



Open network architectures with service orchestration

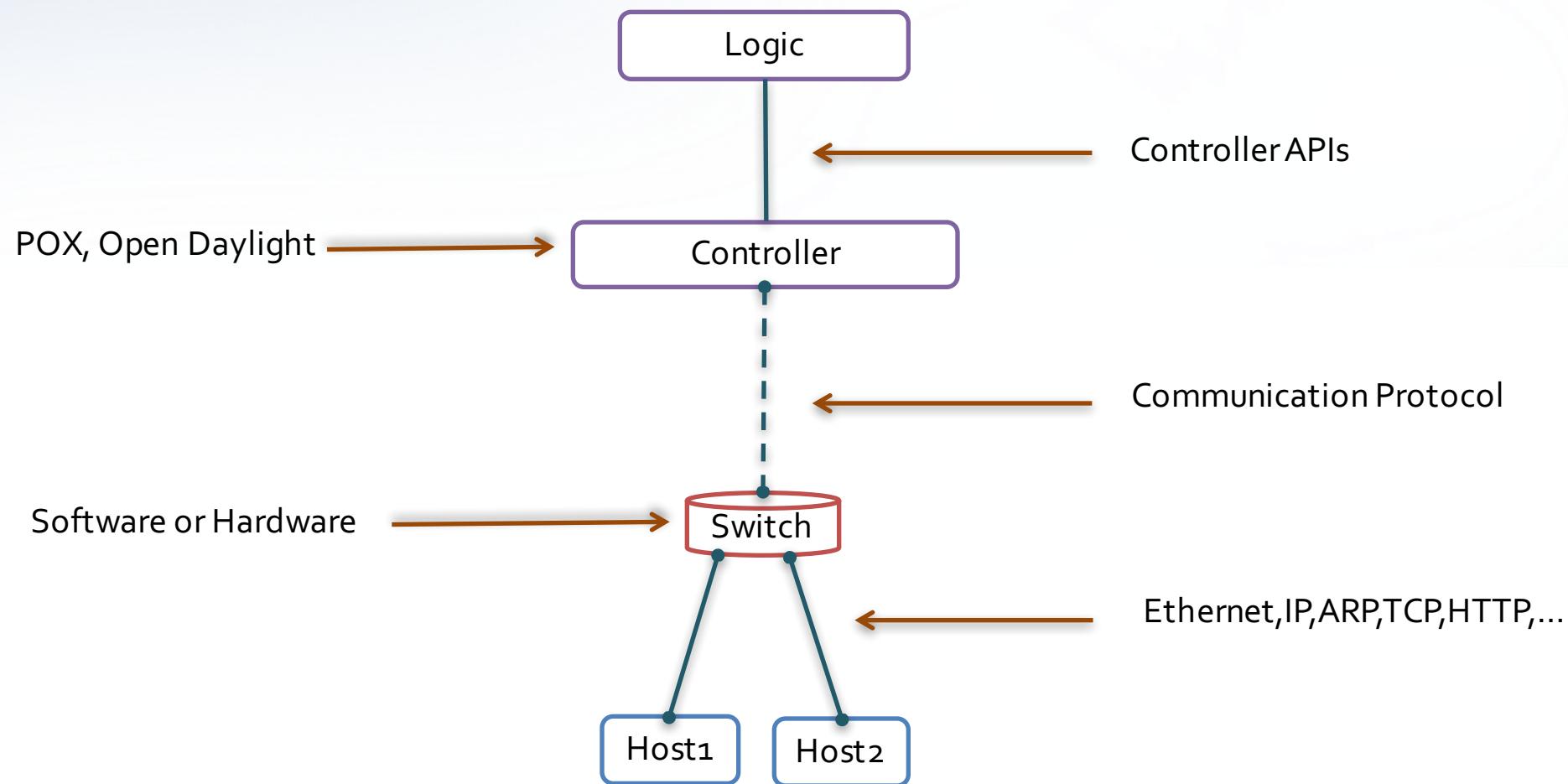
Workshop on SDN networking

Yannis Nikoloudakis Pasiphae Lab - TEIC

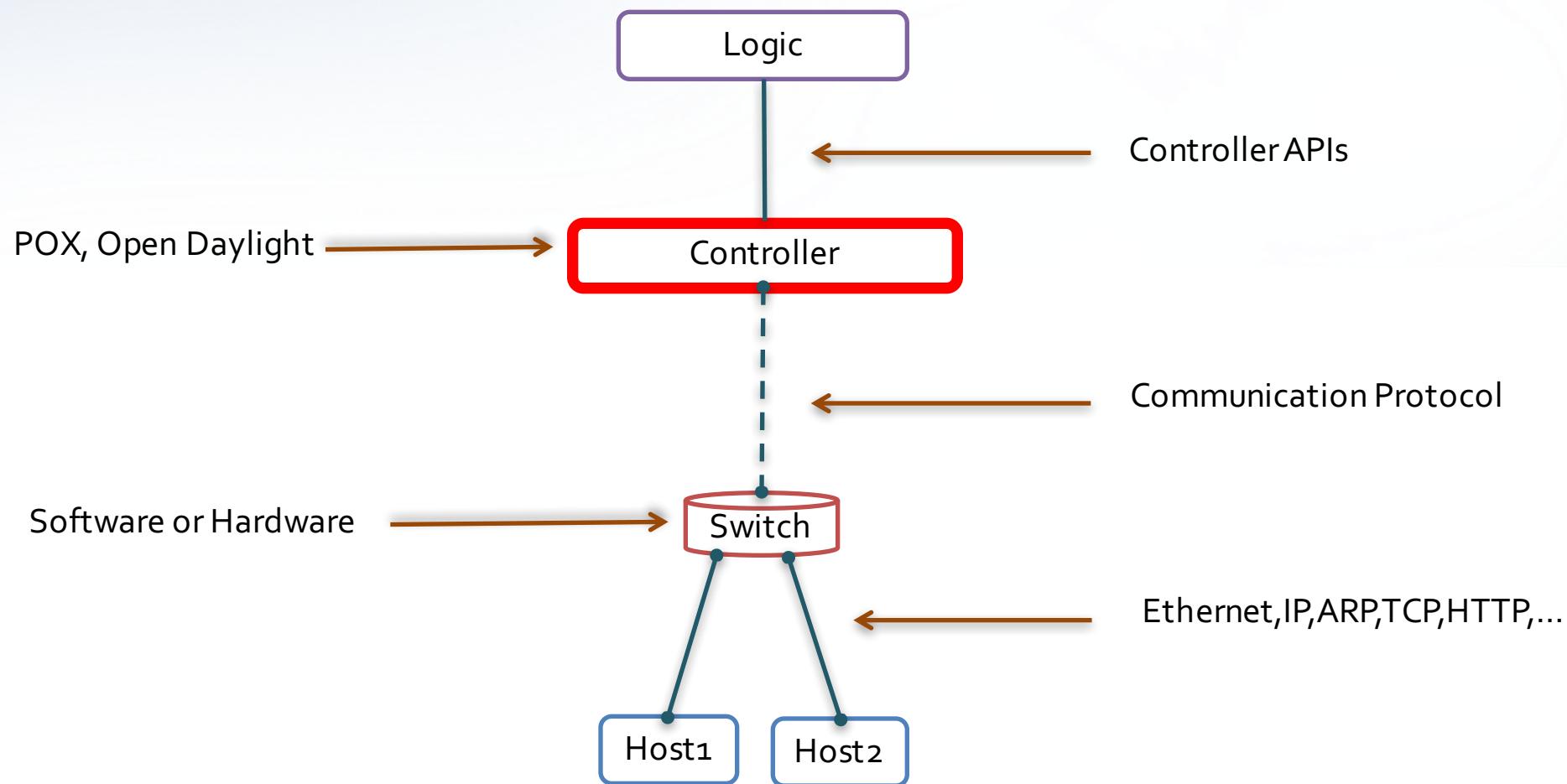
Agenda

- SDN Paradigm
- SDN Controllers
- OpenFlow
- SDN Switch
- Mininet
- Demo / Mininet Walkthrough

SDN Paradigm



SDN Paradigm



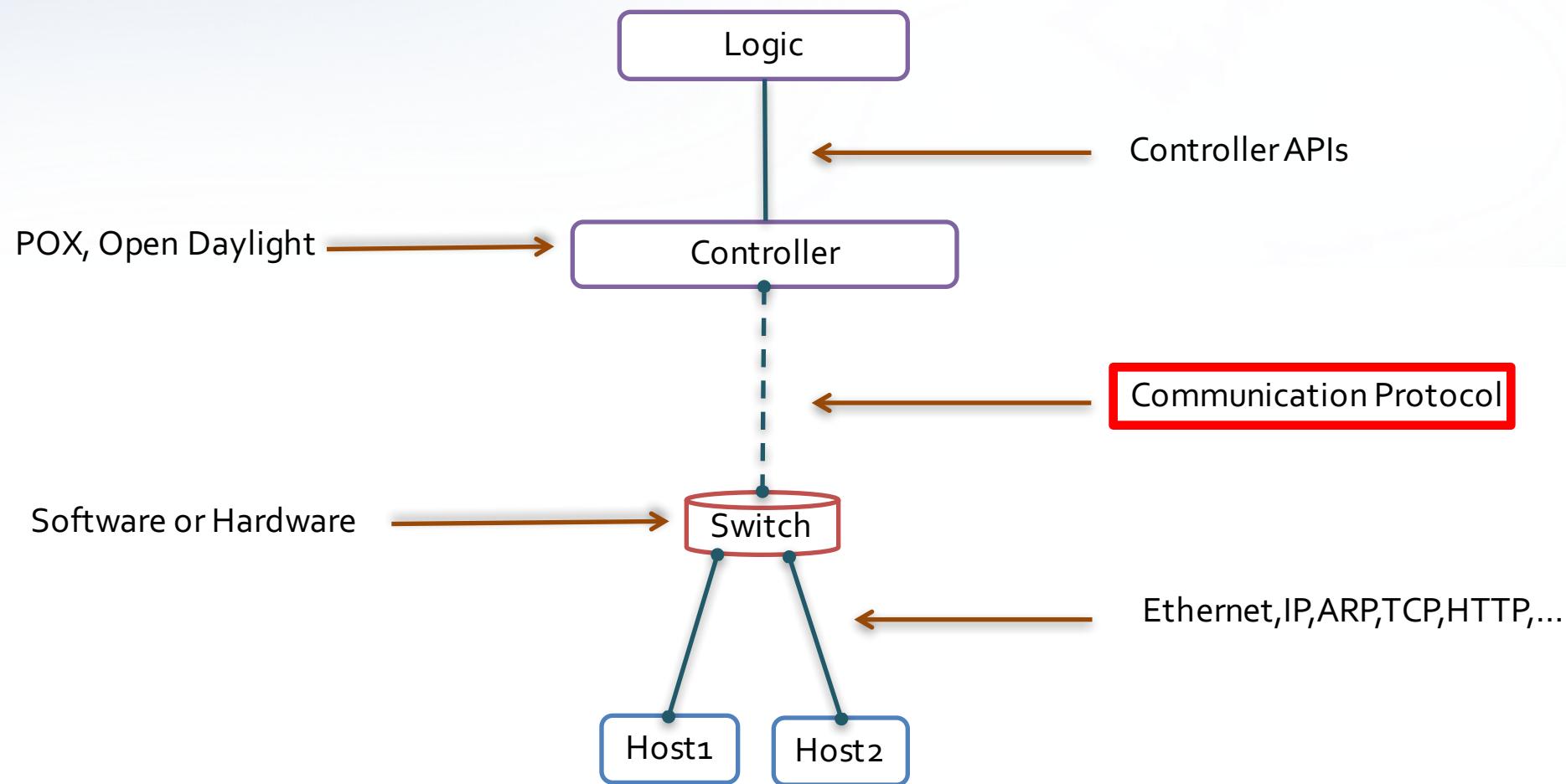
SDN Controllers

- The "*brains*" of the network
- Act as strategic control points inside the SDN network
- Manage flow control to the switches/routers below(via **southbound APIs**)
- Manage the applications' and business logic above (via **northbound APIs**)

SDN Controllers

- Open Daylight (Java)
- Project Floodlight (Java)
- ONOS (Java)
- NOX/POX (Python)
- Ryu Controller (Python)

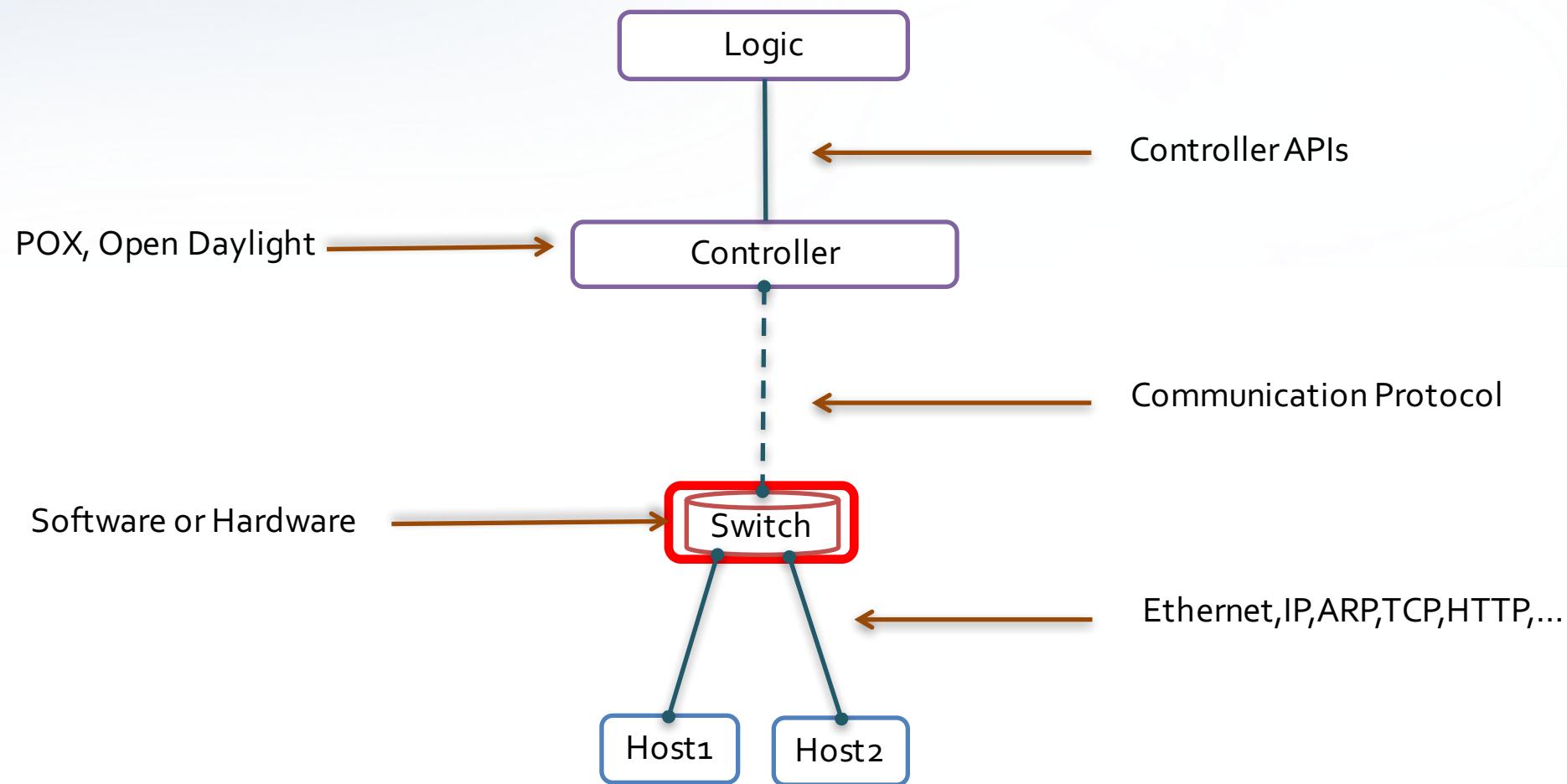
SDN Paradigm



OpenFlow

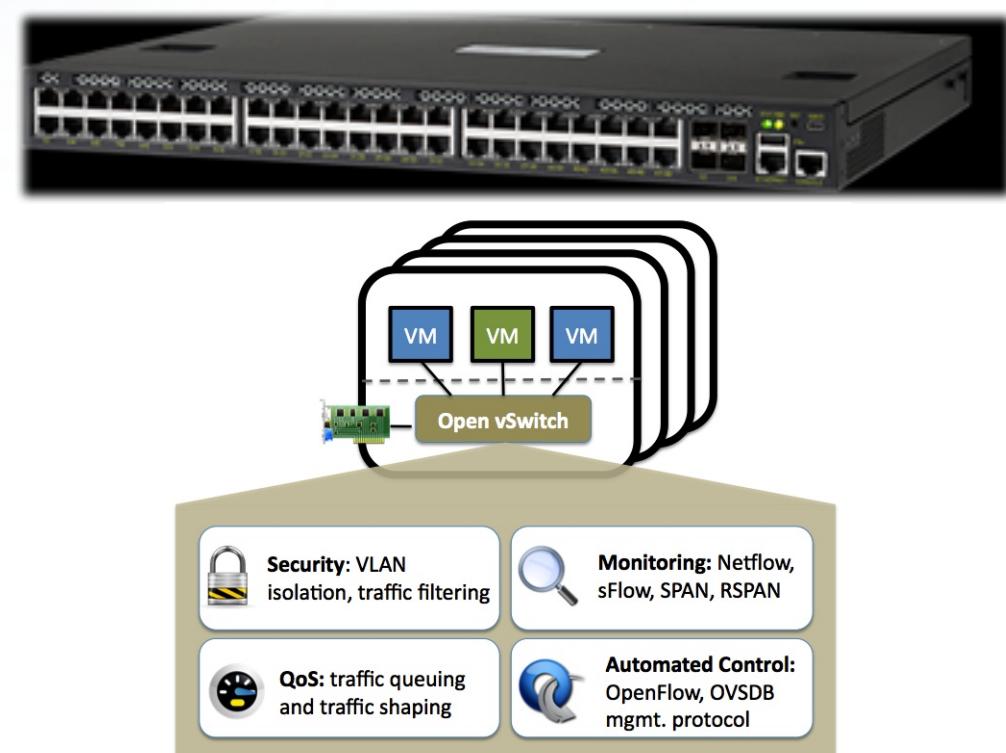
- One of the first SDN standards
- The communication protocol that enables an SDN Controller to directly interact with the forwarding plane of network devices (switches, routers) physical or virtual.

SDN Paradigm

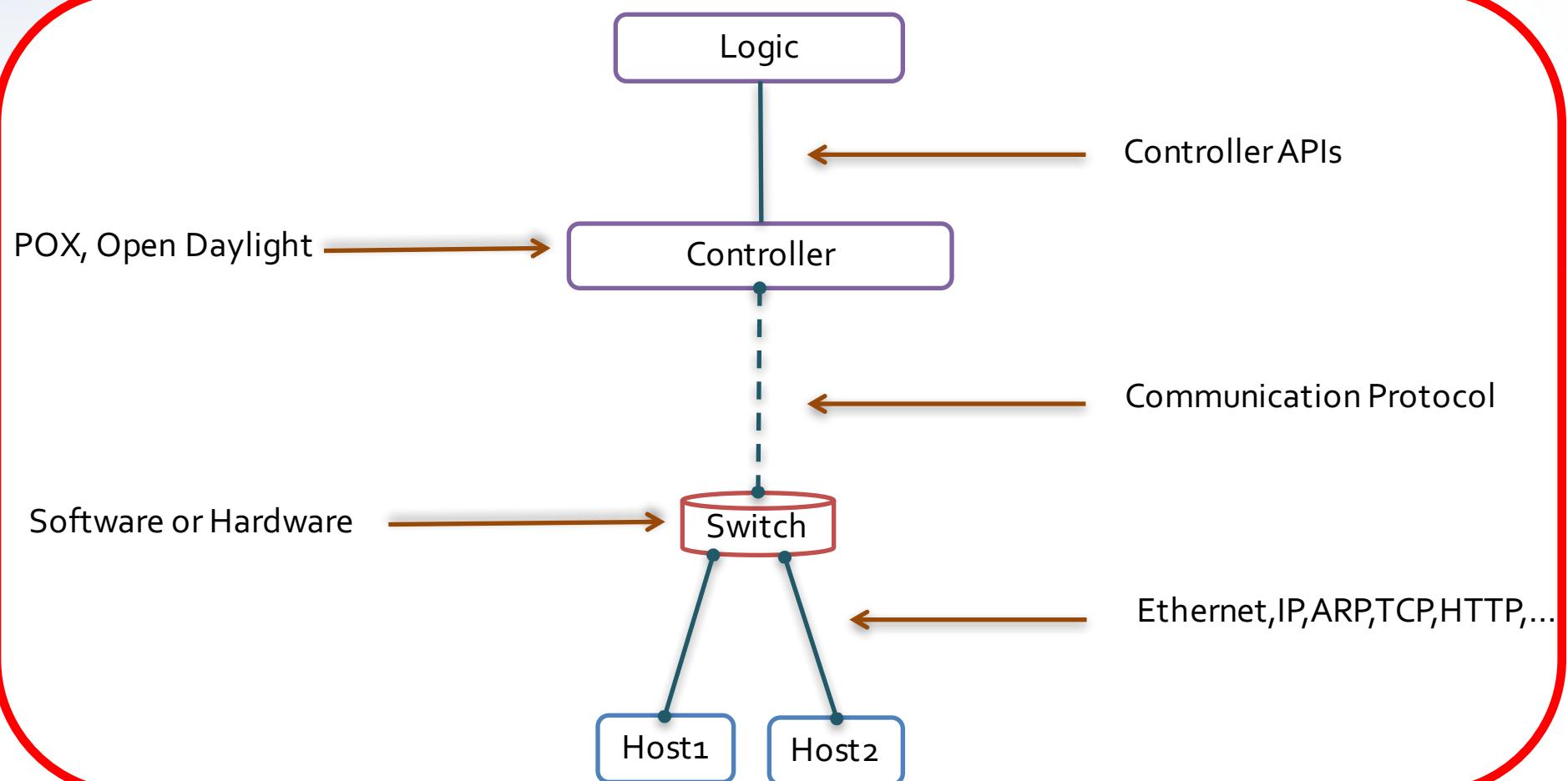


SDN Switch

- Physical
 - Pica8
 - Juniper
 - cisco
- Virtual
 - Open vSwitch



SDN Paradigm



Mininet

- Docker-based Emulation platform
- Rapid and lightweight prototyping of large networks on a single machine
- OS-level virtualization
 - Isolated network namespace
 - Constrained CPU usage on isolated namespace
- CLI and Python APIs

Demo

Prerequisites

- Linux operating System (pref. Ubuntu)
- Oracle VirtualBox Hypervisor
- Internet Connection
- Intermediate Python knowledge
- Familiarity with Linux environment
- SSH client

Demo

- Download Mininet Virtual Appliance
 - URL:
 - <https://drive.google.com/file/d/oB81Pl7s8lqu8SUUhjSWZRMIVWYnc/view?usp=sharing>
 - <https://drive.google.com/open?id=oB81Pl7s8lqu8SUUhjSWZRMIVWYnc>

Demo

Setup Virtual Machine

1. Start VirtualBox
2. Select File>Import Appliance and select the .ova file
3. Press the “Import” button
4. Check network interfaces*

VM User name – **mininet**

VM Password - **mininet**

Demo

Walkthrough

- `$ sudo mn -h`
 - Displays help about Mininet's startup options
- `$ sudo mn`
 - Starts Mininet, creating a default network topology (1 controller, 1 switch, 2 hosts)
- `mininet> help`
 - Displays available Mininet's CLI commands

Demo

Walkthrough

- mininet> nodes
 - Displays all created nodes
- mininet> net
 - Displays Links
- mininet> dump
 - Displays all information about node

Demo

Walkthrough

- mininet> h1 ifconfig –a
 - Display all network interfaces of host #1
- mininet> s1 ifconfig –a
 - Display all network interfaces of switch #1
- mininet> h1 ping –c1 h2
- mininet> pingall

Demo

Run Wireshark on Mininet VM

Ubuntu-host\$ ssh -X mininet@<ip-of-MininetVM>

MininetVM\$ sudo wireshark &

- Start capturing network traffic on **lo** interface
- In Wireshark Filter box, enter the keyword “**of**” and apply to show only OpenFlow network traffic
- Start a new terminal:

Ubuntu-host\$ ssh -X mininet@<ip-of-MininetVM>

Demo

Run Wireshark on Mininet VM

```
MininetVM$ sudo mn
```

```
mininet>h1 ping -c1 h2
```

- Observe the OpenFlow traffic displayed in Wireshark
- Elaborate

Demo

Walkthrough

- mininet> h1 python –m SimpleHTTPServer 80 &
 - Run a simple http server on h1 as a background process
- mininet> h2 wget –O – h1
 - Perform simple HTTP request
- mininet> h1 kill %python
 - Kill python process

Demo

Regression tests

- mininet> sudo mn --test pingpair
 - Runs Mininet, runs an all-pair-ping and tears down the whole topology
- mininet> sudo mn --test iperf
 - Runs Mininet, runs iperf server on one host, runs iperf client on the other host and parses the bandwidth achieved

Demo

Regression tests

- mininet> sudo mn --test iperf --link tc,bw=10
- mininet> sudo mn --test iperf --link
tc,delay=100ms,loss=20

Demo

Custom Topologies

- \$ sudo nano ~/mininet/custom/topo-2sw-2host.py
 - Review custom topology provided for testing purposes by Mininet

Run Mininet with Custom Topology

- \$ sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo --test pingall

Demo

Custom Topologies

```
from mininet.topo import Topo
class MyTopo( Topo ):
    "Simple topology example."
    def __init__( self ):
        "Create custom topo."
        # Initialize topology
        Topo.__init__( self )
        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )
        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )
    topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Demo

Using a Remote Controller

MininetVM\$ sudo mn --controller=remote,ip=<controller-ip>,port=<controller-port>

ControllerVM\$ cd ~/pox

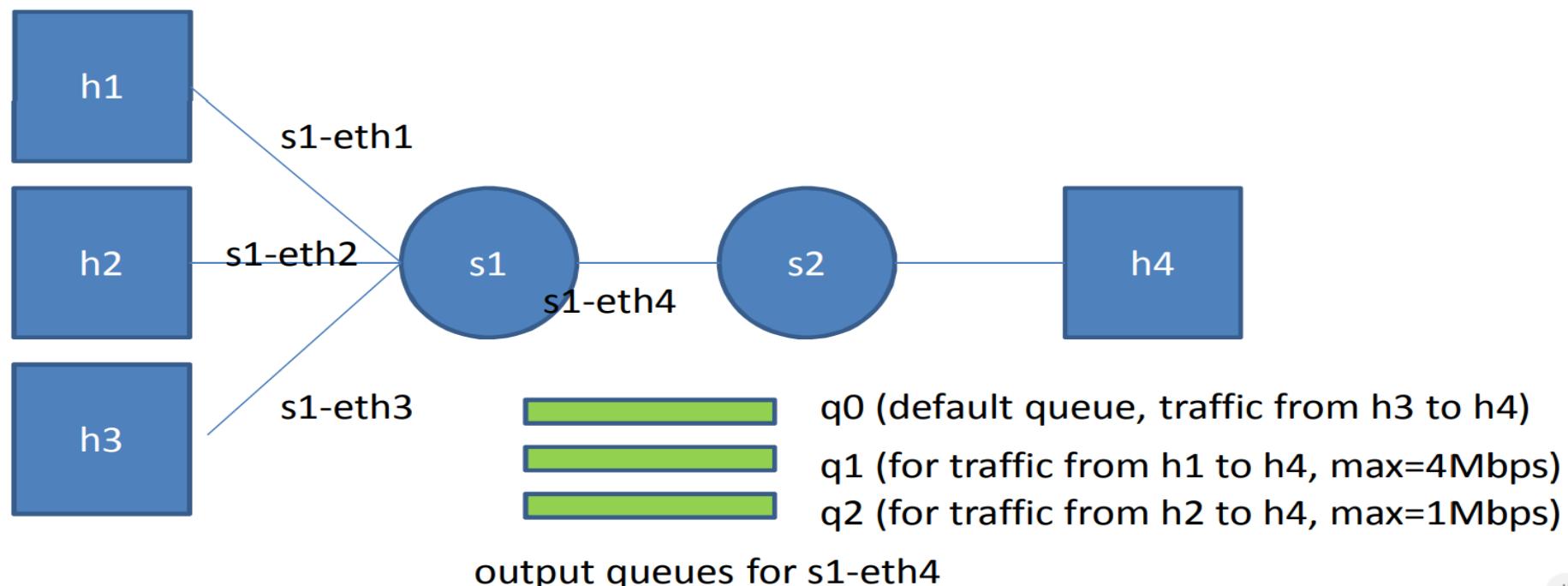
ControllerVM\$./pox.py forwarding.l2_learning

Demo

Defining QoS Rules

- Scenario
 - Create a custom topology
 - 1 Controller
 - 2 Switches
 - 4 Hosts

Demo



Demo

- \$ sudo nano ~/mininet/custom/4h-2w.py
- elaborate

Demo

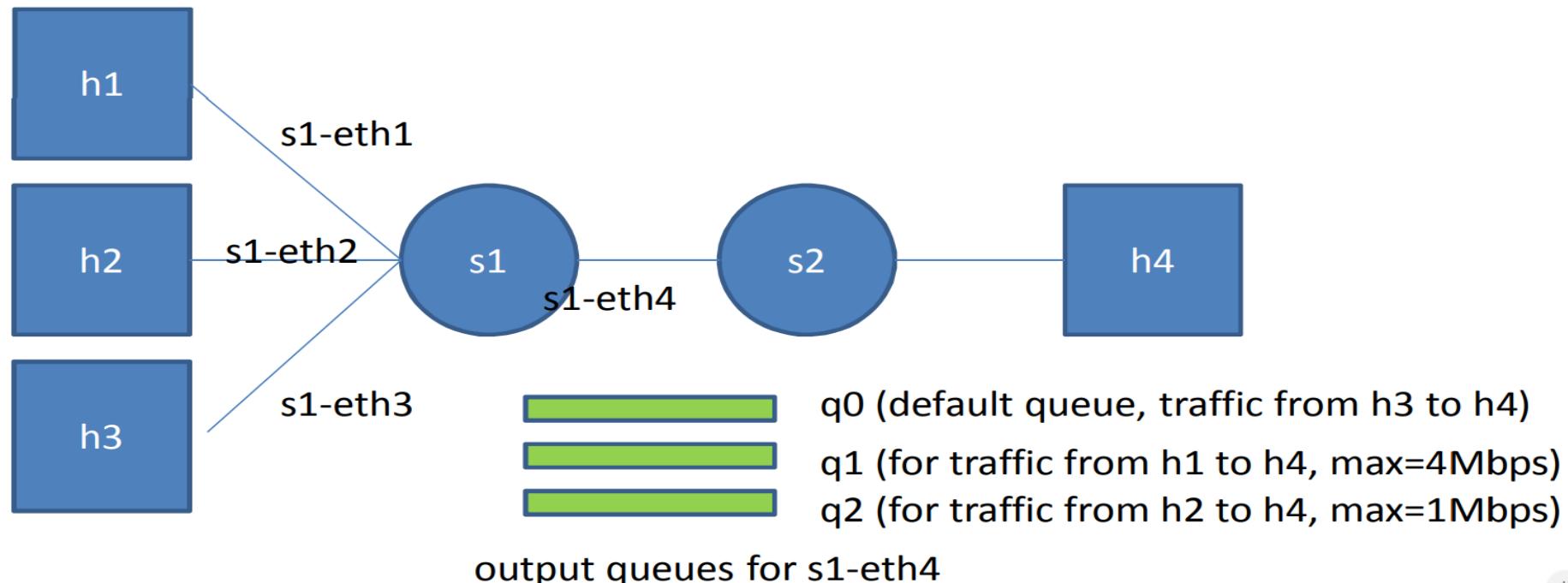
```
from mininet.topo import Topo
class MyTopo(Topo):
    def __init__(self):
        # Initialize topology
        Topo.__init__(self)

        # Add hosts and switches
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        h3 = self.addHost('h3')
        h4 = self.addHost('h4')
        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')

        # Add links
        self.addLink(h1, s1)
        self.addLink(h2, s1)
        self.addLink(h3, s1)
        self.addLink(s1, s2)
        self.addLink(s2, h4)

topos = {'mytopo': (lambda: MyTopo())}
```

Demo



Demo

- \$ sudo nano ~/pox/ext/workshop.py
- Elaborate
- Ubuntu-Host\$ ssh mininet@<mininet-vm-ip>
- Start mininet emulation without any QoS rules
- Mininet-VM\$ cd ~/pox
- Mininet-VM\$./pox.py workshop
- Start a new terminal
- Ubuntu-Host\$ ssh mininet@<mininet-vm-ip>
- Mininet-VM\$ sudo mn --custom ~/mininet/custom/4h-2w.py -- topo=mytopo --controller=remote

Demo

- mininet> xterm h₁ h₂ h₃ h₄
- H₄\$ iperf -s -p 4000 & iperf -s -p 5000 & iperf -s -p 6000
- H₁\$ iperf -c 10.0.0.4 -p 4000
- H₂\$ iperf -c 10.0.0.4 -p 5000
- H₃\$ iperf -c 10.0.0.4 -p 6000
- Elaborate on the results

Demo

- Start a new terminal
- Ubuntu-Host\$ ssh mininet@<mininet-vm-ip>
- Mininet-VM\$ sudo ovs-vsctl -- set Port s1-eth4 qos=@newqos --
id=@newqos create QoS type=linux-htb other-config:max-
rate=1000000000 queues=o=@q0,1=@q1,2=@q2 --id=@q0
create Queue other-config:min-rate=1000000000 other-
config:max-rate=1000000000 --id=@q1 create Queue other-
config:min-rate=4000000 other-config:max-rate=4000000 --
id=@q2 create Queue other-config:min-rate=1000000 other-
config:max-rate=1000000

Demo

- Perform the previous steps
- Elaborate on the results

References

- [1] "POX Wiki - Open Networking Lab - Confluence." [Online]. Available: <https://openflow.stanford.edu/display/ONL/POX+Wiki>. [Accessed: 24-Jul-2017].
- [2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," 2008.
- [3] "Mininet Walkthrough - Mininet." [Online]. Available: <http://mininet.org/walkthrough/#part-1-everyday-mininet-usage>. [Accessed: 24-Jul-2017].