ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Switch-Controller interactions in a multi-controller environment

Chiara Grasselli

LAB. OF NETWORK PROGRAMMABILITY AND AUTOMATION - PROGRAMMABLE NETWORKING (A.Y. 2024/2025)

Multi-controller scenario

- Switches can be connected to one or multiple controllers.
- Advantages:
  - Improved reliability/resilience in case of failure
  - Improved load-balancing
  - Handover between controllers managed by controllers themselves
    - Is this (always) true?

Controller roles

- Available roles are of 3 types:
  - **EQUAL** (default)
  - **MASTER**
  - **SLAVE**

- A switch:
  - can have *only one* MASTER controller, and one (or more) EQUAL and/or SLAVE controller(s) at a time;
  - cannot request the role of a controller to be changed.
- A controller can ask for its role to be changed via an **OFPT_ROLE_REQUEST** message.

Controller roles in detail

1. **EQUAL** (default):
   - Has full access to the switch
   - Receives all asynchronous messages (e.g., packet-in, flow-removed)
   - Can send controller-to-switch messages (i.e., is able to modify the state of the switch)
2. **MASTER**:
   - There is *only one* controller in this role
   - Has full access to the switch
3. **SLAVE**:
   - Has read-only access to the switch
   - Does not receive asynchronous messages; receives only port-status messages
   - Cannot send any controller-to-switch messages

Note: If you need only one controller able to make changes to switches, no controller should be in the EQUAL role (choose SLAVE instead).

Hands-on session:
multi-controller environment

Before starting

- If not available on your environment, install **curl**
  - sudo apt install curl
- If you use the virtual machine provided for the course, just log in with the following credentials:
  - User: ubuntu
  - Password: labnetprog25

Starting Ryu controller(s)

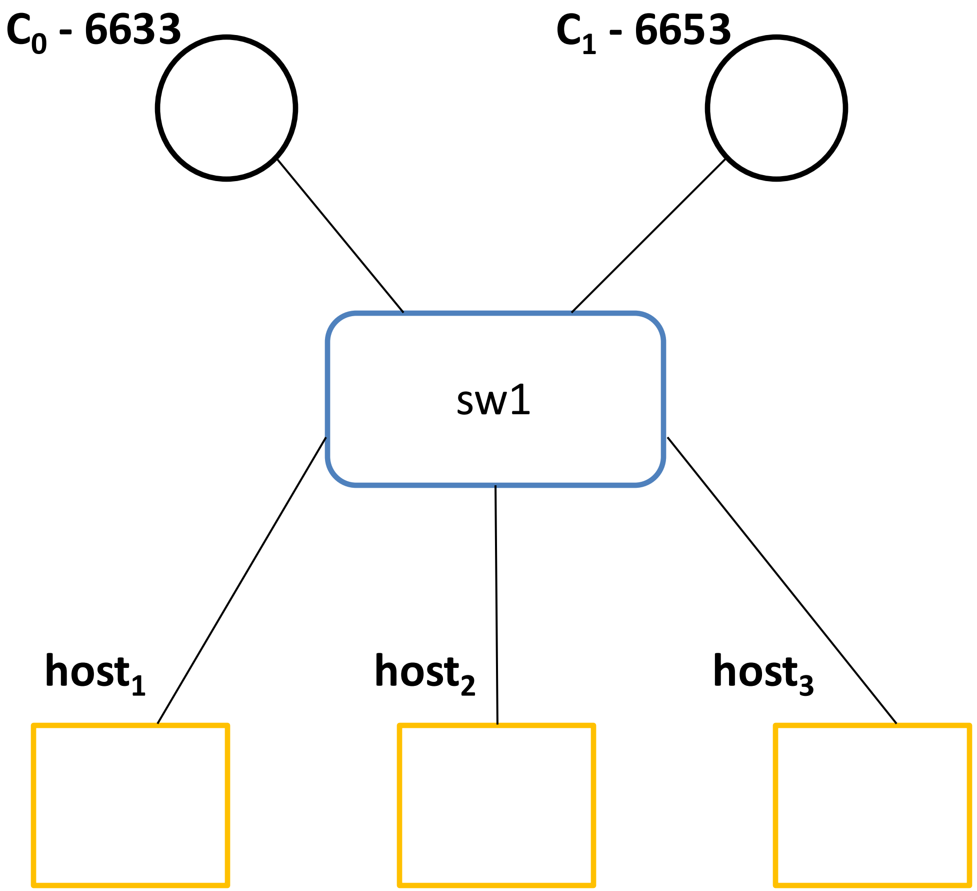
To start the controller on a TCP port different from the default one (6653 or 6633), use the following command:

```
ryu-manager --ofp-tcp-listen-port OFP_TCP_LISTEN_PORT <ryu-app_name.py>
```

where OFP_TCP_LISTEN_PORT is a free port of your choice

Note: Wireshark dissector works only on OpenFlow well-known ports (i.e., 6633 and 6653)

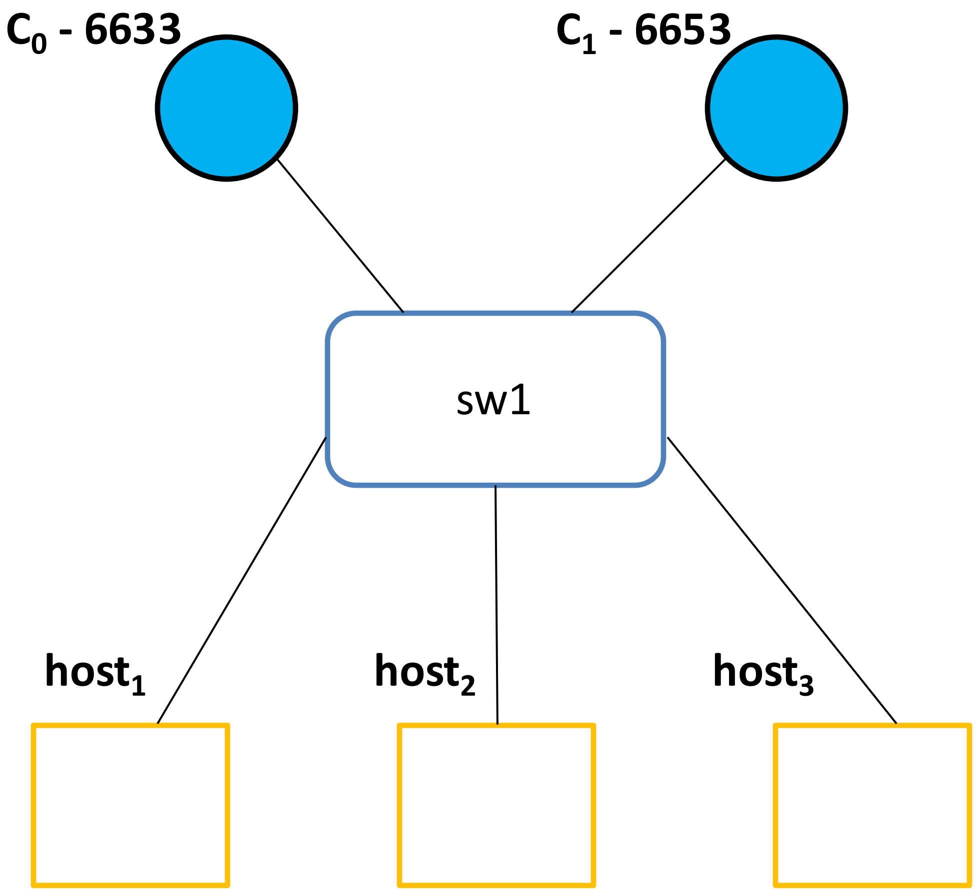
Use case topology



EQUAL controller
MASTER controller
SLAVE controller

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Practical session 1: C_0 and C_1 in EQUAL mode



EQUAL controller
MASTER controller
SLAVE controller

Practical session 1: goal

The goal is to verify the behavior of controllers C_0 and C_1 and prove the resilience in a simple topology scenario.

1. Start a ping between host1 and host2, and check:
   i) OpenFlow rules on switch sw1;
   ii) controller status: **sudo ovs-vsctl list controller**;
   iii) messages with Wireshark.
   <u>What do you notice?</u>
2. Stop one of the two Ryu instances, and make a ping between host1 and host3.
   <u>What do you expect?</u> Check as before.
3. Ping between all pairs of hosts.

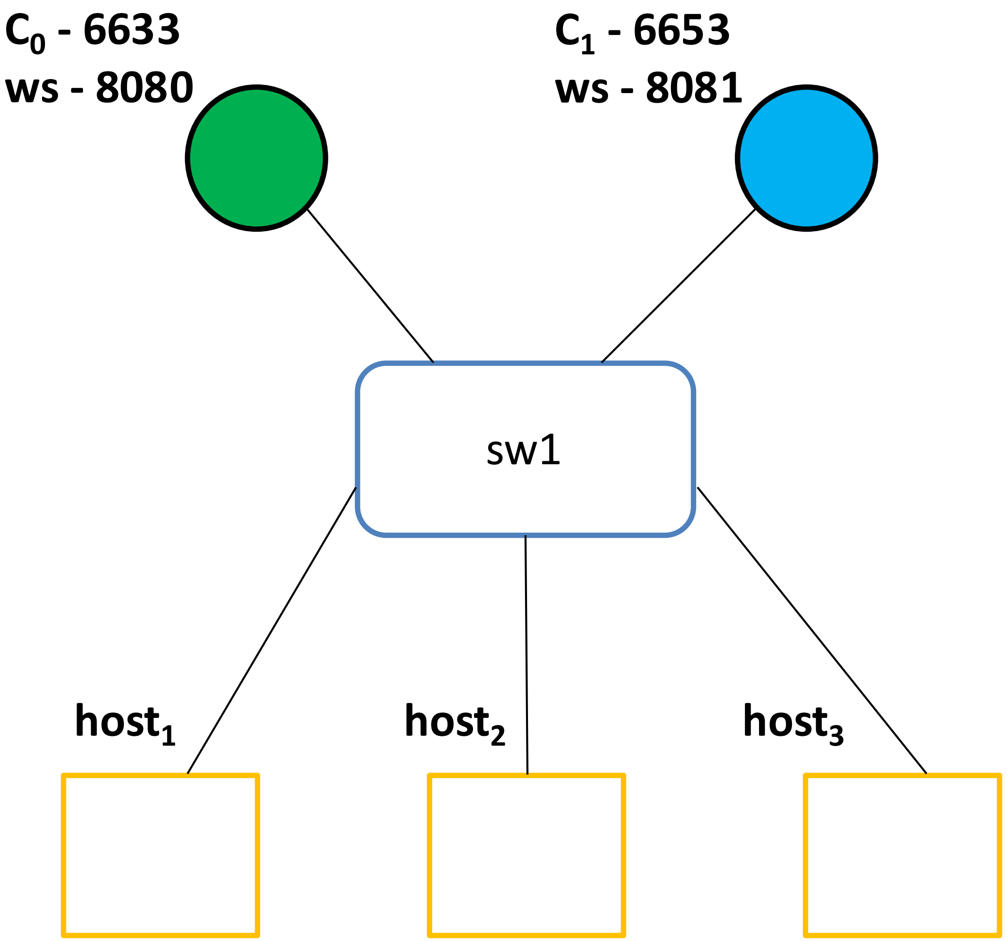
   <u>How many rules do you expect to be added in total?</u>

Practical session 1

C_0 and C_1 start in EQUAL mode, as default.

1. Start two instances of Wireshark and set a filter for each TCP port:
   - **tcp.port == 6633**, **tcp.port == 6653**
2. Run two instances of Ryu simple learning switch (OpenFlow version 1.3) by indicating *explicitly* the OpenFlow TCP listen port, i.e., 6633 for one instance and 6653 for the other one:
   - **ryu-manager --ofp-tcp-listen-port 6633 simple_switch_13.py**
   - **ryu-manager --ofp-tcp-listen-port 6653 simple_switch_13.py**
3. Start Mininet custom topology:
   - Check if topology is an executable; if not: sudo chmod +x 1switch_3host_ext_cntlr.py
   - **sudo python3 1switch_3host_ext_cntlr.py**

Practical session 2: C_0 requires MASTER role to sw1



EQUAL controller
MASTER controller
SLAVE controller

Practical session 2: goal

The goal is to verify controllers and protocol behavior.

Check:
i) controller status;
ii) messages on instance 6633 with Wireshark.
<u>What do you notice?</u>

Practical session 2

C_0 and C_1 start in EQUAL mode, as default.

1. Run two instances of Ryu simple learning switch as in the previous practical section, but this time together with the REST application (default port is 8080):
   - **ryu-manager --ofp-tcp-listen-port 6633 simple_switch_13.py ofctl_rest.py**
   - **ryu-manager --ofp-tcp-listen-port 6653 --wsapi-port 8081 simple_switch_13.py ofctl_rest.py**

   Note: use **--wsapi-port** to set a different port for the REST app
2. Start Mininet custom topology:
   - **sudo python3 1switch_3host_ext_cntlr.py**

Practical session 2: Ryu's REST APIs

- Interact with the controllers using Ryu's REST APIs
  - REST APIs: https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html
- Generate HTTP REST requests on the bash command line:
  - curl **<options> <url>**
    - **-X <method>** : REST method, e.g., GET (collect) or POST (create)
    - **-d <data>** : to be used in conjunction with -X POST to specify content
  - **JSON**: text-based data interchange format [https://www.json.org/]
    - If using JSON format, you can validate it with: https://jsonformatter.org/
    - { } for key:value pairs (objects), [ ] for lists (arrays)

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Practical session 2: Ryu’s REST APIs

1. Get information from C_0 and C_1 via REST APIs.

   For instance, check:

   i. Controlled switches (Datapath IDs)
      - URI: /stats/switches

   ii. Switch flow rules
      - URI: /stats/flow/<dpid>

   iii. Controller role
      - URI: /stats/role/<dpid>

Practical session 2: change of role

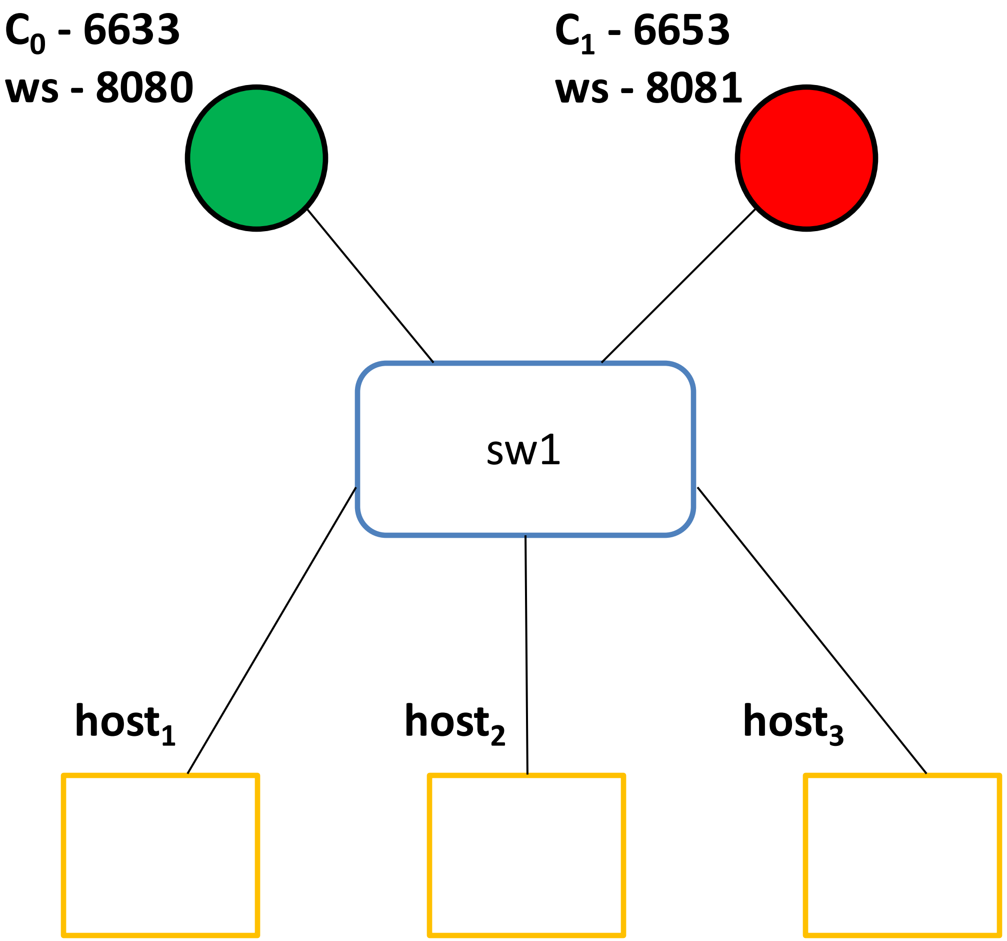
1. Change C_0 role from EQUAL to MASTER and verify protocol behavior:

   curl -d @role-req-toMaster-format.json -X POST http://localhost:8080/stats/role

   What happens?

Note: Do not stop Mininet and Ryu instances at the end of this session, so that the next practical session will start from this condition.

Practical session 3: C_1 requires SLAVE role to sw1



EQUAL controller
MASTER controller
SLAVE controller

Practical session 3: goal

The goal is to verify controllers and protocol behavior.

Check:
i) controller status;
ii) messages on instance 6653 with Wireshark
What do you notice?

Practical session 3

C_0 is in MASTER mode, C_1 is in EQUAL mode.

1. Restart Wireshark captures
2. Change C_1 role from EQUAL to SLAVE and verify protocol behavior:

   ```
   curl -d @role-req-toSlave-format.json -X POST http://localhost:8081/stats/role
   ```

Note: Do not stop Mininet and Ryu instances at the end of this session, so that the next practical session will start from this condition.

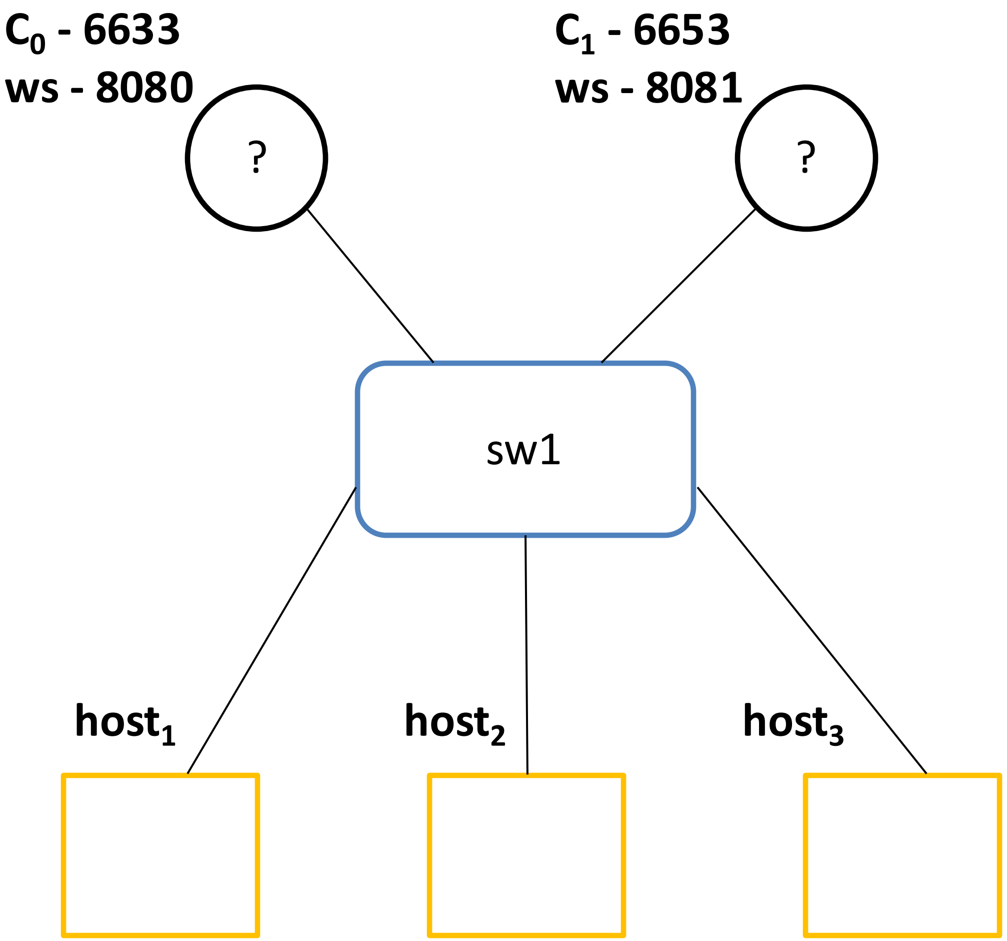
Practical session 4: C_1 now requires MASTER role to sw1

C_0 is in MASTER mode, C_1 is in SLAVE mode. Trigger a role change to MASTER for C_1.

1. Restart Wireshark captures before triggering the change
2. Trigger the role change and check Wireshark connections

What happens?

Practical session 4: C_1 now requires MASTER role to sw1



EQUAL controller
MASTER controller
SLAVE controller

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA