TÉCNICO LISBOA

Computer Networks — 2023/24, Q2		**Assignment:**	Lab 7
Software-Defined Networking (SDN)		**Version:**	1.0
Issued:	2023-12-04	**Submission Due:**	2023-12-10

Submission note. You will need to submit all the answers to this lab's questions in Moodle, under "Lab 7: SDN".

1 Goals

- Configure an SDN topology with Mininet.
- Explore Reactive and Proactive flow rules.
- Implement a Load Balancer on an SDN topology.

2 Exercises

Important. After pulling the updated lab files, execute the following command in a terminal `/home/rc/lab-files/07-lab-sdn/setup.sh`, in order to install additional required dependencies you'll need during this lab. After the script finishes, you can close the terminal and advance to the first section.

2.1 Mininet

Software-Defined Networking is an approach to computer networks in which the control logic is separated from the forwarding elements, such as routers and switches. By separating the control plane from the data plane, an SDN infrastructure provides increased flexibility and easier network management. Network applications are programmed and executed in the controller, thus abstracting the lower-level interaction with the network routers and switches, which become simple forwarding devices.

An SDN controller is a logically centralized element responsible for relaying the network configuration instructions from the network applications to the forwarding devices. In a sense, each controller functions as a traditional operating system, abstracting the lower-level details of each specific network implementation.

The connection between an SDN controller and the networking hardware in the data plane is defined as the Southbound interface. The most common southbound interface protocol is OpenFlow, allowing the direct access to the data plane (and its manipulation) from a controller.

Mininet is a network emulator (similar to CORE) which allows the creation of a virtual topology of hosts and switches supporting OpenFlow. It can be easily coupled with an SDN controller to perform network simulations.

On this section, you will configure different Mininet topologies and explore the insertion of flow rules using SDN controllers.

The default Mininet command initializes a simple topology with two hosts connected to a single switch, and a controller connected to the same switch, as can be observed in the image below:

```
sudo mn
```

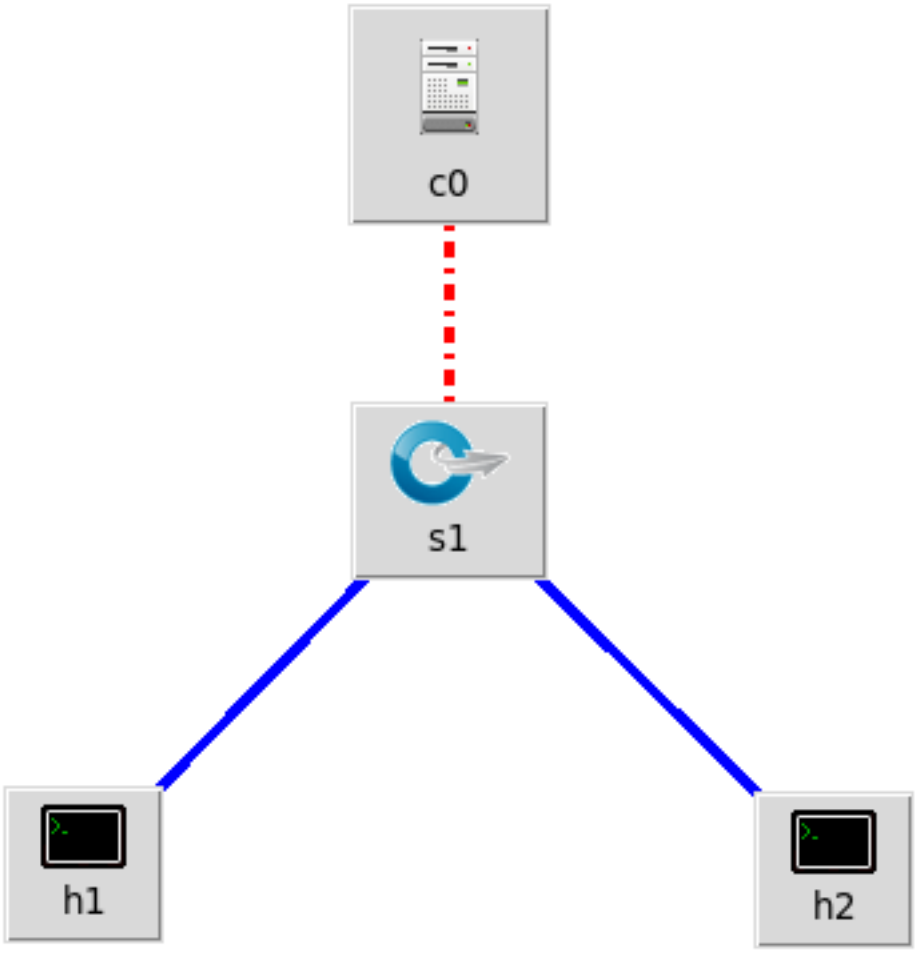

Figure 1: Mininet topology: Simple

To see the active nodes in a topology:

```
>mininet>nodes
```

To see the active links in a topology:

```
>mininet>links
```

The help command presents you more useful commands:

```
>mininet>help
```

To exit Mininet, you can use the following command at any time in the Mininet CLI:

```
>mininet>exit
```

> **Hint:** In case Mininet breaks for any reason, you can perform a complete cleanup through a terminal: `sudo mn -c`

In order to see the flow tables for switch s1, you can use `ovs-ofctl`, a command line utility than allows for sending OpenFlow messages, analyzing info on active switches and manually inserting flow entries.To obtain the flow tables for s1, execute the following command in another terminal:

```
sudo ovs-ofctl dump-flows s1
```

Initially, the flow table is empty.

To generate traffic in the topology, execute a `ping` command on the Mininet CLI for host 1 to host 2:

```
mininet>h1 ping h2
```

Q1. You should see that the ping time for the first attempt is larger than in the subsequent connections. What is the reason for this difference?

Now use the `pingall` command on the Mininet CLI to make all hosts sequentially ping one another. Afterwards, If you re-examine the flow tables for switch s1 again, you should now be able to observe several rules installed by the controller, due to the generated network traffic.

This type of flow insertion is defined as *Reactive*: the flows are installed after a packet arrives on a switch where no rules are found that match. As such, the packet is then sent to the controller, which creates and installs the necessary flow rule for it to be matched and sent to a destination.

An alternative approach is *Proactive* flow insertion, where a flow rule is proactively inserted on the switches **before a packet arrives**.

To test this approach, start Mininet with the topology seen in the image below: 4 hosts; 4 OpenFlow switches, each connected to a host; all switches are connected in a line; the switches are configured to connect to a remote controller (which must be initialized outside Mininet, **and will not be initialized at this point**).

```
sudo mn --topo linear,4 --mac --switch ovsk --controller remote
```

On the mininet CLI, try to ping from h1 to h2:

```
mininet>h1 ping -c3 h2
```

Computer Networks — 2023/24, Q2		**Assignment:**	Lab 7
Software-Defined Networking (SDN)		**Version:**	1.0
Issued:	2023-12-04	**Submission Due:**	2023-12-10

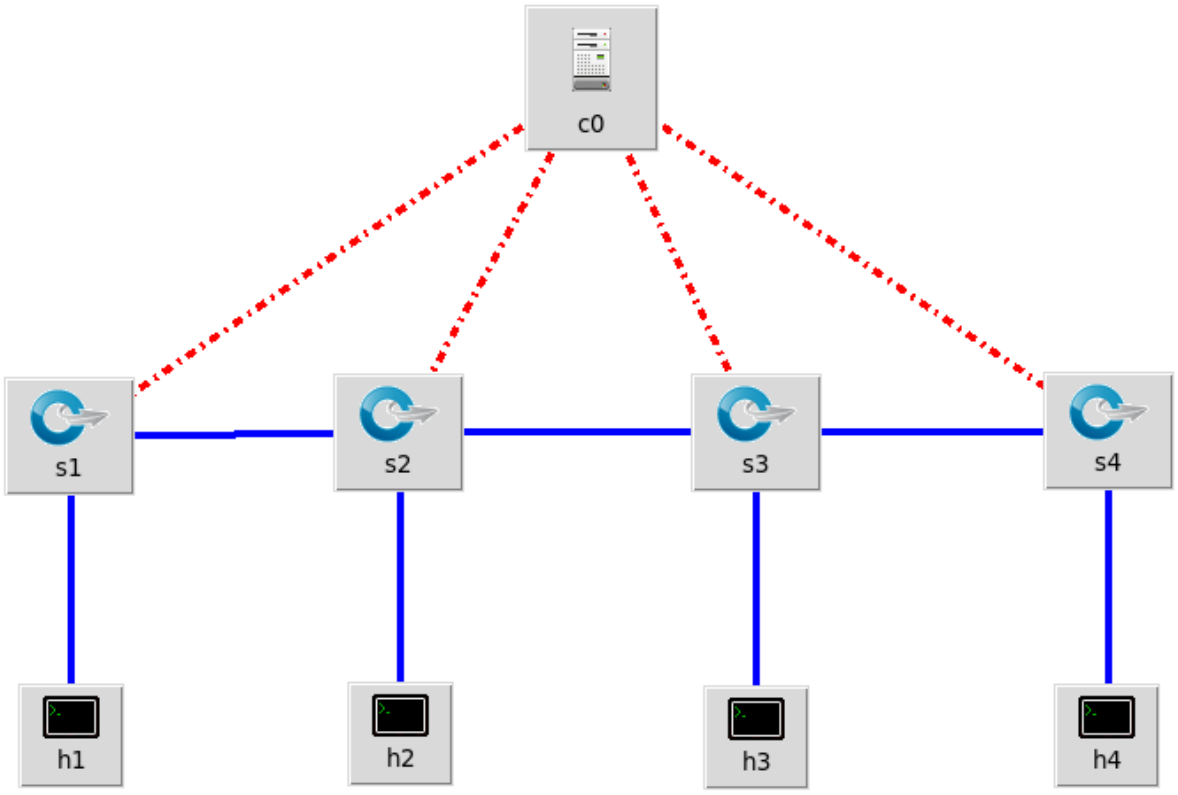

Figure 2: Mininet topology: Linear

Q2. Do the pings get any replies? Why, or why not?

The `ovs-ofctl` utility can be used to manually install flow rules. Run the following commands to forward any packets arriving at s1's port 1 to port 2:

On another terminal, run the following commands to allow ARP requests on s1 and s2:

```
sudo ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,action=flood
sudo ovs-ofctl add-flow s2 dl_type=0x806,nw_proto=1,action=flood
```

Afterwards, run the following command to forward any packets with the following src and dst MAC from s1's port 1 to port 2:

```
sudo ovs-ofctl add-flow s1 \
in_port=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,actions=output:2
```

Q3. Try to ping from h1 to h2 again. What happens now? Do you need more rules for a ping to work? If so, which?

Mininet can be used with a variety of SDN controllers. One of which is Ryu, a python-based controller that provides a powerful API for developing SDN applications. In the following questions you will use Ryu as the SDN controller.

Open two terminals and start the Ryu controller and a Mininet topology with the following commands:

```
ryu-manager ryu.app.simple_switch_13
```

```
sudo mn --topo single,3 --mac --controller remote --switch=ovsk
```

On the Mininet CLI, execute a ping command between h1 and h2:

```
mininet> h1 ping -c3 h2
```

Afterwards, stop the Ryu controller instance, but keep running the mininet instance.

Q4. In this situation, will a ping from h1 to h2 still get any replies? What about from h1 to h3? Explain.

2.2 SDN Load Balancing

Load Balancing, in a networking context, is the process of distributing traffic across multiple servers, in order to optimize the response time and avoid overloading any single server.

An algorithm that allows the implementation of a simple load balancing mechanism is *Round Robin*. With this algorithm, the client requests are forwarded to each server in turn, following a defined list. Whenever the last element is reached, the *Round Robin* algorithm goes back to the top of the list.

In this section, you will use the Ryu SDN controller to implement a load balancing mechanism between 6 client hosts and 3 server hosts. The corresponding Mininet topology can be observed in the image below. The 6 clients access a virtual server IP (10.0.10.10), after which the controller install flows which seamlessly forward the traffic from the client to the chosen server and also the reverse connection. The clients are unable to directly access any of the servers, they can only do so through the virtual server.

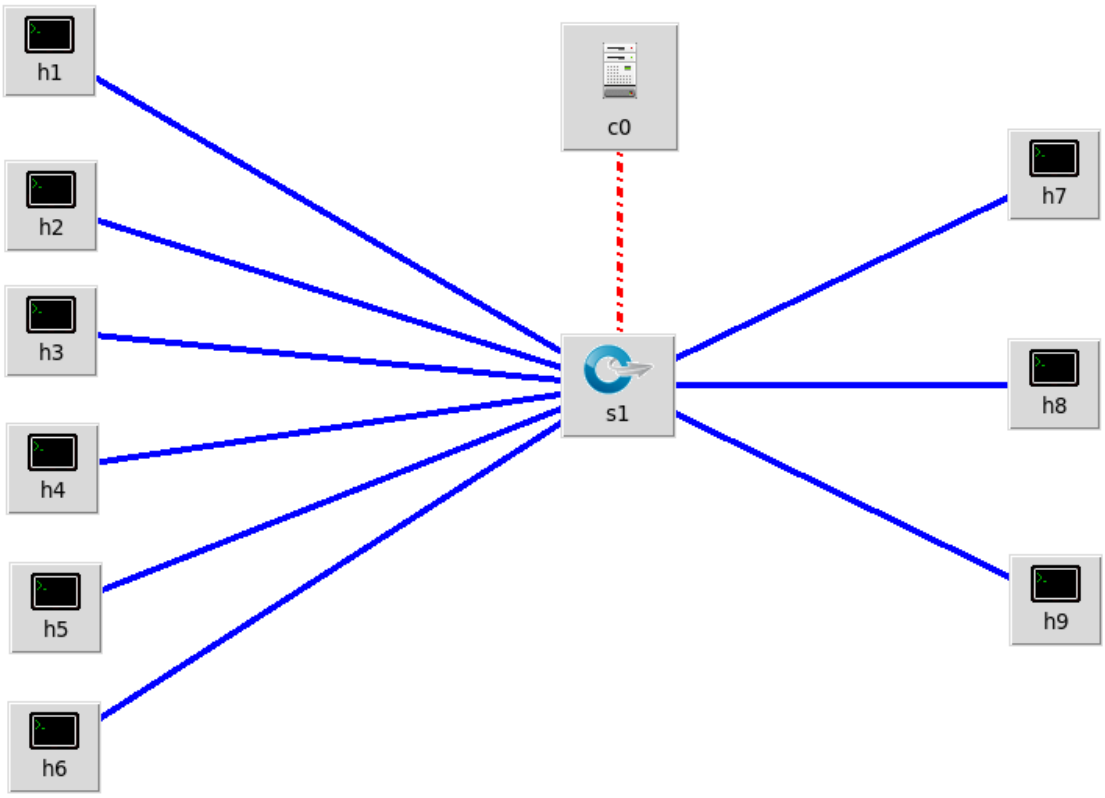

Figure 3: Mininet topology: load balancing

A skeleton implementation file for this load balancer, `loadbalancer.py`, is available on the lab folder. Open this file in your preferred text editor and complete its implementation, following the highlighted instructions in the TO-DO comments.

Q5. Finish the `loadbalancer.py` implementation, following the instructions in the comments to add the missing flow rule parameters. In the answer, show your implementation with the completed TO-DO portions (The TO-DOs are in the `add_flow` function).

To test the complete load balancer, open two terminals: the first will execute the Ryu controller with the `loadbalancer.py` file, while the second will run the Mininet topology.

On terminal 1, run the following command to start the controller on port 6653:

```
ryu-manager ~/rc/lab-files/07-lab-sdn/loadbalancer.py --ofp-tcp-listen-port 6653
```

On terminal 2, start the Mininet topology shown above:

```
sudo mn --topo single,9 --mac --controller remote --switch ovsk
```

To test the load balancer, send a packet from one of the client hosts to the virtual server IP. For instance:

```
mininet>h1 ping 10.0.10.10
```

If the load balancer is functioning correctly, the client should get replies from one of the servers. On the Ryu terminal is also displayed the next server to be selected. Try to ping from another host to the virtual server and verify that the connection is indeed going to the next server selected using *Round Robin*.

Q6. Using the `ovs-ofctl` utility, show the flow table rules for switch s1, after accessing the virtual server from three different hosts.