1. Grading for SDN Project

The grading of the project is composed of the following four parts.

1.1. Code

Please hand in the whole code folder (`sdn-script`). The grading is composed of the following three parts:

- Submit in time
- comments
- Functions (See **3. Testing** for details)

1.2. Readme

1.3. Report

- Background: You need to show your understanding for the following three parts and how they work in your project
    - The understanding for **SDN**
        - Especially, the relationship between controller and switch
    - The understanding for **RYU**
    - The understanding for **mininet**
- Implementation: You need to explain your code here. You should put on some important code as well
- Testing: You need to show how you test your code and analyze the result (with picture)
- Contribution: Please specify the contributions of group members
- Conclusion:
    - What you have learned in this project
    - Have you met any problems and how did you solve them
    - etc.

1.4. Presentation

You should hand in the slide before your presentation.

- Slide: Please give brief presentation of your project based on your report
- Random testing: (See **3. Testing** for details)
    - Basic test:
        - Connection of the network: we will randomly choose two topology (one for acyclic graph while another of the graph with loop)
        - Topology output: it requires the **controller** to output the information

        - Shortest path: it requires the **controller** to output the information
    - Change topology:
        - Connection of the network: we will randomly choose two topology and **change the topology**
        - Topology output: it requires the **controller** to output the information
        - Shortest path: it requires the **controller** to output the information

1.5. Bonus

a. Implement flooding without loops (essentially calculate and install a spanning tree for broadcasts).

b. Implement ECMP on networks with multiple paths (determine the number of paths between two nodes) and install rules that match on, say, even and odd TCP ports!

The first is recommended. If you would like to implement the second, please give the detailed testing steps as well.

2. What to hand in

- [ ] Slide
- [ ] Codes
- [ ] readme
- [ ] Report

3. Testing

3.1 Check connections: `pingall`

Different topology	With loop or not
Single n	No
Linear n	No
Tree n	No
Assign 1	NO
Triangle	Yes
Mesh n	Yes
Some loops	Yes

3.2 Check connections with topology change: `pingall`

Topology	Disconnection
Triangle	Break link between any two switches (i.e. s1-s2/s1-s3/s2-s3)
Mesh n(4)	Randomly break 1-3 links
Some loops	Randomly break 1 link

3.3 Check the shortest path

- The controller should **output the path**
- Check the flow table:
  - `dpctl dump-flows`: check flow table of all switches
  - `sh ovs-ofctl dump-flows s1`: check the specific switch

3.4 Check the topology

The controller should **output the topology**

3.5 Extra

Testing for **Flooding without loops:**

- Use Wireshark to capture the package and to see whether there is a broadcast storm
- It requires the controller to output **spanning tree**