CHALMERS
UNIVERSITY OF TECHNOLOGY

Distributed Computing and Systems
Chalmers university of technology

Distributed Systems I

Lab Introduction

TAs

Beshr Al Nahas

beshr

Charalampos (Babis) Stylianopoulos

chasty

Christos Profentzas

chrpro

Valentin Poirot

poirotv

@chalmers.se

Lab time: Monday 15:15-17:00, Thursday 08:00-9:45
Ask your questions during the lab time!

Lab Introduction

- Lab Introduction I (Nov. 2):
  - Labs overview, tools we use, Lab 1 (1h)
  - Introduction to RESTful, Python skeleton (1h)

- Lab Introduction II (Nov. 9):
  - Lab 2 introduction
  - Solution costs
  - Concurrency in Python

Labs Presentation

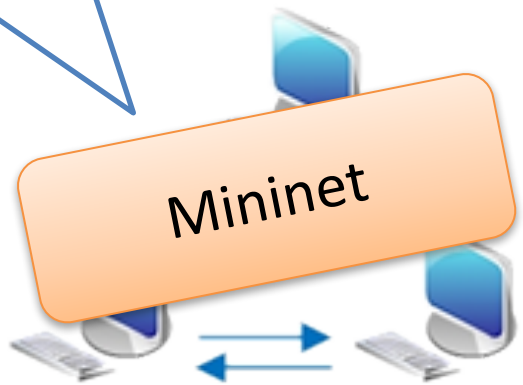
Labs in a nutshell

Lab overview

You will build an application over this network

During the labs you will make this application:

- Reliable
- Consistent
- Efficient
- Fault-tolerant
- …



Mininet

You will use a platform for emulating computers around the world

Labs in a nutshell

- Design and implement a distributed system
- **RESTful distributed blackboard**
  - Restful (web) Clients send notes to any server
  - Servers do distributed systems magic to provide a reliable, consistent, efficient, fault-tolerant service
    - You will learn the required distributed systems magic with Olaf and practice it in the lab exercises

How we will do it

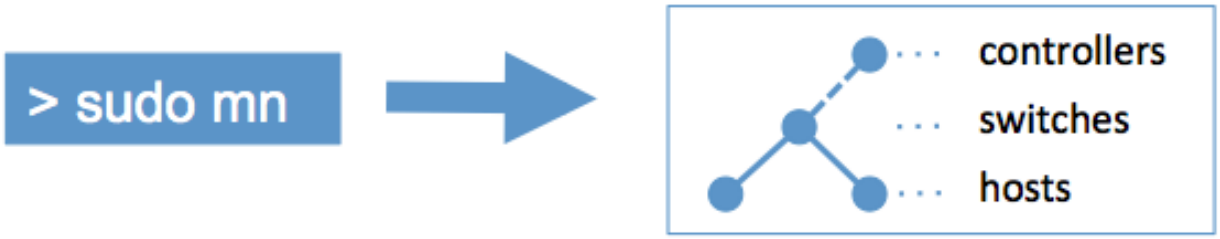
- Incremental steps
- Explore different design choices:
  - Lab1 – *naïve* - make it work ☺
  - Lab2 – *centralized* - strong consistency
  - Lab3 – *leaderless* - eventual consistency
  - Lab4 – *resilient* to malicious servers

The platform: Mininet

Mininet

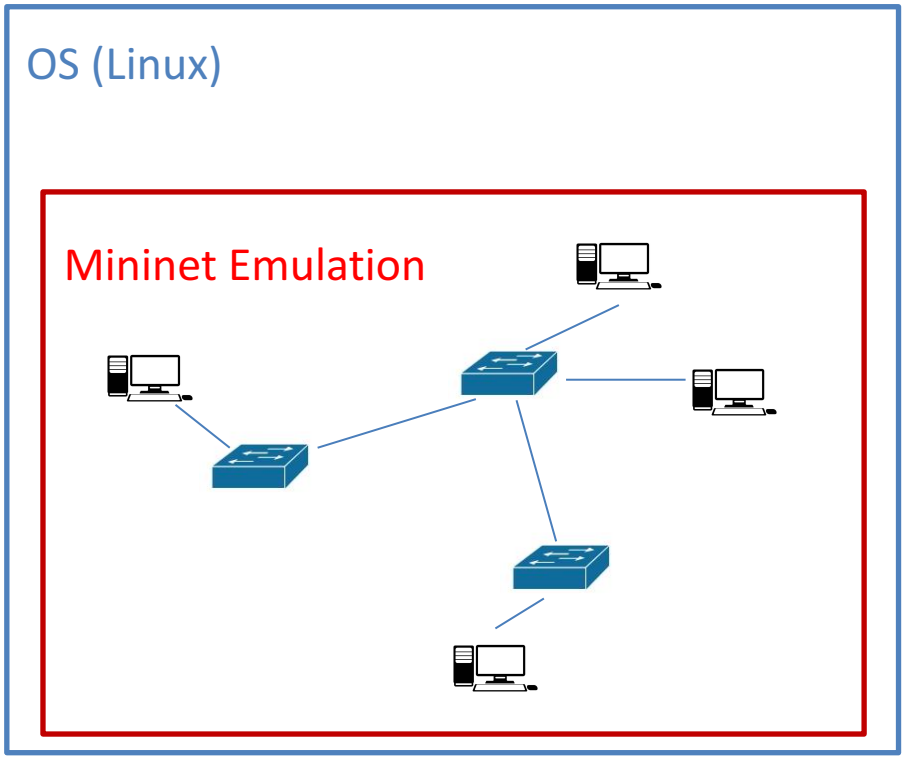
An Instant Virtual Network on your Laptop (or other PC)

- Network Emulator
  - Emulate hosts (machines), switches, controllers, and links
  - On one PC
- Used in research
- Handles large scale networks



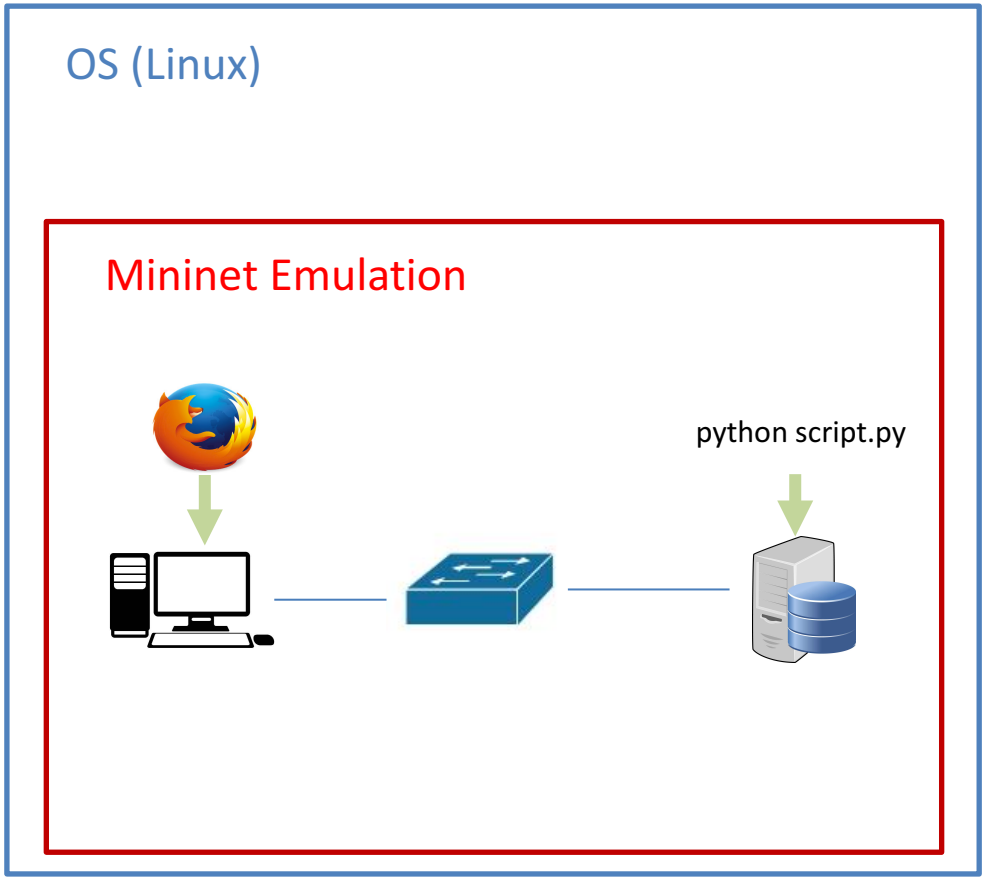
Mininet: How does it work?

- Container-based virtualization
- Node Emulated as a process on PC
  - Can not access the other processes locally
  - But on the Emulated network
  - Uses the host network stack

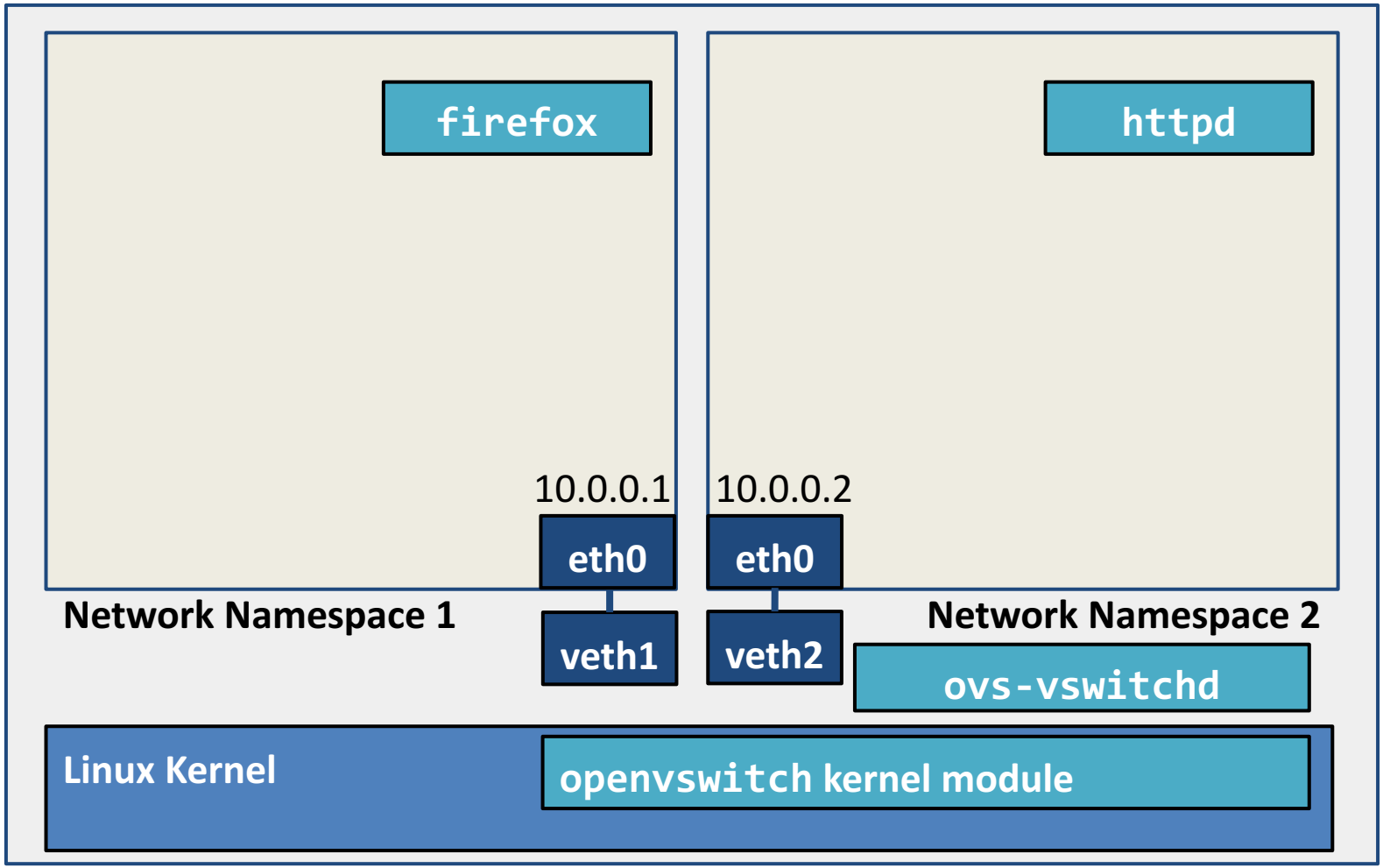


Mininet: How does it work?

- Each simulated machine can be seen as a Linux machine
  - We can run any program on a simulated machine
    - shell commands, web browser, …
  - Every machine = Linux process



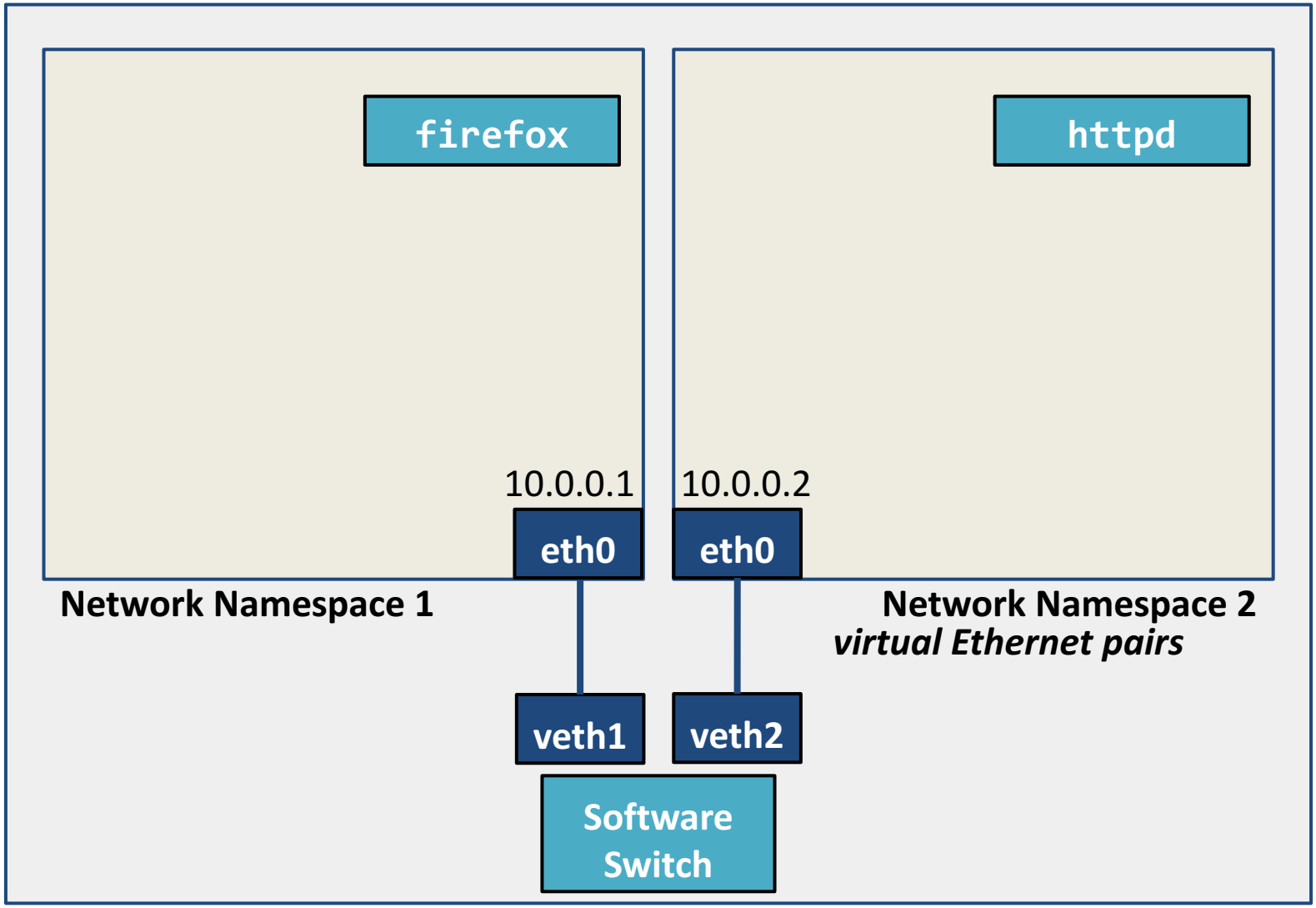
Very Simple Network using Lightweight Virtualization



Server (or VM!)

Credits: SIGCOMM 2014 tutorial

Mechanism: Network Namespaces and Virtual Ethernet Pairs



Credits: SIGCOMM 2014 tutorial

Some basic commands

- Running tests
  - pingall
  - iperf
  - h1 ping h2
  - h1 ifconfig
  - *hostname shell_command*
- Stopping the simulation
  - *exit*
- You network crashed?
  - sudo mn -c
    - Will clear the mininet config files
    - Might become useful if you try to do your own script!

Mininet example topologies

- Run predefined technologies:
  - sudo mn --topo linear,k=4,n=2
  - sudo mn --topo tree,depth=3,fanout=2
- Use specific parameters

  sudo mn -- topo linear, 3 --link tc, bw=10, delay=10ms, jitter=5ms
  - --link specifies the link attributes
  - tc uses the TCLink classes, which has modifiable attributes
  - bw (bandwidth), in Mbps
  - delay, jitter, *must* include `ms´!

Some useful links

- How to install mininet http://mininet.org/download/
- Walkthrough from the basic commands to custom scripts http://mininet.org/walkthrough/
- SIGCOMM 2014 tutorial
  - https://docs.google.com/a/onlab.us/presentation/d/1Xtp05lLQTEFGICTxzV9sQl28wW_cAZz6B1q9_qZBR_8/edit
- Some code examples (advanced): https://github.com/mininet/mininet/tree/master/examples

The tools we use here

The software we use in the labs

Qemu: machine emulator and virtualizer

- Open source
- Mainly through command line
- To start the VM:
  - qemu-system-i386 mininet-vm-i386 -enable-kvm
  - -m 1G to add more memory
  - see the prelab to use the mouse

Mininet in Qemu

- If you want a desktop, install xinit and lxde
- Or just use the X display:
  - add *-redir tcp:2222::22* when launching qemu
  - from the machine's terminal, use *ssh -Y mininet@localhost -p 2222*
- You should install Firefox (or anything else) on the VM

Administrative stuff

How to happily pass the course and make us happy

Lab deadlines

- **Pre-assignment**: November 6, 23:59
- **Lab 1**: November 13, 23:59
  - Demo: Nov. 9 & 13
- **Lab 2**: November 23, 23:59
  - Demo: Nov. 19 & 23
- **Lab 3**: December 7, 23:59
  - Demo: Dec. 3 & 7
- **Lab 4**: January 7, 23:59
  - Demo: Dec. 14

Demo your solution to us before submission

Hard deadlines: Up to 2 weeks late → penalty

Lab logistics

- 4 Labs x 10 points each = 40 points
  - Each lab has several tasks
- PASS = 31/40 points

- Late submissions:
  - within 1 week after the deadline
    → -1 point from your score on that lab
  - within 2 weeks after the deadline
    → -3 points from your score on that lab
  - Not accepted after 2 weeks

Hand in

- Code
  - Well structured
  - Well documented

- Results
  - In a video
    - 2-4 minutes screencast to demonstrate your results
    - Some screencast software: http://en.wikipedia.org/wiki/Screencast
    - Screencast software in Lab rooms: RecordMyDesktop
  - **Or** in a report (as a pdf file)

Do you favor writing over video?

- Include in the report the same information as in the video

- That is
  - Document that your solution works with screenshots and explanations
  - **All stages of the execution** should be screenshot and included in your documentation

Grading

- 10 points for the mandatory tasks
  - Solution -4 points
  - Code structure -2 points
  - Code documentation -2 points
  - Report (video or written) -2 points
  - You need to provide all the parts above
    - Do not submit code without a video or written report
  - The points represent the penalty if we do not like something
- Bonus tasks give bonus points

Code Structure and Documentation

- *“Code is written primarily to be read by humans. It has to be acceptable to the compiler too, but the compiler doesn’t care about how it looks or how well it is written.”*

- In your professional life, you will mostly use, fix and improve code that exists...

Code Structure and Documentation: Guidelines

- Descriptive variable names
  (‘a’, ‘b’, ‘apa’ are not descriptive…)
- Comment blocks of code that are doing something subtle or that is not obvious.
- Document **what** each function does (not how), arguments, returned values, side effects etc. (see Python API).
- If you can’t do this in a few sentences …
  - you may need to rethink your abstraction.

The task – external slides

LAB 1

Questions?