KAIST

[EE323] Assignment #3

Router Implementation

TA : Dohyun Kim, Kwangsoo Park
(EE323TA@gmail.com)

2020.05.11

Intro

- **You are going to mimic a “ROUTER”**
  - Given a **static** network topology & routing table
  - No hardware router!, but **software** one !

Intro

- **You will be able to understand below things**
  - How does a router handle **ARP** packets ?
  - When does a router send **ICMP** packets ?
  - How does a router use a **routing table** and send received packets ?

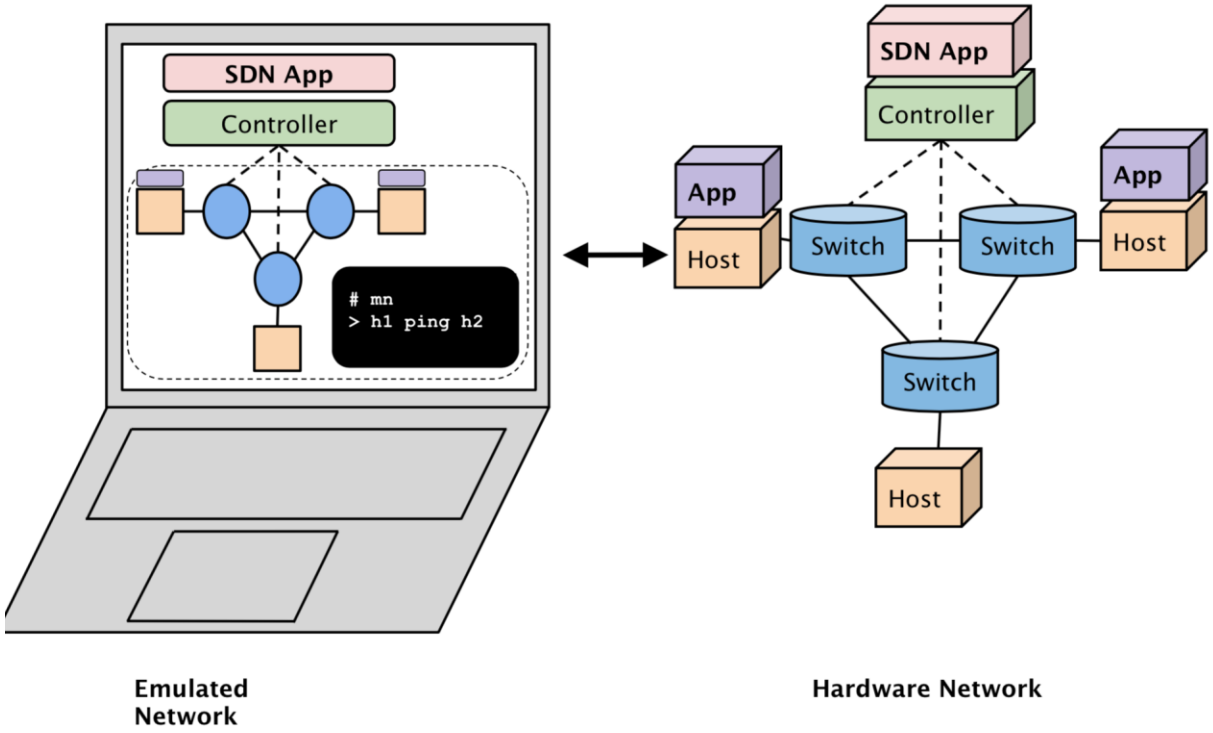
Intro

- **How to do it**
  - Where will my routing logic run ?
  - Where will the traffic come from ?
  - How will I test my code?

We're going to leverage "network emulation"

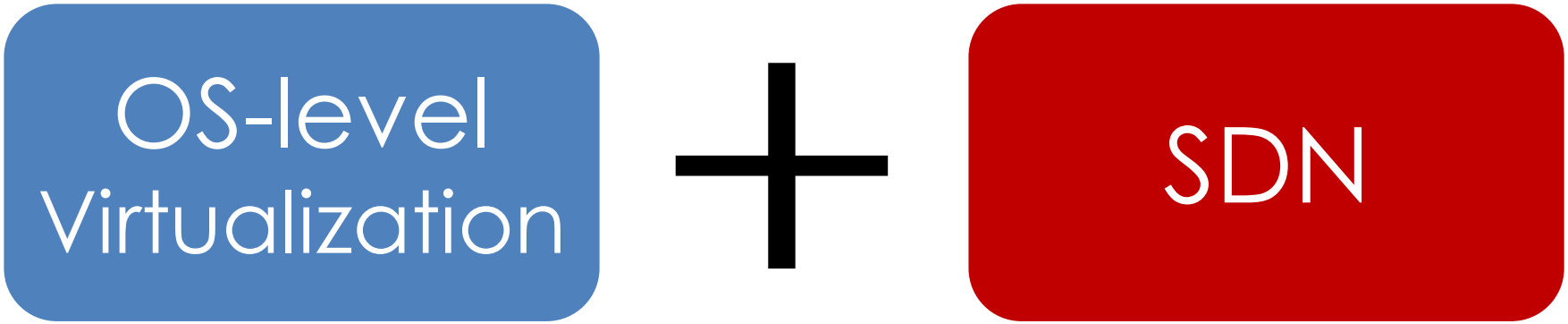
Background

- **Mininet**
  - An awesome network emulation tool



Background

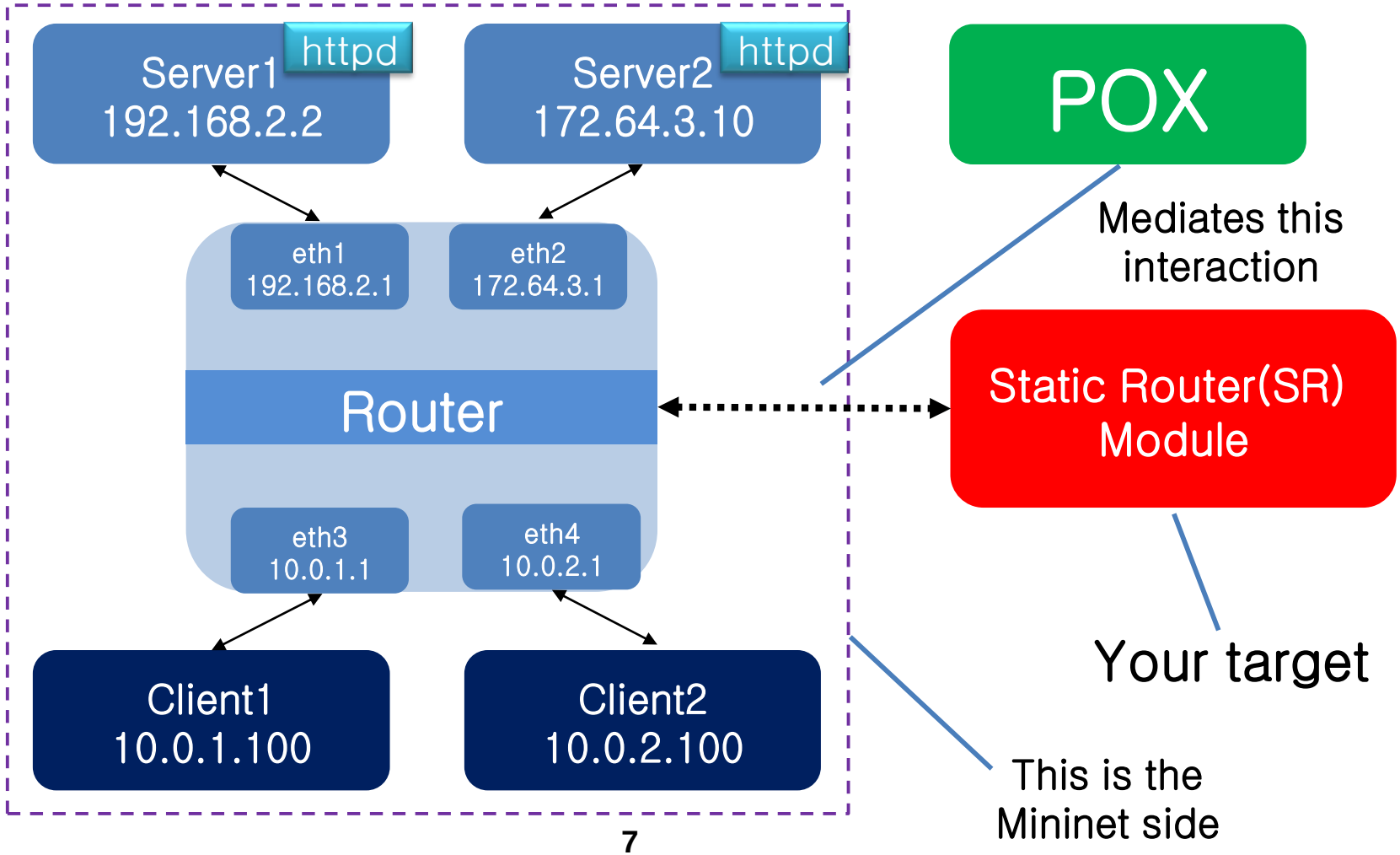
- **Simply run any network you want**
  - With the help of OS-level virtualization and SDN
  - We don't need hardware devices
    - Router, Client PC, Server …
  - You don't need to study Mininet & SDN in this assignment#3



* SDN : Software-defined Networking

Background

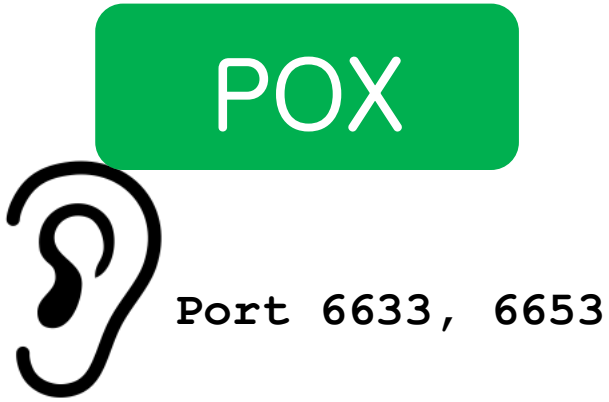
- **The target network architecture**



- **Let’s see what’s happening**

Background

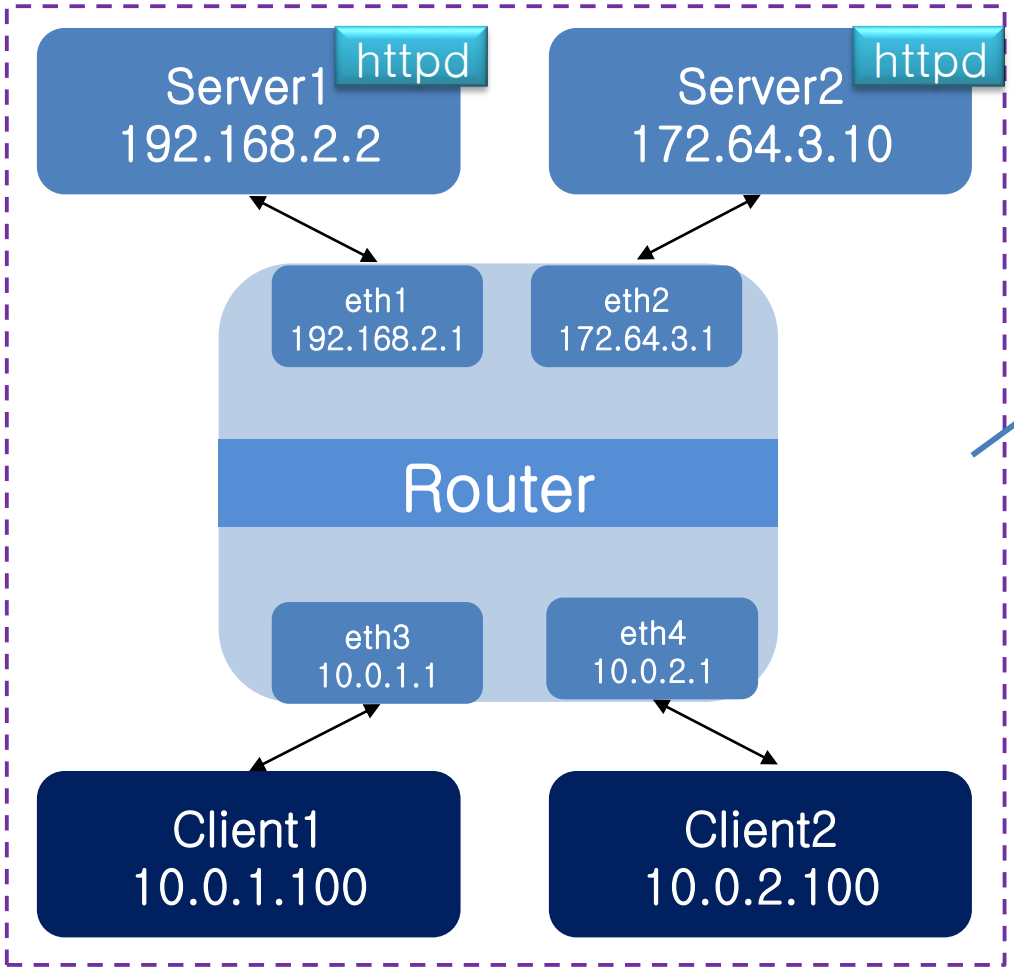
- **./run_pox.sh**

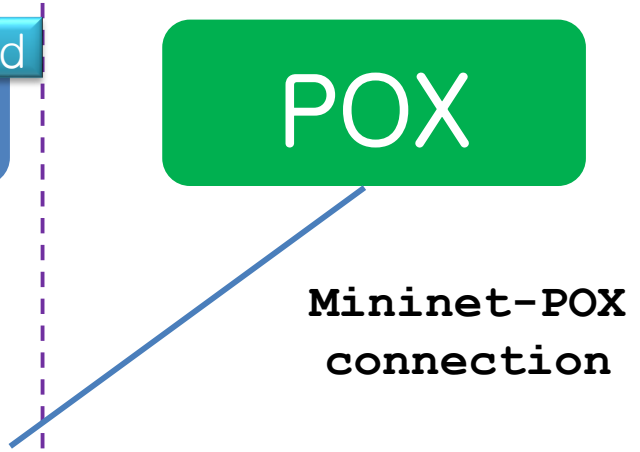


  - When you run POX, it listens port 6633, 6653

Background

- **./run_mininet.sh**

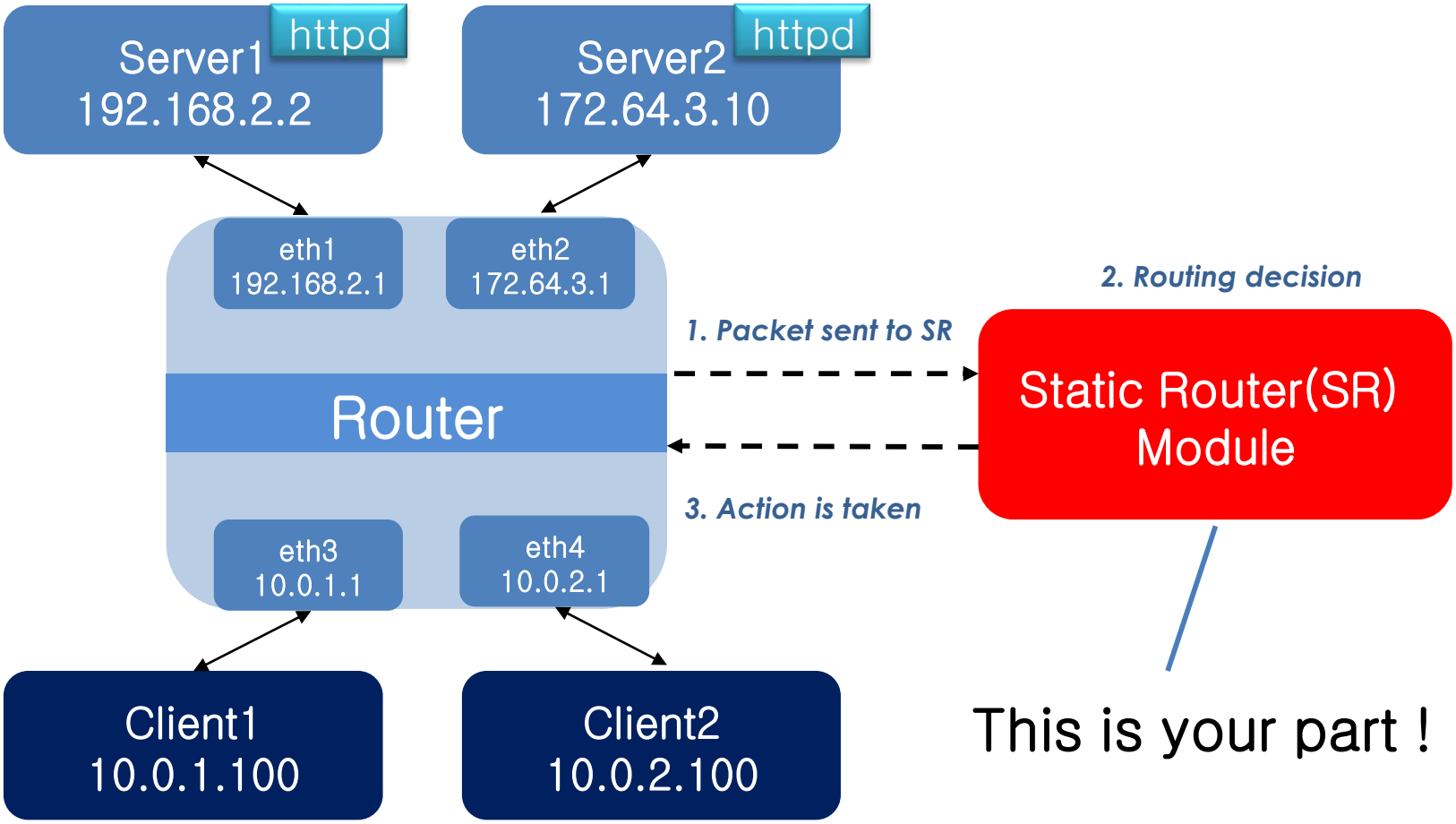




- When you run Mininet, it makes a virtualized network environment
- And **`sw0`** ***attaches*** to either **`port 6633 or 6653`** of POX

Background

- **./router/sr**



Description

- **High-level requirements**
  - You should make sr module
    - sr_router.c
      - ip_black_list()
      - sr_handlepacket()
    - sr_arpcache.c
      - sr_arpcache_handle_arpreq()

Description

- **High-level requirements**
  - Should enable below functions
    - Client1=>router
      - Ping
      - Traceroute
    - Client1=>Server
      - Ping
      - Traceroute
      - Downloading a file using wget (via http://)
    - Client2=>Router, Server
      - Packets blocked (Source ip blacklist)

Description

- **Summary: What your routing logic needs to do**
  - Route Ethernet frames between the client and HTTP servers
  - Handle ARP request and replies
  - Handle traceroutes
    - Generate *Time Exceeded Message*
  - Handle TCP/UDP packets sent to one of the router's interfaces
    - Generate ICMP *Port Unreachable*
  - Respond to ICMP echo requests
  - Maintain an ARP cache

Implementation

- **1. Virtual machine setting**
  - We will use virtualbox(Any OS is okay
  - Download virtualbox 6.0
    - tps://www.virtualbox.org/wiki/Downloads
  - Get VM Image
    - Go to : https://www.dropbox.com/s/quv296teq5kimgj/ee323_assignment3.ova?dl=0
    - Download ee323_assignment3.ova
  - Import image(.ova) using virtualbox and execute
    - Id : ee323 // Pw : ee323
    - Root Pw : root

https://www.virtualbox.org/

Implementation

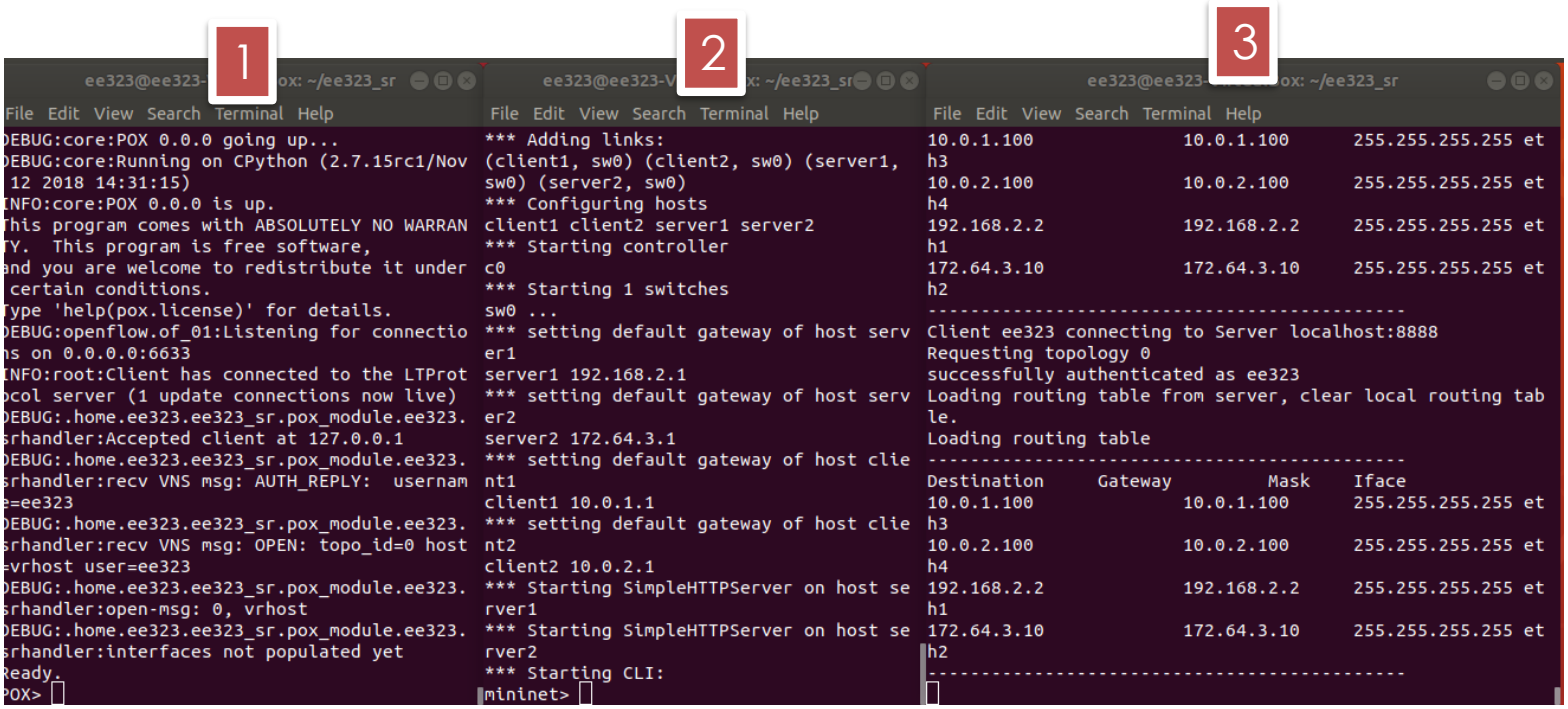
- **2. Program test**
  - $cd ee323_sr
  - $ls

```
ee323@ee323-VirtualBox:~/ee323_sr$ ls
auth_key      http_server2  lab3.py     router          run_pox.sh
config.sh     IP_CONFIG     pox         rtable          sr_solution
http_server1  killall.sh    pox_module  run_mininet.sh
```

  - Create 3 terminals
    - Terminal 1 : $./run_pox.sh
    - Terminal 2 : $./run_mininet.sh
    - Terminal 3 : $./sr_solution
      - Sr_solution is an example program (Not a real solution)

Implementation

- **2. Program test**
  - At terminal 2
    - $ client1 ping 192.168.2.2
    - $ client1 wget http://192.168.2.2
    - $ client1 traceroute server1



Implementation

- **3. When you make program**
  - Execute 3 terminals
    - Terminal1 : $ ./run_pox.sh
    - Terminal2 : $ ./run_mininet.sh
    - Terminal3 : $ ./sr
  - sr?
    - In directory “~/ee323_sr/router”
    - Complete “sr_router.c” and “sr_arpcachec.c”
    - $ make
    - $ ~/ee323_sr/router/sr

Implementation

- **Required Functionality description**
  - –The router must successfully route packets between the Internet and the application servers.
  - –The router must correctly handle ARP requests and replies.
  - –The router must correctly handle traceroutes through it (where it is not the end host) and to it (where it is the end host).
  - –The router must respond correctly to ICMP echo requests.
  - –The router must handle TCP/UDP packets sent to one of its interfaces. In this case the router should respond with an ICMP port unreachable.
  - –The router must maintain an ARP cache whose entries are invalidated after a timeout period (timeouts should be on the order of 15 seconds).
  - –The router must queue all packets waiting for outstanding ARP replies. If a host does not respond to 5 ARP requests, the queued packet is dropped and an ICMP host unreachable message is sent back to the source of the queued packet.
  - –The router must not needlessly drop packets (for example when waiting for an ARP reply)
  - –The router must enforce guarantees on timeouts--that is, if an ARP request is not responded to within a fixed period of time, the ICMP host unreachable message is generated even if no more packets arrive at the router.
  - –The router must block packets from (src IP 10.0.2.* / subnet mask 255.255.255.0) and print log
    - •You must use bitwise calculation

Submission

- You should submit **compressed ‘router’ folder** including makefile and all the source files
  - Submitted file should be YourStudentID_assign3.tar.gz

Tips & Caution

- The most important thing is to understand the assignment
  - **Read the assignment document carefully**!!
- And then, understand source codes
  - There are several header files, source files
- Be careful on the endianness
  - Network byte order and host byte order is different.
  - Be familiar with ntohs(), ntohl(), htons() or htonl()

Question?