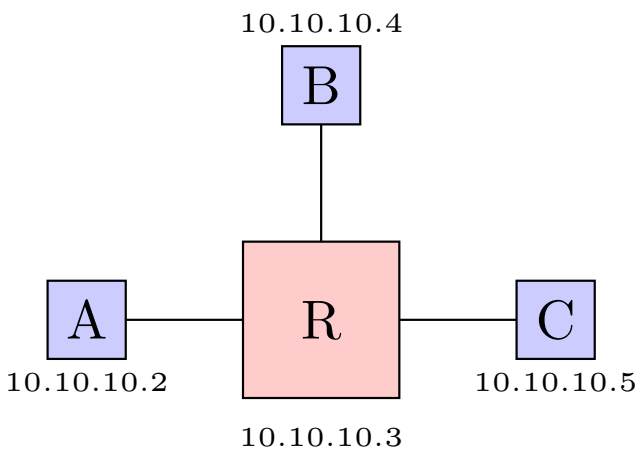
[4] Rahim Masoudi and Ali Ghaffari. "Software defined networks: A survey". In: *Journal of Network and Computer Applications* 67 (2016), pp. 1–25.

Creating an OpenFlow Switch

- Uses the tool Open vSwitch
- Uses the concept of **bridging**
  - Combines network segments into a single network

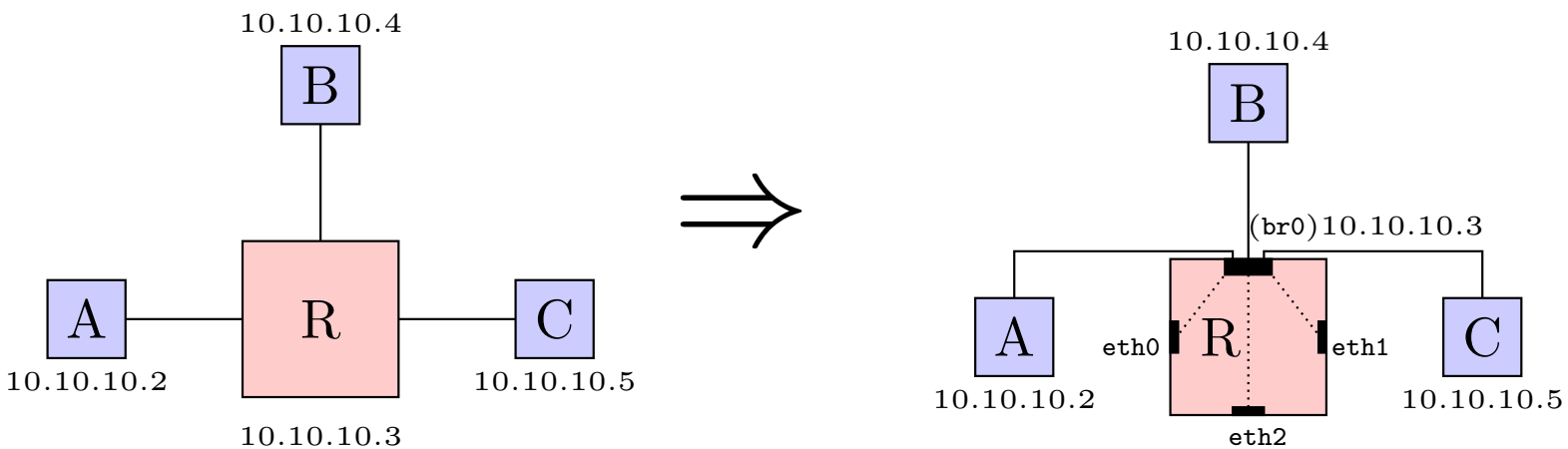
Creating an OpenFlow Switch

- Uses the tool Open vSwitch
- Uses the concept of **bridging**
  - Combines network segments into a single network



Creating an OpenFlow Switch

- Uses the tool Open vSwitch
- Uses the concept of **bridging**
  - Combines network segments into a single network



Installation and set up

- Installation (Debian/Ubuntu Linux)

  $ sudo apt-get install openvswitch-switch

- Creating bridge

  $ sudo ovs-vsctl add-br br0

- Make interfaces down

  $ sudo ifconfig ethx 0.0.0.0 down

- Add ports (interfaces) to the bridge

  $ ovs-vsctl add-port br0 ethx

- Make interfaces up

  $ sudo ifconfig ethx up

- Make bridge up

  $ sudo ifconfig br0 10.10.10.20 netmask 255.255.255.0 up

Open vSwitch commands

1. **ovs-appctl**: Configure and run Open vSwitch daemon
   - $ ovs-appctl help
   - $ ovs-appctl bridge/dump-flows <bridge>
2. **ovs-ofctl**: Flow table management
   - $ ovs-vsctl add-flow <bridge> <flow>
   - $ ovs-vsctl show <bridge>
3. **ovs-vsctl**: Query and configure ovs-vswitchd
   - Bridge commands
   - Controller commands
   - Database commands
   - $ ovs-vsctl add-br <bridge>
   - $ ovs-vsctl set-controller br0 tcp:10.10.10.1:6633

Control plane

- Brain of the network
- Network Operating System + Applications
- Network control functions become application programs
- SDN Controller, the major component
- Examples
  - NOX (C++/Python)
  - POX (Python)
  - **Ryu** (Python)
  - Floodlight (Java)
  - ovs-controller (C)

Ryu SDN controller

- Open source controller (Available in GitHub)
- Licensed under Apache 2.0 License
- Component based architecture
  - L2 Switch
  - Firewall
  - Snort
  - Topology
- Object Oriented + Event-driven approach

Ryu manager

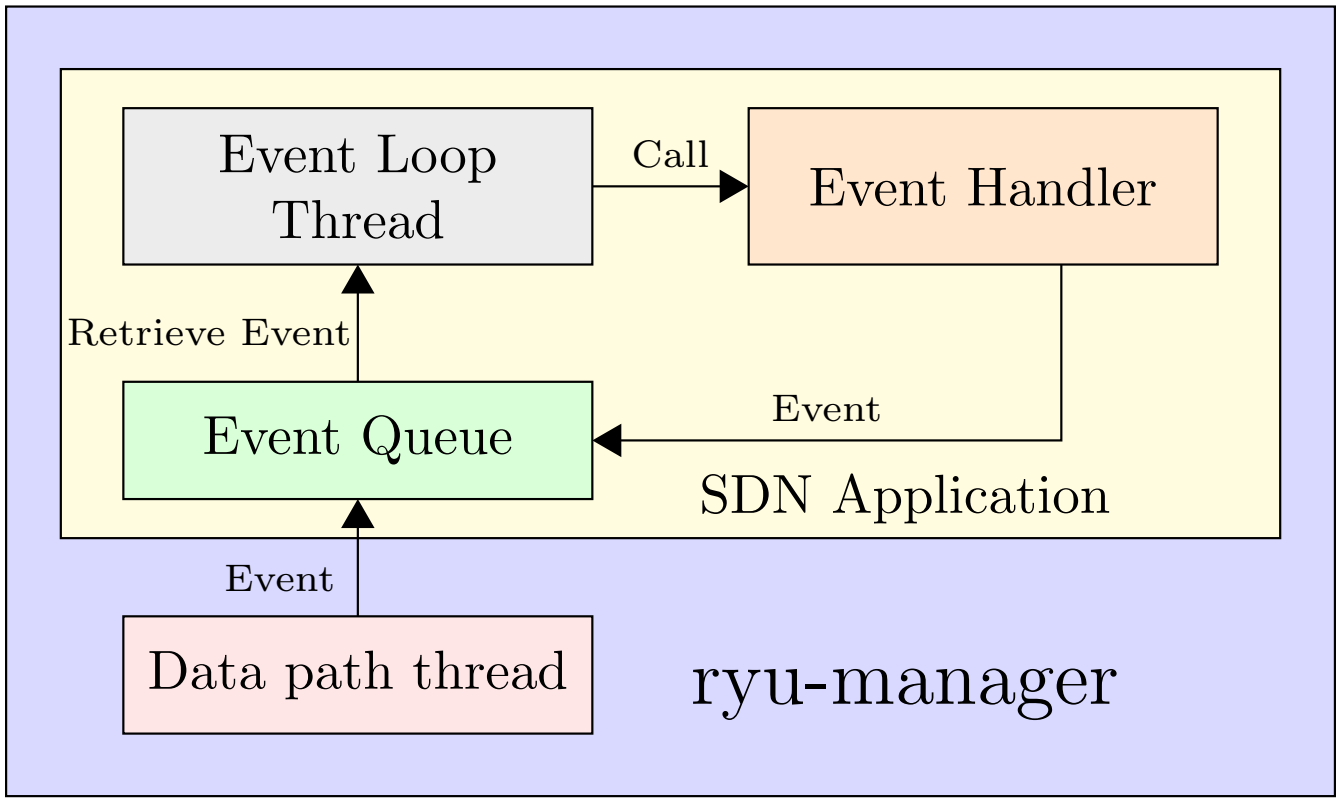


Figure 5: Ryu process

Structure of a Ryu application

```
class AppName(app_manager.RyuApp):
  OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

  def __init__(self, *args, **kwargs):
    super(AppName, self).__init__(*args, **kwargs)

  ## Code for OpenFlow Switch feature extraction
  @set_ev_cls(ofp_event.EventOFPSwitchFeatures,
      CONFIG_DISPATCHER)
  def switch_features_handler(self, ev):
    ## Statements for switch feature extraction

  ## Code to handle packet_in messages
  @set_ev_cls(ofp_event.EventOFPPacketIn,
      MAIN_DISPATCHER)
  def _packet_in_handler(self, ev):
        ## Statements for controller logic
```

Ryu installation

- Install dependencies

  $ sudo apt-get install git python-dev python-setuptools python-pip

- Download Ryu

  $ git clone https://github.com/osrg/ryu.git

- Install Ryu

  $ cd ryu

  $ sudo pip install .

- Run the Ryu controller

  $ PYTHONPATH=. ./bin/ryu-manager ryu/app/AppName.py

Reference materials

Open vSwitch

- Official documentation
- Pica8 OvS commands reference (PDF Version)
- Video lectures by David Mahler

Ryu SDN Framework

- https://osrg.github.io/ryu/
- Documentation

Questions?

sarath.babu.2014@ieee.org

Thank you.