

Software Defined Networking

Lab Work Introduction



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Jeremias Blendin, **Christian Koch**



Peer-to-Peer Systems
Engineering Lab (PS)

<http://www.ps.tu-darmstadt.de/teaching/ws1617/sdn/>

PS - Peer-to-Peer Systems Engineering Lab
Dept. of Electrical Engineering and Information Technology
Technische Universität Darmstadt
Rundeturmstr. 12, D-64283 Darmstadt, Germany
<http://www.ps.tu-darmstadt.de/>

1. Organizational Issues

❖ Support:

- Moodle Forum!
- Lab Work: Jeremias Blending, Christian Koch
 - By e-mail [jblending|ckoch]@ps.tu-darmstadt.de
 - Room: S3|19 7 or 8 (**only upon appointment!**)

2. Exercise Overview

Oct 25	(CK)	Introduction / Exercise 1 Hand-out
Nov 1	(JR)	Exercise 1 Discussion / Exercise 2 Hand-out
Nov 8	(CK)	Lab Work 1 Introduction
Nov 15	(CK)	Lab Work 1 Discussion / Lab Work 2 Introduction
Nov 22	(JR)	Exercise 2 Discussion / Exercise 3 Hand-out
Nov 29	(MW)	Exercise 3 Discussion / Exercise 4 Hand-out
Dec 6	(JB)	Lab Work 2 Discussion / Lab Work 3 Introduction
Dec 13	(CK)	Exercise 4 Discussion / Exercise 5 Hand-out
Dec 20	(JB)	Lab Work 3 Discussion

Christmas Break

Jan 10	(MW)	Exercise 5 Discussion / Exercise 6 Hand-out
Jan 17	(LN)	Exercise 6 Discussion / Exercise 7 Hand-out
Jan 24	(LN)	Exercise 7 Discussion
Jan 31		Consultation hour for the exam
Feb 7		Backup

3. Lab Exercises – Organization

- ❖ Goal: Hands on mechanisms presented in the lecture and theoretical exercise
- ❖ For lab exercise we use a number of different tools, such as Mininet
- ❖ Should be group work! (2-3 persons)
- ❖ Submission of solution before next lab
 - Deadline 16:00 before the next lab
 - Individual submissions by each participant to Moodle!
- ❖ Code must be runnable and adequately solve task
- ❖ Selective code reviewed by supervisors



Our tool to simulate software-defined networks

INTRODUCTION TO MININET

❖ Lab Requirements

- Virtualization Software: **VirtualBox**, VMWare, KVM, etc...
- An **SSH Client**
 - e.g. PuTTY for Windows Users or
 - the built-in OpenSSH for Mac/Linux users

Mininet: Installation

- ❖ Download Mininet:
 - <https://github.com/mininet/mininet/wiki/Mininet-VM-Images> (ca. 1 GB)
- ❖ ZIP file contains an **.ovf** file:
 - You may import it with VirtualBox, VMWare, KVM etc.
- ❖ Change network settings for Mininet to additionally use a **host-only adapter** (VirtualBox):
 - Host-only adapter means: Only your machine can connect to the Mininet VM via this adapter and vice versa.
Do **NOT bridge** adapters!
 - Select VM > Change... > Network > Adapter 2
 - Check „Active“
 - Select „Connected to Host-Only Adapter“

Mininet: Installation (2)

- ❖ Run the Mininet VM
- ❖ Login with username **mininet**, password **mininet**
 - Change your password to a *strong* one for additional security (**passwd**)
- ❖ Run ifconfig, if no device eth1 is present, enter:

```
sudo -s  
nano /etc/network/interfaces
```

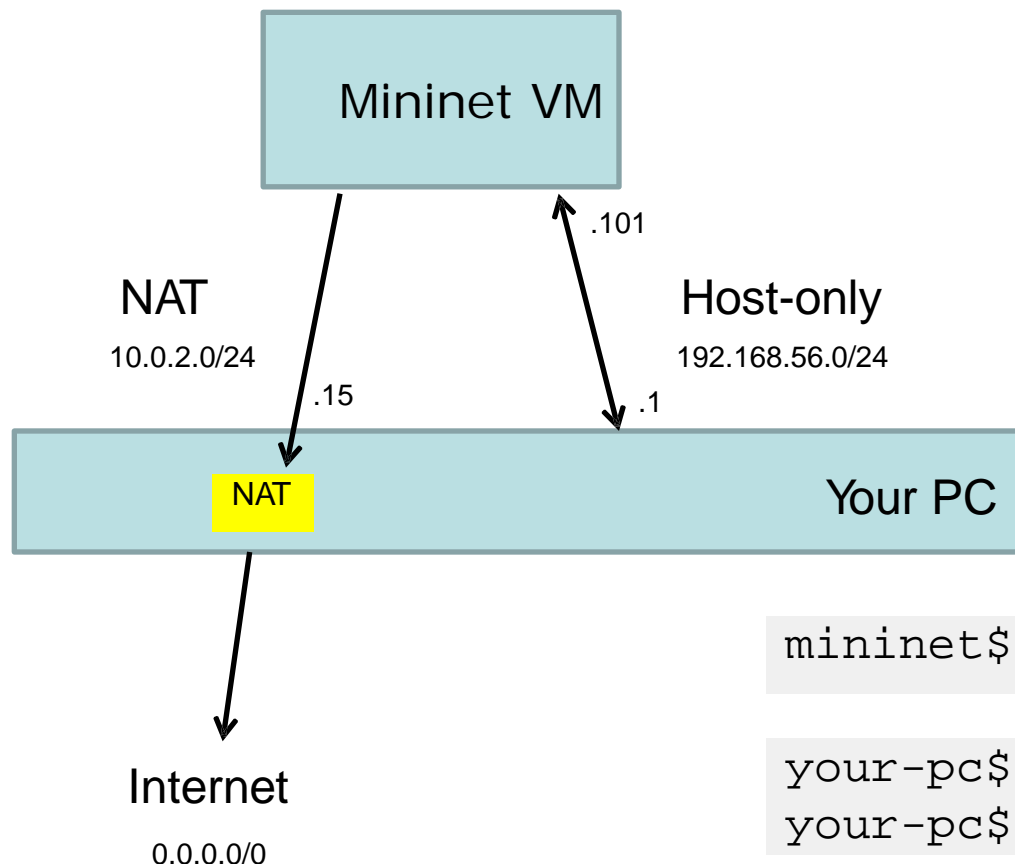
- ❖ In the editor, e.g. nano, append these lines

```
auto eth1  
iface eth1 inet dhcp
```

- ❖ Reboot.

Mininet: Installation (3)

❖ Now, you have the following network configuration:



```
mininet$ ping google.de
```

```
your-pc$ ping 192.168.56.101  
your-pc$ ssh mininet@192.168.56.101
```

Mininet: Initial Topology

❖ Log in to Mininet via ssh (Session 1)

❖ Create our initial topology

```
$ sudo mn --topo single,3 --mac --arp --switch ovsk \
    --controller=remote,ip=127.0.0.1
```

➤ Opens a new shell:

```
mininet>
```

➤ Trying to ping:

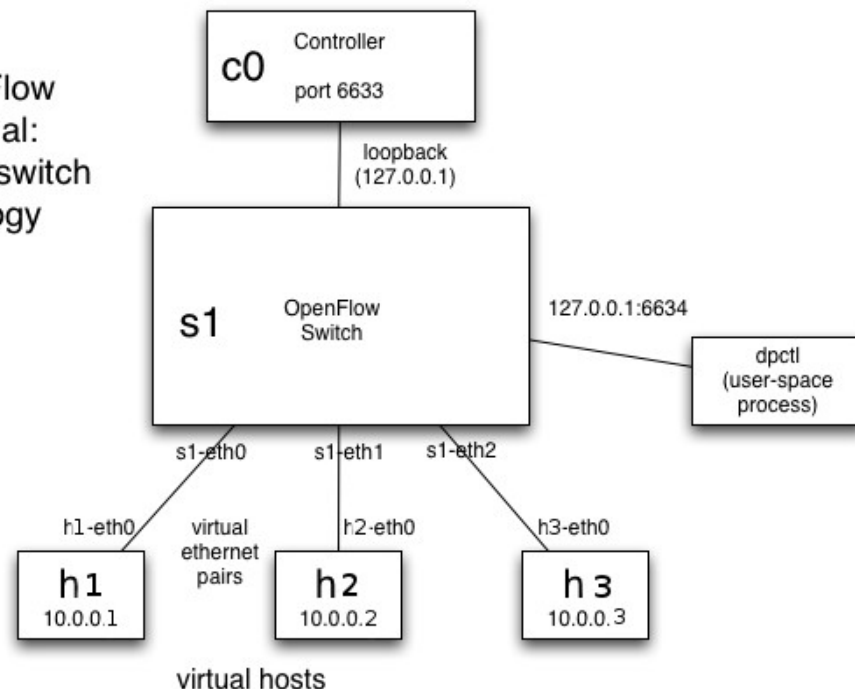
```
mininet> h1 ping 10.0.0.2
```

➤ Will not succeed.

Source of Initial Topology:

<http://pages.cs.wisc.edu/~agember/sdn/session1>
with a correction of a mistake.

OpenFlow
Tutorial:
3hosts-1switch
topology



Mininet: dpctl

- ❖ Tool for manipulating flow rules on a particular OpenFlow switch, preinstalled on the Mininet VM
- ❖ View current rules on our switch s1 with:

```
$ s1 dpctl dump-flows tcp:127.0.0.1:6634
```

- ❖ Enter:

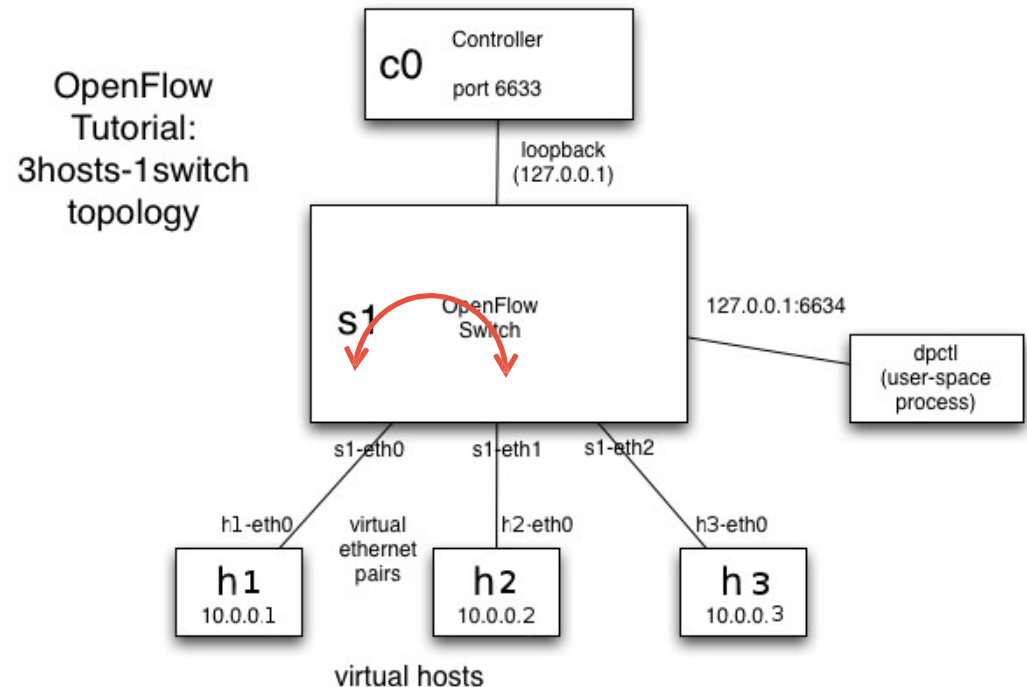
```
$ dpctl --help
```

for a list of commands, or enter „man dpctl“ into Google.

- ❖ **Advice:** Get yourself familiar with the commands.

Mininet Example Task: Passthrough

- ❖ The OpenFlow switch shall pass every packet from h1 to h2, and vice versa.
 - No matching necessary



Mininet: Passthrough - Solution

- ❖ Log in via another SSH session (Session 2)
 - E.g. open a new PuTTY window (Alternative: use **screen**)
- ❖ On the switch s1, create two flows:
 - Whatever enters Port 1 (h1) should leave Port 2 (h2)
 - Whatever enters Port 2 (h2) should leave Port 1 (h1)
- ❖ For that, use dpctl:

```
$ s1 dpctl add-flow tcp:127.0.0.1:6634 \  
in_port=1,idle_timeout=0,actions=output:2
```

```
$ s1 dpctl add-flow tcp:127.0.0.1:6634 \  
in_port=2,idle_timeout=0,actions=output:1
```

```
$ s1 dpctl dump-flows tcp:127.0.0.1:6634
```

(idle_timeout=0
prevents flows from
being deleted after
60 seconds)

Mininet: Passthrough - Solution

- ❖ In the first SSH session (with the mininet prompt), try to ping between both hosts again:

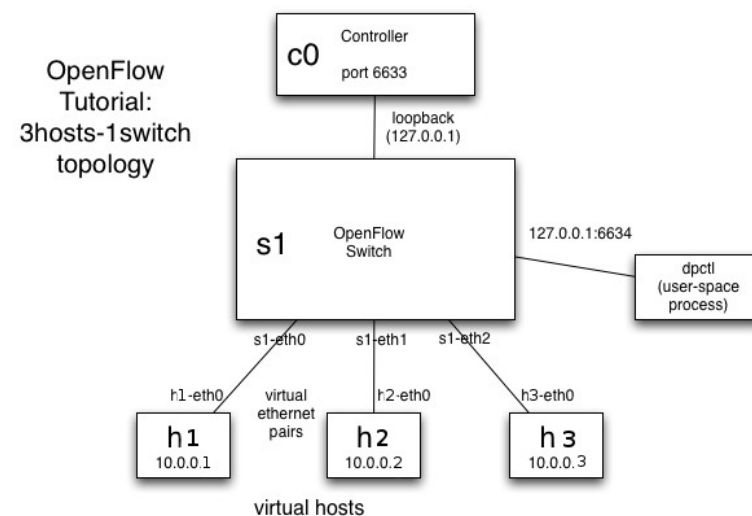
```
mininet> h1 ping 10.0.0.2
```

```
mininet> h2 ping 10.0.0.1
```

- ❖ It should work now.
- ❖ **Congratulations**, you software-defined your first network!

Lab 1 Task 1: Layer 2 Bridge

- ❖ Define Layer 2 OpenFlow rules, so that all three hosts are able to ping each other.
- ❖ Use MAC address matching
 - You can „hard code“ the MAC addresses
- ❖ Broadcast broadcast frames, and unicast unicast frames. For performance reasons, do not broadcast everything!



Lab 1 Task 2: „Spam Filter“

- ❖ Spam is a serious problem today. If a client is part of a botnet, it may send tons of spam out via SMTP
 - The three hosts should **not be able** to exchange **SMTP traffic**. For that, explicitly filter SMTP traffic.
 - Building on Task 1, define appropriate rules to match and drop SMTP traffic, while not harming other traffic.

Lab 1 Tips (1): Scripting

- ❖ After restarting Mininet, your entered OpenFlow rules will be lost. So create a shell script:

```
$ nano create-lab1-rules.sh
```

- This opens the editor **nano**, editing the shell script with the filename above (feel free to use another editor).
- Enter your Unix shell commands (e.g. `dpctl`), each in a new line.
- Save with **Ctrl+X**, confirm saving with **Y**
- Run with:

```
$ sh create-lab1-rules.sh
```
- This will execute all commands in the script.

Lab 1 Tips (2)

- ❖ dpctl Example: Forward packets matching a destination MAC address:

```
$ s1 dpctl add-flow tcp:127.0.0.1:6634 \  
dl_dst=11:22:33:44:55:66,idle_timeout=0,\  
actions=output:2
```

- ❖ Get yourself familiar with Layer 2 bridging/broadcast and the behavior of the **ARP** protocol, which precedes L3 communication.

Lab 1 Tips (3): Tools

- ❖ tcpdump will output all frames entering or leaving a host. Google „man tcpdump“ for more advanced usage.

```
mininet> h1 tcpdump
```

- ❖ Get yourself familiar with Layer 2 bridging/broadcast and the behavior of the **ARP** protocol, which precedes Layer 3 IP communication.

Lab 1 Tips (4):

You can login with a second terminal window like this:

- `ssh mininet@192.168.56.101`

```
yourPC$ sudo mnexec -a *PID* bash
```

How to find the PID:

- In the terminal where you started mininet:

```
mininet> py h1.pid
```

- Alternatively you can enter: "xterm h1 s1" to start for example two terminals connected to h1 and s1.
However, you have to use the '-X' option on ssh and to use Linux.

- ❖ The Layer 2 rules you have defined should be submitted as a **shell script** to Moodle.
 - It should contain the appropriate rule definitions (e.g. via `dpctl`) to program the switch `s1` with the desired behavior.
- ❖ After creating the initial topology **and** running the shell script you have submitted, the setup should show the desired behavior.
- ❖ It is sufficient if you submit an all-in-one shell script containing the behavior of both Task 1 and Task 2.

Good Luck!