TÉCNICO LISBOA

Lab 7: Software Defined Networking (SDN)

Exploring the SDN paradigm.

Goals

1. Configure an *SDN topology*
   with *Mininet* and a *controller*
2. Explore *Reactive* and *Proactive* flow rules
3. Implement a Load Balancer on an SDN topology

Evaluation

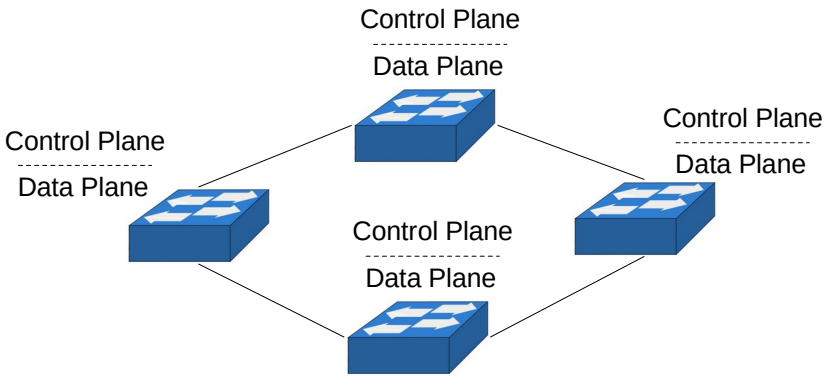
- Where
  - *Moodle: Lab 7: Software-Defined Networking (SDN)*
- Submission due
  - Sunday, December 10, 23h59

The SDN Paradigm

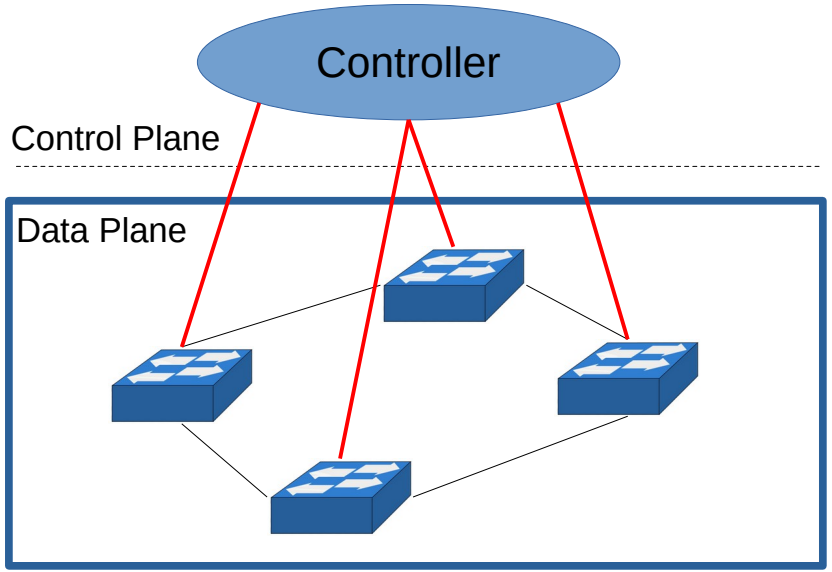
- Separation of the control logic
  from the traffic forwarding elements
  - Decoupling of the control and data planes
- Logically centralized control plane
- Routers/Switches become simple forwarding devices
- Network applications are programmed
  and executed on the controller

The SDN Paradigm

Traditional Networks

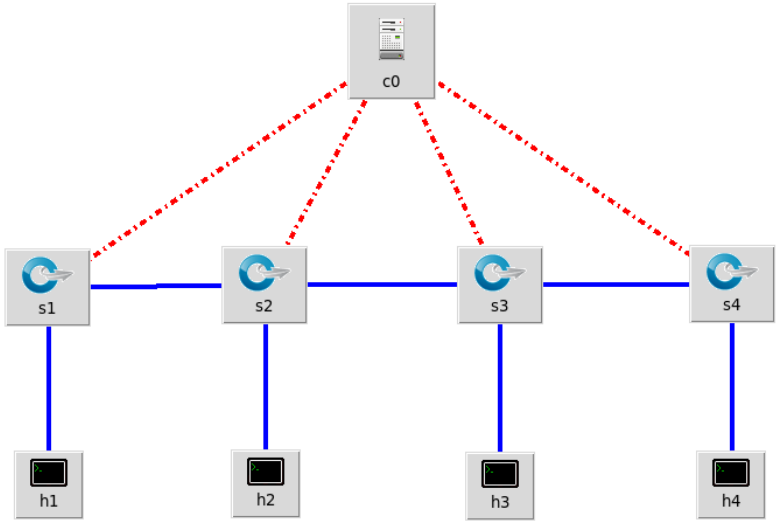


Software-defined Networking



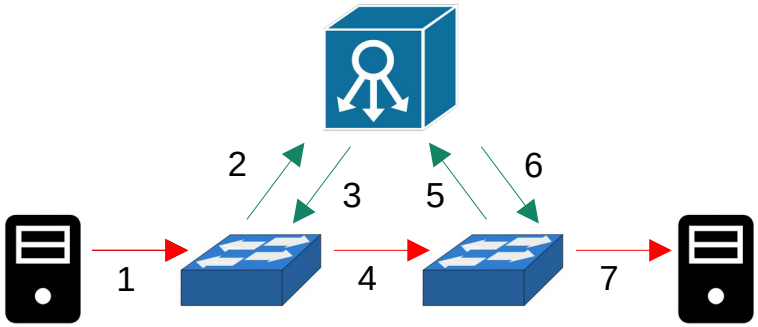
Mininet

- Network emulator that enables the prototyping of SDN topologies
- Supports multiple SDN controllers out-of-the-box: OVS controller, POX, Ryu…
- Extensible Python API

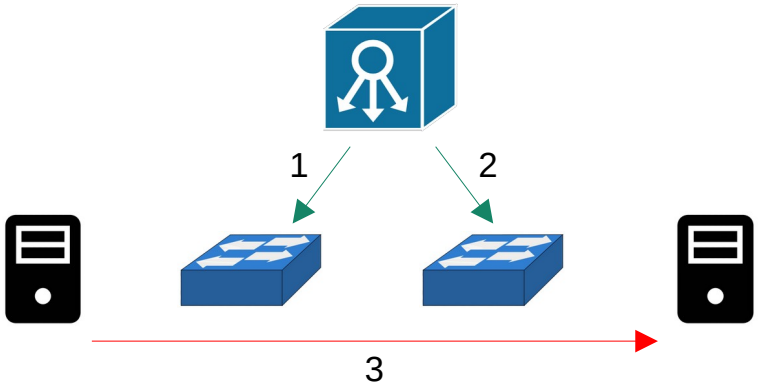


Reactive vs. Proactive Flows

Reactive



Proactive

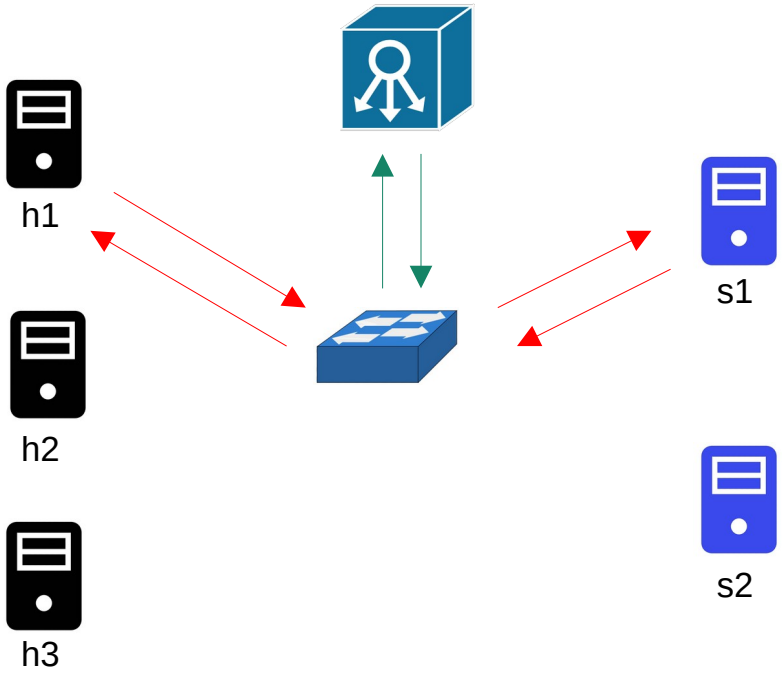


Load Balancing

- Process of distributing traffic across multiple servers
- Goal: optimize the response time
  and avoid overloading any single server
- LB algorithms are widely used
  in large scale network infrastructures

Round Robin

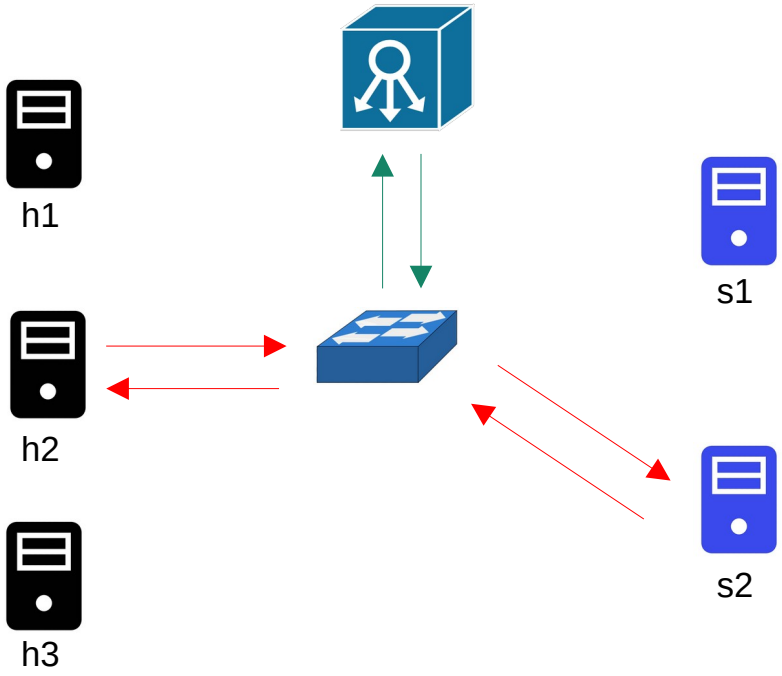
- The client requests are forwarded to each server in turn, following a defined list.
- Whenever the last element is reached, the algorithm goes back to the top of the list.



Server Pool = {**s1**, s2}

Round Robin

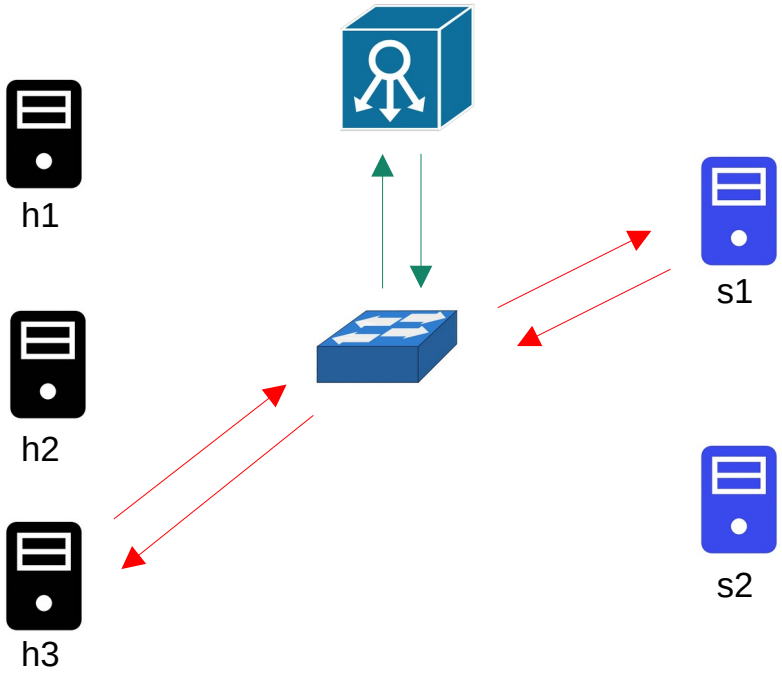
- The client requests are forwarded to each server in turn, following a defined list.
- Whenever the last element is reached, the algorithm goes back to the top of the list.



Server Pool = {s1, **s2**}

Round Robin

- The client requests are forwarded to each server in turn, following a defined list.
- Whenever the last element is reached, the algorithm goes back to the top of the list.



Server Pool = {**s1**, s2}