Mininet

Peng Zhang

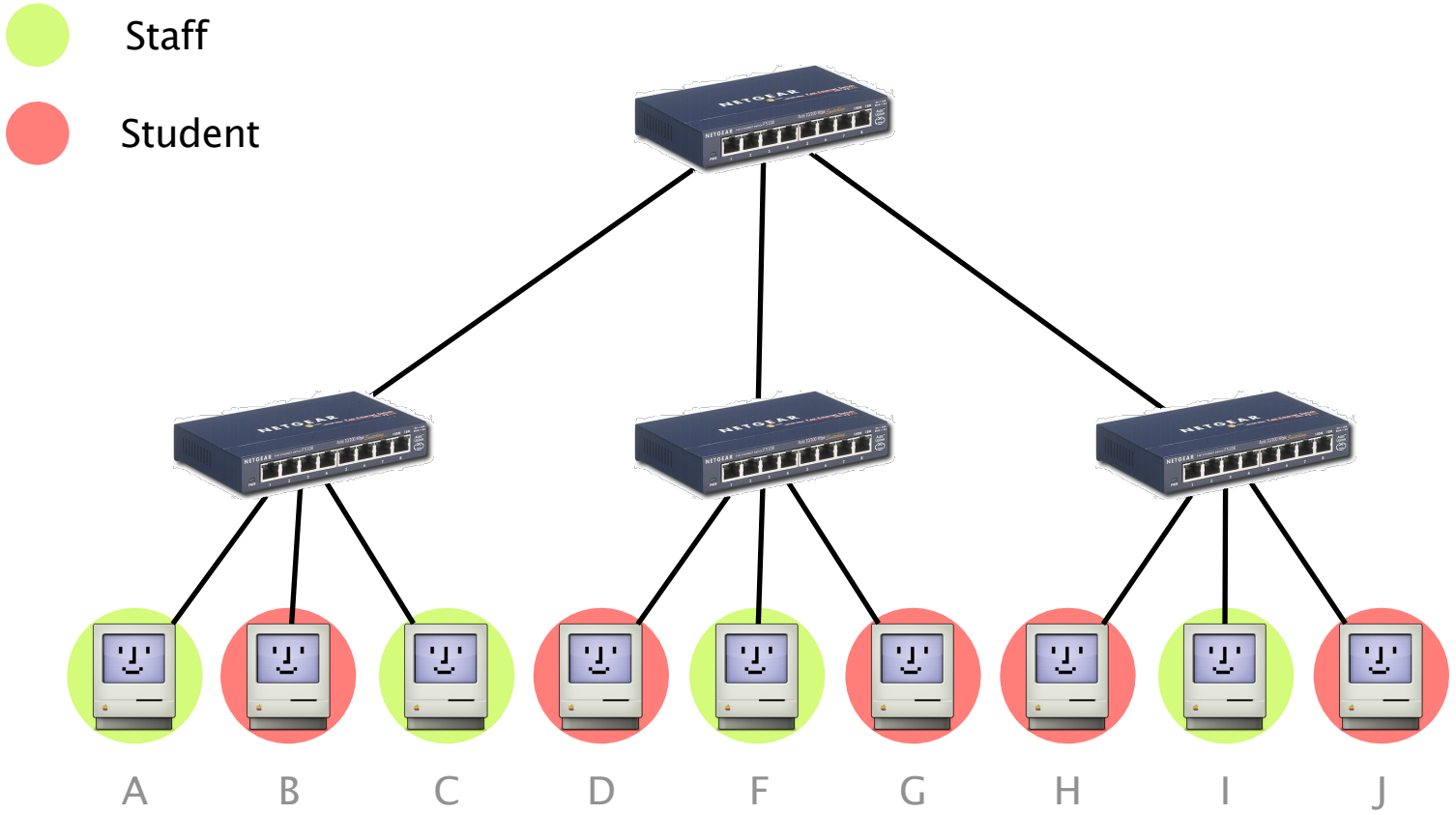
School of Computer Science and Technology

Xi'an Jiaotong University

Spring 2024

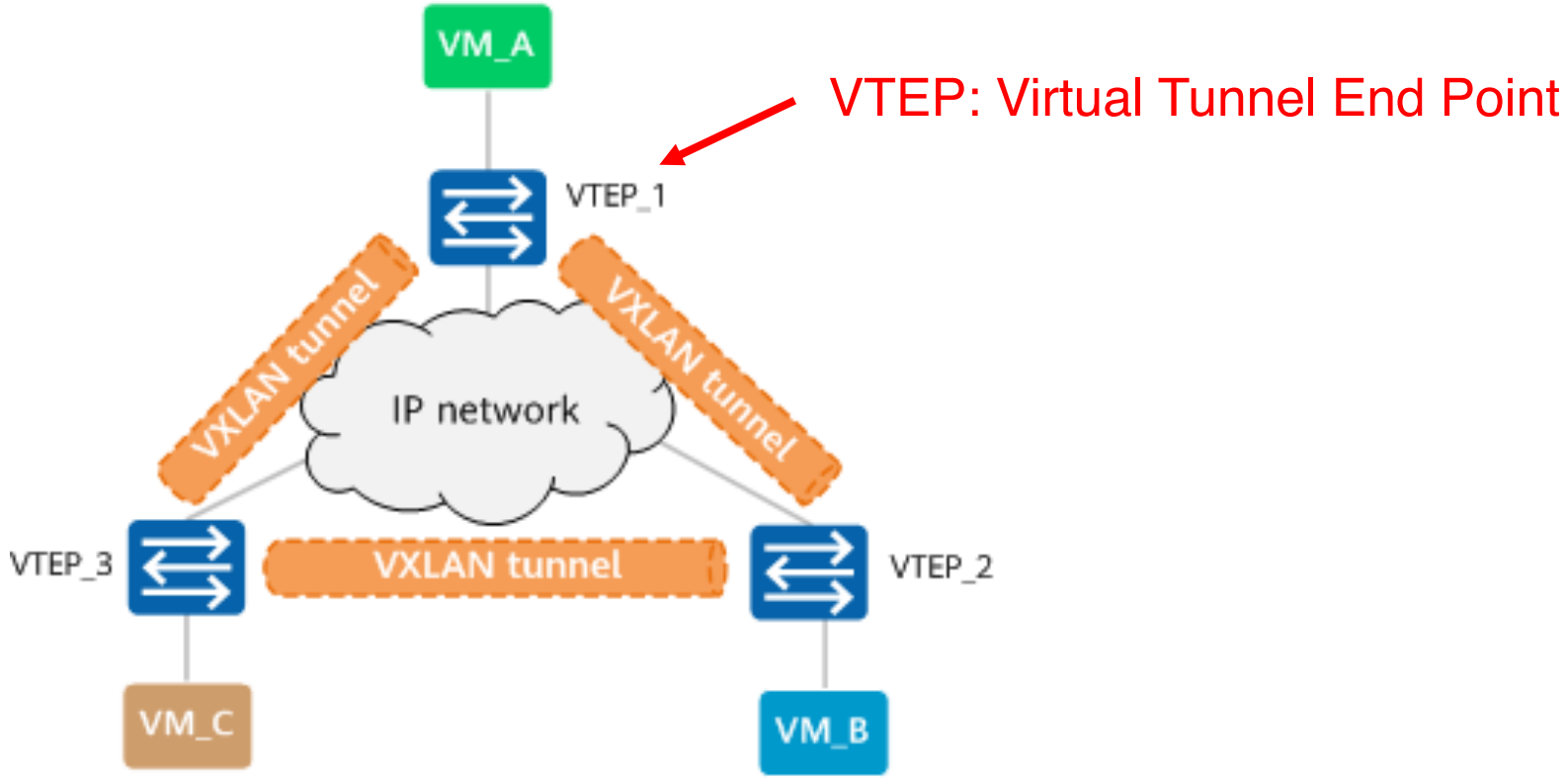
Recap: VLAN

A VLAN logically identifies a set of ports attached to one (or more) Ethernet switches, forming one broadcast domain



Recap: VXLAN

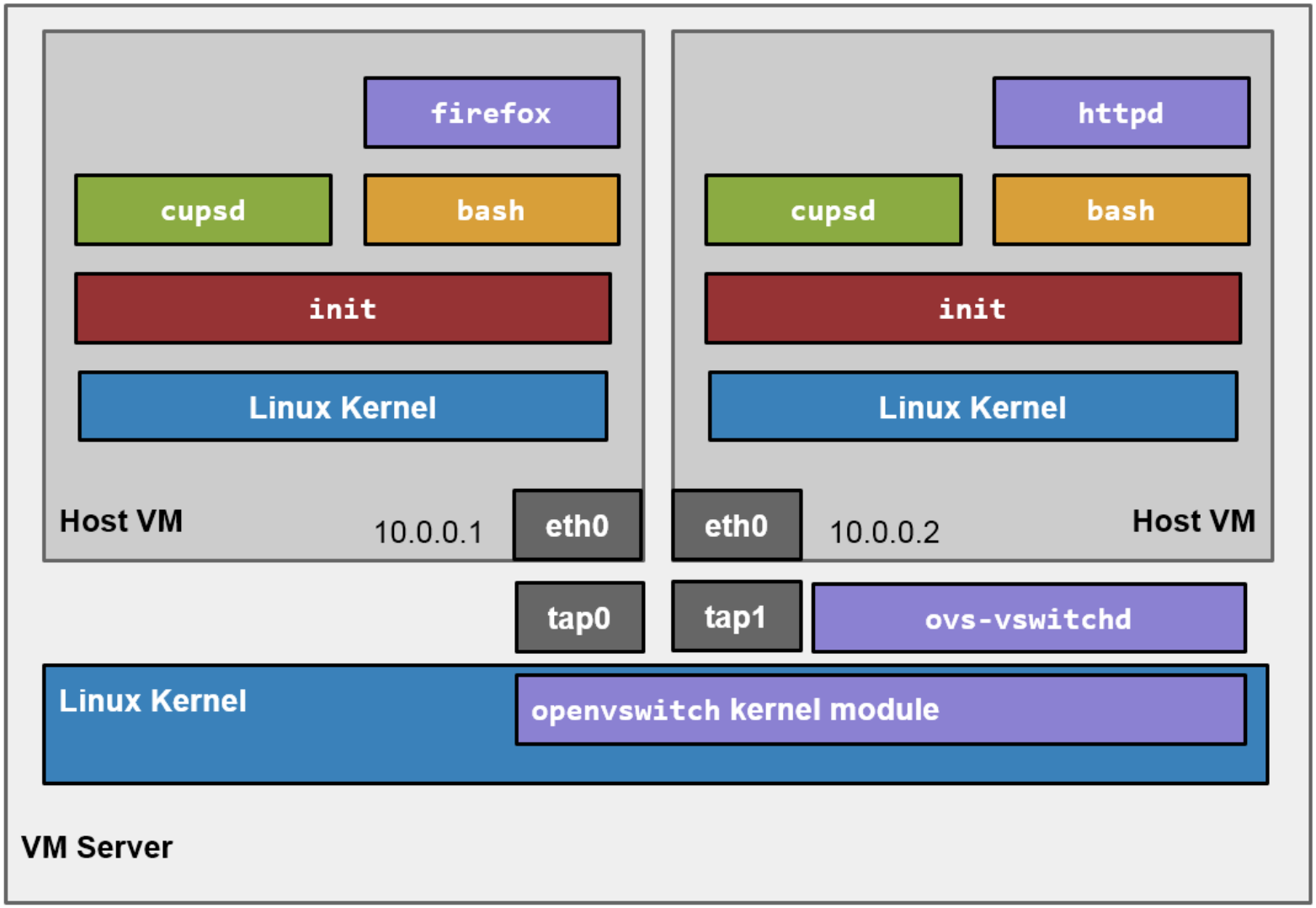
- VXLAN: Virtual Extensible LAN (VXLAN)
  - network virtualization over Layer 3 (NVO3)



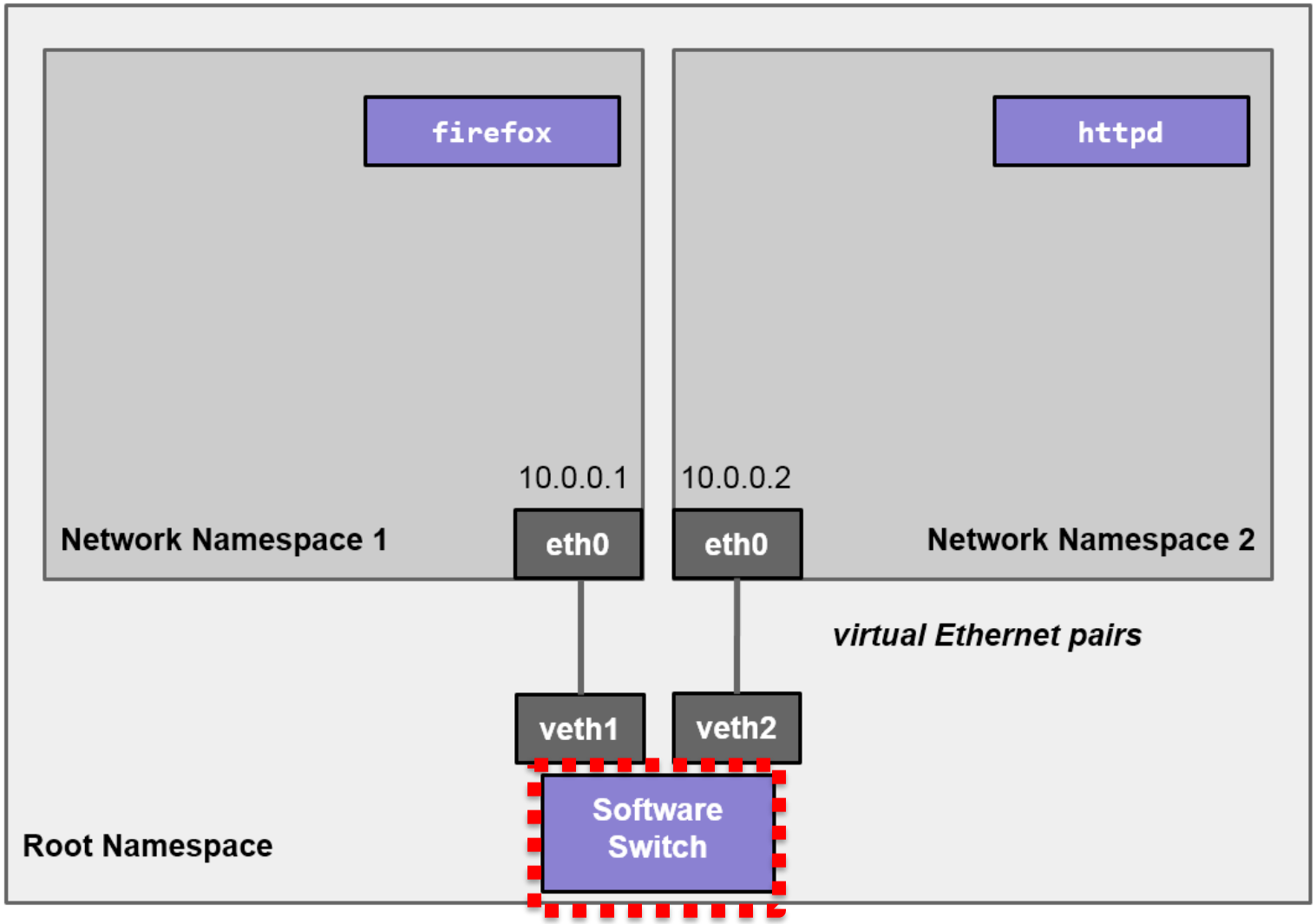
https://datatracker.ietf.org/doc/html/rfc7348

Virtualized Network for Hosts

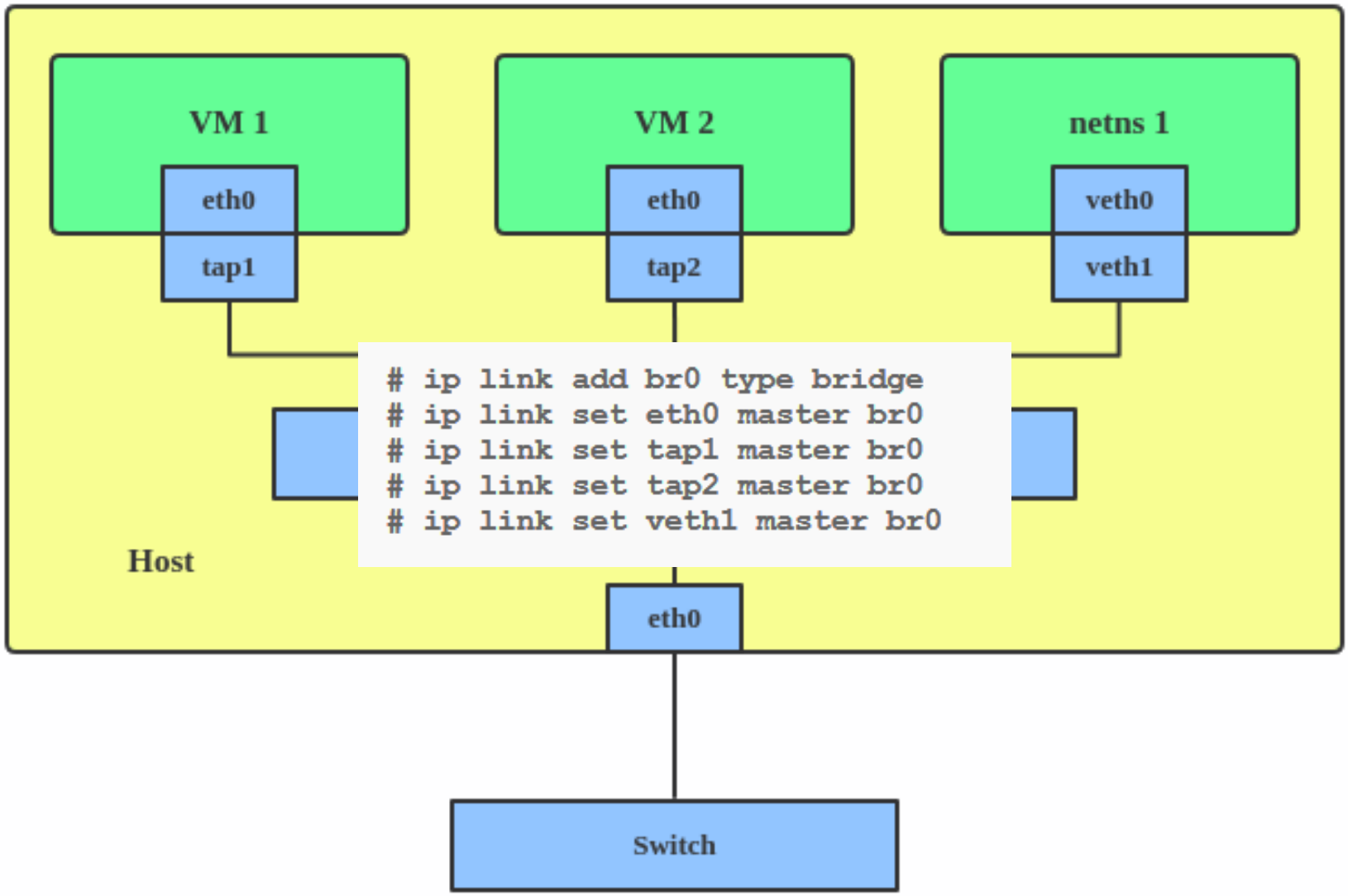
Full Virtualization



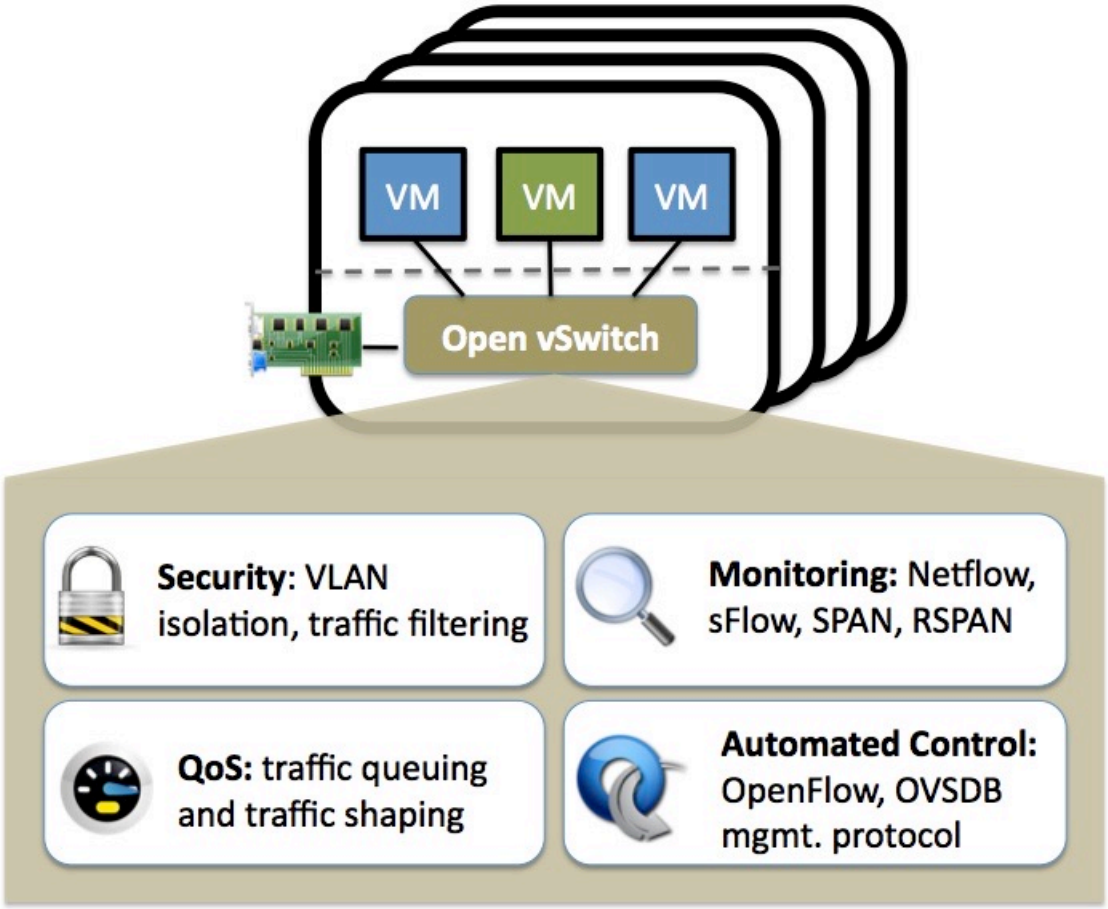
Lightweight Virtualization



Linux Bridge

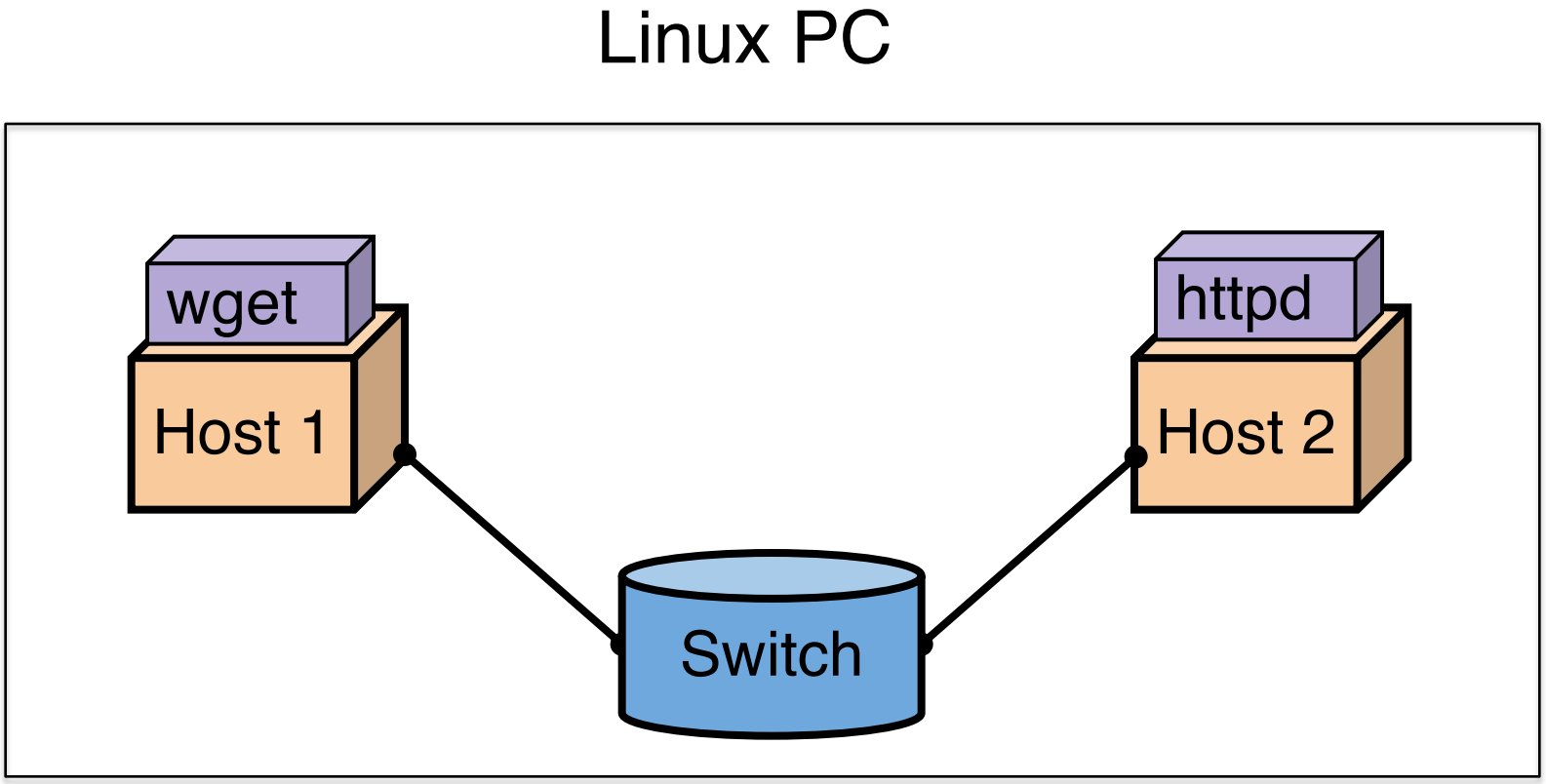


Open vSwitch



A Simple Example

- How to set up the following network on a single Linux PC?

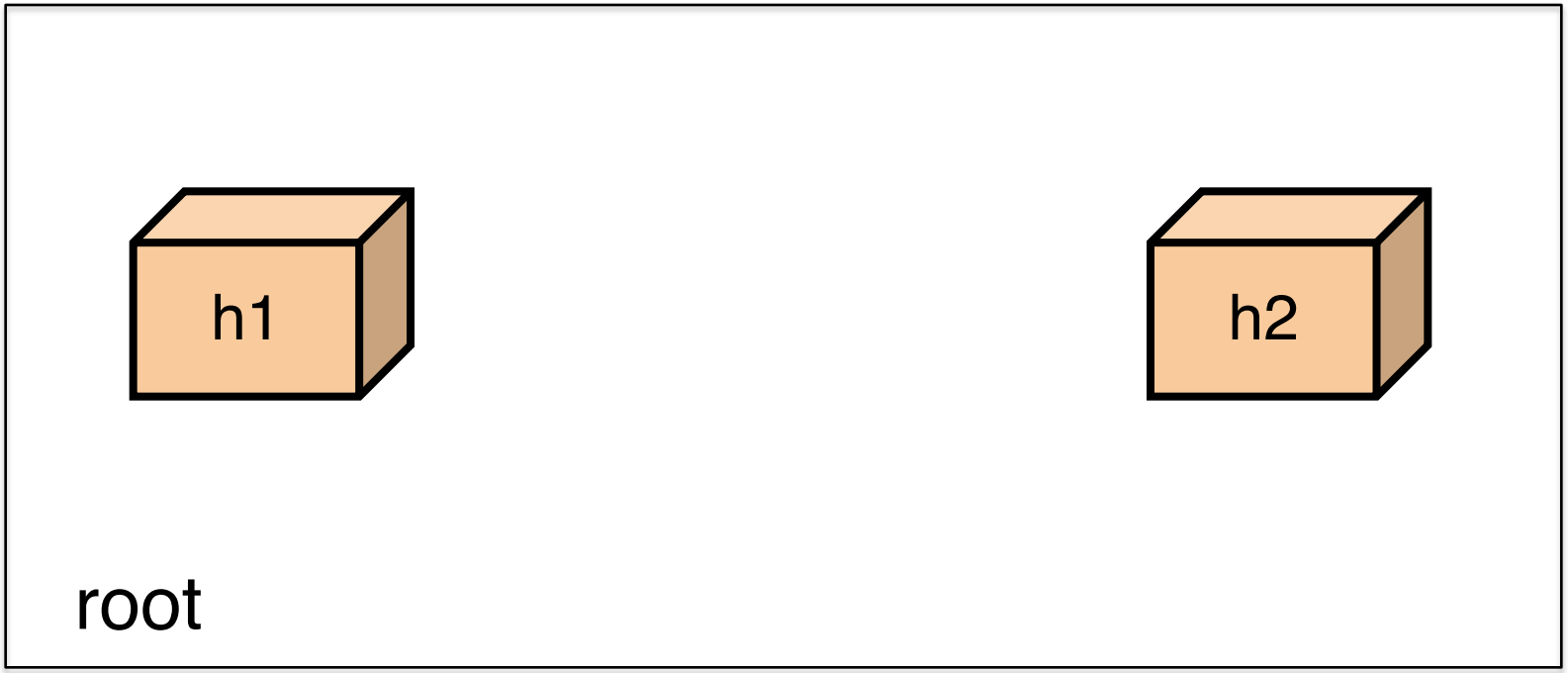


Create Network Namespaces

```
ip netns add h1
ip netns add h2
ip netns show
```

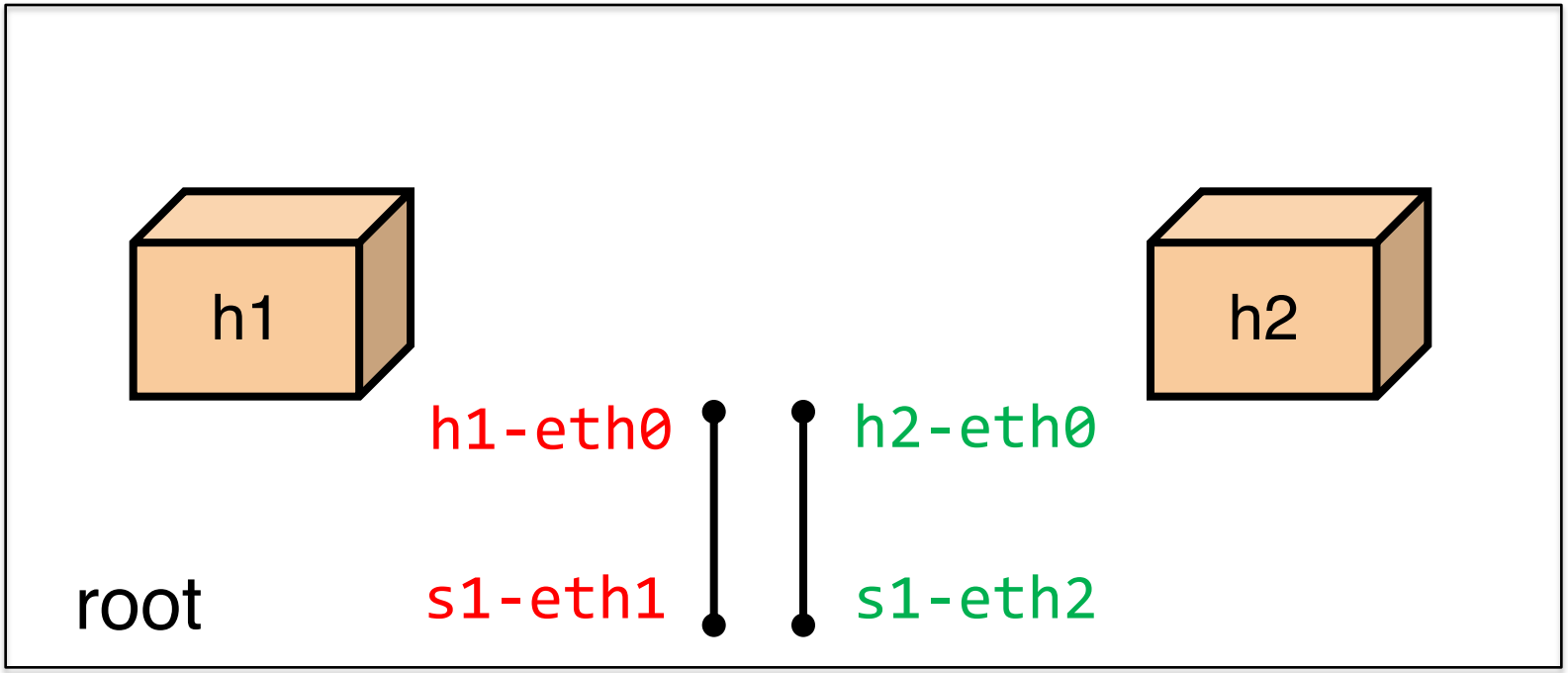
There should be 3 network namespaces:

“h1”, “h2”, and **“root”**



Create Virtual Ethernet Pair

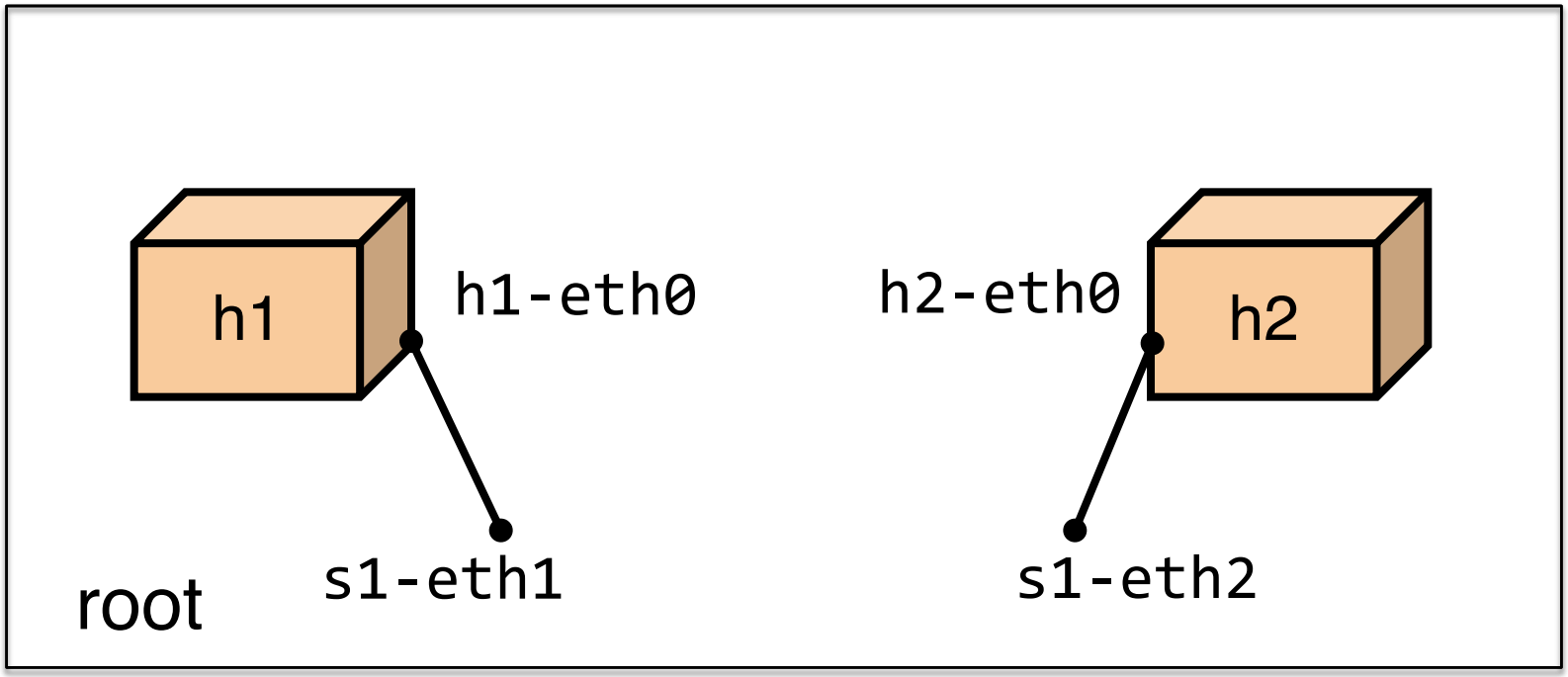
```
ip link add h1-eth0 type veth peer name s1-eth1
ip link add h2-eth0 type veth peer name s1-eth2
ip link show
```



Move Ports into Host Namespaces

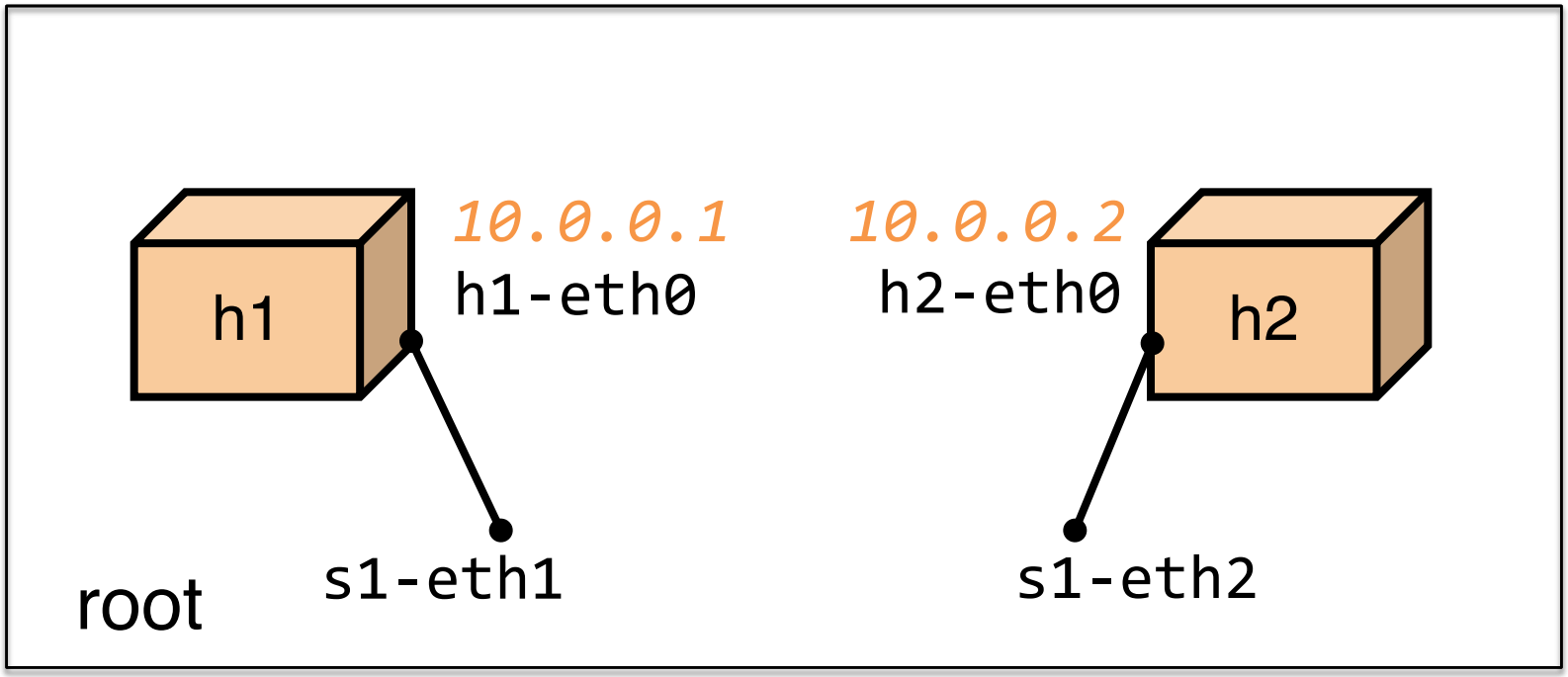
```
ip link set h1-eth0 netns h1
ip link set h2-eth0 netns h2
ip netns exec h1 ip link show
ip netns exec h2 ip link show
```

What if we run **“ip link show” ?**



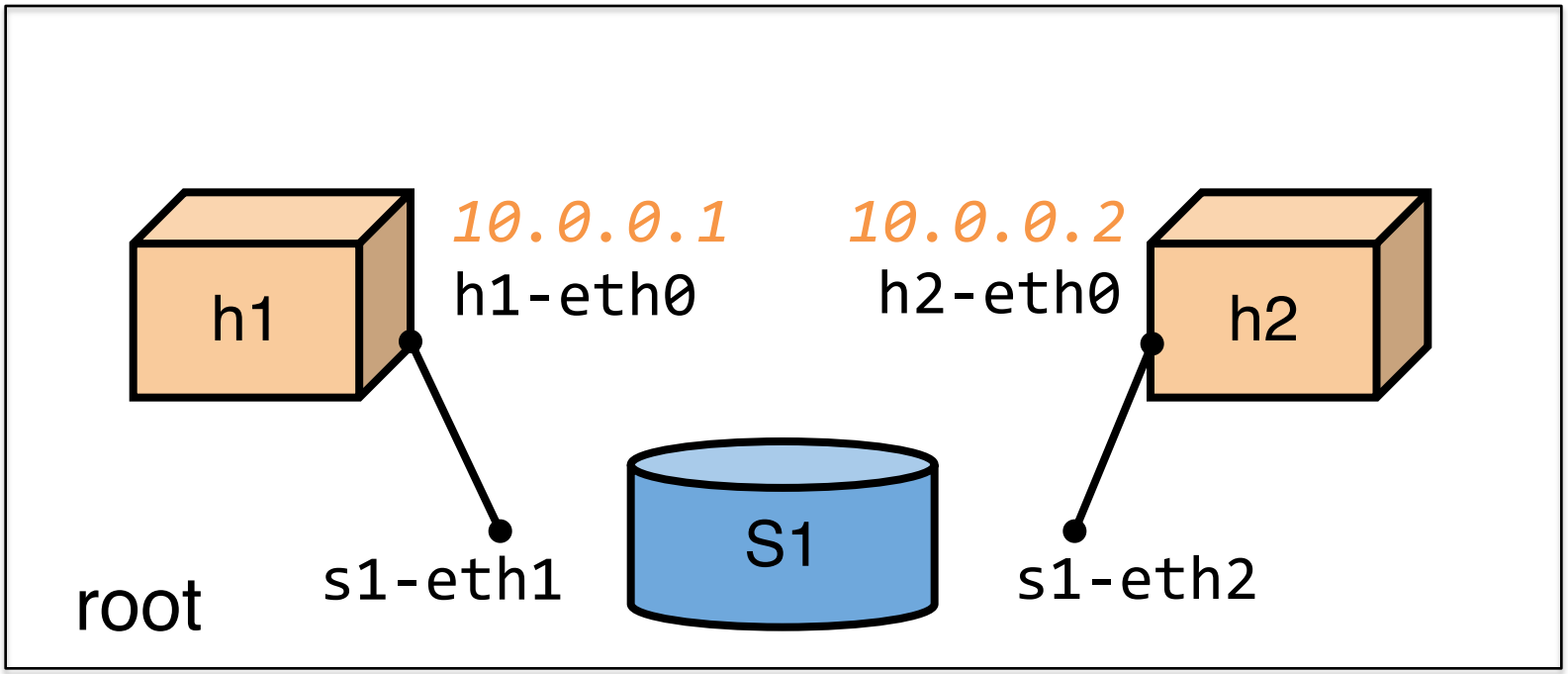
Configure Host Interfaces

```
ip netns exec h1 ifconfig h1-eth0 10.1
ip netns exec h2 ifconfig h2-eth0 10.2
ip netns exec h1 ifconfig lo up
ip netns exec h1 ifconfig lo up
```



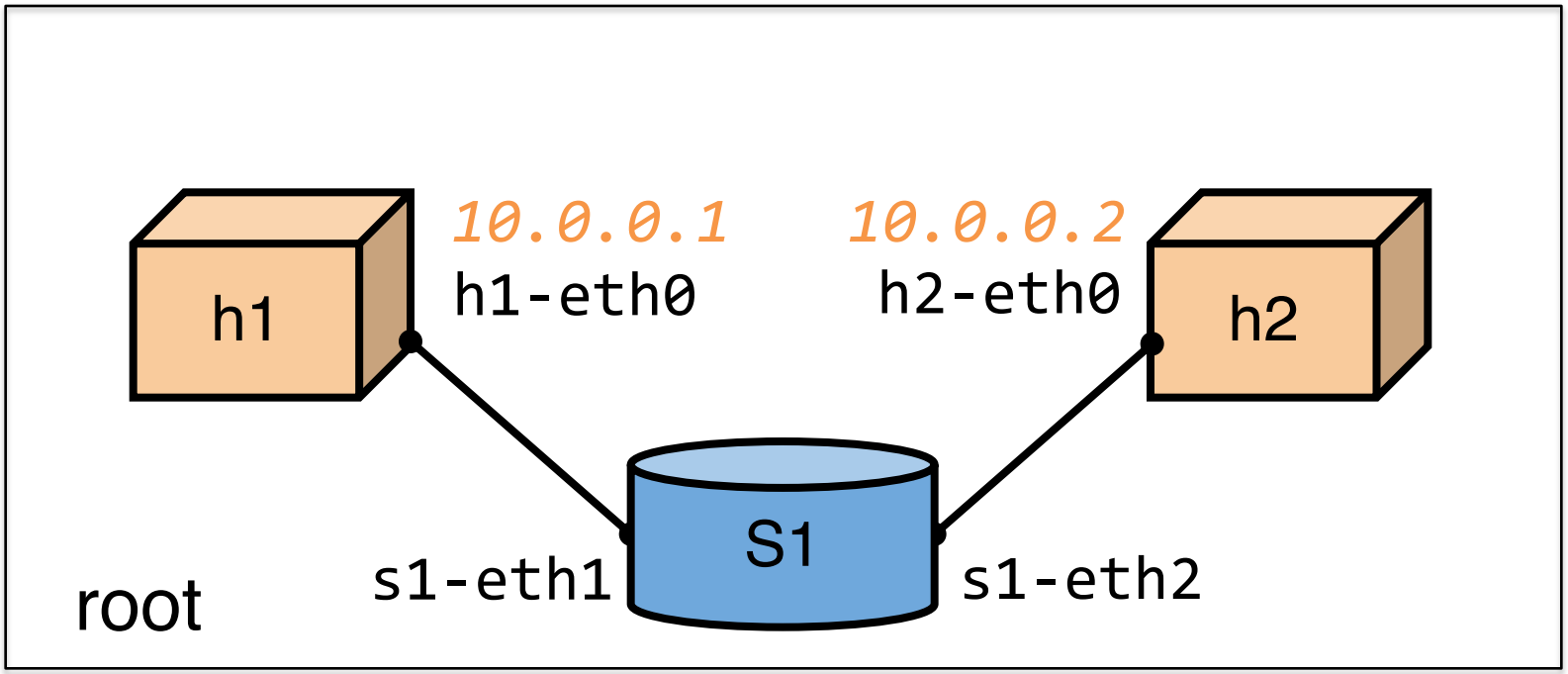
Create Virtual Switch

```
ovs-vsctl show
ovs-vsctl add-br s1
ovs-vsctl show
```



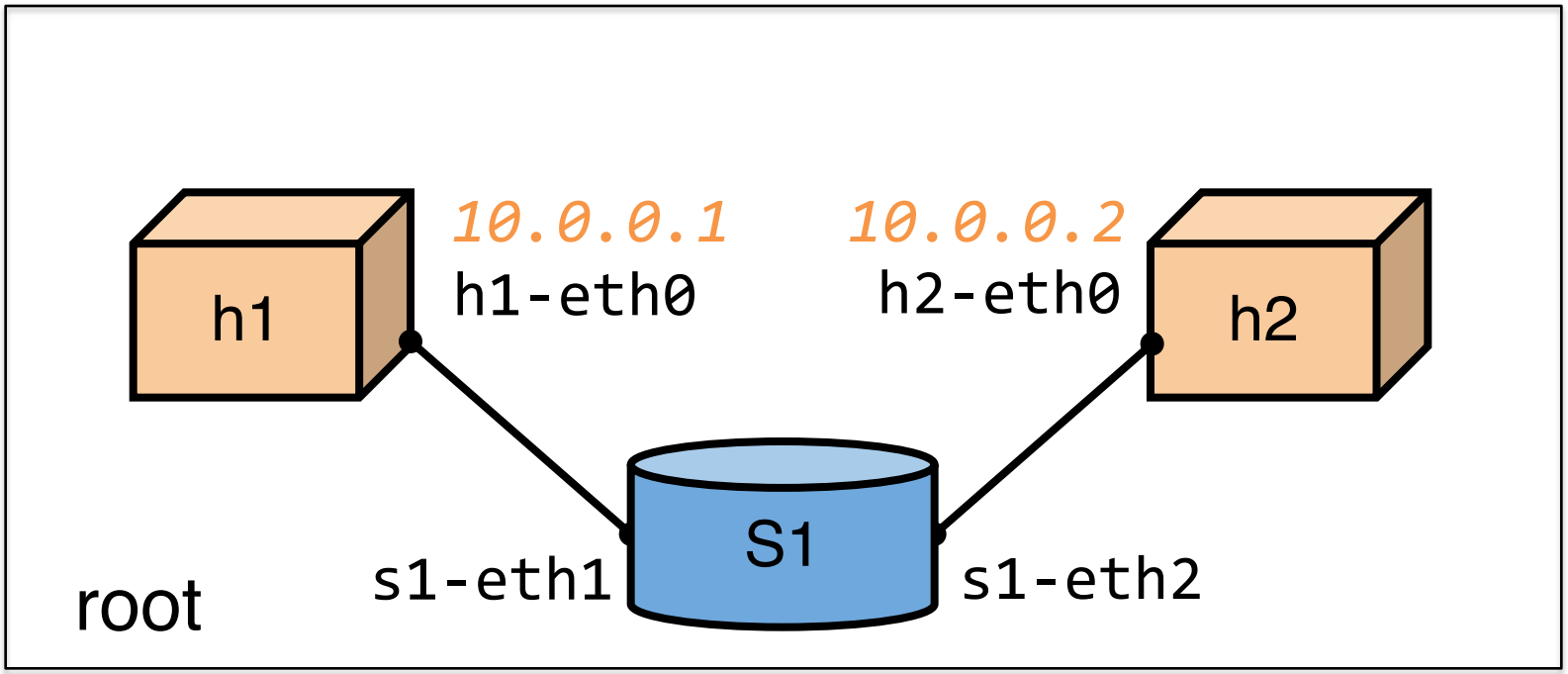
Connect Switch Ports to OVS

```
ovs-vsctl add-port s1 s1-eth1
ovs-vsctl add-port s1 s1-eth2
ovs-vsctl show
```



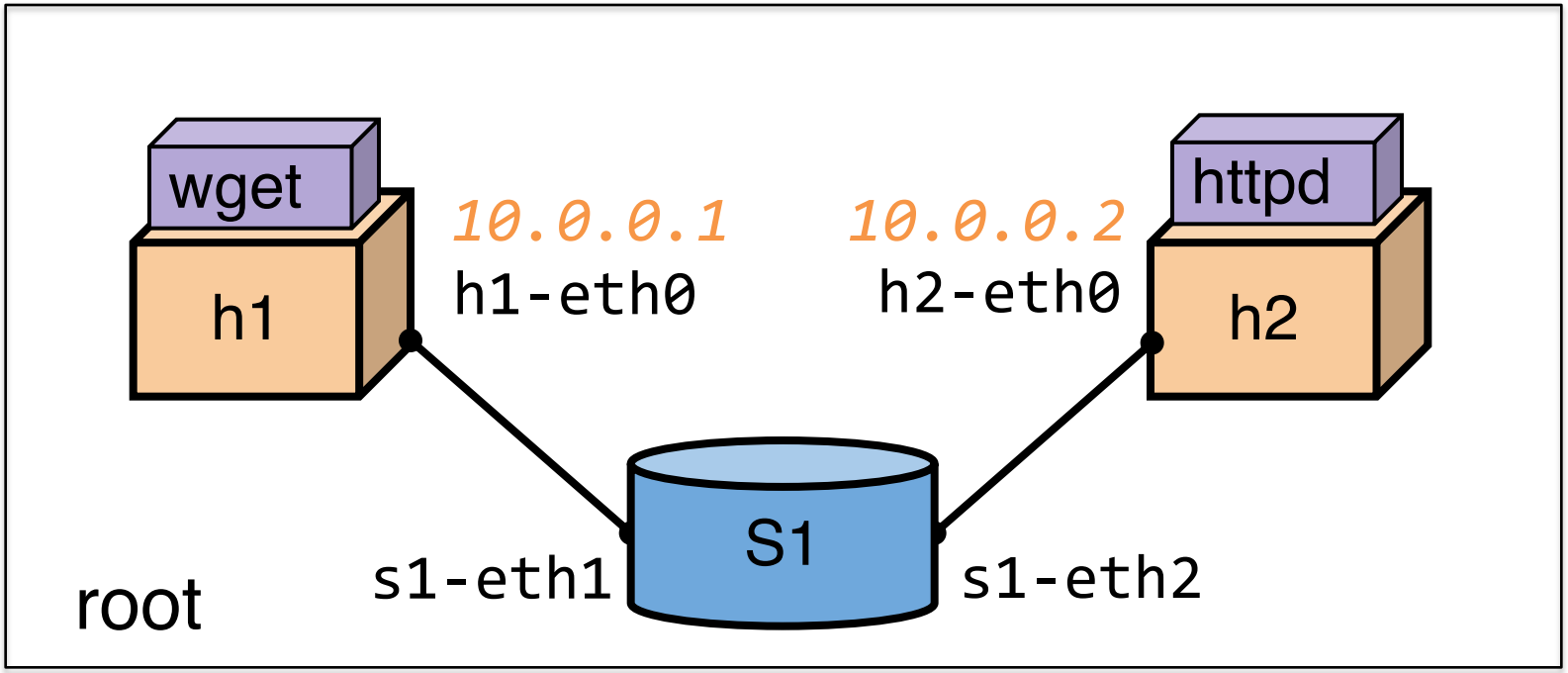
Test The Network

```
ip netns exec h1 ping -c10 10.2
ifconfig s1-eth1 up
ifconfig s1-eth2 up
ip netns exec h1 ping -c10 10.2
```



Test Client and Server

```
ip netns exec h2 python -m SimpleHTTPServer 80 &
ip netns exec h1 wget -O - 10.2
```



A summary

```
# Create host namespaces
ip netns add h1
ip netns add h2
# Create switch
ovs-vsctl add-br s1
# Create links
ip link add h1-eth0 type veth peer name s1-eth1
ip link add h2-eth0 type veth peer name s1-eth2
# Move host ports into namespaces
ip link set h1-eth0 netns h1
ip link set h2-eth0 netns h2
# Configure network
ip netns exec h1 ifconfig h1-eth0 10.1
ip netns exec h2 ifconfig h2-eth0 10.2
# Connect switch ports to OVS
ovs-vsctl add-port s1 s1-eth1
ovs-vsctl add-port s1 s1-eth2
```

What if we need to create a fat tree topology with k=4?

What if we need to create a fat tree topology with k=20 (500 nodes)?

Wouldn't it be Better if

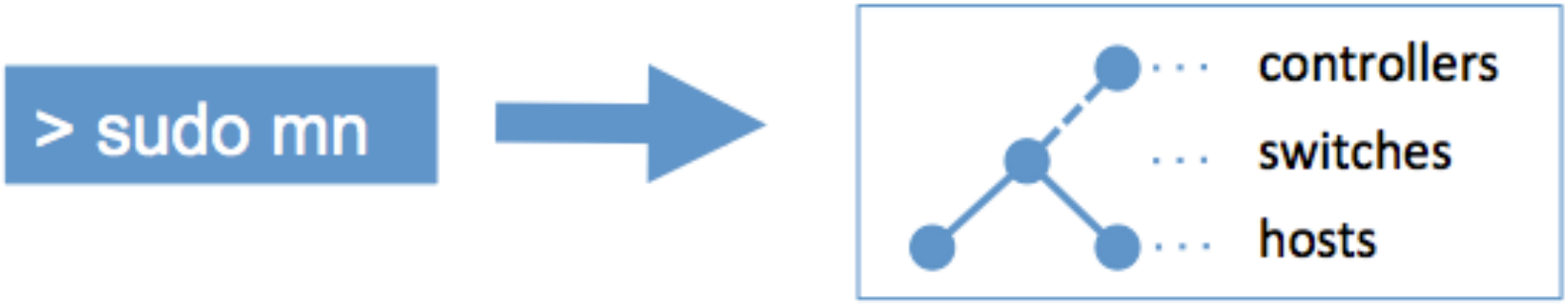
- We had a simple API that did this for us automatically?

- It allowed us to easily create topologies of varying size, up to hundreds of nodes?

Mininet

What is Mininet

- Mininet creates a **realistic virtual network**, running **real kernel, switch and application code**, on a single machine (VM, cloud or native), in seconds, with a single command:



http://mininet.org/

Using the Mininet API

```
# Create host namespaces
ip netns add h1
ip netns add h2
# Create switch
ovs-vsctl add-br s1
# Create links
ip link add h1-eth0 type veth peer name s1-eth1
ip link add h2-eth0 type veth peer name s1-eth2
# Move host ports into namespaces
ip link set h1-eth0 netns h1
ip link set h2-eth0 netns h2
# Configure network
ip netns exec h1 ifconfig h1-eth0 10.1
ip netns exec h2 ifconfig h2-eth0 10.2
# Connect switch ports to OVS
ovs-vsctl add-port s1 s1-eth1
ovs-vsctl add-port s1 s1-eth2
```

```
net = Mininet()
# Create hosts
h1 = net.addHost('h1')
h2 = net.addHost('h2')
# Create switch
s1 = net.addSwitch('s1')
# Create links
net.addLink(h1,s1)
net.addLink(h2,s1)
```

Show Nodes and Links

- **`mininet> nodes`**
  - Display nodes
- **`mininet> net`**
  - Display links
- **`mininet> dump`**
  - Dump information about all nodes

Show Interface Configurations

- **`mininet> h1 ifconfig -a`**
  - h1-eth0
  - lo
- **`mininet> s1 ifconfig -a`**
  - eth0
  - lo

Show the Processes

- **`mininet> h1 ps -a`**
  - Show the processes seen by h1
- **`mininet> h2 ps -a`**
  - Show the processes seen by h2

only the network is virtualized; each host process sees the same set of processes and directories

Test Connectivity

- **`mininet> h1 ping -c5 h2`**
  - Test connectivity between h1 and h2
  - The first ping takes a much longer time

- **`mininet> pingall`**
  - Test all-pair connectivity

Simple Web Server and Client

- `mininet> h1 python -m SimpleHTTPServer 80 &`
- `mininet> h2 wget -O - h1`

As another way

- `mininet> xterm h1 h2`
- `h1> python -m SimpleHTTPServer 80 &`
- `h2> wget -O 10.0.0.1`

Customize Topologies

```
# create the custom topology file
vi ~/mininet/custom/topo-2sw-2host.py

# run mininet with the topology
sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo
```

```python
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def build( self ):
        "Create custom topo."

        # 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() ) }
```

Lab 1: Mininet

- Warm-up
  - Install Mininet using Virtual Box
  - Try Mininet CLI to create and interact with a network
  - Try ovs-vsctl to interact with Open vSwitch (OVS)
  - Use wireshark to capture packets in Mininet
- Task
  - Construct a fattree (k=4) topology using Mininet Python API
  - Make sure all hosts are reachable
  - If not reachable, try to solve it
  - If reachable, show the path between two hosts
  - Don’t use controller! (use ‘--controller=none’ option)

Credits

[1] Te-Yuan Huang et al., Introduction to Mininet, SIGCOMM 2014 Tutorial

[2] Mininet walkthough, http://mininet.org/walkthrough/