Mininet and Open vSwitch

Open vSwitch Fall Conference
November 16, 2015

Bob Lantz
Open Networking Laboratory

Mininet and Open vSwitch

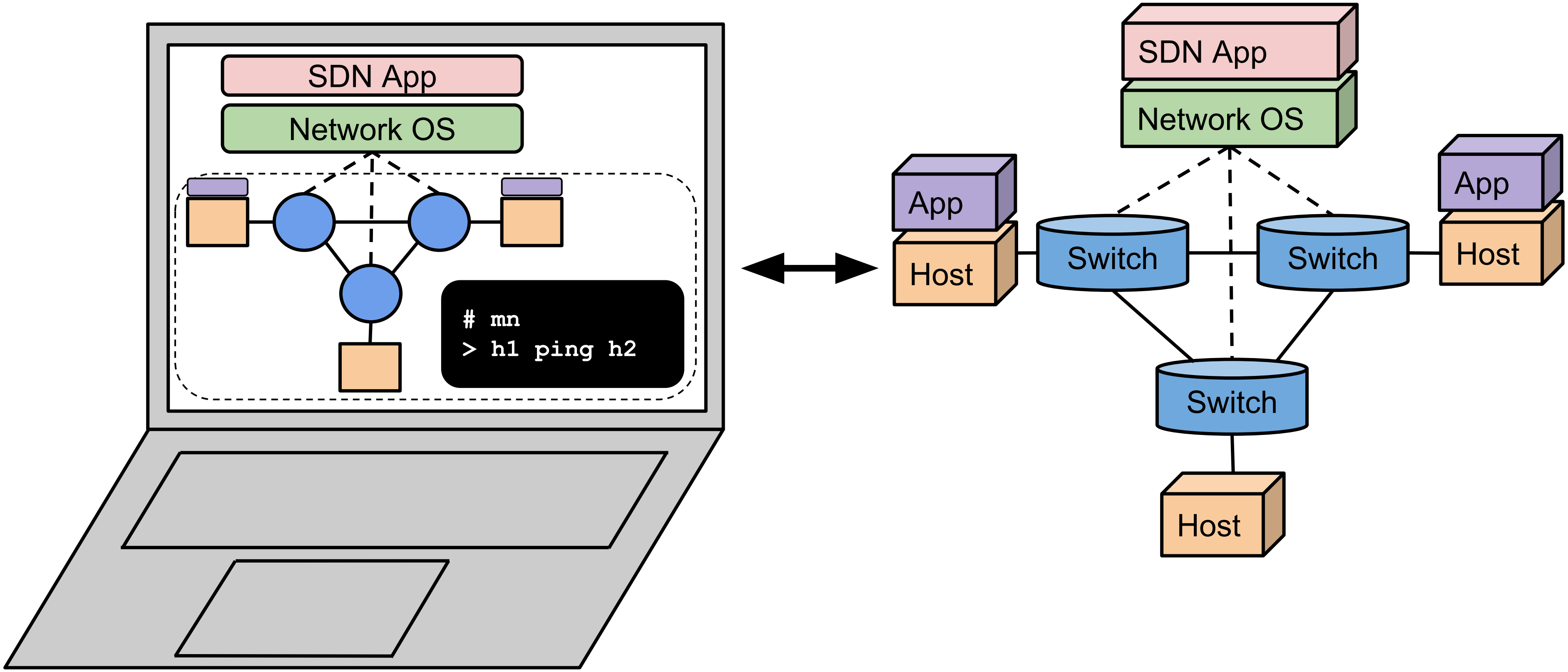
Development Platform for SDN

Processes in Namespaces

Mininet Demo and API

Experiences with Open vSwitch

A Development Platform for OpenFlow/SDN



```
# mn
> h1 ping h2
```

Developer Laptop

Hardware Network

Mininet and Open vSwitch

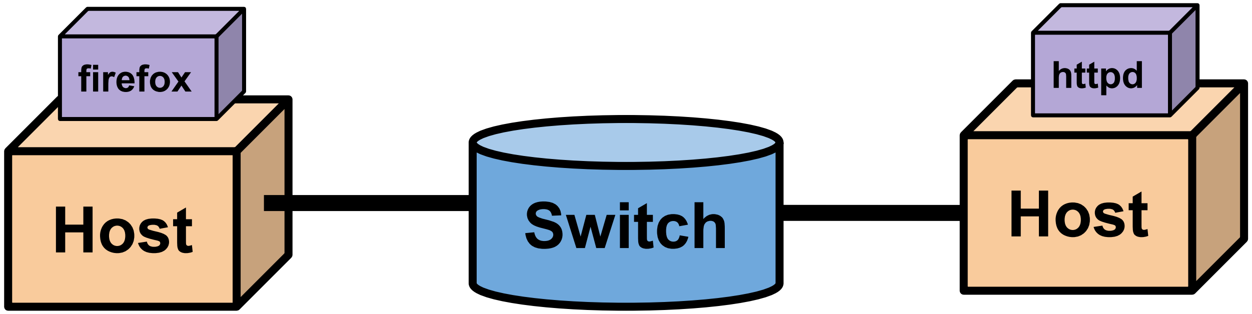
Development Platform for SDN

Processes in Namespaces

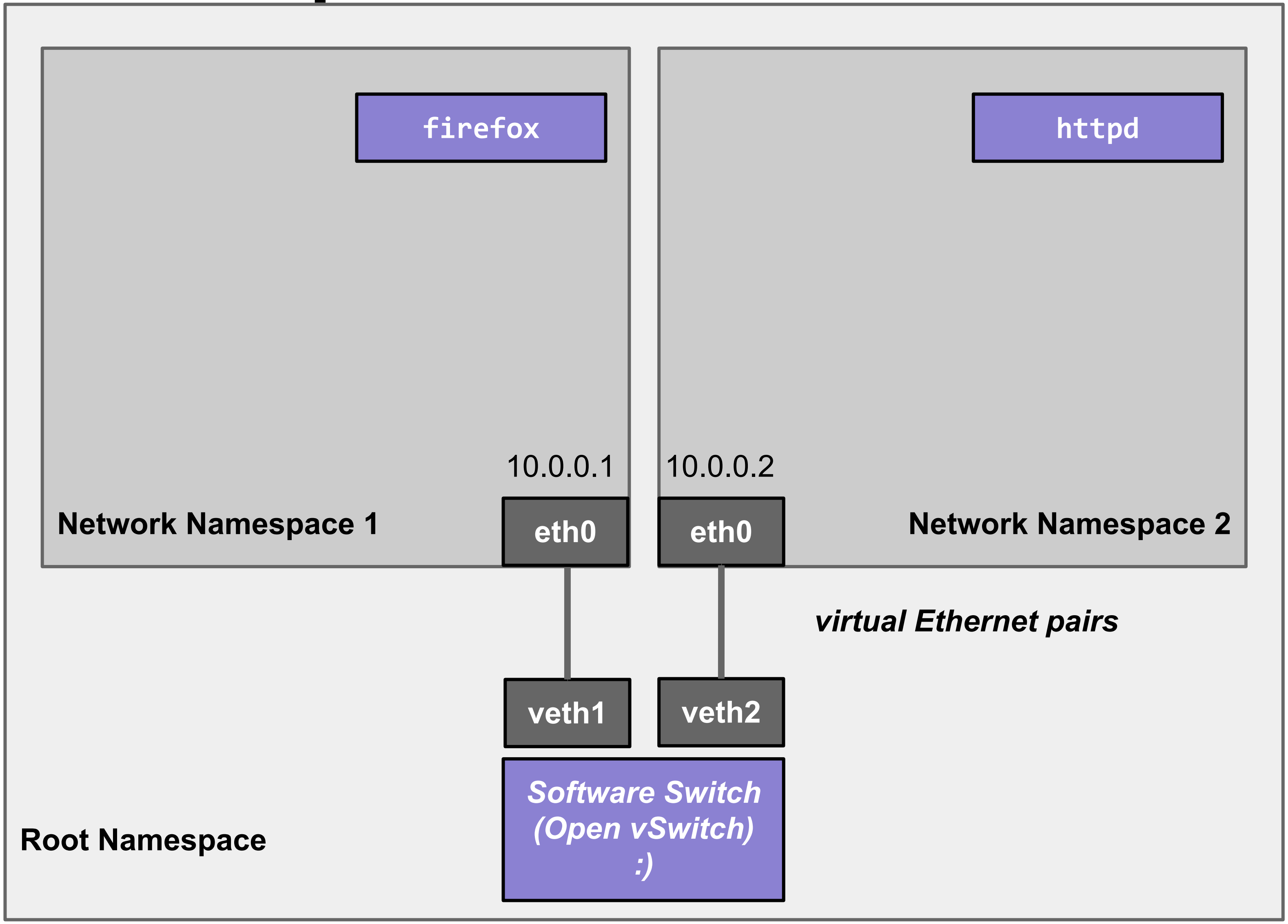
Mininet Demo and API

Experiences with Open vSwitch

To start with,
a Very Simple (legacy) Network



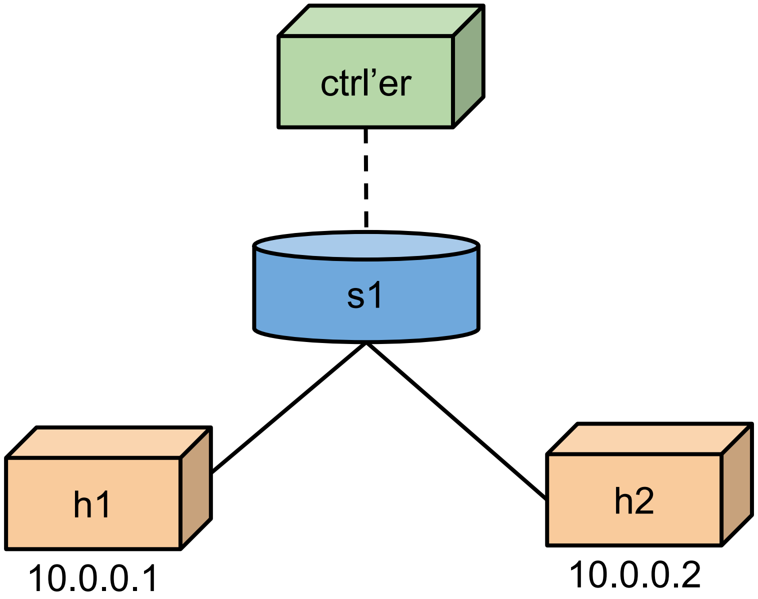
Mechanism: Processes in Network Namespaces



SDN version using Linux commands

```
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
controller ptcp: &
ovs-vsctl show
```

```
# Configure network
ip netns exec h1 ifconfig h1-eth0 10.1
ip netns exec h1 ifconfig lo up
ip netns exec h2 ifconfig h2-eth0 10.2
ip netns exec h1 ifconfig lo up
ifconfig s1-eth1 up
ifconfig s1-eth2 up
# Test network
ip netns exec h1 ping -c1 10.2
```

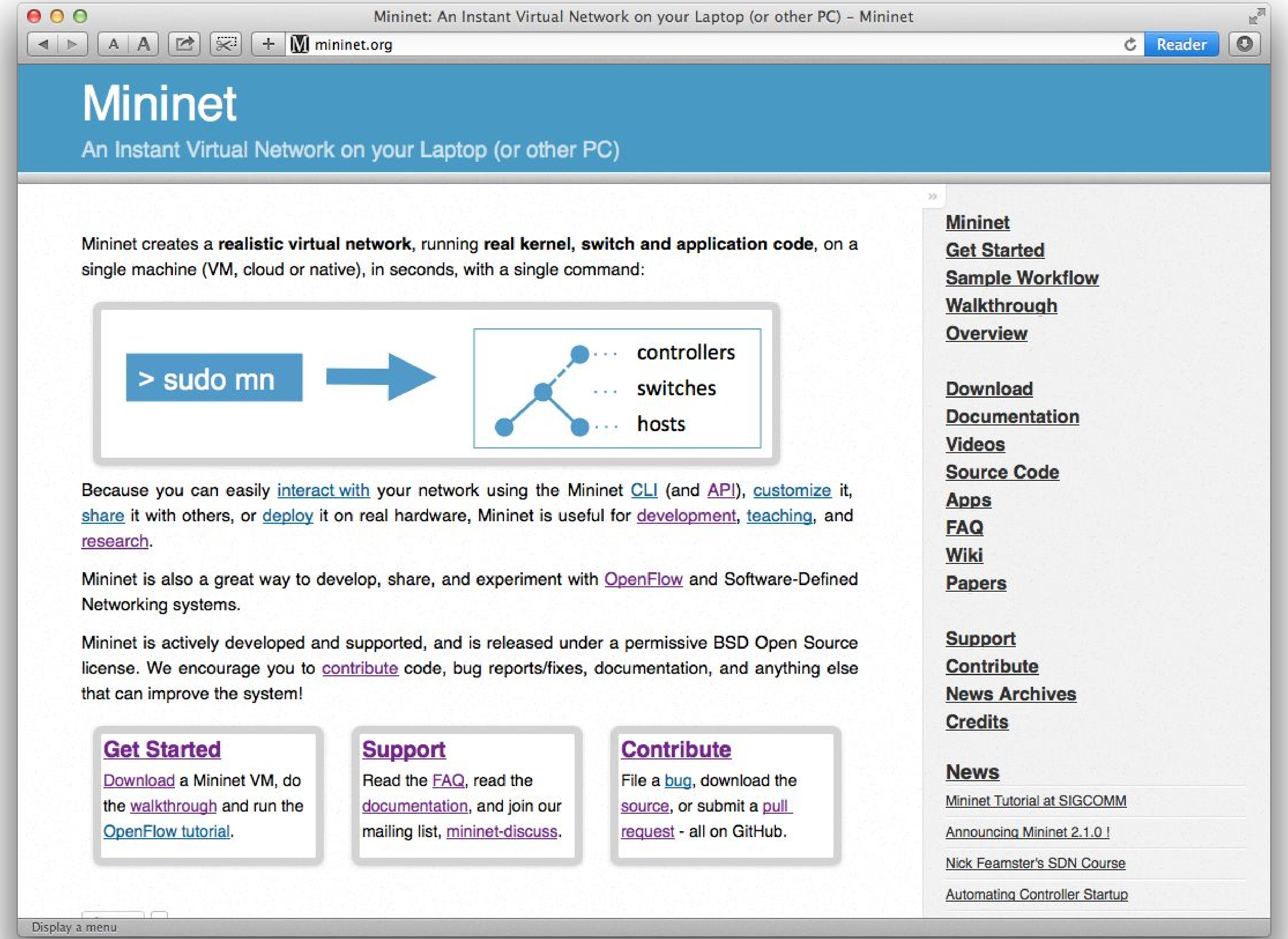


Wouldn’t it be great if...

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 Debian and Ubuntu?



Mininet

An Instant Virtual Network on your Laptop (or other PC)

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:

Because you can easily interact with your network using the Mininet CLI (and API), customize it, share it with others, or deploy it on real hardware, Mininet is useful for development, teaching, and research.

Mininet is also a great way to develop, share, and experiment with OpenFlow and Software-Defined Networking systems.

Mininet is actively developed and supported, and is released under a permissive BSD Open Source license. We encourage you to contribute code, bug reports/fixes, documentation, and anything else that can improve the system!

Get Started

Download a Mininet VM, do the walkthrough and run the OpenFlow tutorial.

Support

Read the FAQ, read the documentation, and join our mailing list, mininet-discuss.

Contribute

File a bug, download the source, or submit a pull request - all on GitHub.

News

Mininet Tutorial at SIGCOMM

Announcing Mininet 2.1.0 !

Nick Feamster's SDN Course

Automating Controller Startup

Mininet and Open vSwitch

Development Platform for SDN

Processes in Namespaces

Mininet Demo and API

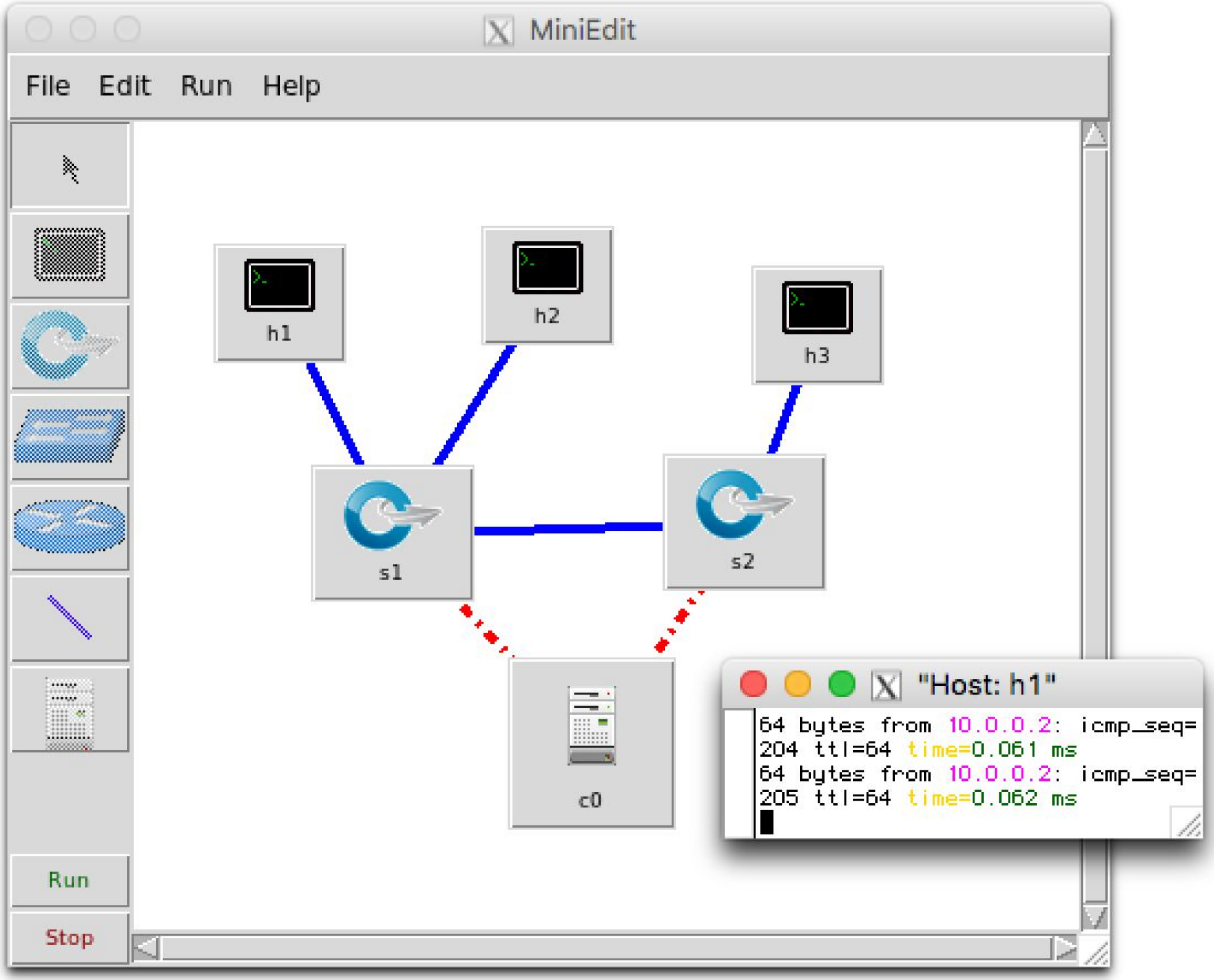
Experiences with Open vSwitch

Mininet command line tool and CLI demo

```
# mn
# mn --topo tree,depth=3,fanout=3 --link=tc,bw=10
mininet> xterm h1 h2
h1# wireshark &
h2# python -m SimpleHTTPServer 80 &
h1# firefox &
# mn --topo linear,100
# examples/miniedit.py
```

Mininet GUI (MiniEdit)

(unfortunately omitted from live presentation!)



```
64 bytes from 10.0.0.2: icmp_seq=
204 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=
205 ttl=64 time=0.062 ms
```

Mininet's Python API

Core of Mininet!! Everything is built on it.

Dynamic Python >> static JSON/XML/etc.

Easy and (hopefully) fun

Python is used for *orchestration*, but emulation is performed by compiled C code (Linux + switches + apps)

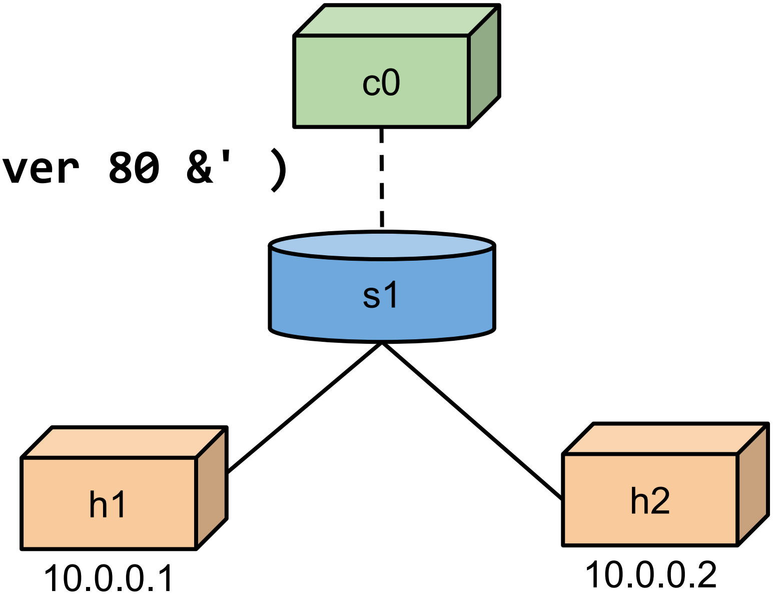
api.mininet.org

docs.mininet.org

Introduction to Mininet

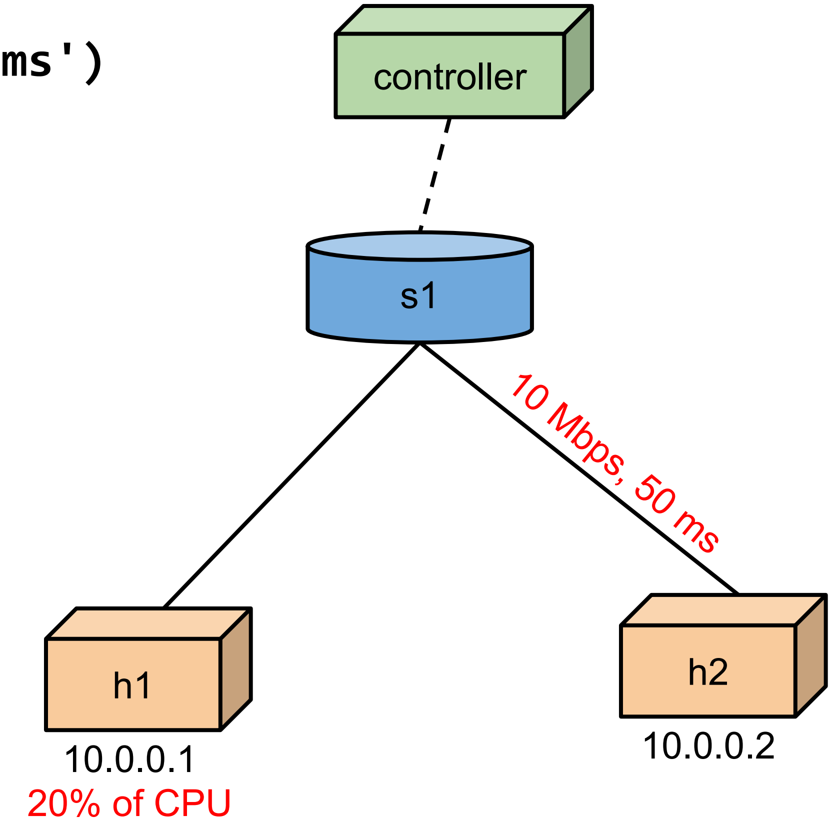
Mininet API basics

```
net = Mininet()                  # net is a Mininet() object
h1 = net.addHost( 'h1' )     # h1 is a Host() object
h2 = net.addHost( 'h2' )     # h2 is a Host()
s1 = net.addSwitch( 's1' )       # s1 is a Switch() object
c0 = net.addController( 'c0' )   # c0 is a Controller()
net.addLink( h1, s1 )            # creates a Link() object
net.addLink( h2, s1 )
net.start()
h2.cmd( 'python -m SimpleHTTPServer 80 &' )
sleep( 2 )
h1.cmd( 'curl', h2.IP() )
CLI( net )
h2.cmd('kill %python')
net.stop()
```



Performance modeling in Mininet

```
# Use performance-modeling link and host classes
net = Mininet(link=TCLink, host=CPULimitedHost)
# Limit link bandwidth and add delay
net.addLink(h2, s1, bw=10, delay='50ms')
# Limit CPU bandwidth
net.addHost('h1', cpu=.2)
```



examples:

reproducingnetworkresearch.wordpress.com

Mininet and Open vSwitch

Development Platform for SDN

Processes in Namespaces

Mininet Demo and API

Experiences with Open vSwitch

Experience with OvS and Mininet

Network emulation is an **incredibly useful application of Open vSwitch**!

Mininet + Open vSwitch gives you an **instant network on your laptop**, for development, testing, research, demos, experimentation... almost **anything you can think of**!

Experience with OvS and Mininet

Initially, **poorer startup and switching performance** than Stanford reference switch (I miss the reference kernel switch!)

Switching performance has improved over time by a factor of 30+

Inclusion in the Linux kernel was a major coup!

Startup performance is still slow due to **`ovsdb`**

OVS **patch links do provide better performance and faster startup** at the expense of losing tcpdump capability and bandwidth limiting using **`tc`**.

Even **batching `ovsdb` commands**, it is still **slow to create large networks** with hundreds/thousands of switches/ports.

Both OvS and Mininet want to use **`tc`**.

How can OvS evolve to improve support for network emulation?

Scaling to **thousands of virtual switches** (many **thousands of ports**!) on a single Linux kernel. (Also long chains of patch links.)

Supporting **configuration of flow tables** (size, match/action support) and **flow pipeline** on individual switches (P4 may help, though it's overkill.)

Even better performance of true OpenFlow switching (closer to memory bandwidth and to netmap/VALE's reported performance)

Accurate **switch port characteristics reporting** from Linux, OpenFlow (currently everything is reported as 10 Gb/s)

Tracking OpenFlow (and **possibly P4**) is essential for enabling the future of networking, including network OS development and network applications (compare with smartphone revolution.)

Can OvS **do all this today**? If not, **how can we get there**?

Thank you

mininet.org

github.com/mininet

reproducingnetworkresearch.wordpress.com