Introduction to Computer Networks

Assignment 4: Solving NAT traversal problem

1. Goal

- Develop two programs, Registration Server, and Client for chatting.
- Develop a solution that allows clients under different NATs to communicate with each other.

2. System Overview

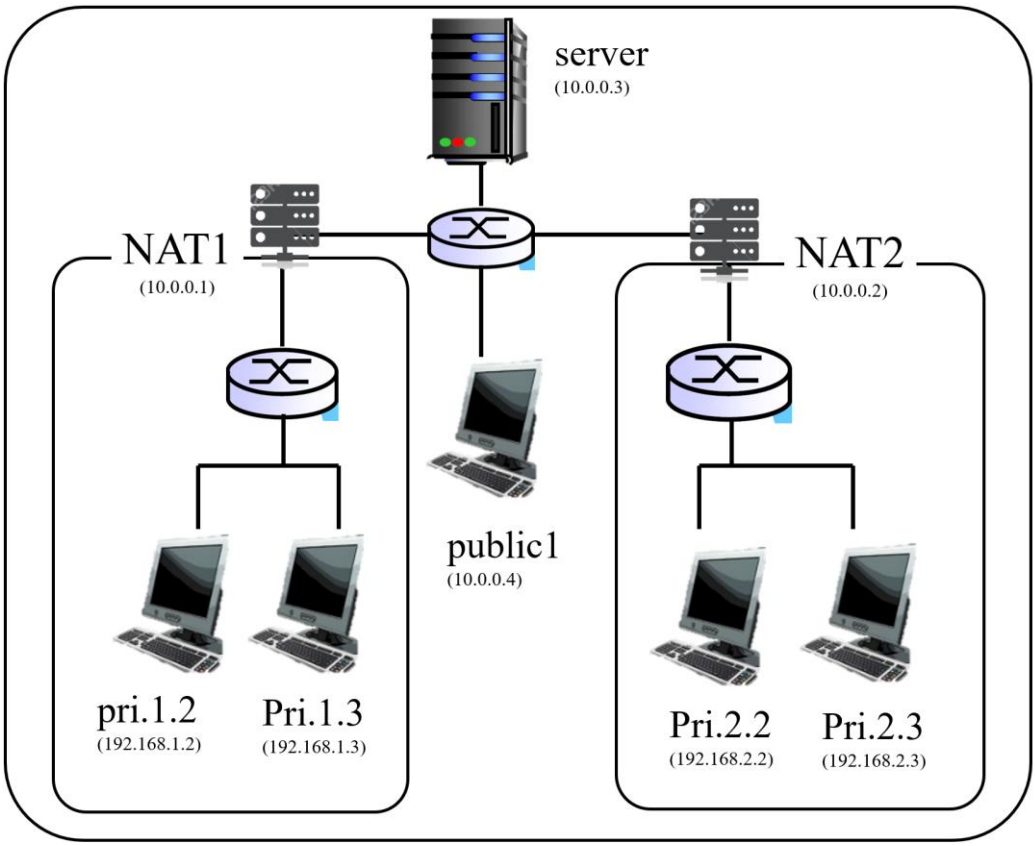

Figure 1. System configuration on a single computer

The Mininet will provide two NAT networks, one public client host and Registration Server host. Each NAT network has multiple private client hosts.

Client programs register their IDs into the Registration Server program and retrieve current registered client information (including myself) from the Server. The client information includes a client ID, and its IP & port number used in the registration.

3. Development environments

- TA will evaluate your results on Mininet. (Mininet can be run on Linux, you need to install Linux or virtual machine).
- Use Python (version 3.6+)
- You have to describe your development environment information in detail in the report. If not, TA cannot evaluate your program and you will get zero points.
- We will give you a VM image which includes Mininet and skeleton codes.
  - "execute_mn.sh" is a script that will run Mininet with NAT that contains preset topology including NAT routers and hosts.
  - In this assignment, "execute_mn.sh" does not need any argument.

    ex) sudo ./execute_mn.sh

  - "execute_mn.sh" automatically runs five client hosts and one server host. After terminals appear you need to run server(server.py) and client(client.py) manually.

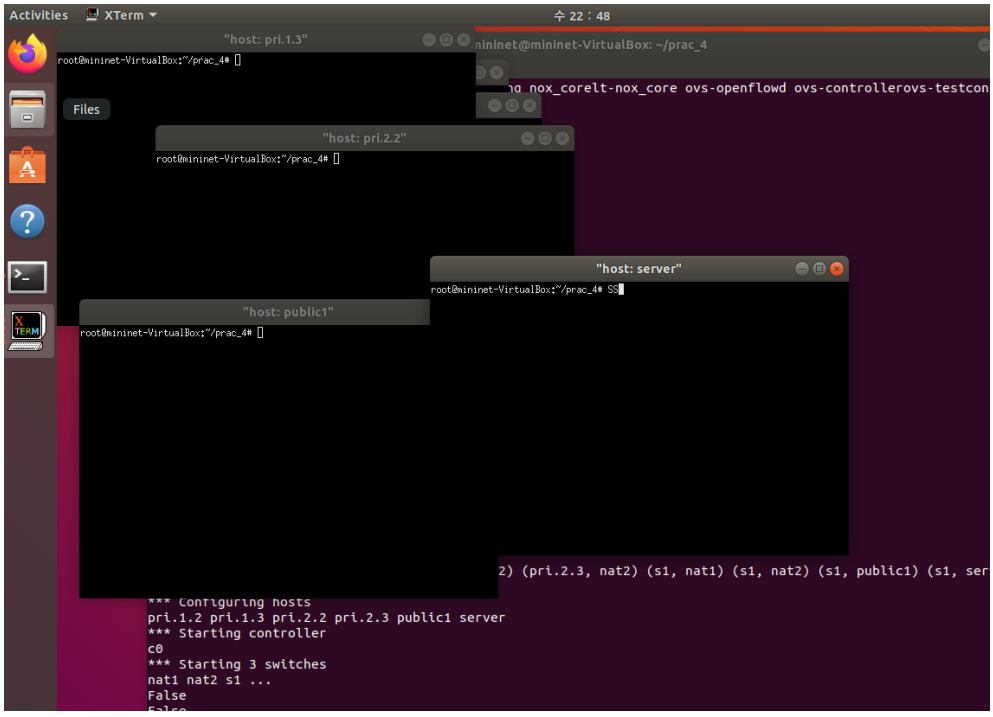

Figure 2. "execute_mn.sh" execute example

- You can terminate the Mininet by pressing **'ctrl + c'** at the main terminal.
- The template codes for client.py and server.py are given, you can modify them for this assignment.
- Mininet's ID and password are both "mininet".

4. Functionalities to implement

- Client program
  - When the client program starts, you enter each client ID. Assume that all clients are set to different IDs. Client IDs don't include any white space.
  - After starting
    - The client program must send a registration request to a server with UDP with port number 10081. The registration request must include Client ID (you can add other information if you need).
    - The client must receive the registration response from the server. The registration response must have the information of currently registered clients (you can also add additional information).
  - The client must be able to handle the server's message about other clients' registration and deregistration.
  - The client displays all registered client information using the command '@show_list'. The client displays the clients list as follows:

    client1 10.0.0.1:59494
    client2 10.0.0.2:53213

    Note: displayed IP addresses and port numbers are captured by the server through each registration process.

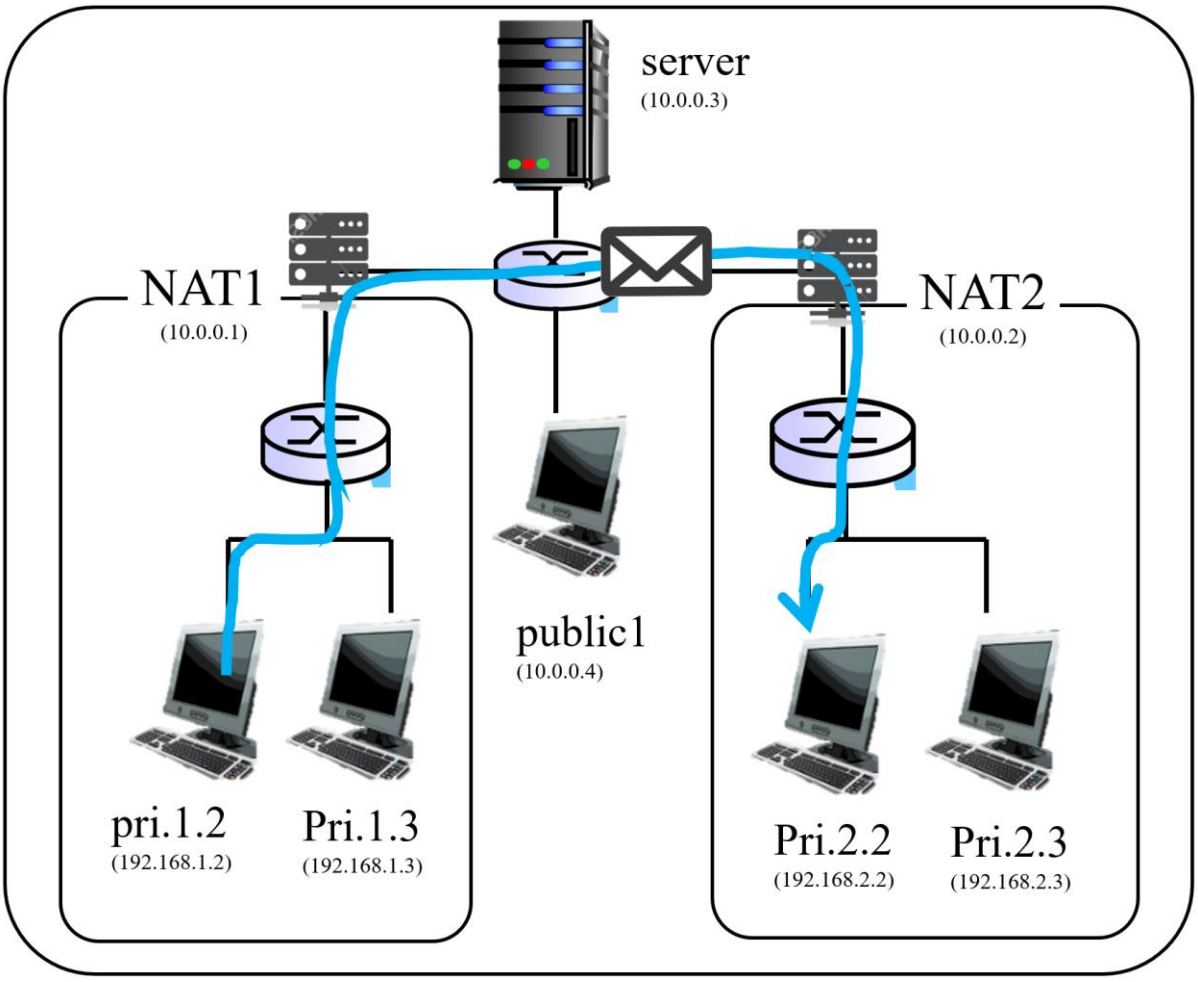

Figure 3. Overview of message transfer directly

- After registration, the client can receive incoming chat messages from other clients directly. (see Figure 3) If the chat messages come, display them with the sending client ID as follows:

  From client2 [I love Network!!!]

- If the client wants to send a message to other clients, use a command '@chat' with client ID of the opponent client as following:

  @chat client2 I love Network too!!!

- The clients can send and receive chat messages repeatedly before deregistration.

- You can use the '@exit' command to send a deregistration request to the server and

terminate the client program.

- Clients must keep sending a registration request to the server for ‘keep alive’ purpose in every 10 seconds. (the Mininet NAT’s default session timeout is 30 seconds)

- Registration Server Program
  - When receiving a registration request via UDP, display it and broadcast the new client information to all registered clients.

    client1 10.0.0.1:59494

  - When receiving a deregistration request, display it and delete the client information in the registration list. Then, broadcast the deregistered client information to all registered clients.

    client2 is deregistered 10.0.0.2:53213

  - Sometimes a client is disconnected without sending a deregistration request. Check ‘keep alive’ registration requests periodically sent from clients. If this request is not sent more than 30 seconds, then display the ‘timeout’, and the client’s IP and Port. And delete the client information in the registration list. Then broadcast it to all registered clients.

    client1 is disappeared 10.0.0.1:59494

- Extended Goal

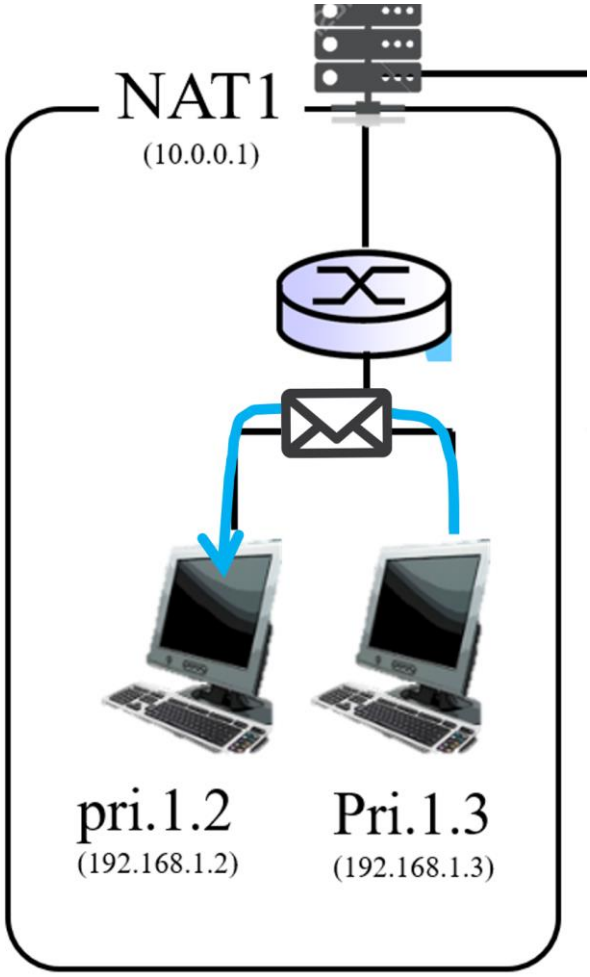

Figure 4. Overview of message transfer in same NAT network

  - If two clients are under the same NAT, Allow the clients communicate with each other using local / private IP addresses (not using the NAT public IP address).

    - You can design/upgrade any protocol to achieve this extended goal.

- Miscellaneous
  - Client ID consists of alphabet and number without any white spaces. The length does not exceed 32 bytes.
  - The size of a chat message is limited in 200 letters.

5. Sample Results

Client 1	Client 2	Registration Server
# client1 starts Enter ID : client1		client1 10.0.0.1:59595
	# client 2 starts Enter ID : client2	client2 10.0.0.2:51515
	@show_list client1 10.0.0.1:59595 client2 10.0.0.2:51515	
From client2 [hello]	@chat client1 hello	
@show_list client1 10.0.0.1:59595 client2 10.0.0.2:51515		
@chat client2 nice to meet you	From client1 [nice to meet you]	
@exit # client1 terminates		client1 is unregistered
	# stop the program by Ctrl-C	<after few seconds> client2 is disconnected

6. Submission

- The deadline is **12.13 (Sun) 23:59**.
  - For delayed submissions, a penalty of -15 points applies every 24 hours. After 72 hours, you get zero points.
  - In the case of plagiarism, you will get ***F***.
- Submit a zip file including a **report** and two (client.py and server.py) program files to iCampus
  - The report file format should be PDF.
  - Name the Report file as follows ***StudentID_Name.pdf*** (ex: 2020001_홍길동.pdf)
  - The report has to include the following things.
    1) Describe your development environment information in detail (compilers/interpreter versions, compile options, used extra library etc.)
    2) Present how to design your assignment such as data structures and algorithms.
    3) Explain how to run both sender and receiver programs including the screen capture.

6. Scoring

- Total 100 points
  - 20 points: The client registers to the registration server.
    - The client sends registration request to the server
    - The server displays the client information
  - 20 points: The client and server can handle other client's registration and deregistration.
    - The server broadcasts new registration and deregistration information to all registered clients.
    - When a client receives the broadcasting information, update the registration list.

- 20 points: The client can send, receive, and display a chat message.
    - Clients must chat directly with each other without relay of the server.
- 10 points: Enable clients under the same NAT can communicate using the same private network prefix (without NAT assistance)
- 10 points: For ‘@exit’ command
    - Clients send the deregistration request to the server and terminates.
    - The server displays the request and deletes the client in the registration list.
- 10 points: For client disappearance without sending a deregistration request.
    - The server must detect & display a client disappearance, and delete the client information in the registration list.
- 10 points: Report
    - 10 points for the well-written documentation.

7. Q&A

- Leave your questions on the google sheet.