Introduction to the labs

Andrea Bianco, Paolo Giaccone
http://www.telematica.polito.it/

Labs

- Lab #1
  - 3 hours
  - Configuration
  - SDN and network routing
- Lab #2
  - 3 hours
  - Performance
  - QoS support: mainly scheduling
- Lab #3
  - 3 hours
  - Detailed analysis and implementation
  - Simulation of algorithms
- It is possible to work on the labs at home
  - assistance is provided only in presence during the lab

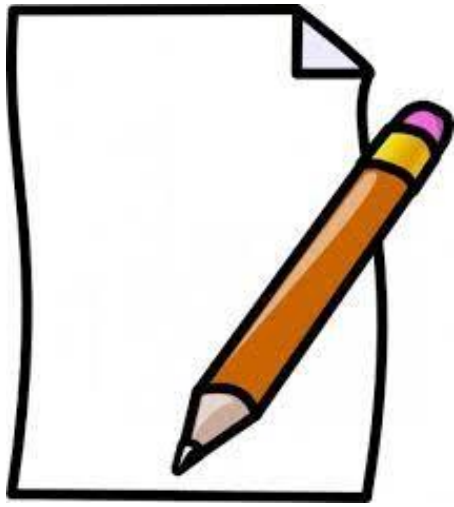
- **The exam starts with a question on the lab!**

Lab logistics

- LED2 - Dept. of Electronics and Telecommunications
  - 2nd floor south ”scavalco” on C.so Castelfidardo
  - https://www.polito.it/mappe?bl_id=TO_CIT11&fl_id=XP02&rm_id=065
- Wednesday 16:00-19:00 in LED2
  - November 19th, December 3rd, December 10th
- Please arrive 5 minutes earlier, so you can start the lab on time
- Join your group as listed here
  - https://docs.google.com/spreadsheets/d/1LOV8OxxxNcgDrU_v1TxjosKnLoRt8vxEPNXNrOCFZFY/edit?gid=0#gid=0
- Will use crownlabs
  - Working in a virtual environment
  - Using a Linux Virtual Machine
  - Using Linux terminal and shell commands
    - (see notes available as teaching material)

Lab detailed instructions

- 3 pdf files are provided, one for each lab
- Print the pdf at home and bring it into the lab
  - Better one copy for each student
- Bring pen and papers to take notes
  - Required to be able to follow the lab
  - Needed to discuss labs during examination
  - Notes can be taken on tablet or PC



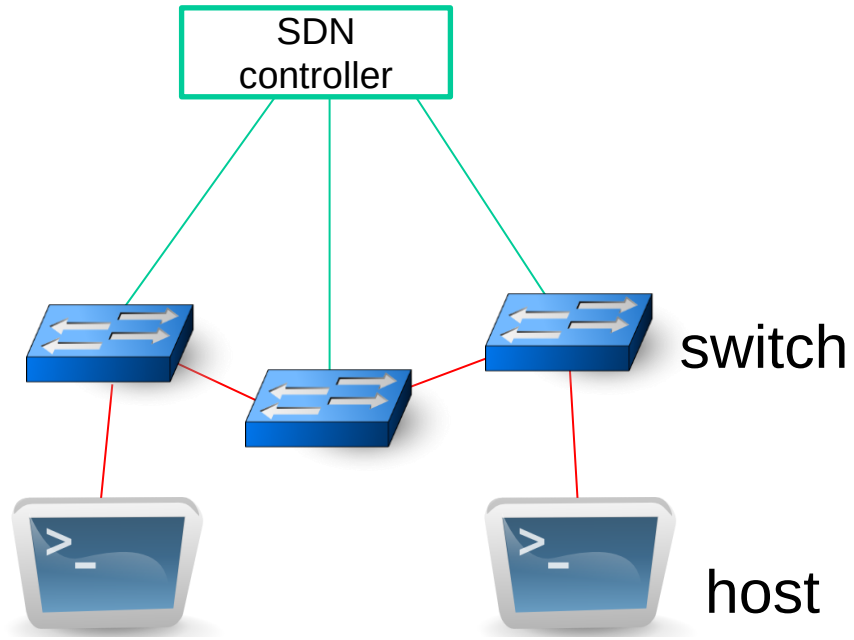
Software tools

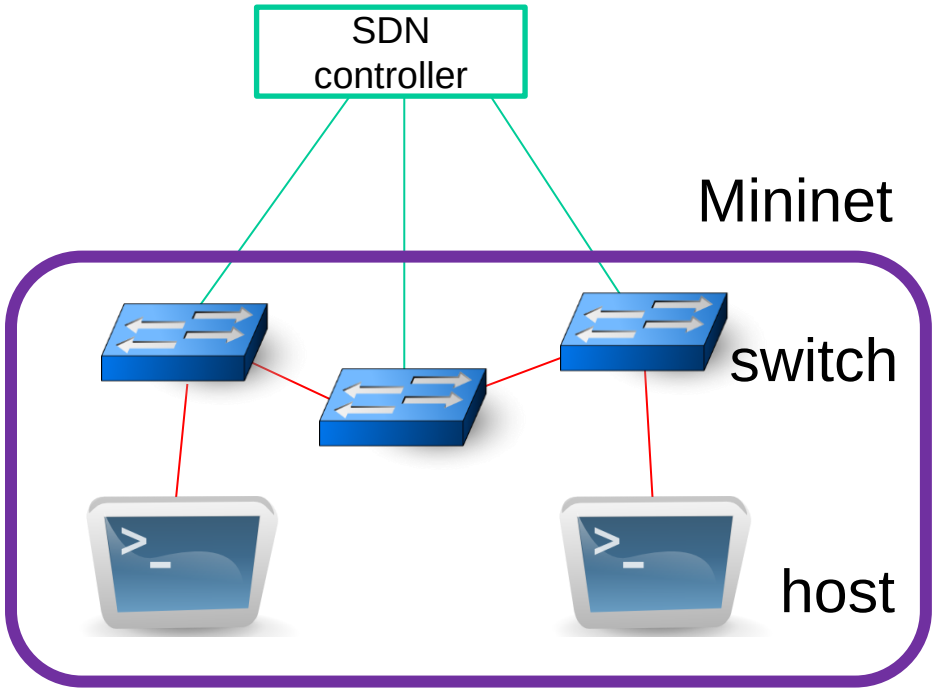
- In crownlabs VM
  - Linux shell commands
  - Browser
  - Text editor
  - Linux network applications
    - Traffic generators

- For the first two labs
  - Main software tool: Mininet
  - SDN Network emulator

Mininet

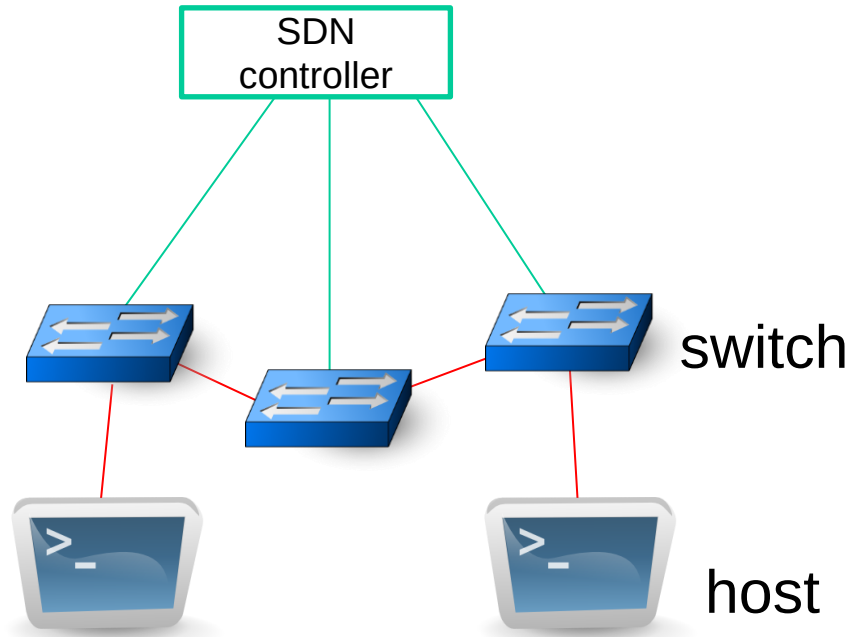
- Network emulator



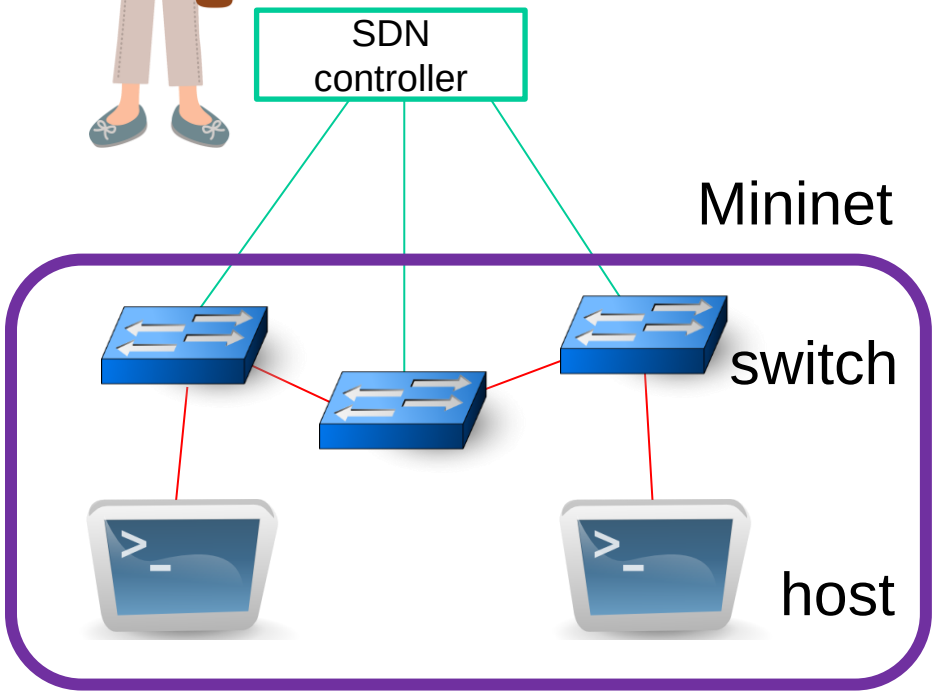


Mininet

- Network emulator



you will act as an SDN controller



Mininet

- It is a SDN emulator
  - Permits to create a topology containing nodes
    - host
    - SDN switch
- Linux container/process for each node
- Command line interface CLI
  - global commands for the emulator
  - local commands to access/configure nodes

Mininet global commands

- In the selected/created topology
  - **nodes** display nodes in the current topology
    - h1 h2 s1
    - h1 -> host 1
    - s1 -> switch 1
  - **links** display links
    - h1-eth0<->s1-eth1
  - **net** display a summary of all the nodes and links
    - h1 h1-eth0:s1-eth1
  - **dump** dump information about all nodes
    - <Host h1: h1-eth0:10.0.0.1 pid=123456>

Mininet local commands

- local commands for the nodes
  - if the first typed string is a host, switch or controller name, the command is executed on that node
- **h1 ifconfig** provides the list of the network interfaces attached to h1
- **h1 ping h2** sends ICMP packets from h1 to h2
- **h1 iperf3 –c 10.0.0.1** uses iperf to test the bandwidth towards 10.0.0.1

- **sh** allows to run a (Linux) command outside mininet, while mininet is running; e.g., mininet> **sh ls**

Network monitoring/performance tool

- Ping test host reachability
  - Exploits ICMPs echo (request and reply) messages
  - Measures the average round trip time

  - PING www.ietf.org (64.170.98.32) 56(84) bytes of data.
  - 64 bytes from mail.ietf.org (64.170.98.32): icmp_seq=0 ttl=66 time=169 ms
  - 64 bytes from mail.ietf.org (64.170.98.32): icmp_seq=1 ttl=66 time=172 ms
  - 64 bytes from mail.ietf.org (64.170.98.32): icmp_seq=2 ttl=66 time=174 ms
  - 64 bytes from mail.ietf.org (64.170.98.32): icmp_seq=3 ttl=66 time=177 ms
  - --- www.ietf.org ping statistics ---

  - 4 packets transmitted, 4 received, 0% packet loss, time 3008ms
  - rtt min/avg/max/mdev = 169.290/173.427/177.398/2.994 ms, pipe 2

Network monitoring/performance tool

- **iperf3** test the available bitrate between two hosts
- client server application
  - client: generates the traffic (TCP/UDP)
  - server: receives the traffic (TCP/UDP)

- **iperf3 –c dest_IP** run the test as client
- **iperf3 –s** run the test as server
  - by default, each host is already running it in background
- many options are available: **iperf3 --help**

Initial steps

- Navigate to crownlabs: https://crownlabs.polito.it
- Select Login @Polito
- Select PoliTOSSO at the bottom
- Use your University credentials
- Select the Computer Network Design workspace
- Create the VM instance
- Connect to the VM instance
- …
- Execute lab
- ...
- STOP and DELETE the VM instance at the end (transferring data if needed)!!!

Lab#1 – Initial step

- **Proceed step by step. Do not skip!**

- Shell command overview
  - Shortcuts

- Become familiar with Mininet and its commands
  - two hosts and one (software) Openflow switch
  - understand the performance of software switches

Lab #1 – Single switch

- Topology discovery
  - in terms of node graph and IP addresses
- Test connectivity
- Add the proper match-action rules in the flow table to route the traffic
  - based on source port only
  - based on destination port only
- Observe the traffic through **tshark**

- Remind the notation: IP 1.2.3.4/24 = 1.2.3.4/255.255.255.0

Lab #1 – Linear two switches

- Topology discovery
  - in terms of node graph and IP addresses
- Test connectivity
- Fix the routing

Lab #1– Mesh topology

- Topology discovery
  - in terms of node graph and IP addresses
- Test connectivity and fix the routing
- Multipath routing
  - route UDP and TCP flows between the same pairs of hosts on different paths
- Dynamic routing
  - Flow rerouting from main path to a backup
  - “flow-mod messages” issued “by the controller”
    - investigate whether the sequence of the messages matters or not for a completely transparent rerouting process (i.e., no losses)
- Fault-tolerant rerouting (Optional)
  - detect a link failure and apply the backup path

Lab #2

- Analyze performance of various schedulers
  - FIFO, Round Robin, Weighted Round Robin, Priority
- In different scenarios
  - Underload and overload
  - Two flows
    - single bottleneck
  - Multiple flows
    - Dingle bottleneck
    - Mesh network
- Compare with max min fair solution
- Have a look at transient behaviour (optional)

Lab #3

- Analyze various schedulers
  - RR, DRR, WRR; Virtual Clock, WFQ …
- Interactive python environment
- Need to set up some parameters to understand scheduler behaviour
- Write some code to customize scheduler behaviour