Introduction to Computer Networks

Assignment 4: Solving NAT traversal problems

1. Goal

- Develop two programs; Registration Server and Client for chatting.
- Develop the UDP hole-punching solution for NAT traversals.

2. System Overview

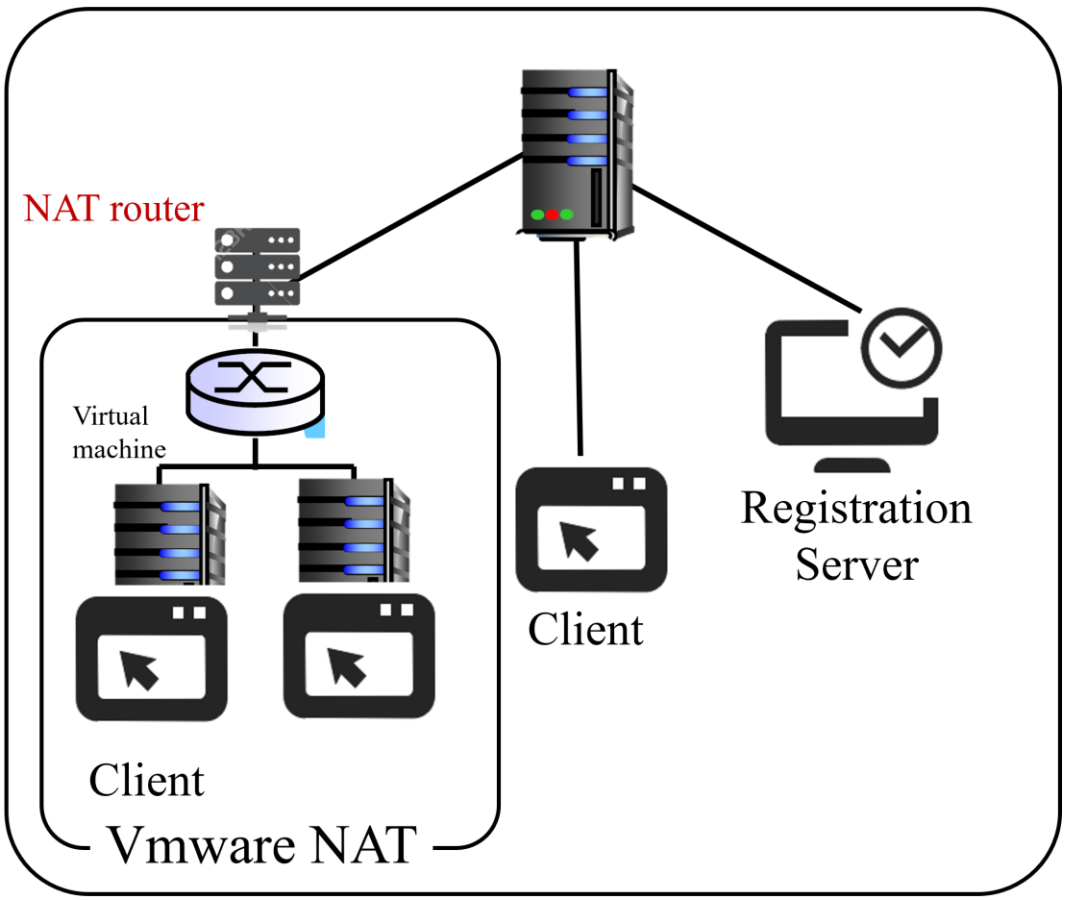

Figure 1. System configuration on a single computer

Using `VMware`, you can deploy a NAT environment. If you install multiple (at least two) VMs, they are under the same NAT. Run a client program on each VM, and run one client and a registration server on your host computer (outside NAT) .

Client programs register their IDs into the Registration Server program and retrieve current registered client information (including myself) from the Server. The clients can talk directly to other clients.

3. Development environments

- You can use C/C++ or Python (version 3.6+).
- For this assignment you must use `VMware` for NAT and virtual machines. Guideline is as follows.
  - You can install `VMware` on Linux or Window systems. `VMware` provides a NAT environment by default.
  - Install `VMware` : https://www.vmware.com/kr/products/workstation-player/workstation-player-evaluation.html
  - Install Ubuntu Virtual Machine (this guideline is using ubuntu desktop but you may use ubuntu server which is a lighter version depending on your computer capacity): https://theholmesoffice.com/installing-ubuntu-in-vmware-player-on-windows/
    - Ubuntu Server version : https://ubuntu.com/download/server
  - How to check my private IP address in Linux of Virtual Machine
    - If necessary, Install net-tools by type follows : 'sudo apt install net-tools'
    - Check private IP by the command : 'ifconfig'
- You have to describe your development environment information and how to run your programs in detail in the report. If not, TA may not evaluate your program properly and you will get zero points.

4. Functionalities to implement

- Client program
  - When starting the client program, enters the client ID and a server IP address. Assume that all clients have different IDs.
  - When starting the client program
    - The client program binds a UDP socket in the port number of 10081.
    - The client program sends a registration request with a client ID to the registration server using the server IP and 10080 port number.
    - When the client is registered for the first time, the server sends the registration list containing all registered client information.
  - The client must display all registered clients using the command '@show_list' as follows: (must follow the output format)

    client1 10.0.0.6:59465
    client2 10.0.0.5:45692

  - After the registration, the client must receive and update other client registration or unregistration information forwarded by the server.
  - After the registration, the client program can send/receive messages to/from other clients directly. If the message comes, display it with the sending client ID as follows:

    From client2 [I love Network!!!]

  - If the client wants to send a message to another client, use a command '@chat' with the opponent client ID as following;

    @chat client2 I love Network too!!!

  - Of course, clients can send and receive messages repeatedly.

- Client must send a keep-alive message to the registration server for ‘keep-alive’ purpose in every 10 seconds. (It is also required to maintain the NAT entry)

- You can use the ‘@exit’ command to send the unregistration request to the server and terminate the client program.

- Registration Server Program

  - Bind a UDP socket in the port number of 10080

  - When receiving a registration request display it, and send the current registration list to the client. Must forward the registration information to all other registered clients.

    client1 10.0.0.6: 59465

  - When receiving an unregistration request, display it and delete the client in the registration list. Must forward the unregistration information to other registered clients.

    client2 is unregistered 10.0.0.5: 45692

  - A client may be terminated without sending an unregistration request. To detect this, the server checks ‘keep alive’ messages periodically sent from clients. If this message is not received more than 30 seconds, display the ‘off-line’, and delete the client in the registration list. Must forward the off-line information to other registered clients.

    client1 is off-line 10.0.0.6: 59465

- Extended Goal

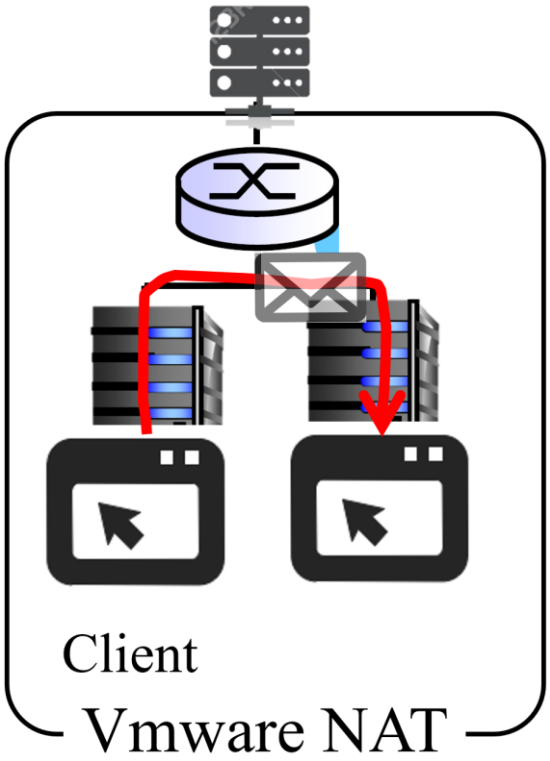

Figure 2. Overview of message transfer in same NAT network

  - If both clients are under the same NAT, let the clients communicate each other with private IP addresses (not using the NAT public IP address).
  - You can design/upgrade the protocol to achieve this extended goal.

- Miscellaneous
  - Assume that client IDs consist of alphabet and number without any white spaces. And its length does not exceed 32 bytes.
  - Assume that the size of a chatting message is less than 200 letters.
  - Using a Wireshark, you can check source and destination IP addresses & port numbers of packets.

5. Sample Results

Client 1	Client 2	Registration Server
# client1 starts		client1 7.7.7.7:77
@ show _list client1 7.7.7.7:77		
	# client 2 starts	client2 8.8.8.8:88
	@ show _list client1 7.7.7.7:77 client2 8.8.8.8:88	
@show_list client1 7.7.7.7:77 client2 8.8.8.8:88		
From client2 [hello]	@chat client1 hello	
@chat client2 nice to meet you	From client1 [nice to meet you]	
@exit # client1 terminates		client1 is unregistered
	@ show _list client2 8.8.8.8:88	
	# stop the program by Ctrl-C	# checking time-out client2 is off-line

6. Submission

- The deadline is **6.14 (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 receive 0 points for the first time and ***F*** for the second.
- Submit a zip file including a **report** and two (sender and receiver) program sources to iCampus
- Submit a zip file including a **report** and two program (client and server) source codes (and Makefile for C/C++) to iCampus
  - Name the zip file ***StudentID_Name.zip*** (ex: 2018001_홍길동.zip)
  - The report file format should be PDF, and the file name is also ***StudentID_Name.pdf*** (ex: 2020101_홍길동.pdf)
  - The report has to include the following things;
    1) Describe your development environment information in detail (versions of operating systems, languages, compilers/interpreter versions, compile options)
    2) Present how to design your assignment such as data structures and algorithms.
    3) Explain how to run both server and client programs including the screen capture.

7. Scoring

- Total 100 points
  - 20 points: The client registers to the registration server.
    - The client sends registration request to the server
    - The server display the client information
  - 10 points: The client sends 'keep alive' message every 10 seconds.
  - 10 points: The client receives the information of other clients from the server.
    - When the client is registered for the first time, receive the registration list

from the server.

- After the registration, receive individual client registration or unregistration information from the server.

- 20 points: The client can send and receive a chat message (and display it)
  - Clients must communicate each other directly without the server's relaying.
- 10 points: For '@exit' command
  - Clients must send the unregistration request to the server, and terminate
  - The server display it, delete the client in the registration list, and forward the information to other registered clients.
- 10 points: For client termination without sending an unregistration request.
  - The server detects the termination and display it, delete the client in the registration list, and forward the information to other registered clients.
- 10 points: Enable clients, under the same NAT, to communicate each other using private IP addresses (NOT traversing through the NAT).
- 10 points: Report
  - 10 points for the well-written documentation.

8. Q&A

- Leave your questions on the google sheet