CHALMERS
UNIVERSITY OF TECHNOLOGY

Distributed Computing and Systems
Chalmers university of technology

Distributed Systems I

Lab Introduction

TAs

Christos Profentzas **chrpro@chalmers.se**
Maria A. Romero mariaag@student.chalmers.se
Erik Bergsten erikbergsten94@gmail.com
Badi A. B. Iskhandar azzarfan@student.chalmers.se

Labs in a nutshell

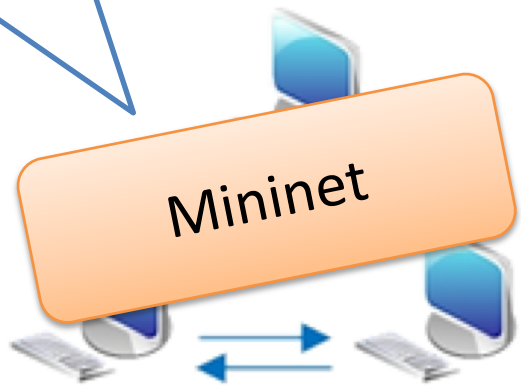
Lab overview

You will make this application:

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

You will build an message blackboard over this network

Mininet



Mininet is a simulation tool

Labs in a nutshell

- **RESTful distributed message board**
  - Restful (web) Clients send messages to any server
  - Servers are distributed systems with reliable, consistent, and fault-tolerant service

How we will do it

- Group of at least 2 (max 3) persons (strict rule)
- Incremental steps:
  - Lab1 – *naïve* - make it work ☺
  - Lab2 – *centralized* - strong consistency
  - Lab3 – *leaderless* - eventual consistency
  - Lab4 – *fault tolerance – some* servers fail

The tools we use here

The software we use in the labs

Virtual Machines

- You can download our VM

  or

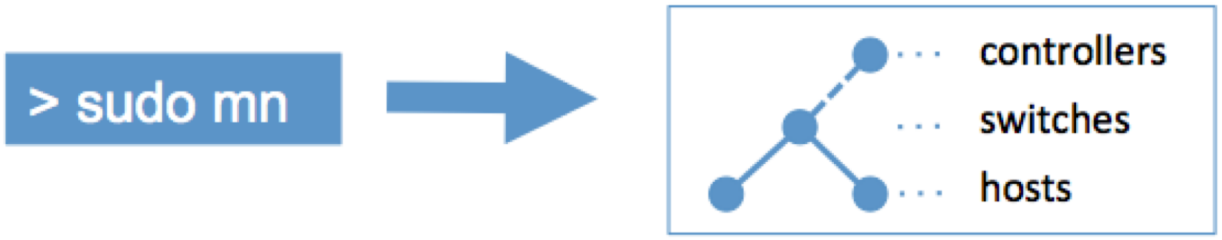
- You can install the software local (no support from TA)

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



Some basic commands

- Running tests
  - pingall
  - xterm node1 (terminal to node 1)
- Stopping the simulation
  - *exit*
- You network crashed / lab1.py errors?
  - sudo mn -c
    - Clear the mininet config files

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

On the code side

- Python 2.7
- Any web server you like, we recommend Bottle (Flask is fine too)
- We are giving you a simple skeleton, that you can work on to create lab 1

Hand in

- Code
  - Well structured
  - Well documented
- Demo
  - Video 5-10 minutes screencast to demonstrate your solution

Lab deadlines

- **Lab 1**: November 15
- **Lab 2**: November 29
- **Lab 3**: December 13
- **Lab 4**: January 3

CHALMERS
UNIVERSITY OF TECHNOLOGY

Distributed Computing and Systems
Chalmers university of technology

Distributed Systems I
Lab Introduction - part 2

API: What, Why?

- API = Application Programming Interface
- Allows you to quickly add functionality/data that others have created.
- Allows frontend developers and backend developers to agree on a common interface

Functions

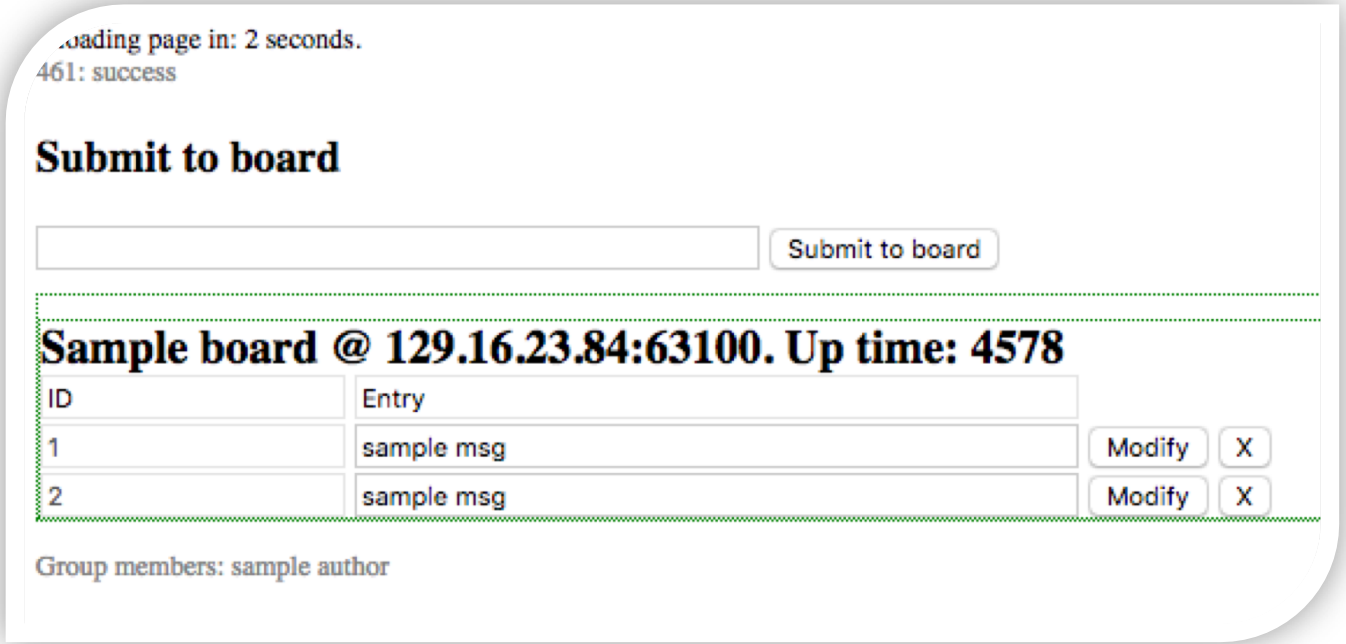
- View the board
- Add a new entry
- Delete an entry

An example API

- GET /board
- POST /entries
- DELETE /entries/entryID

The Web Browser as a GUI

- Web applications need integration between client side (HTML/HTTP) and server side



The (distributed) board API

- Each function has a name and parameters
- REST: HTTP method + URL

Functions	Example API	Parameters	Returns
View the board's contents	GET /board	None	The whole board start page : html
Retrieve entries only	GET /entries	None	List of available entries (not the full page) : html
Add a new entry	POST /entries	entry : text	Status
Retrieve one entry	GET /entries/entryID	None	The entry : html
Modify an entry	PUT /entries/entryID	entry : text	Status
Delete an entry	DELETE /entries/entryID	None	Status

Sending DELETE and PUT requests

- HTML forms supports only **GET** or **POST** requests (No **DELETE** or **PUT**)
  - Use JS to send the request
  - Or for the sake of this course, change the API to use GET or POST
    - Use extra **parameters**

Functions	API	Parameters	Returns
Add a new entry	POST /entries	entry : text	Status
Modify an entry	PUT /entries/entryID	entry : text	Status
Delete an entry	DELETE /entries/entryID	None	Status
Modify or Delete an entry	*POST /entries/entryID*	entry : text *delete: logical*	Status

Python 2.7

BOTTLE: PYTHON WEB FRAMEWORK

Bottle API

```
from bottle import Bottle, run
app = Bottle()

@app.route('/hello')
def hello():
        return "Hello World!"

run(app, host='localhost', port=8080)
```

A POST example

```
@app.post('/board')
  def client_add_received():
      new_entry = request.forms.get('entry')
       Do_something(new_entry)
```

A GET example

```
@app.get('/board')
    def get_board():
        return
template(‘board.tpl',board_title=‘title')
```

References

- *API Crash Course*, Patrick Murphy at CWU Startup Club, http://cwustartup.com/APICrashCourse.pptx
- *Building Web Services the REST Way*, Roger L. Costello, http://www.xfront.com/REST-Web-Services.html
- *REST Architecture Model: Definition, Constraints and Benefits*, Ricardo Plansky, http://imasters.expert/rest-architecture-model-definition-constraints-benefits/
- *API Integration in Python - Part 1*, Aaron Maxwell, https://realpython.com/blog/python/api-integration-in-python/
- https://en.wikipedia.org/wiki/Representational_state_transfer
- http://www.w3schools.com/html