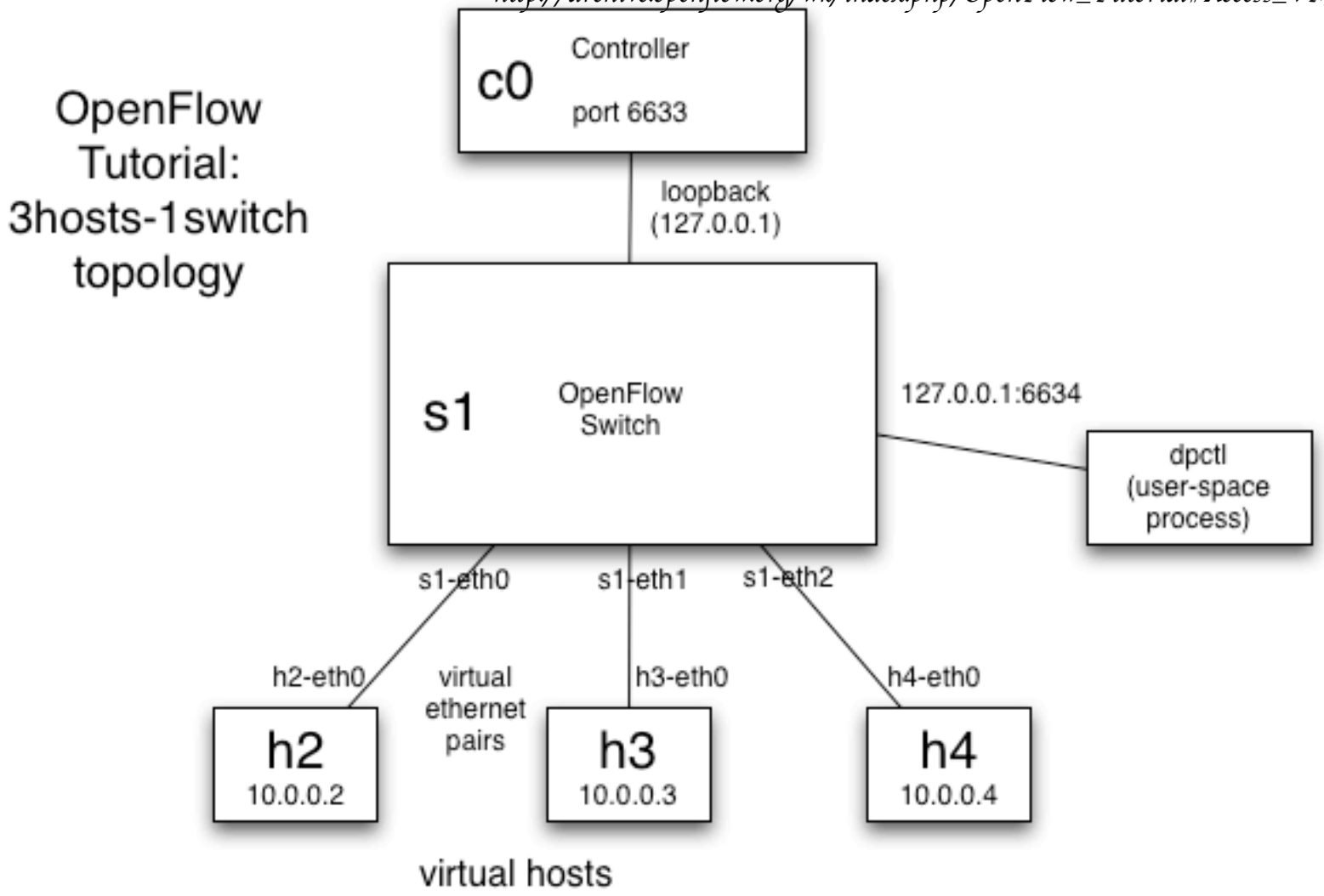
MININET BASED STUDIES

Mininet Demo

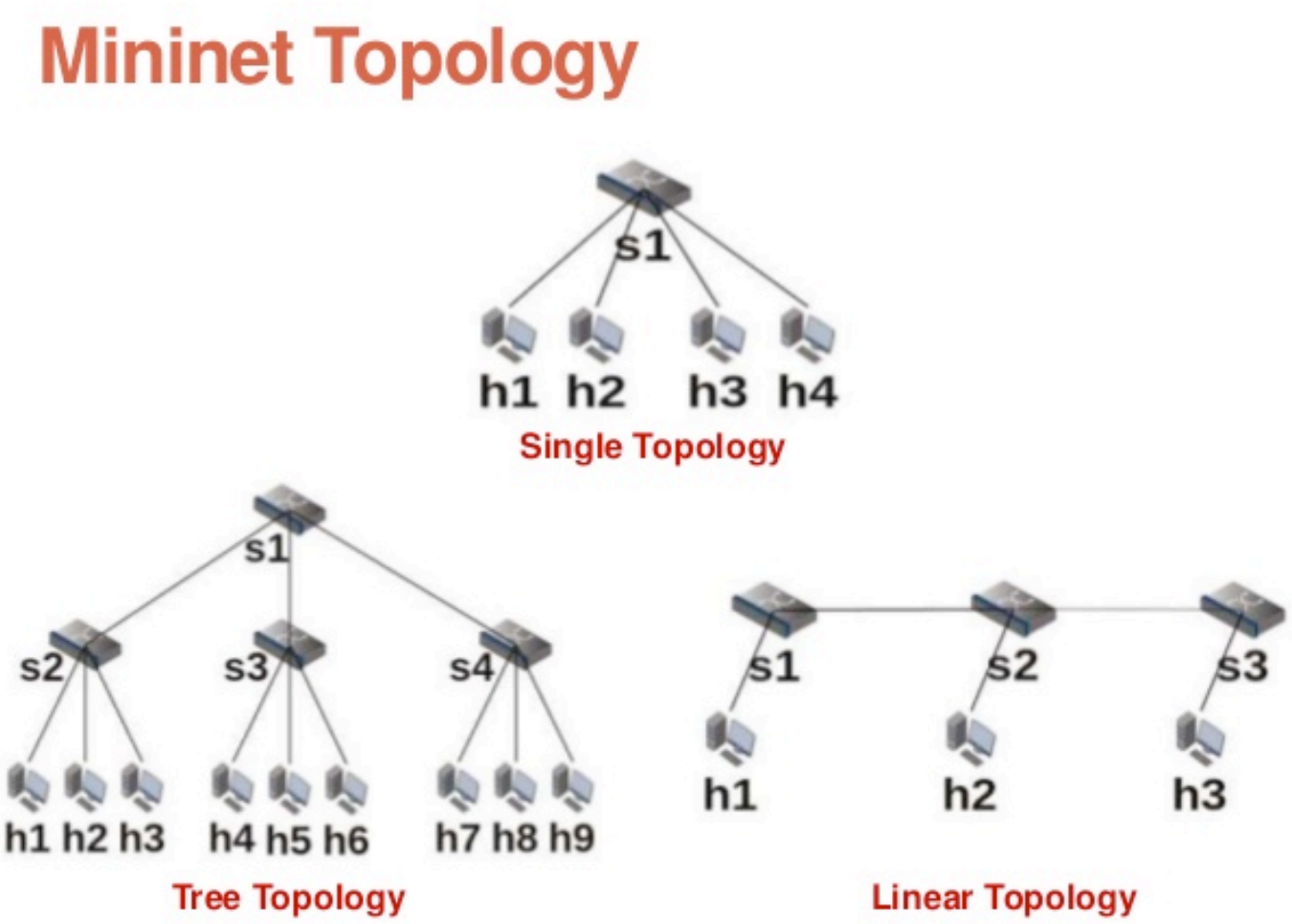
- "Mininet creates a **realistic virtual network**, running **real kernel, switch and application code**, on a single machine (VM, cloud or native" – mininet.org
  - Interact with network using CLI or API
  - Useful for experiments with OpenFlow/SDN
  - BSD Open Source license
- Mininet VM is a good way to get started
  - Can also build on Linux
- Can interact with SDN controllers (Floodlight, POX, etc.)

Mininet Topology

http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial#Access_VM_via_SSH



Different Topologies



POX Controller Programming

- Python interface
  - Write scripts in Python to program controller
- When a connection to a switch starts, a ConnectionUp event is fired
- **connection.send()**
  - To send OF messages from controller to switch
- **ofp_action_output class**
  - out_action = of.ofp_action_output(port = of.OFPP_FLOOD)
  - out_action = of.ofp_action_output(port = 2)

POX Documentation

- https://openflow.stanford.edu/display/ONL/POX+Wiki

Launch

```
def launch (): # Definition Start
    def start_switch (event):
    L2Hub(event.connection)

core.openflow.addListenerByName("ConnectionUp", start_switch) # Definition End
```

Parsing packets

```
packet = event.parsed
src_mac = packet.src
dst_mac = packet.dst
if packet.type == ethernet.IP_TYPE:
    ipv4_packet = event.parsed.find("ipv4")
    # Do more processing of the IPv4 packet
    src_ip = ipv4_packet.srcip
    src_ip = ipv4_packet.dstip
```

Packets parsed by pox/lib

- arp, dhcp, dns
- eapol, eap
- ethernet, icmp
- igmp, ipv4
- llc, lldp, mpls
- rip
- tcp, udp, vlan

Sending message to switch

```
 msg = of.ofp_packet_out()              # Create packet out message
 msg.buffer_id = event.ofp.buffer_id    # Use the incoming packet as the
data for the packet out

 msg.in_port = packet_in.in_port        # Set the in_port so that the switch
knows

# Set match fields
msg.match = of.ofp_match.from_packet(packet)

# Add an action to send to the specified port

action = of.ofp_action_output(port = of.OFPP_FLOOD)
msg.actions.append(action)

# Send message to switch
self.connection.send(msg)
```

Events gen. from switch message

- FlowRemoved, FeaturesReceived
  ConnectionUp, FeaturesReceived
  RawStatsReply, PortStatus
  PacketIn, BarrierIn, SwitchDescReceived,
  FlowStatsReceived, QueueStatsReceived,
  AggregateFlowStatsReceived,
  TableStatsReceived, PortStatsReceived

POX Programming, contd.

- **ofp_match class**
  - Objects specify packet header fields and input port to match on
    - dl_src - The data link layer (MAC) source address
    - dl_dst - The data link layer (MAC) destination address
    - in_port - The packet input switch port
  - All fields are optional
  - Example:
    - match = of.ofp_match()
    - match.in_port = 3

POX Prog., contd.

- **ofp_packet_out OpenFlow message**
  - instructs a switch to send a packet
  - Packet constructed by switch (OR) packet earlier switch by switch to Controller for processing
  - Example

```
msg = of.ofp_packet_out()
newaction = of.ofp_action_output(port = out_port)
msg.actions.append(newaction)

# Send message to switch
self.connection.send(msg)
```

POX Prog., contd.

- **ofp_flow_mod OpenFlow message**
  - instructs a switch to install a flow table entry
  - Entries match packet fields and execute actions
  - Components of entry
    - idle_timeout - Number of idle seconds before the flow entry is removed. Defaults to no idle timeout.
    - hard_timeout - Number of seconds before the flow entry is removed. Defaults to no timeout.
    - actions - A list of actions to perform on matching packets (e.g., ofp_action_output)
    - priority - When using non-exact (wildcarded) matches, this specifies the priority for overlapping matches. Higher values are higher priority. Not important for exact or non-overlapping entries.
    - match - An ofp_match object. By default, this matches everything, so you should probably set some of its fields
  - fm = of.ofp_flow_mod()
  - fm.match.in_port = 3
  - fm.actions.append(of.ofp_action_output(port = 4))

Mininet, Contd.

Switch Types

- User-space Switch (Slow)
- Kernel-space Open Virtual Switch (Fast)
- Hardware Switch supporting OpenFlow
- FPGA Switches

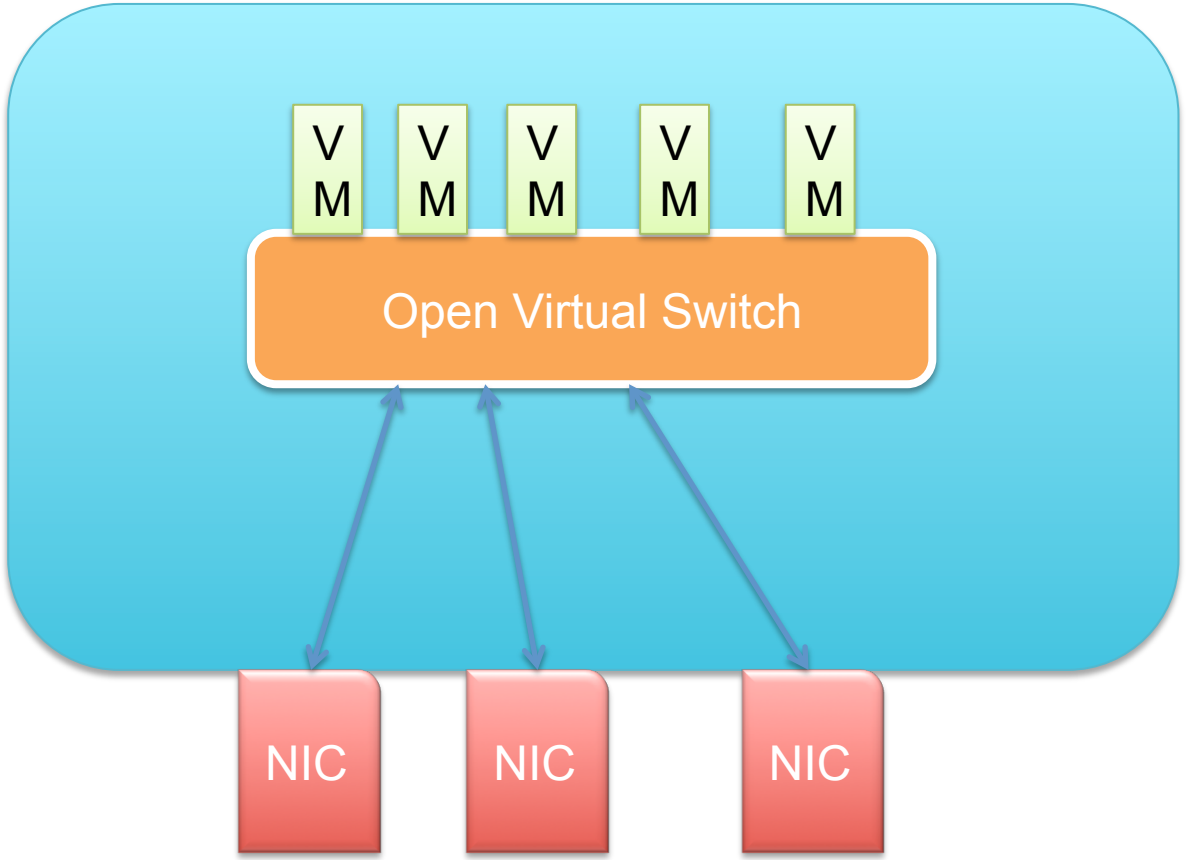
Controller

- Local Default Controller
- Local NOX/POX/Floodlight etc. Controller
- Remote Controller

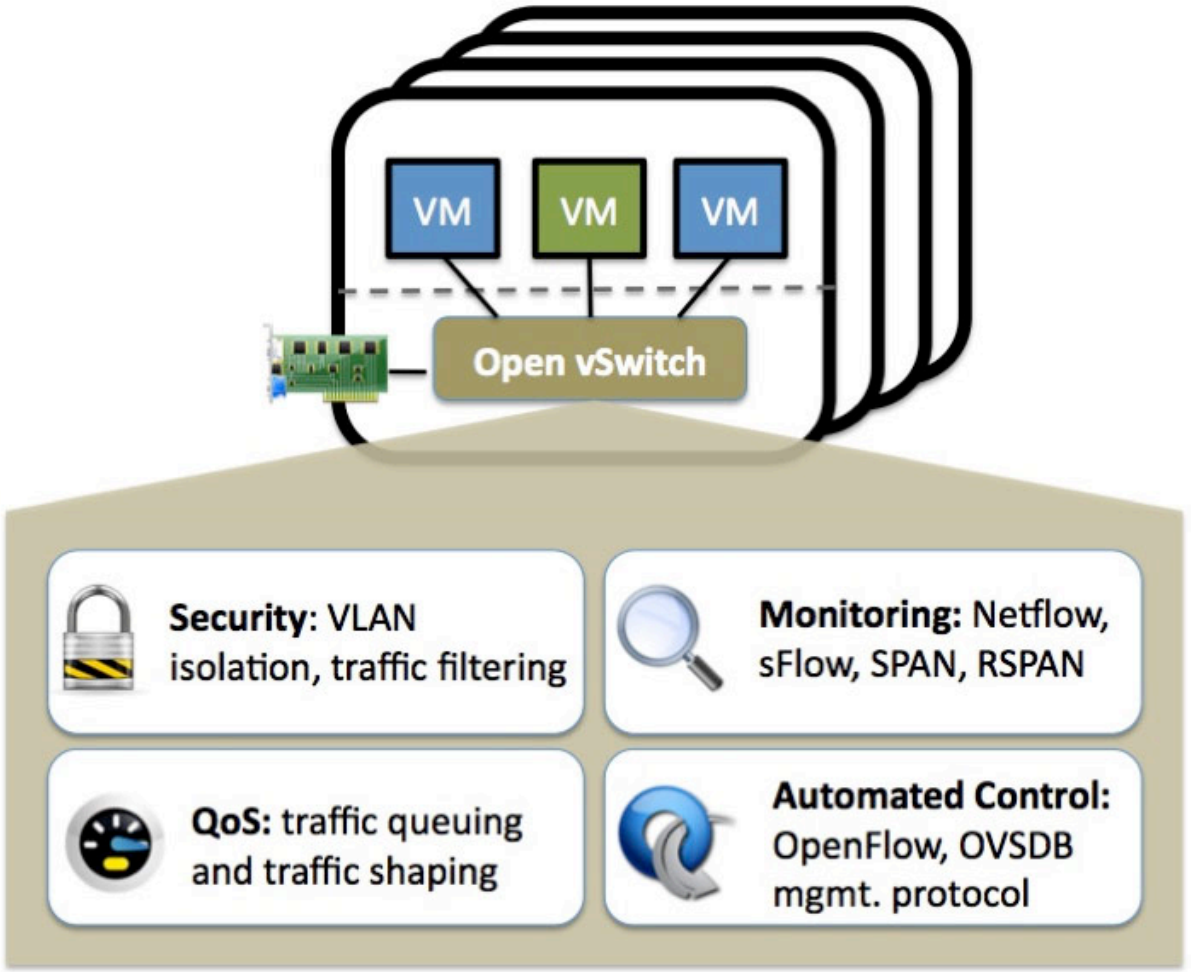
Open Virtual Switch (OVS)

- http://openvswitch.org/
- Apache 2.0 License
- “production quality, multilayer virtual switch”
- “Supports distribution across multiple physical servers similar to VMware's vNetwork distributed vswitch or Cisco's Nexus 1000V”
- Runs a soft switch inside the hypervisor, or as part of software stack of switching h/w
- Default switch in Xen Server, Xen Cloud
- Integrated into OpenStack, OpenNebula, oVirt
- Kernel datapath is distributed with Linux, with packages for Ubuntu, Debian, and Fedora

OVS, conceptual rep.



OVS



Source: https://www.sdxcentral.com/wp-content/uploads/2014/09/open-vswitch-sdn-virtualization.jpg

NETWORK VIRTUALIZATION

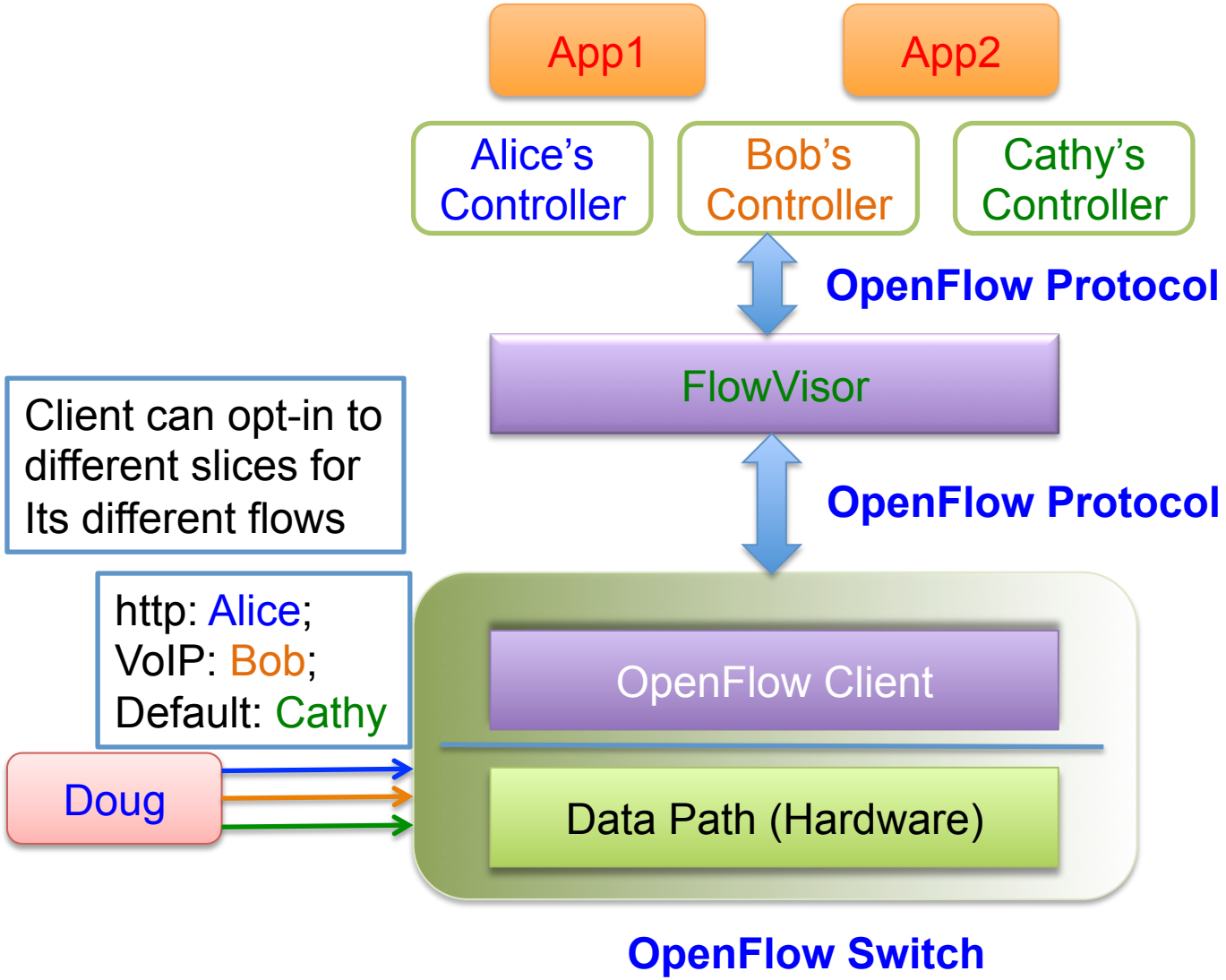
Network Virtualization

- Compute & Storage Virtualization in place
- Virtualization of Networks is up next
  - Going beyond VLANs
- Creation of “Virtual Network Elements” at different levels
  - Switches, Networks, etc.
- Virtualization Enabling Systems Available
  - FlowVisor, RouteFlow, VMware NSX, etc.

Stanford's FlowVisor

- Provides a software-based "slicing layer" between forwarding and control planes
- Policy language to map flows to slices
  - Each slice controls a set of flows, "flowspace"
  - Resources sliced in terms of: BW, Topology, Device CPU, Forward Table Entries, etc.
- Can operate on deployed networks
  - The production network can be a testbed
- Multiple Controllers exist above FlowVisor
  - Each controller uses its logic on its slice

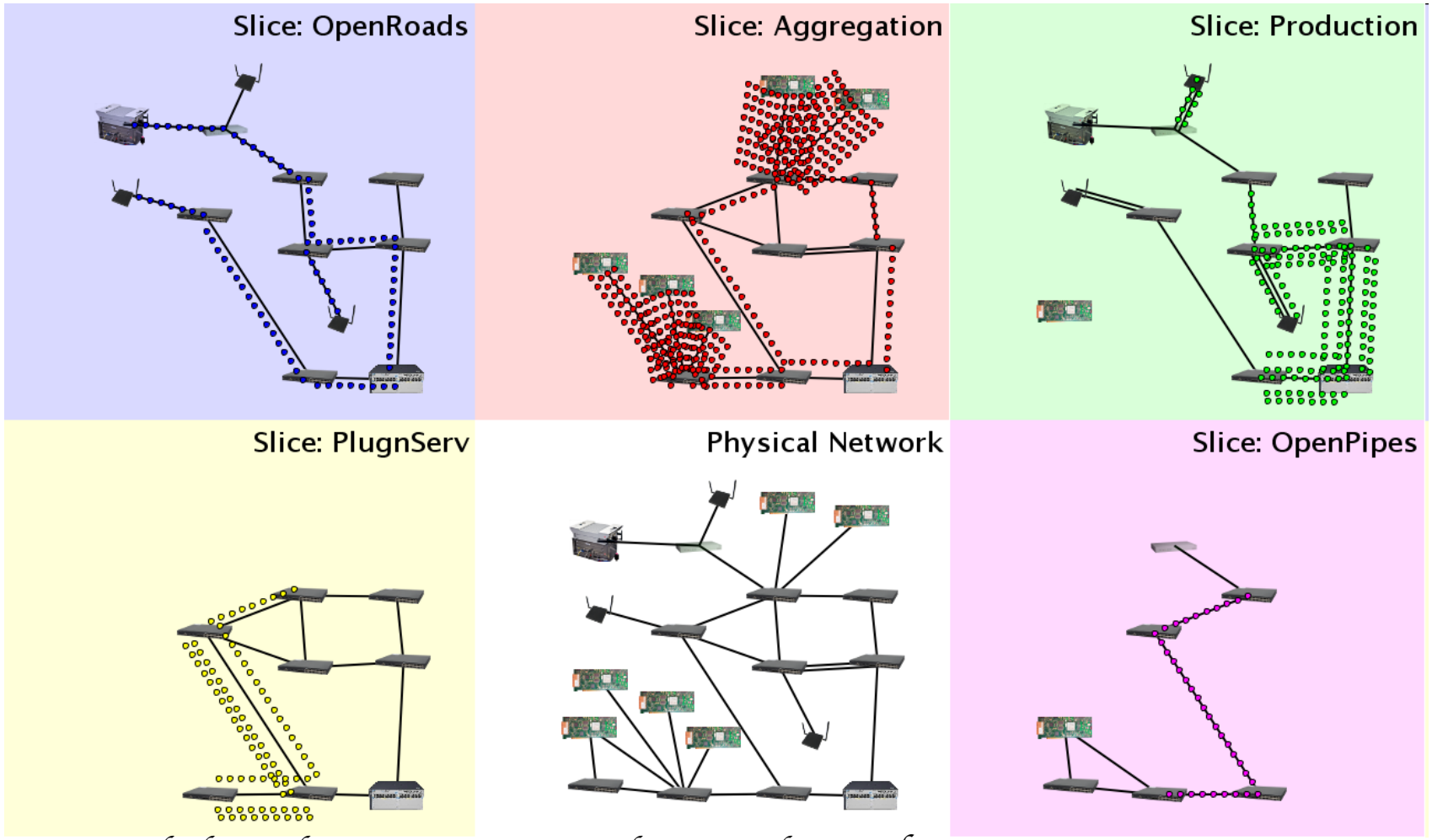
Flow Visor, Contd.



FlowSpace

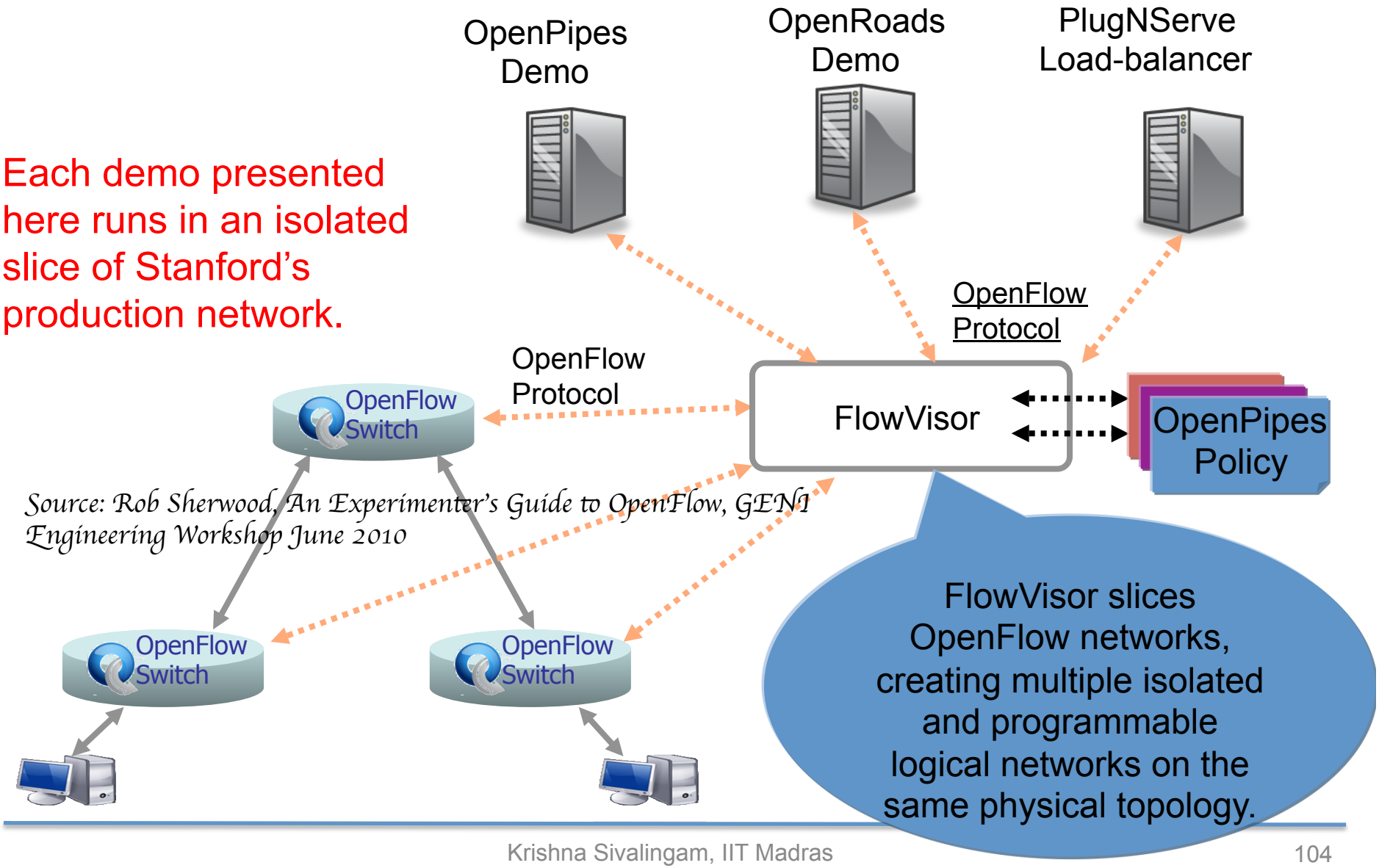
- Defined by collection of packet headers and assigned to “Slices”
  - Source/Destination MAC address
  - VLAN ID
  - Ethertype
  - IP protocol
  - Source/Destination IP address
  - Source/Destination port number
  - ToS/DSCP

Slices



Source: Rob Sherwood, An Experimenter's Guide to OpenFlow, GENI Engineering Workshop June 2010

FlowVisor Creates Virtual Networks



OpenVirteX (OVX): Network Virtualization

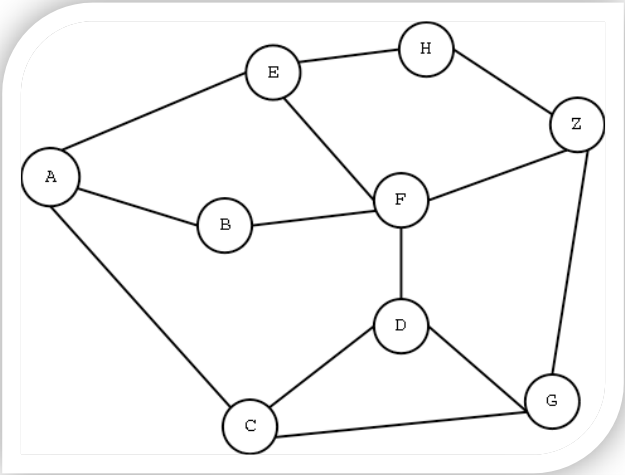
- From ON.Lab at Stanford
- Single Infrastructure provider
  - Multiple tenants, each specifies network and resource connectivity
- OVX supports multiple virtual networks
  - Helps specify tenant's topology and addressing
- OVX resides between physical hardware and virtual network controllers
  - Create isolated virtual networks
  - Use tenant's choice of Network OS
  - Use entire address space!

ON.LAB OVX Architecture

Network Operating System Layer

OpenVirteX

Network Virtualization Layer



Physical Infrastructure

VMWare NSX

- Provides network virtualization for software-defined data center
  - Based on Nicira Acquisition (2012)
- Treats physical network as a pool of transport capacity
- Virtual Networks can be created
  - Logical switches, routers, firewalls, load balancers, etc.
  - Programmatically created, provisioned and managed
- https://www.vmware.com/products/nsx/features.html