Mininet: a network in a laptop

Laboratorio di Internet

Outline

- Introduction
- Mininet
- TUN, TAP and Veth
- Container VS Virtual Machine
- Namespaces
- Open vSwitch
- Creating a simple network
- Full system virtualization
- Lightweight Virtualization
- Creating a network in Linux
- Creating a network in Mininet

Introduction

- Learning by doing is memorable and leads to mastery
- In computer systems courses, this means building, modifying, using, and experimenting with working systems
- Networking (and distributed systems) courses require complicated testbeds including multiple servers and switches

Hardware Testbed	fast, accurate: "ground truth"	expensive, shared resource? hard to reconfigure, hard to change, hard to download
Simulator	inexpensive, flexible, detailed (or abstract!), easy to download, virtual time (can be "faster" than reality)	may require app changes, might not run OS code, detail != accuracy, may not be "believable", may be slow/non-interactive
Emulator	inexpensive, flexible, real code, reasonably accurate, easy to download, fast/interactive usage	slower than hardware, experiments may not fit, possible inaccuracy from multiplexing

Introduction

- Flexible
  - new topologies and new functionality are defined in software, using familiar languages and operating systems
- Deployable
  - deploying a functionally correct prototype on hardware-based networks and testbeds does not require changes to code or configuration
- Interactive
  - managing and running the network occurs in real time, as if interacting with a real network

Introduction

- Scalable
  - the prototyping environment scales to networks with hundreds or thousands of switches on only a laptop
- Realistic
  - prototype behavior represents real behavior with a high degree of confidence
- Share-able
  - self-contained prototypes can be easily shared with collaborators, who can then run and modify our experiments

Mininet

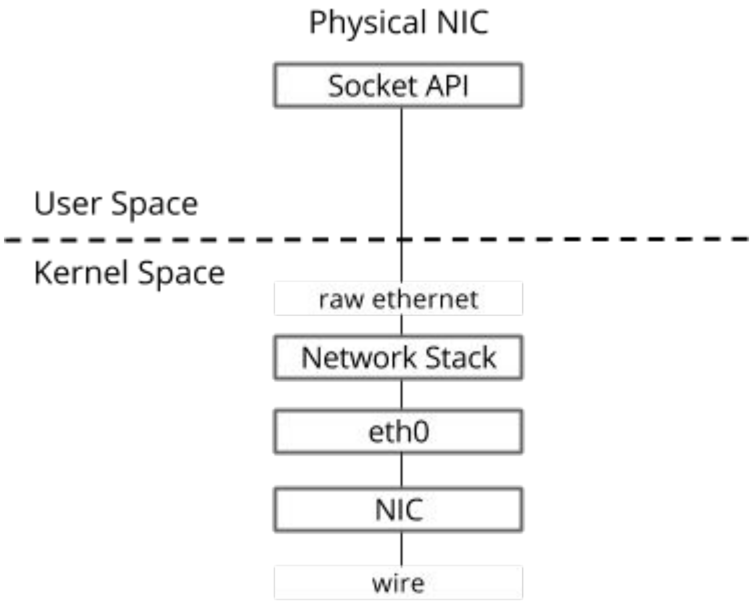
- Mininet is a system for rapidly prototyping large networks on the constrained resources of a single laptop
- The lightweight approach of using OS-level virtualization features, including processes and network namespaces, allows it to scale to hundreds of nodes
- The greatest value of Mininet will be supporting collaborative network research, by enabling self-contained SDN prototypes which anyone with a PC can download, run, evaluate, explore, tweak, and build upon

Mininet

- Mininet reaches all these goals by using lightweight virtualization
- Users can
  - implement a new network feature or entirely new architecture
  - test it on large topologies with application traffic
  - deploy the exact same code and test scripts into a real production network
- Mininet runs surprisingly well on a single laptop by leveraging Linux features
  - processes and virtual Ethernet pairs in network namespaces
    - allows to launch networks with gigabits of bandwidth and hundreds of nodes (switches, hosts, and controllers)

TUN, TAP and Veth

- Computer systems typically consist of a set networking devices (eth0, eth1 etc.)
- These network devices are associated with a physical network adapter, who is responsible for placing the packets onto the wire
- In the world of virtual networking, a degree of internal plumbing is required to patch, tunnel and forward packets within the system
- This "internal plumbing" is built using virtual networking devices, such as - TUN, TAP and Veth Pairs



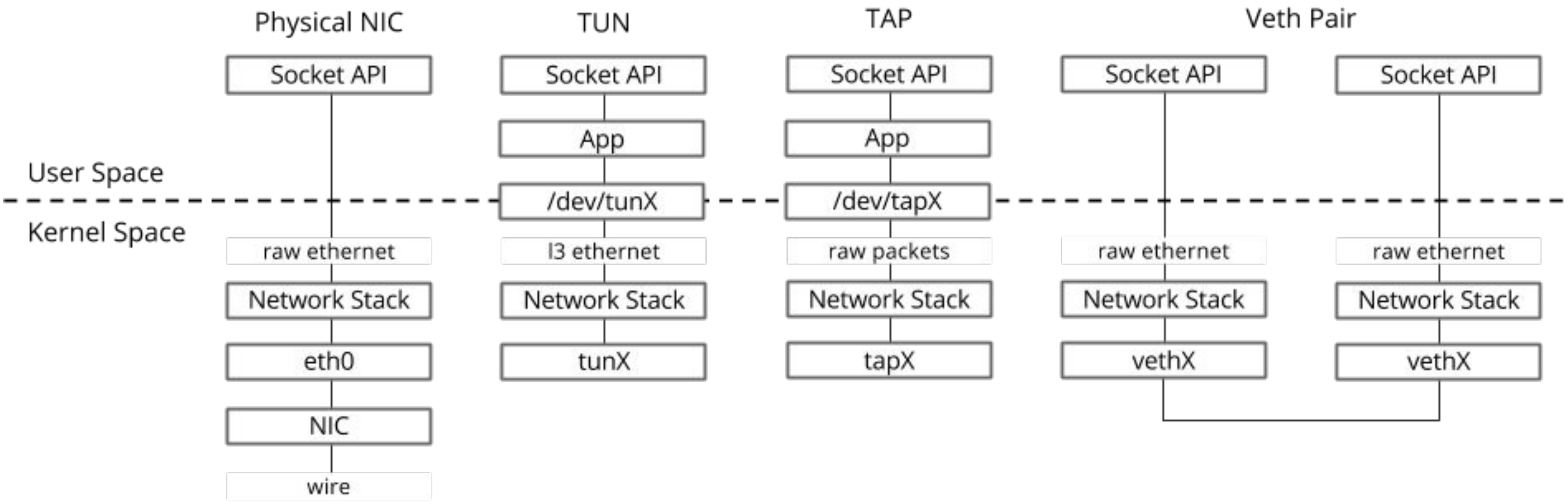
TUN, TAP and Veth

- TUN/TAP provides packet reception and transmission for user space programs
- It can be seen as a simple Point-to-Point or Ethernet device
  - instead of receiving packets from physical media, receives them from user space program
  - instead of sending packets via physical media writes them to the userspace program
- In other words, the TUN/TAP driver builds a virtual network interface on your Linux host
  - you can assign an IP to it, analyze the traffic, route traffic to it etc
- TUN (tunnel) devices operate at layer 3
  - sends and receives only IP packets
- TAP (network tap)
  - operates much like TUN however, it can work also with raw ethernet packets

TUN, TAP and Veth

- Veth devices are built as pairs of connected virtual ethernet interfaces
  - can be thought of as a virtual patch cable
  - what goes in one end will come out the other
- This makes veth pairs ideal for connecting different virtual networking components together such as
  - Linux bridges
  - OVS bridges
  - LXC containers
  - etc

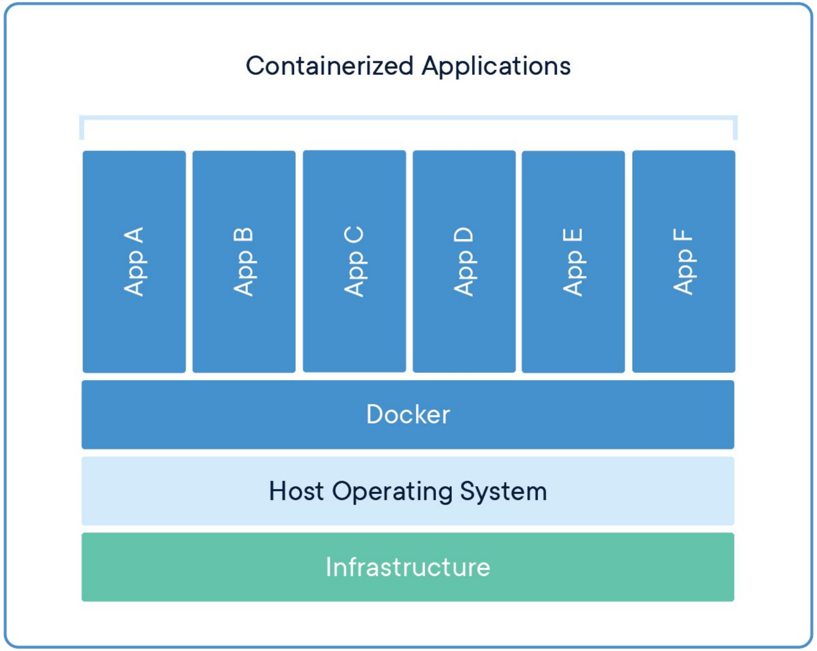
TUN, TAP and Veth

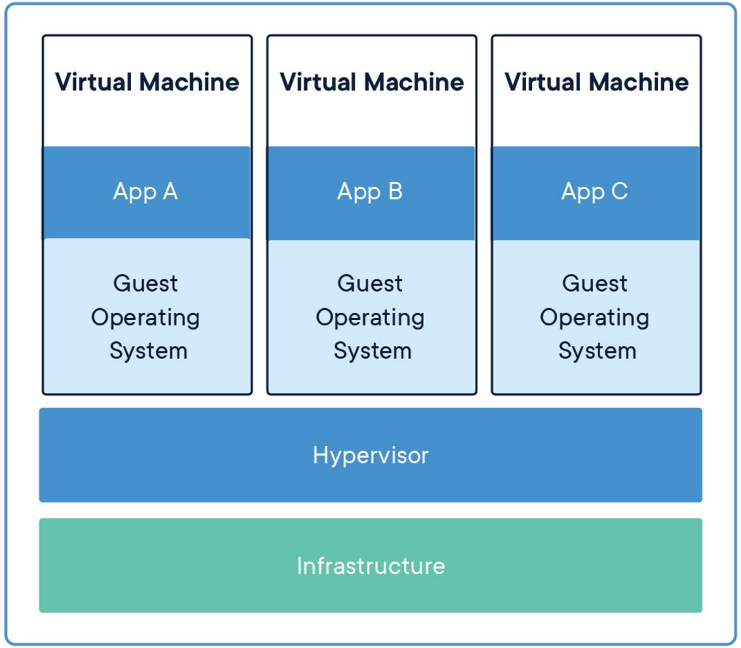


Container VS Virtual Machine

- CONTAINERS
  - Containers are an abstraction at the app layer that packages code and dependencies together
  - Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space
  - Containers take up less space than VMs (typically tens of MBs in size)
- VIRTUAL MACHINES
  - Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers
  - The hypervisor allows multiple VMs to run on a single machine
  - Each VM includes a full copy of an operating system, the application, necessary binaries and libraries (taking up tens of GBs)

Container VS Virtual Machine





Namespaces

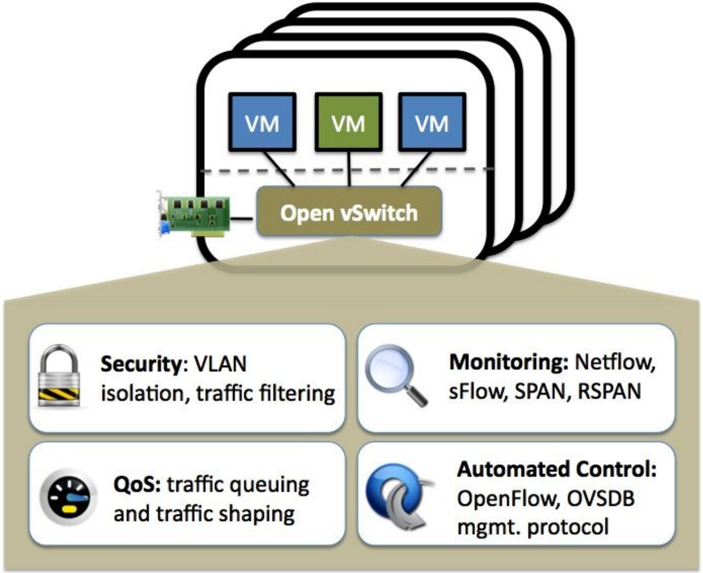
- Namespaces and cgroups are two of the main kernel technologies most of the new trend on software containerization rides on
  - cgroups are a metering and limiting mechanism
    - they control how much of a system resource (CPU, memory) you can use
  - namespaces limit what you can see
    - thanks to namespaces processes have their own view of the system's resources
- The Linux kernel provides 6 types of namespaces: pid, net, mnt, uts, ipc and user
  - a process inside a pid namespace only sees processes in the same namespace

Namespaces

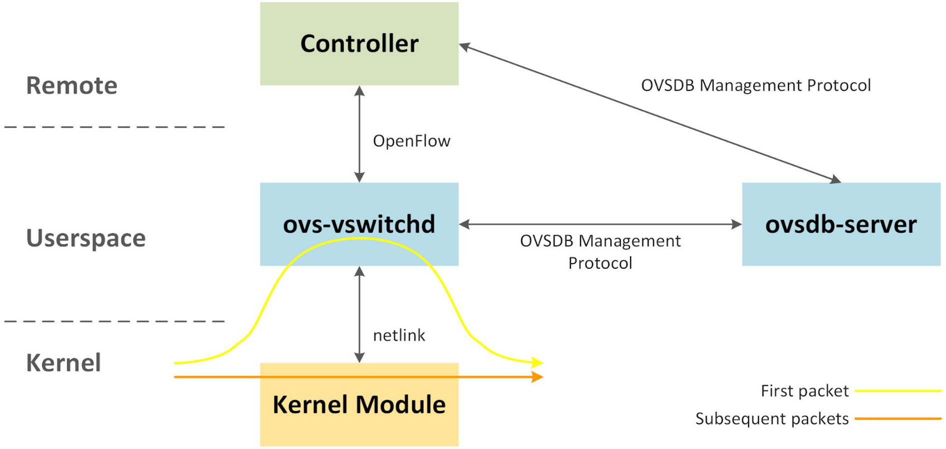
- Network namespaces provide a brand-new network stack for all the processes within the namespace
  - that includes network interfaces, routing tables and iptables rules
- One of the consequences of network namespaces is that only one interface could be assigned to a namespace at a time
  - if the root namespace owns eth0, which provides access to the external world, only programs within the root namespace could reach the Internet
- The solution is to communicate a namespace with the root namespace via a veth pair
- What to do when we have several network namespaces?

Open vSwitch

- Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license
- It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols

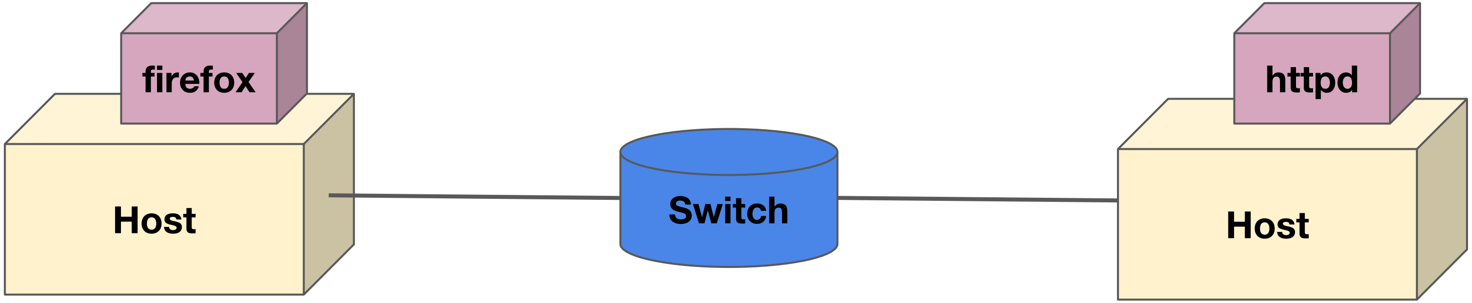

VM
VM
VM
Open vSwitch
Security: VLAN isolation, traffic filtering
Monitoring: Netflow, sFlow, SPAN, RSPAN
QoS: traffic queuing and traffic shaping
Automated Control: OpenFlow, OVSDB mgmt. protocol

Open vSwitch

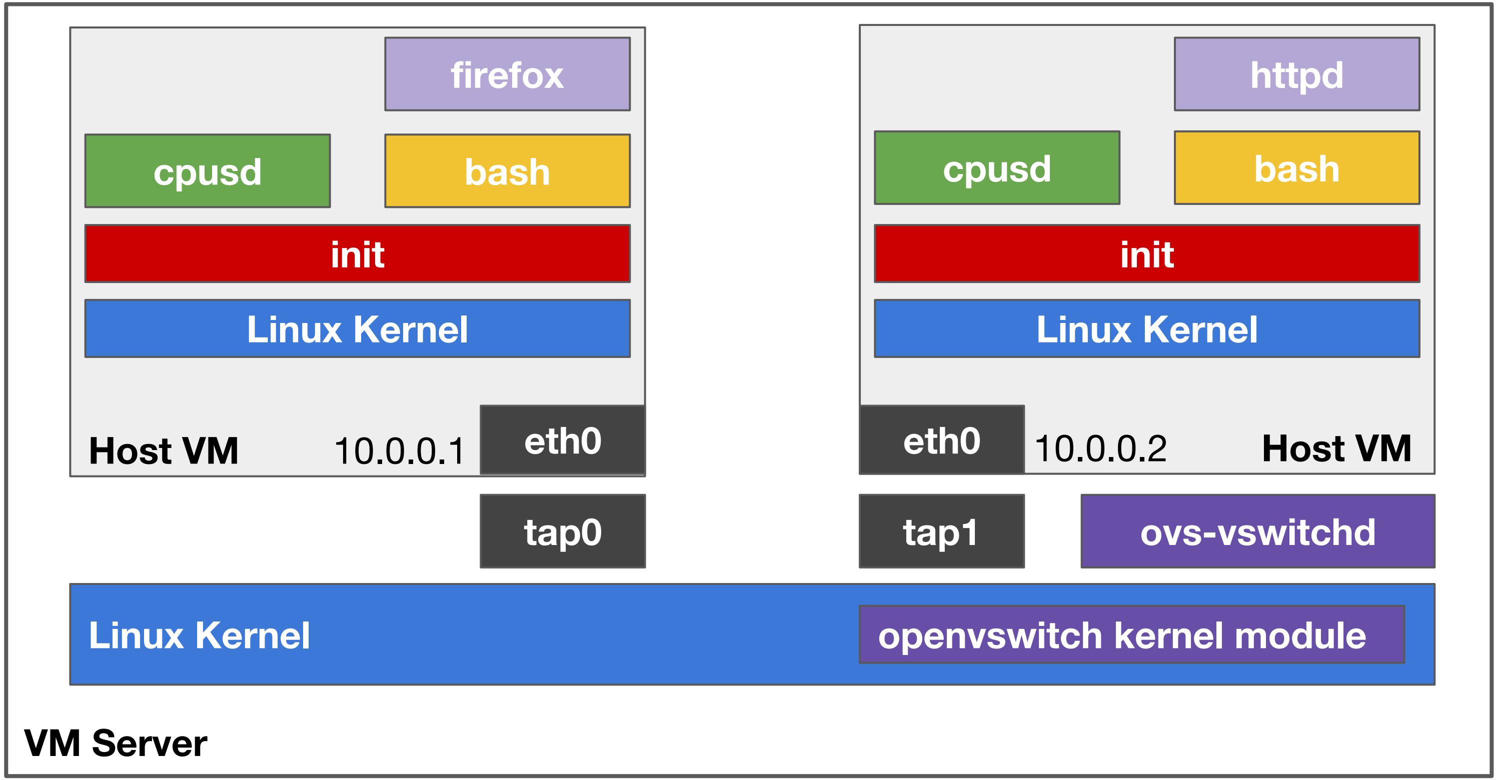


- Decision about how to process packet made in userspace
- First packet of the flow goes to ovs-vswitchd, following packets hit cached entry in kernel

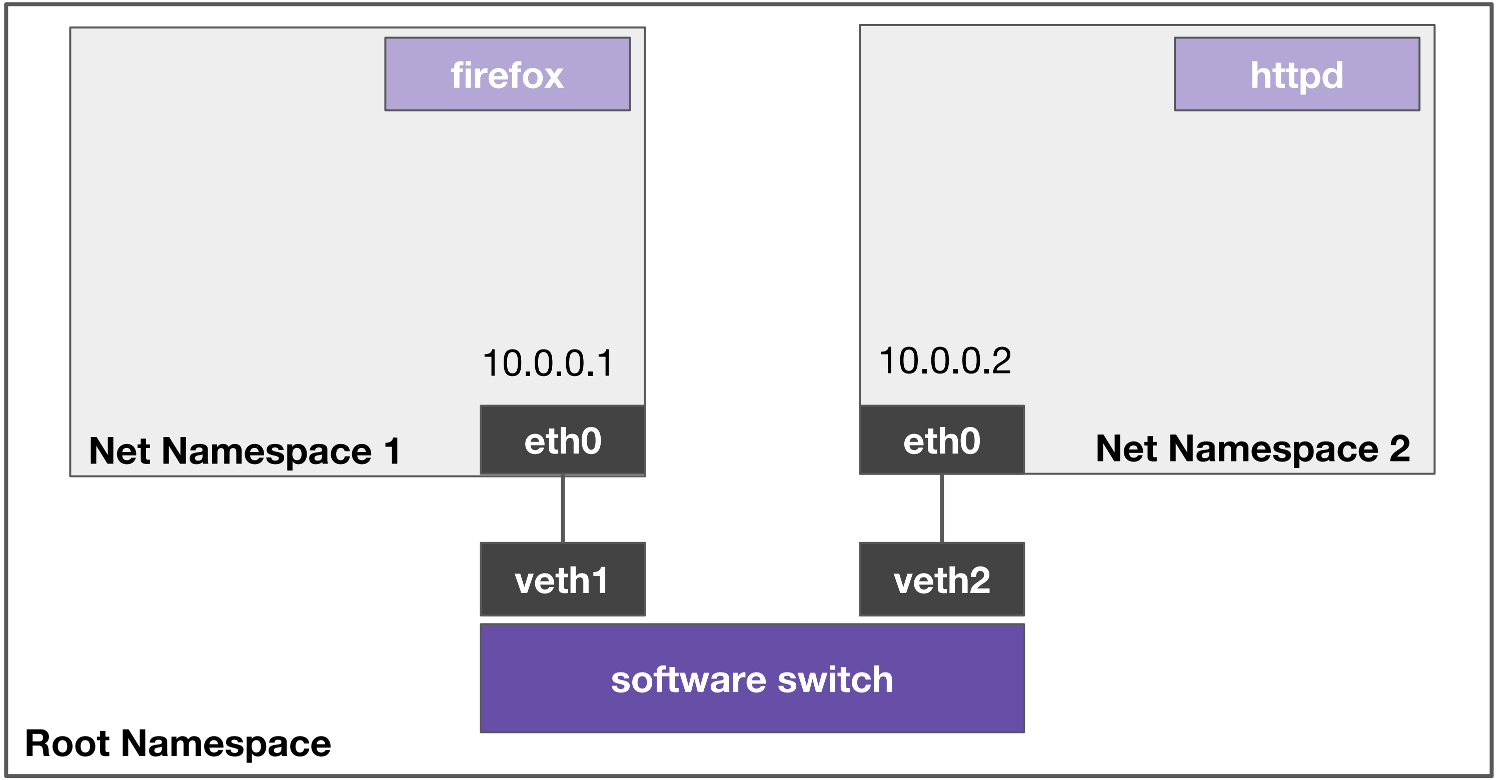
Creating a simple network



Full system virtualization



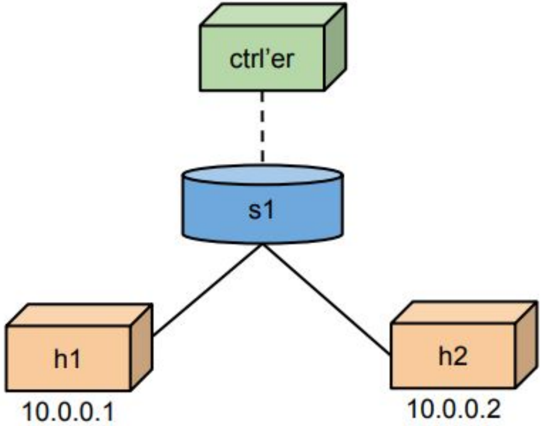
Lightweight Virtualization



Creating a network in Linux

```
sudo bash
# 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
ip link show
# Move host ports into 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
# Connect switch ports to OVS
ovs-vsctl add-port s1 s1-eth1
ovs-vsctl add-port s1 s1-eth2
ovs-vsctl show
# Set up OpenFlow controller
ovs-vsctl set-controller s1 tcp:127.0.0.1
ovs-controller ptcp: &
ovs-vsctl show
```

```
# Configure network
ip netns exec h1 ifconfig h1-eth0 10.0.0.1
ip netns exec h1 ifconfig lo up
ip netns exec h2 ifconfig h2-eth0 10.0.0.2
ip netns exec h1 ifconfig lo up
ifconfig s1-eth1 up
ifconfig s1-eth2 up
# Test network
ip netns exec h1 ping -c1 10.0.0.2
```



Creating a network in Mininet

- Wouldn’t it be great if…
  - We had a simple command-line tool and/or API that did this for us automatically?
  - It allowed us to easily create topologies of varying size, up to hundreds of nodes, and run tests on them?
  - It was already included in Ubuntu?
- Mininet offers all these services!
  - the previous network scenario can be easily built with the simple `mn` command