Project3: Proxy Server and Network Address Translation (NAT)

M2608.001200 Introduction to Data Communication Networks (2022 Fall)

Instructors: Prof. Kyunghan Lee

TAs: Gibum Park, Jongseok Park, Sanghyeon Han

Due date: Dec. 24, 2022, 11:59 AM.

Notes:

- This project is optional. The project score will be given by adding up the two highest scores among projects 1, 2, and 3.
- Be aware of plagiarism. You can use the ETL for QnA, but do **NOT** discuss it with your classmates **directly**.
- **Grading**: 200 pts total
- If you find bugs, typos, or issues in the given code, please report them to the TAs by posting on the ETL. We will look into them and fix them as soon as possible. Once fixed, we will upload the new version on the ETL by naming the proper version number for the code and making an announcement on the ETL. Please pay attention to it and make sure you are using the latest version before submission.
- If you do not want other students to see your questions, you can always make them private.
- You'll be deducted 10% per 12 hour delay. We do not accept submissions after 11:59 pm on the 26th.

Objective

A proxy server is a server that clients use to access other computers. A client who requests some services connects to the proxy server. Then, the proxy server forwards the request to a different server that has available the requested service. There are many benefits of using a proxy server. The main advantage of proxy is anonymizing. The client can exchange data with the remote server without a direct connection. Therefore, it can hide the client's IP address from the remote server.
Network Address Translation (NAT) also can make the client anonymous. NAT is a method of mapping an IP address into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device. It is usually used to translate private IP to public IP to protect the private network.
The goal of this project is to implement a simple HTTP proxy server and create NAT rules. With this project, you can understand how proxy and NAT work and the difference between a proxy and NAT.

1. Implementation of a Simple HTTP Proxy Server

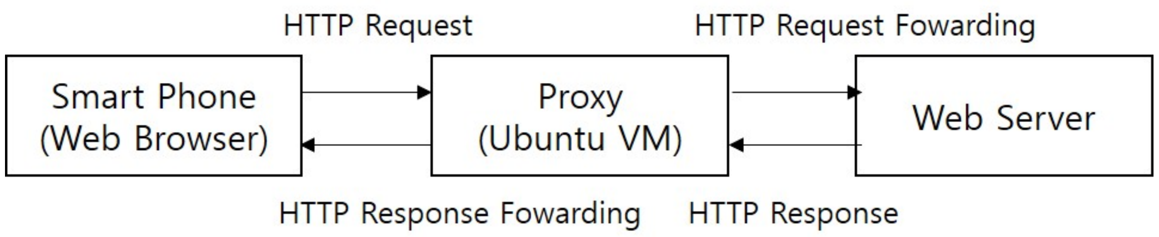


Figure 1: Overview of the HTTP proxy server

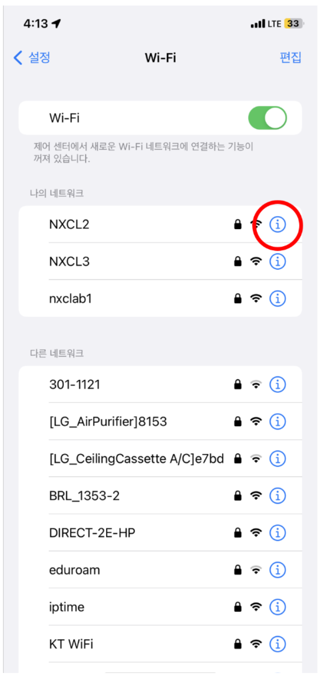
1) Preparation

On windows WSL (You can skip this part if you are using native Linux machine or mac)

1. open PowerShell as administrator
2. Type command “netsh interface portproxy add v4tov4 listenport=7777 listenaddress=0.0.0.0 connectport=7777 connectaddress=172.29.203.200” (replace 172.29.203.200 with IP address of your VM)
3. open control panel(제어판) => Window Defender firewall(Window Defender 방화벽) => advanced setting(고급설정) => inbound rules(인바운드 규칙)=> New rules(새 규칙) => Port(포트)=>Specific local port 7777(특정로컬포트) => Allow connection(연결허용) => next => put the name in the name field(“7777_port” for example)
   (NOTE: delete this rule after the project)

On smartphones

1. Connect VM and smartphone to the same AP (e.g., kt_SNU)
2. When you connect the smartphone to the AP, activate the advanced proxy option in the WiFi menu.



4:13 LTE 33

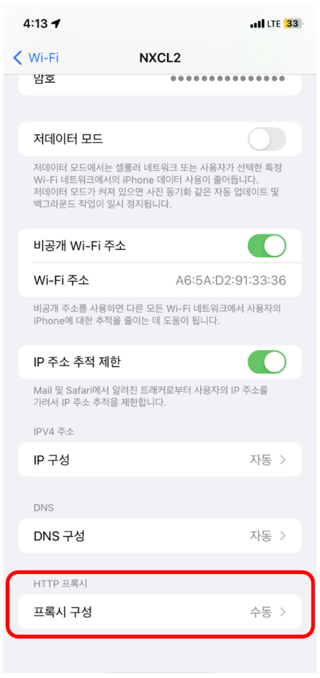
설정 | Wi-Fi | 편집

Wi-Fi

제어 센터에서 새로운 Wi-Fi 네트워크에 연결하는 기능이 꺼져 있습니다.

나의 네트워크
- NXCL2
- NXCL3
- nxclab1

다른 네트워크
- 301-1121
- [LG_AirPurifier]8153
- [LG_CeilingCassette A/C]e7bd
- BRL_1353-2
- DIRECT-2E-HP
- eduroam
- iptime
- KT WiFi



4:13 LTE 33

Wi-Fi | NXCL2

암호 ••••••••••••••••

저데이터 모드

저데이터 모드에서는 셀룰러 네트워크 또는 사용자가 선택한 특정 Wi-Fi 네트워크에서의 iPhone 데이터 사용이 줄어듭니다. 저데이터 모드가 켜져 있으면 사진 동기화 같은 자동 업데이트 및 백그라운드 작업이 일시 정지됩니다.

비공개 Wi-Fi 주소

Wi-Fi 주소 A6:5A:D2:91:33:36

비공개 주소를 사용하면 다른 모든 Wi-Fi 네트워크에서 사용자의 iPhone에 대한 추적을 줄이는 데 도움이 됩니다.

IP 주소 추적 제한

Mail 및 Safari에서 알려진 트래커로부터 사용자의 IP 주소를 가려서 IP 주소 추적을 제한합니다.

IPV4 주소

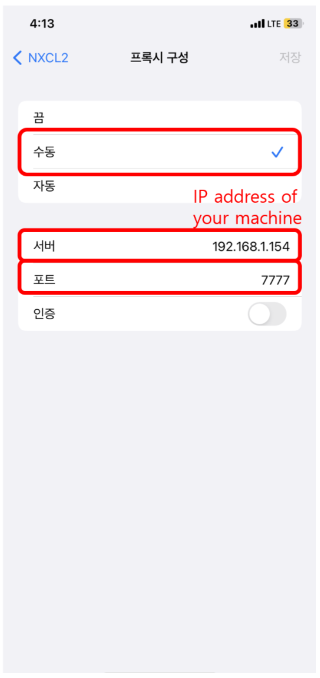
IP 구성 자동 >

DNS

DNS 구성 자동 >

HTTP 프록시

프록시 구성 수동 >



4:13 LTE 33

NXCL2 | 프록시 구성 | 저장

끔
[✓ 수동]
자동

IP address of your machine

서버 192.168.1.154

포트 7777

인증

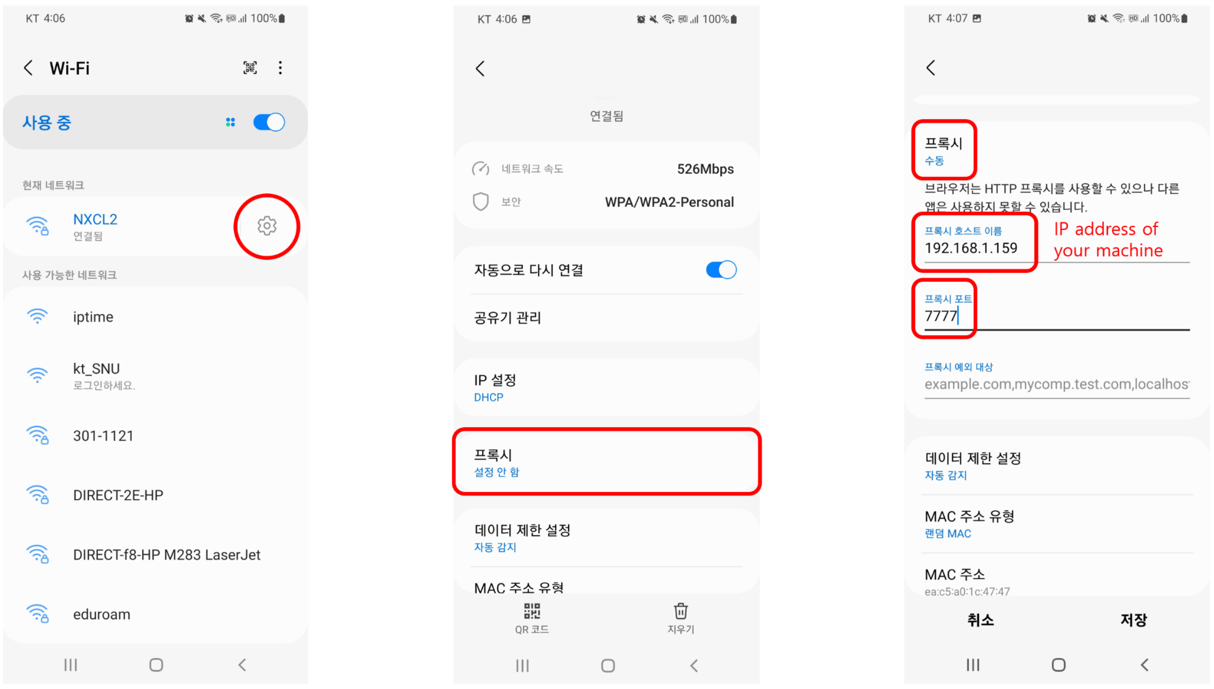
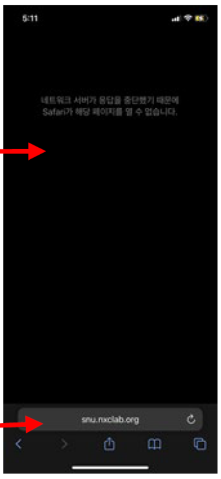
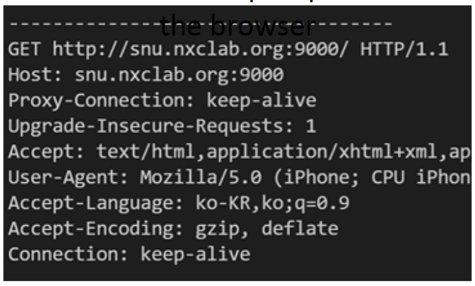


Figure 2: WiFi proxy setting

Run skeleton code

1. Compile skeleton code (server.c) and run the proxy server.
   1.1. Type make in terminal
   1.2. Type ./server to run server.
2. Type in the server address ("http://snu.nxclab.org:9000") in the web browser and check that the HTTP proxy server receives the request successfully from the browser. (※ test with "http://" request not "https://" because you need to implement complicated functionality for https requests)
3. You can check that the proxy server shows requests from the browser. (snu.nxclab.org:9000)


Browser cannot show the page yet because forwarding is not implemented in the skeleton code.
Send request to http://snu.nxclab.org:9000
5:11
네트워크 서버가 응답을 중단했기 때문에 Safari가 해당 페이지를 열 수 없습니다.
snu.nxclab.org



```
-------------------------------
GET http://snu.nxclab.org:9000/ HTTP/1.1
Host: snu.nxclab.org:9000
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,ap
User-Agent: Mozilla/5.0 (iPhone; CPU iPhon
Accept-Language: ko-KR,ko;q=0.9
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Figure 3: Example of sending a request to the web server via skeleton proxy server.

2) Implementation of request and response forwarding

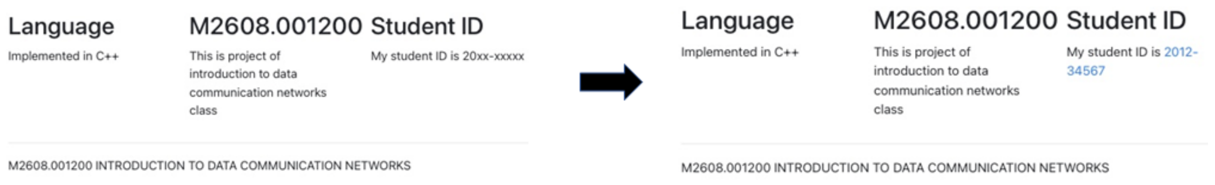
a) Forward received HTTP request to the web server (test with http://snu.nxclab.org:9000)
   i) HInt1: The HTTP request has a hostname in the path. Take the IP address with the hostname and open a socket with the address. (use gethostbyname() function)
   ii) Hint 2: You should modify the path of the request header before forwarding it to a web server. e.g. "GET http://snu.nxclab.org:9000/ HTTP 1.1" -> "GET / HTTP/1.1"

b) Receive responses from the web server and forward them to the browser.

c) If you implement all of these functionalities well, the smartphone browser will show the web page. (Please check if the browser receives images and JS/CSS files correctly. It is essential for your grade)

3) Implementation of content modification in the proxy server

a) Modify HTML file before response forwarding. Then, the web browser in your smartphone shows the modified content. Replace the student id 20xx-xxxxx with your student id.



Language
Implemented in C++
M2608.001200 Student ID
This is project of introduction to data communication networks class
My student ID is 20xx-xxxxx
M2608.001200 INTRODUCTION TO DATA COMMUNICATION NETWORKS

Language
Implemented in C++
M2608.001200 Student ID
This is project of introduction to data communication networks class
My student ID is 2012-34567
M2608.001200 INTRODUCTION TO DATA COMMUNICATION NETWORKS

Figure 4: Example of content modification in proxy server.

b) Change the order of the images (you can change it however you want)

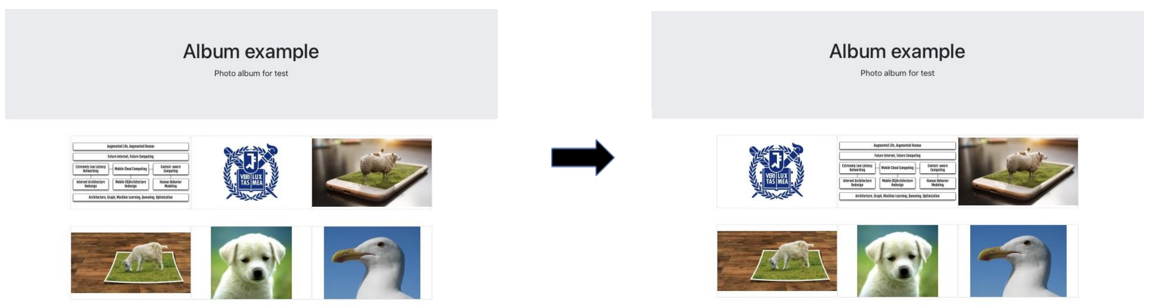


Figure 5: Example of content modification in proxy server.

2. Network Address Translation (NAT)

The concept of NAT in a router, which we learned in the class, assigns a random port to a public address to manage private IP. But NAT in this project changes one network address to the other literally. It just modifies the destination, the source address, or the port number of the packets from the client and delivers them to the server. By Figure 6, you can understand the mechanism of this simple proxy more easily. You will set up your own Ubuntu as an NAT proxy and send a request through your proxy for receiving web pages by your smartphone. You will capture the packets in your proxy and report how this proxy modifies the packets and discuss the impact of this proxy.

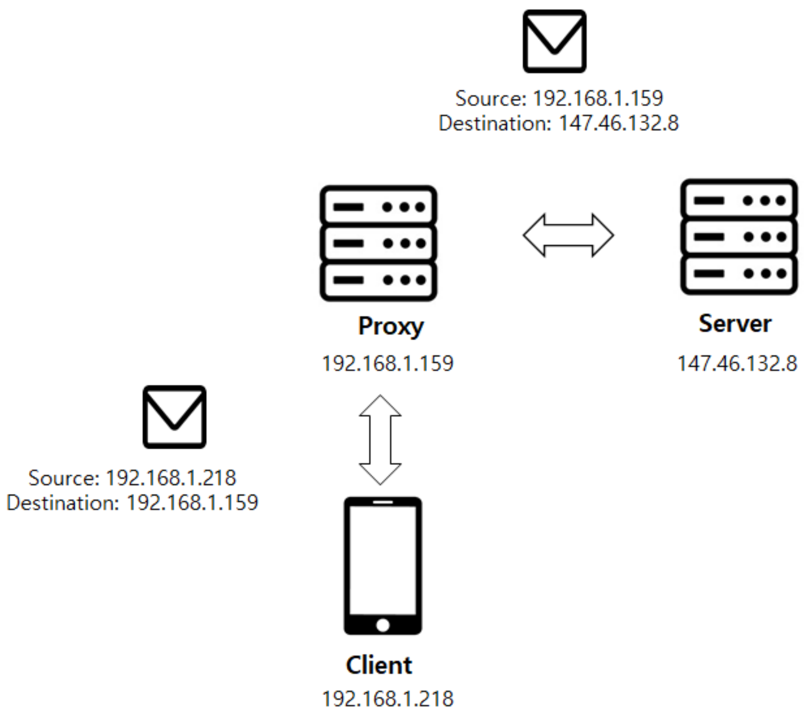


Figure 6: Client sends a packet to a proxy and the proxy modifies source and destination of packets. Then deliver the packet to the server. Then, with this proxy server, the client and the server do not need to know their IP address. Therefore, this proxy hides the server from the client.

Background Knowledge: iptables

(You should use pfctl instead of iptables on your macOS. Its usage is described in the green box)

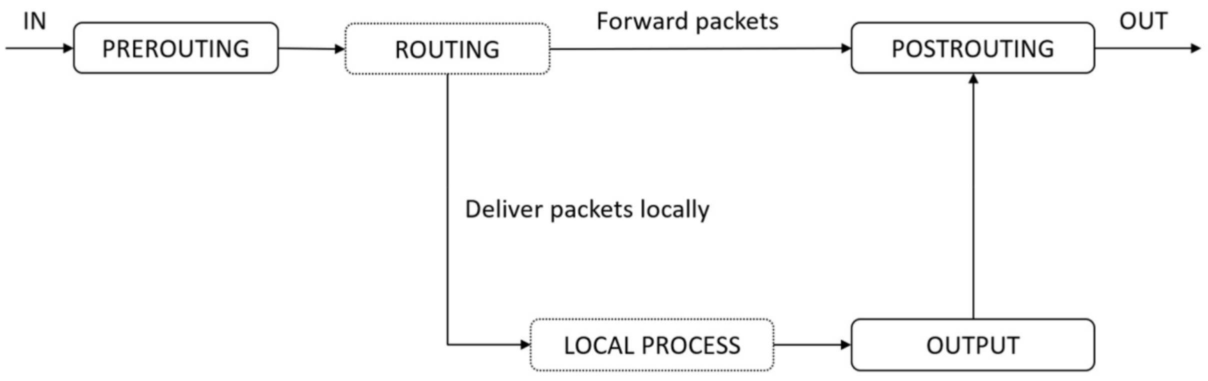


Figure 7: Overview of iptables. PREROUTING chain is responsible for packets that just arrived at the network interface. If you append a rule at PREROUTING, the rule applies to packets before being routed. After the packets are routed, they pass the POSTROUTING chain and then leave through the network interface. Therefore, if you append a rule at POSTROUTING, the rule applies to packets after being routed.

You will use the command utility 'iptables' to create NAT rules in the 'nat' table. This table has three predefined chains: PREROUTING, OUTPUT, and POSTROUTING, as in Figure 7. In this project, you will use PREROUTING and POSTROUTING chains. **(Depending on the version, the iptables command might not work properly. In that case, replace iptables with iptables-legacy.)**

```
How to set rules
-t nat - select table 'nat' for configuration NAT rules.
-A [chain] - append a rule to the corresponding chain.
-i [network interface] - select network interface for input packets.
-o [network interface] - select network interface for output packets.
-j [action] - set action for packets.
--to-destination [IP address] - set destination IP address.
--to-source [IP address] - set source IP address.
```

e.g. Modify destination of packets to 1.2.3.4

```
Iptables -t nat -A PREROUTING -j DNAT –to-destination 1.2.3.4 e.g.
or
Iptables-legacy -t nat -A PREROUTING -j DNAT –to-destination 1.2.3.4 e.g.
```

Ex) Modify the source of packets to 1.2.3.4

```
Iptables -t nat -A POSTROUTING -o wlp3s0 -j SNAT –to-source 1.2.3.4
or
Iptables-legacy -t nat -A POSTROUTING -o wlp3s0 -j SNAT –to-source 1.2.3.4
```

*** DNAT (Destination Network Address Translation): Modify destination of the packets.**
*** SNAT (Source Network Address Translation): Modify the source of the packets.**

1) Enabling IP Forwarding

To set up your Ubuntu as a NAT proxy, you should enable IP forwarding.

One line command

```
sudo sysctl -w net.ipv4.ip_forward=1
```

or

```
sudo echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Reset your iptables (or replace with iptables-legacy)

```
sudo iptables -F
sudo iptables -X
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
```

on macOS :

enable IP forwarding.

```
sudo sysctl -w net.inet.ip.forwarding=1
```

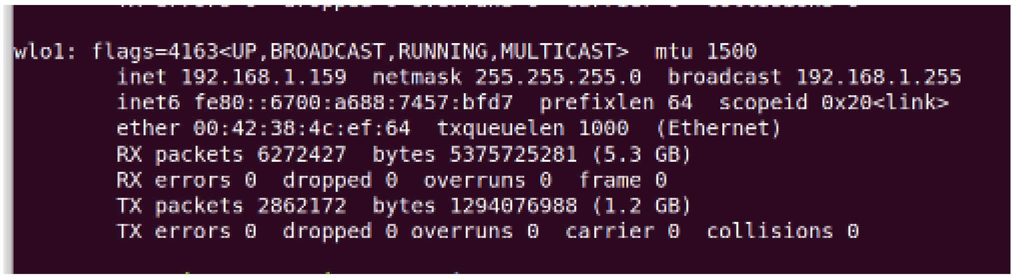
Revert to Default Settings on /etc/pf.conf

```
sudo pfctl -F all -f /etc/pf.conf
```

(Flush(-F) all setting and load(-f) /etc/pf.conf)

2) Set Forwarding Rules on your Proxy

- Connect your proxy (Ubuntu VM) and smartphone to same WIFI network.
- Take the IP address and network interface name (e.g. wlo1 or wlp3s0) of your proxy with "ifconfig" command line.



```
wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.159  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::6700:a688:7457:bfd7  prefixlen 64  scopeid 0x20<link>
        ether 00:42:38:4c:ef:64  txqueuelen 1000  (Ethernet)
        RX packets 6272427  bytes 5375725281 (5.3 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2862172  bytes 1294076988 (1.2 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- (Rule1) Append DNAT rule to the PREROUTING Table

```
sudo iptables -t nat -A PREROUTING -j DNAT -p tcp --to-destination 147.46.132.8:9000
or
sudo iptables-legacy -t nat -A PREROUTING -j DNAT -p tcp --to-destination 147.46.132.8:9000
```

*** 147.46.132.8 : web page address (use this for testing)**

- (Rule2) Append SNAT rule to the POSTROUTING Table

```
sudo iptables -t nat -A POSTROUTING -j SNAT -o eth0 --to-source 192.168.1.159
or
sudo iptables-legacy -t nat -A POSTROUTING -j SNAT -o eth0 --to-source 192.168.1.159
```

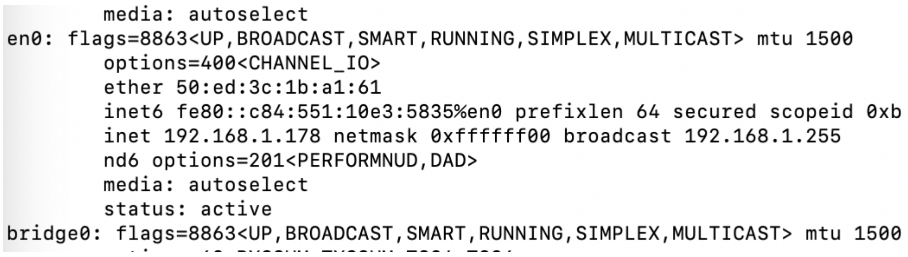
*** 192.168.1.159 : change this IP to your VM's IP.**

- Check that you appended two rules correctly

```
sudo iptables -t nat --list
or
sudo iptables-legacy -t nat --list
```

on macOS

- Connect your proxy (macOS machine) and smartphone to the same WIFI network.
- Take the IP address and network interface name (e.g. en0) of your proxy with "ifconfig" command line.



```
        media: autoselect
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 50:ed:3c:1b:a1:61
        inet6 fe80::c84:551:10e3:5835%en0 prefixlen 64 secured scopeid 0xb
        inet 192.168.1.178 netmask 0xffffff00 broadcast 192.168.1.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
```

- Append 2 rules at once

```
echo " rdr on en0 inet proto tcp from 192.168.1.164 to 192.168.1.178 port 7777 -> 147.46.132.8
port 9000 nat on en0 inet proto tcp from 192.168.1.164 to 147.46.132.8 port 9000 ->
192.168.1.178 " | sudo pfctl -ef -
```

***192.168.1.164: change this IP to your smartphone IP**
***192.168.1.178: change this IP to your macOS machine IP**
***147.46.13.8: web page address (use this for testing)**
command above will add two rules
rdr on en0 inet proto tcp from 192.168.1.164 to 192.168.1.178 port 7777 -> 147.46.132.8 port 9000
(Act like prerouting on iptables)
and
nat on en0 inet proto tcp from 192.168.1.164 to 147.46.132.8 port 9000 -> 192.168.1.178
(Act like postrouting on iptables)

- Check that you appended two rules correctly

```
sudo sudo pfctl -s nat
```

- Revert to Default Settings on /etc/pf.conf

```
sudo pfctl -F all -f /etc/pf.conf
```

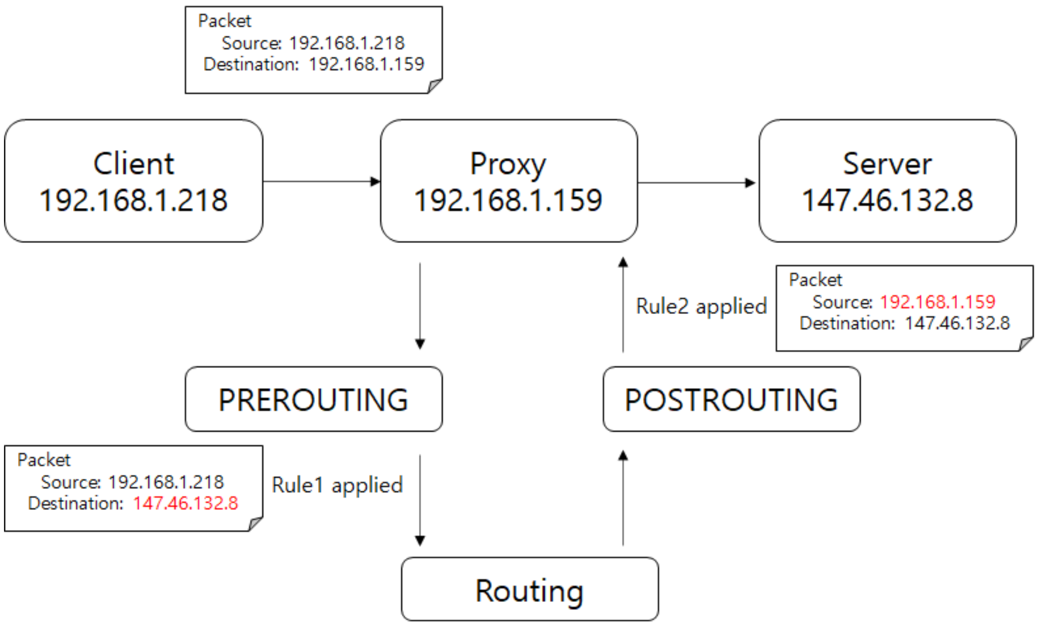


Figure 8: Overview of iptables operation.

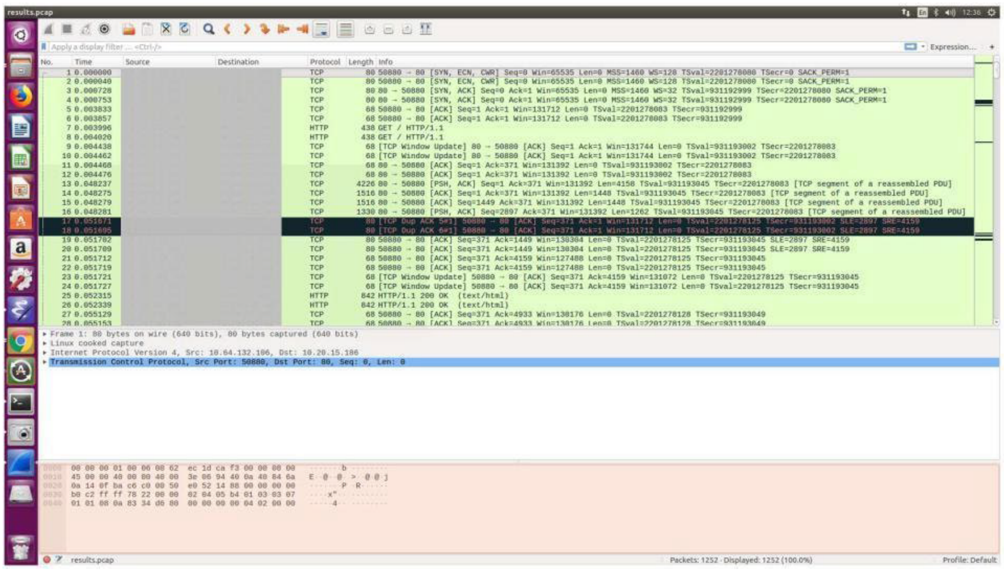
3) Packet capture with tcpdump & wireshark

(NOTE: this process will not be scored you can skip this part if you want)
You can capture packets which pass through network interface with 'tcpdump' and show the results with 'wireshark.'
After setting up the rules, execute 'tcpdump' with following command line and request to proxy by your smartphone browser.

```
sudo tcpdump -i eth0 tcp port 9000 -w results_1.pcap
```

(Same command on macOS)

If your browser show web page successfully, stop 'tcpdump'(press Ctrl+C).
Drag results.pcap to Wireshark in Windows/MAC



```
results.pcap
Apply a display filter ... <Ctrl-/>                                          Expression...
No.  Time      Source  Destination  Protocol Length Info
 1 0.000000   [redacted]            TCP    80 50880 → 80 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=128 TSval=2201278080 TSecr=0 SACK_PERM=1
 2 0.000040                         TCP    80 50880 → 80 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=128 TSval=2201278080 TSecr=0 SACK_PERM=1
 3 0.000728                         TCP    80 80 → 50880 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=32 TSval=931192999 TSecr=2201278080 SACK_PERM=1
 4 0.000753                         TCP    80 80 → 50880 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=32 TSval=931192999 TSecr=2201278080 SACK_PERM=1
 5 0.003833                         TCP    68 50880 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2201278083 TSecr=931192999
 6 0.003857                         TCP    68 50880 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2201278083 TSecr=931192999
 7 0.003996                         HTTP  438 GET / HTTP/1.1
 8 0.004020                         HTTP  438 GET / HTTP/1.1
 9 0.004438                         TCP    68 [TCP Window Update] 80 → 50880 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=931193002 TSecr=2201278083
10 0.004462                         TCP    68 [TCP Window Update] 80 → 50880 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=931193002 TSecr=2201278083
11 0.004468                         TCP    68 80 → 50880 [ACK] Seq=1 Ack=371 Win=131392 Len=0 TSval=931193002 TSecr=2201278083
12 0.004476                         TCP    68 80 → 50880 [ACK] Seq=1 Ack=371 Win=131392 Len=0 TSval=931193002 TSecr=2201278083
13 0.048237                         TCP  4226 80 → 50880 [PSH, ACK] Seq=1 Ack=371 Win=131392 Len=4158 TSval=931193045 TSecr=2201278083 [TCP segment of a reassembled PDU]
14 0.048275                         TCP  1516 80 → 50880 [ACK] Seq=1 Ack=371 Win=131392 Len=1448 TSval=931193045 TSecr=2201278083 [TCP segment of a reassembled PDU]
15 0.048279                         TCP  1516 80 → 50880 [ACK] Seq=1449 Ack=371 Win=131392 Len=1448 TSval=931193045 TSecr=2201278083 [TCP segment of a reassembled PDU]
16 0.048281                         TCP  1330 80 → 50880 [PSH, ACK] Seq=2897 Ack=371 Win=131392 Len=1262 TSval=931193045 TSecr=2201278083 [TCP segment of a reassembled PDU]
17 0.051671                         TCP    80 [TCP Dup ACK 5#1] 50880 → 80 [ACK] Seq=371 Ack=1 Win=131712 Len=0 TSval=2201278125 TSecr=931193002 SLE=2897 SRE=4159
18 0.051695                         TCP    80 [TCP Dup ACK 6#1] 50880 → 80 [ACK] Seq=371 Ack=1 Win=131712 Len=0 TSval=2201278125 TSecr=931193002 SLE=2897 SRE=4159
19 0.051702                         TCP    80 50880 → 80 [ACK] Seq=371 Ack=1449 Win=130304 Len=0 TSval=2201278125 TSecr=931193045 SLE=2897 SRE=4159
20 0.051709                         TCP    80 50880 → 80 [ACK] Seq=371 Ack=1449 Win=130304 Len=0 TSval=2201278125 TSecr=931193045 SLE=2897 SRE=4159
21 0.051712                         TCP    68 50880 → 80 [ACK] Seq=371 Ack=4159 Win=127488 Len=0 TSval=2201278125 TSecr=931193045
22 0.051719                         TCP    68 50880 → 80 [ACK] Seq=371 Ack=4159 Win=127488 Len=0 TSval=2201278125 TSecr=931193045
23 0.051721                         TCP    68 [TCP Window Update] 50880 → 80 [ACK] Seq=371 Ack=4159 Win=131072 Len=0 TSval=2201278125 TSecr=931193045
24 0.051727                         TCP    68 [TCP Window Update] 50880 → 80 [ACK] Seq=371 Ack=4159 Win=131072 Len=0 TSval=2201278125 TSecr=931193045
25 0.052315                         HTTP  842 HTTP/1.1 200 OK  (text/html)
26 0.052339                         HTTP  842 HTTP/1.1 200 OK  (text/html)
27 0.055129                         TCP    68 50880 → 80 [ACK] Seq=371 Ack=4933 Win=130176 Len=0 TSval=2201278128 TSecr=931193049
28 0.055153                         TCP    68 50880 → 80 [ACK] Seq=371 Ack=4933 Win=130176 Len=0 TSval=2201278128 TSecr=931193049

▸ Frame 1: 80 bytes on wire (640 bits), 80 bytes captured (640 bits)
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.64.132.106, Dst: 10.20.15.186
▸ Transmission Control Protocol, Src Port: 50880, Dst Port: 80, Seq: 0, Len: 0

0000  00 00 00 01 00 06 00 62  ec 1d ca f3 00 00 00 00   .......b ........
0010  45 00 00 40 00 00 40 00  3e 06 94 40 0a 40 84 6a   E..@..@. >..@.@.j
0020  0a 14 0f ba c6 c0 00 50  e0 52 14 88 00 00 00 00   .......P .R......
0030  b0 c2 ff ff 78 22 00 00  02 04 05 b4 01 03 03 07   ....x".. ........
0040  01 01 08 0a 83 34 d6 80  00 00 00 00 04 02 00 00   .....4.. ........

results.pcap                         Packets: 1252 · Displayed: 1252 (100.0%)          Profile: Default
```

Figure 9: Example of Wireshark

4) Report

So far, you have implemented a simple HTTP proxy server and NAT rules. In the report, you should explain how you implemented them in detail. Then, you should list the pros and cons of proxy and NAT and compare them. Finally, you need to suggest a simple web page caching HTTP proxy with pseudocode. If you have implemented HTTP proxy and NAT, you can understand them enough to write your report.

3. Grading Policy

- **[HTTP Proxy Server] Forwarding: 40%**
- **[HTTP Proxy Server] Content Modification: 30%**
- **[NAT] Report: 30%**

4. Submission

- **Make a directory “proxyserver_<student_id>” and compress to a zip file.**
- **A zip file should contain the report and your codes.**
- **Submit the zip file to ETL.**