CAN201 Introduction to Networking

# Networking Project

| | |
|---|---|
| Contribution to Overall Marks | **40%** |
| Submission Deadline of Part I | **Sunday 20th Nov. 2022, 23:59** |
| Submission Deadline of Part II | **Sunday 18th Dec. 2022, 23:59** |
| Type | **Team coursework** |
| Learning Outcome | **[A] [B] [C] [D]** |

## How the work should be submitted?

- **SOFT COPY ONLY!**
- Every team leader must submit the work through Learning Mall.

## Specification of Part II (20% of overall marks)

This part of the networking project aims to use Mininet to create a simple SDN network topology and emulate a traffic control function through using the SDN flow entry. Assuming that the client side only knows the service running on server 1 and communicates with server 1 (without knowing the existence of the service on server 2). However, the SDN controller can manipulate (forward/redirect) the traffic without the awareness of the client. The detailed project tasks are specified as follows. For Part II, the client side program and server side program will be provided.

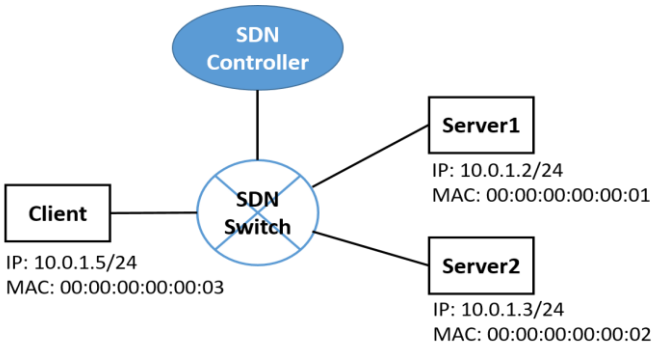


Figure 1. A simple SDN network topology

**Task 1**

Use Mininet Python library to create a Python file to build a simple SDN network topology as Fig. 1 shows. Note that Client uses IP address 10.0.1.5/24, Server1 uses IP address 10.0.1.2/24, and Server2 uses 10.0.1.3/24. Also, Client, Server1 and Server2 need to use the MAC address as Fig. 1 presents.

**Task 2**

Program and run an SDN controller application using Ryu framework (see Task 4.1 and Task 5.1) and make sure every node (i.e., Client, Server1 and Server2) are reachable with each other. In other words, they can ping with each other. Notice that any flow entry (excluding the table-miss flow entry) should set an idle timeout of 5 seconds.

**Task 3**

Apply the given socket client program (client.py) and the given socket server program (server.py) to this SDN network topology. Specifically, run server.py on both Server1 and Server2, and run client.py on Client. With that, use the socket client side on Client to send traffic to the socket server side on Server1. Notice that wait 5 seconds after ping (the idle timeout mentioned in Task 2) and then start run the client.py, which is to make sure that the flow entry caused by ICMP ping packets has been removed.

**Task 4**

**4.1**. Program an SDN controller application that is able to create a flow entry after receiving the first (TCP SYN segment caused) Packet_In SDN packet (from the SDN switch to the SDN controller), then install the flow entry to the SDN switch, and then send out the Packet_Out SDN packet that contains the TCP SYN segment, whereby all the following traffic sent from Client to Server1 is forwarded to Server1.

**4.2**. With task 4.1, use Wireshark/Tcpdump on Client to capture the packets and then calculate the networking latency (from the first SYN segment till the last ACK segment indicating the TCP 3-way handshake is done).

**Task 5**

**5.1**. Program an SDN controller application that is able to create a flow entry after receiving the first (TCP SYN segment caused) Packet_In SDN packet (from the SDN switch to the SDN controller), then install the flow entry to the SDN switch, and then send out the Packet_Out SDN packet that contains the TCP SYN segment, whereby all the following traffic sent from Client to Server1 is redirected to Server2.

**5.2**. With task 5.1, use Wireshark/Tcpdump on Client to capture the packets and then calculate the networking latency (from the first SYN segment till the last ACK segment indicating the TCP 3-way handshake is done).

## *Submission:*

**Codes:**

- >= Python 3.6;
- The whole implementation includes multiple Python scripts as follows:
  1) Network topology Python file (which is used to create the SDN network topology for completing **Task 1**). Please name it "networkTopo.py".
  2) The Ryu SDN controller Python program (for performing **Task 2** and **Task 4.1**). Please name it "ryu_forward.py".
  3) The Ryu SDN controller Python program (for performing **Task 2** and **Task 5.1**). Please name it "ryu_redirect.py".

**Project Report:**

- A cover page with your full names (pinyin for Chinese student; name on your passport for international student) and student IDs of the whole team;
- 3 ~ 5 pages (including everything such as the reference), double columns, using the IEEE template provided;
- PDF format, LaTeX is recommended, IEEE template;

- Including:
  - **Abstract**
  - **Introduction**: project task specification (introduce some background about SDN and describe the task of this project, **do not** copy from this document and use your own words), challenge (identify the research/development problems you are going to address), practice relevance (come up with the potential applications with your proposal, e.g., load balance, secure traffic control, etc.), contributions (key points that you did for this coursework).
  - **Related Work**: research papers, technical reports, or similar applications that solve or facilitate network traffic redirection.
  - **Design**: the design of you solution, which should include the network system design diagram (and you need to describe it using your own words) based on Fig. 1, the workflow of your solution (in particular, the steps of creating the flow entry, installing the flow entry, etc.), the algorithm (i.e., the kernel pseudo codes of the network traffic redirection function) for the SDN controller.
  - **Implementation**: the host environment where you develop the implementation, such as the host CPU, Memory, Operating System, etc. Also, the development softwares or tools, like the IDE, the Python libraries, the SDN controller software (i.e., Ryu here), etc. Further, steps of implementation (e.g., program flow charts), programming skills (OOP, Parallel, etc.) you used, and the actual implementation of the traffic redirection function. In addition, the difficulties you met and how did you solve them.
  - **Testing and Results**: testing environment (can be more or less the same with your host implementation environment), testing steps (the steps of using the developed Python programs to complete the project tasks 1-4, including snapshots), and testing results, i.e., the networking latency comparison between the forwarding case (**Task 4.2**) and the redirection case (**Task 5.2**), and you should apply figures of bars or curves for showing average performance.
  - **Conclusion**: what you did for this project and any future work for improvement.
  - **Acknowledgement**: individual contribution percentage should be clarified here if the project is a teamwork by using this format: Student1's name (ID) contributes XX% to the project, Student2's name (ID) contributes XX% to the project, and Student3's name (ID) contributes XX% to the project. If there is no clarification of individual contribution, it is considered that all the individual team contributes the same percentage to the project.
  - **Reference** [IEEE format]

## *Meanwhile, you have to follow the compulsory requirement (no tolerance[1]):*

- Only ZIP file is allowed to submit;
- The ZIP file should be named as: CAN201-CW-Part-II-Student1name-Student2name-Student3name
- The ZIP file includes two folders, i.e., "Codes" and "Report". The Codes folder includes all the Python files, and the Report folder includes the report file;
- Python files are: networkTopo.py, ryu_forward.py, ryu_redirect.py;
- The report file should be named as: Report_PartII.pdf;

## *Allowed Python modules:*

*os, sys, shutil, socket, struct, hashlib, math, tqdm, numpy, threading, multiprocessing, gzip, zlib, zipfile,*

[1] It means that if you do not follow the compulsory requirement, your work will be marked as zero.

*time, mininet, ryu.*

## *Marking Criteria*

The following marking scheme is for the team, and every team member shall contribute to the project. Also, several specific rules should be followed:

1. Every team should use the "ACKNOWLEDGMENT" section of the IEEE template to describe the individual contribution(s) using the following format: Student1's name (ID) contributes XX% to the project, Student2's name (ID) contributes XX% to the project, and Student3's name (ID) contributes XX% to the project.
2. If there is no clarification about the individual contributions, it is considered that every team member in the same team has the same contribution percentage and will have the same mark of the CW project.
3. The individual contribution must be in a range: for a 3-person team, it must be 30% - 40% (30% and 40% are included); for a 2-person team, it must be 40% - 60% (40% and 60% are included). If any individual contribution percentage of a team is out of the range (e.g., a 3-person team has the contributions like: 50%, 30%, 20%), the team has to go through a review by the module leader about the contribution discrepancy.
4. The algorithm for calculating individual mark as follows:
    a. Assuming the 3-person team's mark is *m*, student1 contributes *x*%, student2 contributes *y*% and student3 contributes *z*%.
    b. If x>y and x>z, student1 with the highest contribution gets the same mark with the team's mark, i.e., student1 will get *m*. Also, student2 will get ( *y* / *x* ) * *m*; and student3 will get ( *z* / *x* ) * *m*.
    c. If x=y>z, student1 and student2 will have the same mark *m*, and student3 will get ( *z* / *x* ) * *m*.
    d. If x=y=z, all the team members will have the same mark *m*.
    e. The 2-person team's marking criteria is in accordance with the above.

**Report (50%)**

| **Marking Criteria** | **Item** | **Mark** |
|---|---|---|
| Contents (40%) | Abstract | 3% |
| | Introduction | 5% |
| | Related Work | 4% |
| | Design | 8% |
| | Implementation | 7% |
| | Testing and Results | 7% |
| | Conclusion | 3% |
| | Reference | 3% |
| Typography (5%) | Report structure, style, and format | 5% |
| Writing (5%) | Language | 5% |

Marking Scheme:

1. Contents (40%)
    1.1. Abstract (3%)
        - Good (3%)
        - Appropriate (1-2%)
        - No abstract (0%)
    1.2. Introduction (5%)
        - Excellent (5%)

- Lack of necessary parts (1%-4%)
- No introduction (0%)

1.3. Related Work (4%)
- Sufficient (4%)
- Not enough (1%-3%)
- No introduction (0%)

1.4. Design (8%)
- Excellent: adequate and accurate figures and text description (8%)
- Reasonable: clear figures and text description (4%-7%)
- Incomplete: unclear figures and text description (1%-3%)
- No design (0%)

1.5. Implementation (7%)
- Excellent: sufficient details of implementation (7%)
- Reasonable: clear description of implementation (4%-6%)
- Incomplete: unclear description of implementation (1%-3%)
- No implementation (0%)

1.6. Testing and Results (7%)
- Excellent: sufficient testing description, correct experimental results using figures with clear text description and analysis (7%)
- Acceptable: clear testing description, appropriate experimental results using figures with acceptable text description and analysis (3%-6%)
- Incomplete: lack of testing description, experimental results with figures, or text description and analysis (1%-2%)
- No testing and results (0%)

1.7. Conclusion (3%)
- Excellent conclusion (3%)
- Acceptable conclusion (1%-2%)
- No conclusion (0%)

1.8. Reference (3%)
- Excellent reference with the correct IEEE format (3%)
- Incorrect or inconsistent reference format (1%-2%)
- No reference (0%)

2. Typography (5%)
- Beautiful and clear typography: 5%
- Acceptable typography: 2%-4%
- Bad typography: 0% ~ 1%

3. Writing (5%)
- Accurate and concise language: 3%-5%
- Unclear and confusing language: 1% ~ 2%

**Codes (50%)**

Program testing steps:

**1．Forwarding case:**

1.1 Run the networkTopo.py to create the SDN network topology. Check Client, Server1, and Server2 use the correct IP addresses and MAC addresses.

1.2. Run ryu_forward.py on Controller, and use Client to ping Server1's IP address and Server2's IP address.

1.3. Run server.py on both Server1 and Server2, and also run client.py on Client after the previous ICMP ping incurred flow entry's idle timeout (i.e., 5 seconds).

a. Show the flow table on Switch.
b. Show Server1 receives the traffic sent from Client.

**2．Redirection case:**

2.1 Run the networkTopo.py to create the SDN network topology. Check Client, Server1, and Server2 use the correct IP addresses MAC addresses.

2.2. Run ryu_redirect.py on Controller, and use Client to ping Server1's IP address and Server2's IP address.

2.3. Run server.py on both Server1 and Server2, and also run client.py on Client after the previous ICMP ping incurred flow entry's idle timeout (i.e., 5 seconds).

c. Show the flow table on Switch.
d. Show Server2 receives the traffic sent from Client.

Marking scheme:

1. Step 1.1 and 2.1 (10%)
   - Complete topology with correct IP addresses: 10%
   - Incomplete topology or incorrect IP addresses: 3%-9%
   - No networkTopo.py or not executable networkTopo.py: 0%-2%

Note: if no networkTopo.py or the networkTopo.py is not executable, the marking stops here.

2. Step 1.2 (5%)
   - The program ryu_forward.py can work and Client can ping Server1 and Server2: 5%
   - If ryu_forward.py can work but Client cannot ping Server1 or Server2: 2%-4%
   - If no ryu_forward.py or ryu_forward.py is not executable: 0%-1%

Note: if no ryu_forward.py or ryu_forward.py is not executable, no marking for Step 1.3.

3. Step 1.3 (10%)
   - The flow entry can be shown correctly and Server1 can receive traffic: 10%
   - The flow entry cannot be shown correctly or Server1 cannot receive traffic: 4-6%
   - Neither the above: 0%
4. Step 2.2 (10%)
   - The program ryu_redirect.py can work and Client can ping Server1 and Server2: 10%
   - If ryu_redirect.py can work but Client cannot ping Server1 or Server2: 4%-9%
   - If no ryu_redirect.py or ryu_redirect.py is not executable: 0%-3%

Note: if no ryu_redirect.py or ryu_redirect.py is not executable, no marking for Step 2.3.

5. Step 2.3 (15%)
   - The flow entry can be shown correctly and Server2 can receive traffic: 15%
   - The flow entry cannot be shown correctly or Server1 cannot receive traffic: 5-10%
   - Neither the above: 0%