

# HANDS-ON SDN

*Introduction to Software-defined Networking*  
Block Course – 16-20 March 2015

David Koll

# Remarks on Presentations

- We will have 11 presentations on Friday afternoon
  - 15 minutes instead of 20 minutes per presentation
  - 5 minutes Q&A remains unchanged
- 
- Start at 1:30 pm, in order of appearance on the course website
  - Make sure your presentation works in this room beforehand
  - Be present for all presentations and participate in the discussion

# Where we are now

## **You have now learned about:**

- Python programming
- SDN basic principles
  - Basic concepts (CP/DP separation etc.)
  - De-facto standard interfaces (OpenFlow)
  - Controllers (NOX, POX, ...)
  - Virtualization (FlowVisor)

# Where we want to go

## You have now learned about:

- Python programming
- SDN basic principles
  - Basic concepts (CP/DP separation etc.)
  - De-facto standard interfaces (OpenFlow)
  - Controllers (NOX, POX, ...)
  - Virtualization (FlowVisor)
- Put the stuff learned into practice:
  - Implement OpenFlow?
  - Implement controllers?
  - Implement FlowVisor?
  - Rather: *learn how to use and program them!*
    - Hands-on work on state-of-the-art tools

# Detailed Course overview

Day	Morning Session 1	Morning Session 2	Afternoon Session 1	Afternoon Session 2
Mon	Introducing SDN I	Lecture Exercises	Introduction to Python	Python Exercises
Tue	Introducing SDN II	Lecture Exercises	Introduction to Python (cont.)	Python Exercises
Wed	Current Research in SDN	Paper Selection and Reading (Teams)	<b>Hands-On SDN I</b>	<b>SDN Exercises</b>
Thu	<b>Hands-On SDN II</b>	<b>SDN Exercises</b>	<b>Hands-On SDN III</b>	Presentation Preparation
Fri	<b>Hands-On SDN IV</b>	<b>SDN Exercises</b>	Wrap-Up & Free Slot	Presentations

**Hands-On Sessions: putting the concepts learned in the lectures into practice**

# How can we get there?

- Luckily, implementations are available.
  - Switches implementing OF
  - Controllers implementing OF
- So, how do we run them?
  - We don't have a hardware testbed at hand
  - We don't have access to a production network
  - We may want to test different things on different network topologies
  - Simulation?

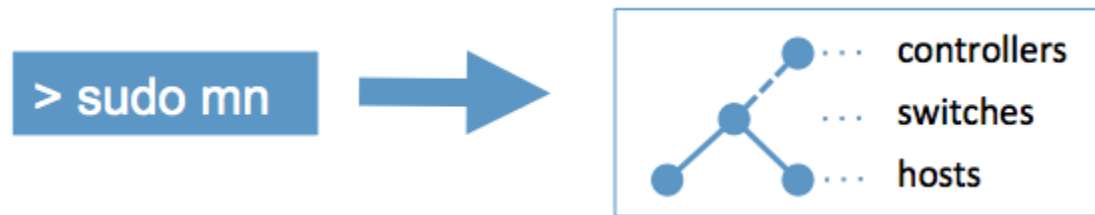


# Emulation of Networks

- Network emulation means to run unmodified code interactively on virtual hardware
- Huge benefit:
  - Can actually port our applications seamlessly to hardware
- Challenges:
  - Scalability: need to model hosts, switches, links, controllers, ...
  - Ease-of-Use: easily allow to create different topologies with varying parameters
  - Accuracy: results have to match results obtained from running same experiment on hardware

# Enter 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”[1]

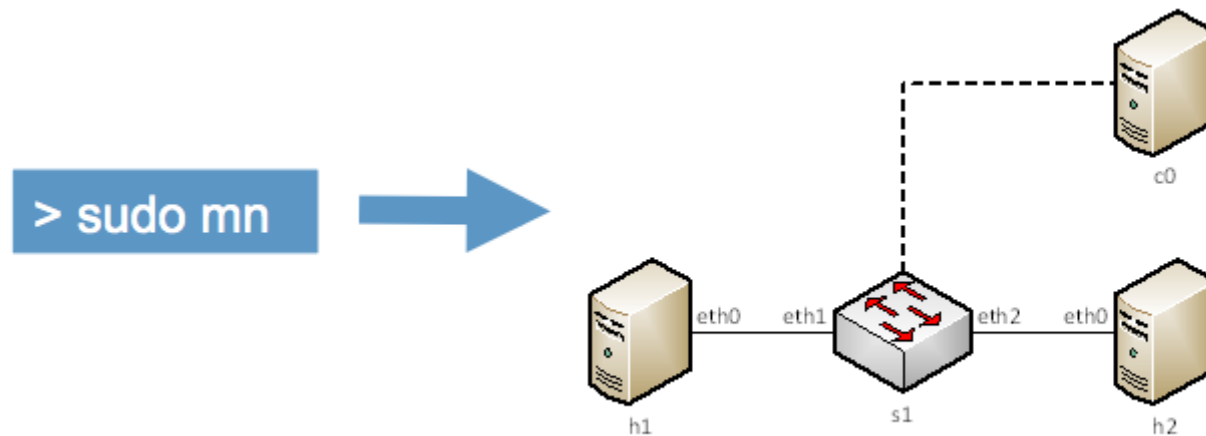


[1] [mininet.org](http://mininet.org) (Paper: Lantz et al.: “A Network in a Laptop: Rapid Prototyping for Software-Defined Networks”, ACM HotNets 2010)



# Enter 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”[1]



[1] mininet.org



# Enter Mininet

Mininet offers CLI & API to interact with the network

**(see demo)**



# Customize Topologies

Mininet is not limited to the very basic setup

**(see demo)**

# Customize 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() ) }
```



# Customize Switches and Controllers

You can connect different switches and controllers

**(see demo)**



# Bring Links Up/Down

Change the topology at runtime

**(see demo)**



# Python Interpreter

Can access each host, switch and controller with Python

**(see demo)**



# Use of Wireshark

We can use Wireshark to debug our network

**(see demo)**



# Limitations?

Limited by single system resources

Limited to Linux kernel (e.g., portability to Windows?)

# Exercise!

Time for Exercise 5 and 6