

Computer Networks Project 3

2016147030 송민석

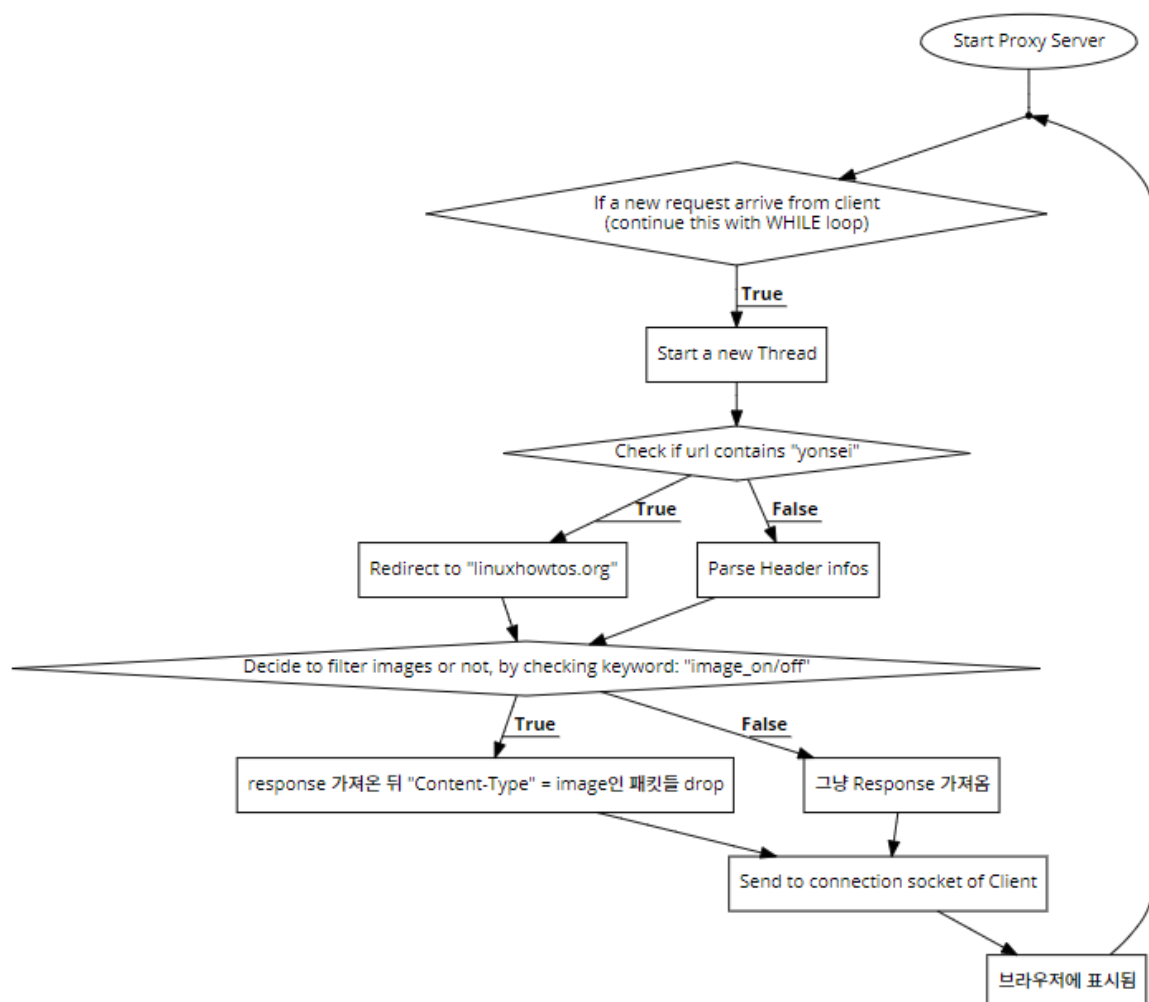
1. 개발 환경

Language : Python 3.8.5

OS : Ubuntu 20.04 LTS (WSL2)

Used (only internal) Libraries : threading, socket, sys, time

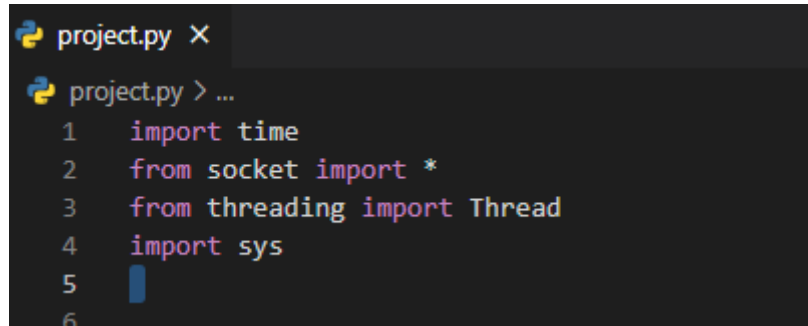
2. Flowchart



project.py에 구현된 Proxy Server의 작동 흐름은 위 flowchart와 같다. 해당 .py파일, 즉 서버가 시작된 후, 이 proxy server는 꾸준히 client로부터 요청을 listen한다. 이후의 과정들은 요청 하나당 하나의 Thread로 묶어서 처리한다. url 주소에 "Yonsei"가 포함된 요청의 경우, linuxhowtos.org로

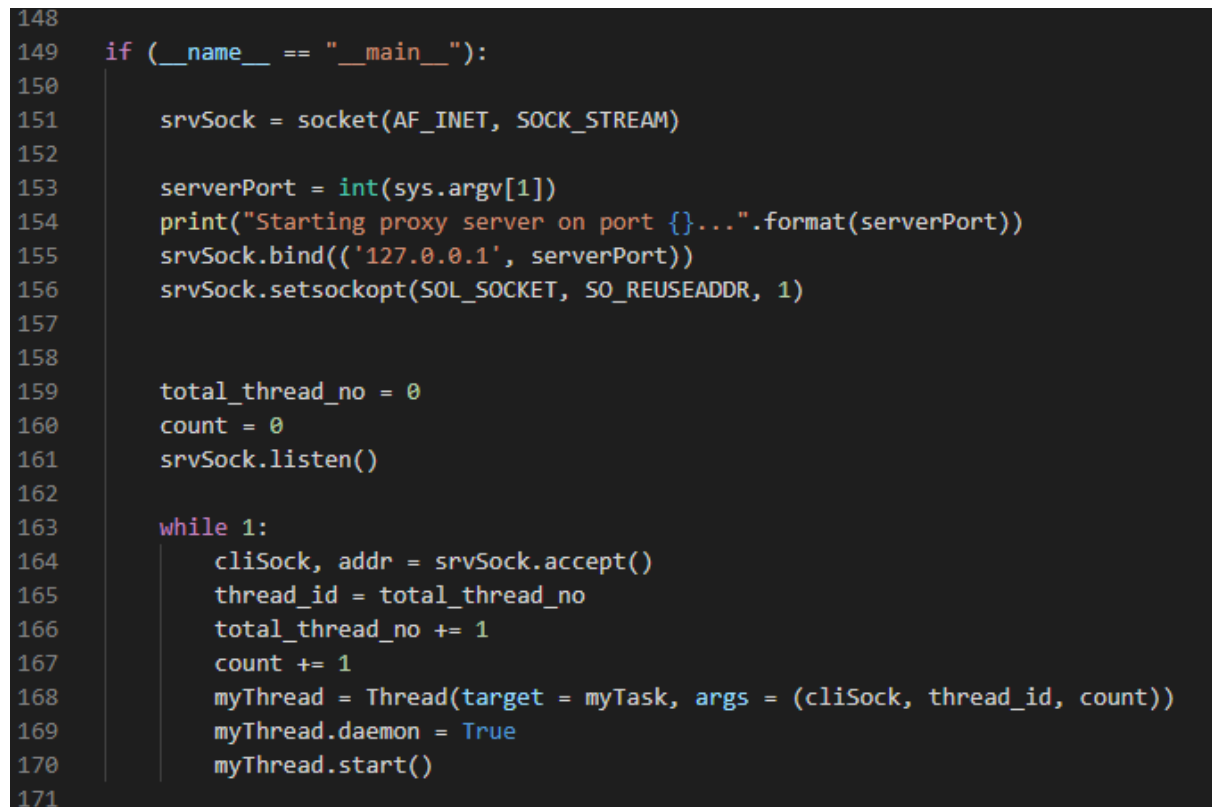
redirect하기 위해 특정 헤더값들을 변경하여 실제 서버에 response를 요청한다. 이 때 image_off가 url에 포함되어 있으면 response들 중에 image type이 아닌 response만을 남겨 클라이언트 측 소켓에 전달한다. 이렇게 두 가지 필터를 거친 패킷 정보들이 최종적으로 브라우저에 표시된다.

3. Block by Block Code Explanation



```
project.py X
project.py > ...
1  import time
2  from socket import *
3  from threading import Thread
4  import sys
5
6
```

필요한 라이브러리들을 import 하는 부분이다.



```
148
149  if (__name__ == "__main__"):
150
151      srvSock = socket(AF_INET, SOCK_STREAM)
152
153      serverPort = int(sys.argv[1])
154      print("Starting proxy server on port {}".format(serverPort))
155      srvSock.bind(('127.0.0.1', serverPort))
156      srvSock.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
157
158
159      total_thread_no = 0
160      count = 0
161      srvSock.listen()
162
163      while 1:
164          cliSock, addr = srvSock.accept()
165          thread_id = total_thread_no
166          total_thread_no += 1
167          count += 1
168          myThread = Thread(target = myTask, args = (cliSock, thread_id, count))
169          myThread.daemon = True
170          myThread.start()
171
```

Main함수는 위와 같다. 지속적으로 client로부터 request를 수신할 메인 서버 소켓인 srvSock 소켓을 생성하여 input으로 주어진 port에 bind하고, while loop을 이용하여 꾸준히 요청을 받아들이는 다. 이후 요청이 accept되는 경우 새 thread를 시작하여 proxy server 이후의 단계를 처리한다.

```

project.py X
project.py > myTask
35 def myTask(connSock, thread_id, count):
36     log = []
37     global total_thread_no
38     urlFilter = "X"
39     imgFilter = "X"
40     try:
41         # CLI ==> PRX
42         data = connSock.recv(8192).decode('utf-8', errors = 'ignore')
43         data_lines = data.split("\r\n")
44         # print("Request Length : {}".format(len(data)))
45         if ("GET" in data) and ("windowupdate" not in data) and ("microsoft.com" not in data):
46             if ("yonsei" in data):
47                 urlFilter = "O"
48                 log.append("{} [Conn: {}/ {}]".format(count, thread_id, total_thread_no))
49                 # print("{} [Conn: {}/ {}]".format(count, thread_id, total_thread_no))
50                 log.append("[ {} ] URL filter | [ {} ] Image filter\n".format(urlFilter, imgFilter))
51                 # print("[ {} ] URL filter | [ {} ] Image filter\n".format(urlFilter, imgFilter))
52                 log.append("[CLI connected to {}:{}]".format(addr[0], addr[1]))
53                 # print("[CLI connected to {}:{}]".format(addr[0], addr[1]))
54                 log.append("[CLI ==> PRX --- SRV]")
55                 # print("[CLI ==> PRX --- SRV]")
56                 data_dict = {}
57                 # print(data)
58                 for line in data_lines:
59                     line_split = line.split(" ")
60                     data_dict[line_split[0]] = line_split[1:]
61                     # print (line)
62                 method_message = data_dict['GET'][0]
63                 log.append("> GET " + method_message)
64                 # print("> GET", method_message)
65                 userAgent = data_dict['User-Agent:']
66                 userAgent_message = ""
67                 for word in userAgent:
68                     userAgent_message += (word+" ")
69                 log.append("> " + userAgent_message)
70                 # print('>', userAgent_message)
71

```

먼저 Client로부터 수신된 request의 정보를 파악하는 부분이다. GET에 대해서만 동작하고, windowupdate등 브라우저로 사용자가 직접 요청한 것이 아닌 요청들을 걸러내기 위해 if문으로 필터링 해주었다. 이후, yonsei가 포함되는 경우 urlFilter의 값을 O로 설정해주어 이후 redirect가 가능하도록 하였다. thread들끼리 print하는 사이에 꼬이지 않도록 log라는 list안에 출력 양식이 line by line으로 append되고 마지막에 한번에 출력하도록 하였다. 이 부분에서는 Request로부터 url, host 등 출력에 필요한 값들을 log에 저장하고, 이 request를 다시 본래 server혹은 redirect할 server로 보낼 수 있도록 준비한다.

```
project.py X
project.py > myTask

72
73     # PRX ==> SRV
74     url = data_dict['GET'][0].split(" ")[0]
75     if ("?image_off" in url):
76         imgFilter = "0"
77     elif ("?image_on" in url):
78         imgFilter = "X"
79     host = data_dict['Host:'][0]
80     outerSock = socket(AF_INET, SOCK_STREAM)
81     outerSock.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
82     req = data.split("\r\n")
83
84     # Redirect if "yonsei" is included
85     if (urlFilter == "0"):
86         url = "www.linuxhowtos.org"
87         host = "linuxhowtos.org"
88         req[0] = "GET http://linuxhowtos.org/ HTTP/1.1"
89         req[1] = "Host: linuxhowtos.org"
90
91     log.append("[SRV connected to {}]:80".format(host))
92     log.append("[CLI --- PRX ==> SRV]")
93     outerSock.connect((host, 80))
94     data_toServer = ""
95     for line in req:
96         data_toServer += line+"\r\n"
97     log.append("> GET " + url)
98     log.append("> " + userAgent_message)
99     outerSock.send(data_toServer.encode("utf-8", errors = 'ignore'))
100
101
```

다음은 프록시 서버로부터 실제 서버로 요청을 전송하는 부분이다. 앞에서 받은 request를 서버로 전달하는데, image_off가 url에 포함되어있으면 imgFilter 변수를 0로 설정하여 이후 과정에서 image를 drop할 수 있도록 하였다. Yonsei가 포함되어 있다면 urlFilter가 0일 것이므로, url, host값을 변경하고 그렇지 않은 경우 그대로 req 변수에 저장된 request정보를 실제 서버와 소통하는 outerSock 소켓에 전달한다.

```
project.py X
project.py > myTask
101
102     # PRX <= SRV
103     log.append("[CLI --- PRX <= SRV]")
104     res = []
105     response_raw = recv_timeout(outerSock)
106     response_string = response_raw.decode("utf-8", errors = 'ignore')
107     response_lines = response_string.split("\r\n")
108     response_dict = {}
109     status = ""
110     contentInfo = ""
111     for i in range(len(response_lines)):
112         line = response_lines[i]
113         if (i == 0):
114             status = line
115             log.append("> "+status[9:])
116         elif ("Content-Type" in line):
117             contentInfo += line[14:]
118         elif ("Content-Length" in line):
119             contentInfo += line.split(" ")[1] + " bytes "
120     log.append("> " + contentInfo)
121     if (imgFilter == '0'):
122         if ("image" in contentInfo):
123             connSock.close()
124             total_thread_no -= 1
125             return
126
```

이후 실제 서버로부터 outerSock이 수신한 response를 parsing한다. recv_timeout 함수는 부분부분 수신된 패킷을 연속적으로 수신하여 return하는 함수이다(밑에 따로 기술한다). Response를 두 줄의 개행 문자로 구분하여 헤더, 실제 data로 분리하고 헤더에서 status 및 content에 대한 info를 log에 저장한다. 이후 imgFilter가 0이면 content-type이 image인 패킷들에 대해 소켓을 close하고 다음 loop으로 넘어감으로써 브라우저 측에서 정보를 수신할 수 없도록 하였다. 그렇지 않은 경우, 모든 패킷을 온전히 저장한다.

```
project.py X
/mnt/c/Users/MinseokSong/Desktop/linux/pj3/project.py

126
127         # CLI <== PRX
128         log.append("[CLI <== PRX --- SRV]")
129         connSock.send(response_raw)
130         log.append(" > "+status)
131         log.append(" > "+contentInfo)
132         connSock.close()
133         log.append("[CLI disconnected]")
134         log.append("[SRV disconnected]")
135         outerSock.close()
136         log.append("-----")
137         res = "\n".join(log)
138         print(res)
139         total_thread_no -= 1
140
```

저장된 response를 client측 socket에 전달하는 부분이다. 출력에 필요한 정보들을 log에 마찬가지로 저장하고, 전처리(이미지 필터링 등)된 response를 그대로 client측 소켓인 connSock에 send한다. 이후 두 소켓을 닫고, 모든 log를 출력한 뒤 thread 숫자를 줄여준다.

```
141         except KeyboardInterrupt as e:
142             print (e)
143             sys.exit()
144
145         except Exception as e:
146             print (e)
147             return
```

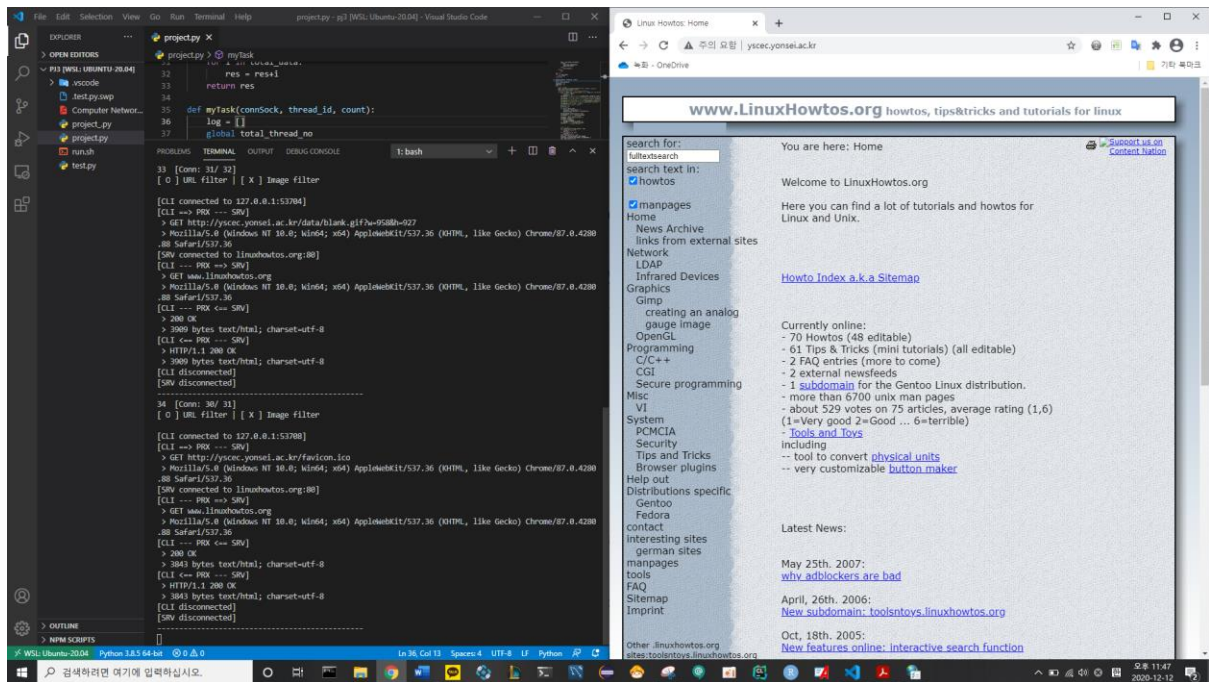
실행 중 Ctrl+C가 입력되면 즉시 프로그램을 종료하고, 다른 error 발생 시 일단은 다음 request를 수신하는 다음 loop로 넘어가도록 하였다.

```
project.py X
project.py > myTask
6
7 def recv_timeout(the_socket,timeout=2):
8     the_socket.setblocking(0)
9
10    total_data=[]
11    data=''
12
13    begin=time.time()
14    while True:
15        if total_data and time.time()-begin > timeout:
16            break
17        elif time.time()-begin > timeout*2:
18            break
19
20        try:
21            data = the_socket.recv(65535)
22            if data:
23                total_data.append(data)
24                begin=time.time()
25            else:
26                time.sleep(0.1)
27        except:
28            pass
29
30    res = b""
31    for i in total_data:
32        res = res+i
33    return res
```

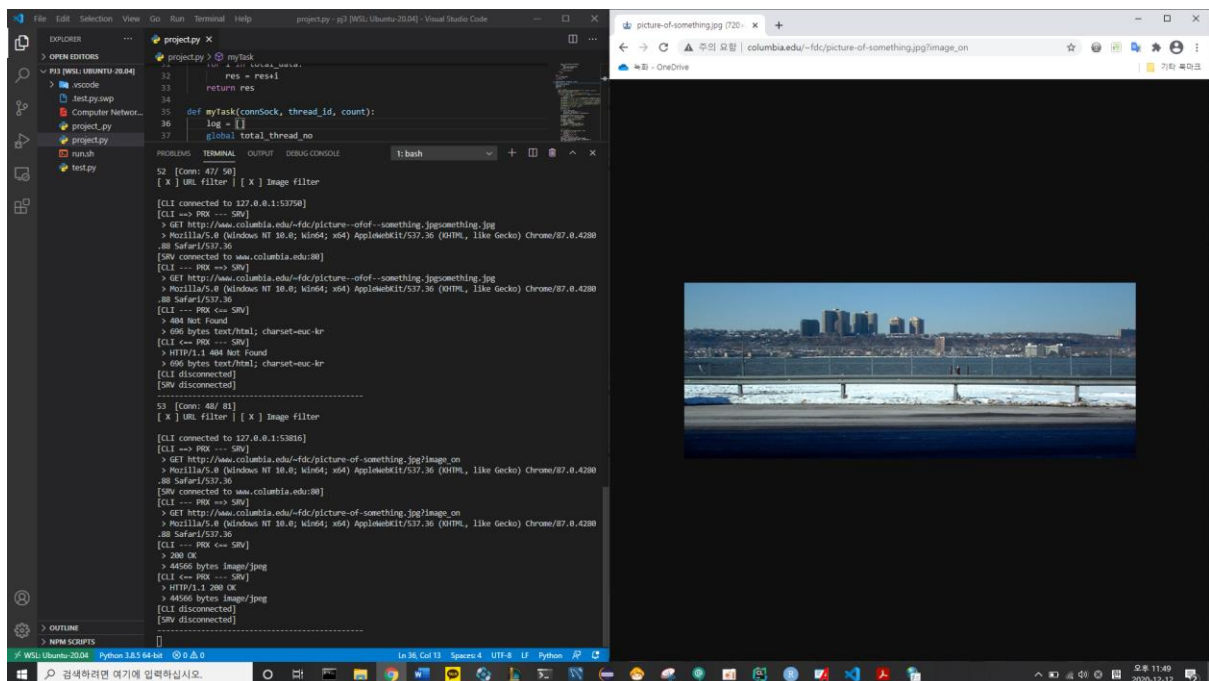
위에서 서술한 패킷을 연속적으로 수신하는 함수이다. 다음 data, 즉 다음 패킷이 계속 수신될 경우 loop를 통해 계속 수신하여 결과에 더해주고, 일정 시간동안 패킷이 수신되지 않을 경우 기다렸다가 수신을 종료하여 이미지와 같은 큰 용량의 response를 연속적으로 온전히 수신할 수 있도록 한다. 이는 개발 커뮤니티의 문서를 참조하여 구현하였다.

4. 작동 예시

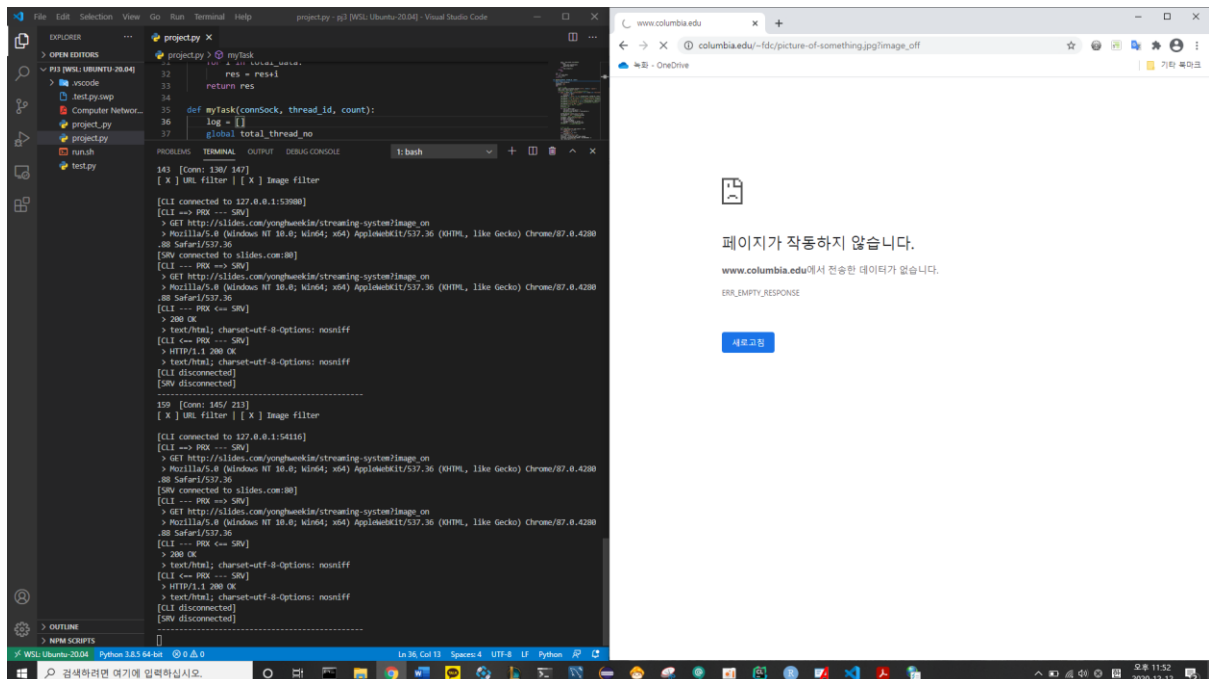
1. "Yonsei"가 포함된 yssec홈페이지로 접속을 시도했을 경우 linuxhowtos.org로 이동하는 모습



2. image_on인 상태로 http://www.columbia.edu/~fdc/picture-of-something.jpg?image_on에 접속을 시도한 경우



3. image_off인 상태로 http://www.columbia.edu/~fdc/picture-of-something.jpg?image_off 에 접속을 시도한 경우(이미지만 있으므로 아무것도 response가 넘겨지지 않아 페이지가 없다고 출력)



5. Multithreading 적용에 따른 성능(시간) 비교

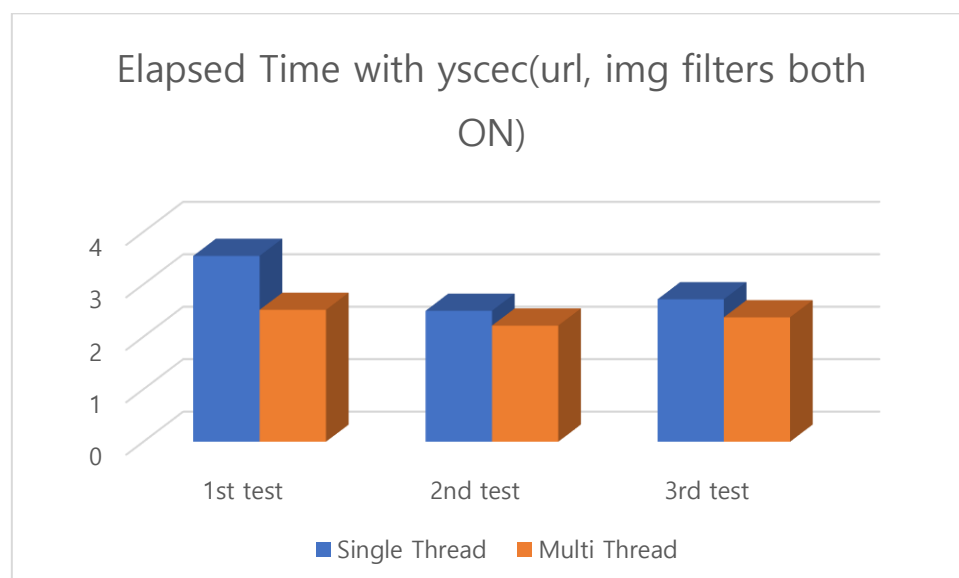
측정을 위해 잠시 아래와 같이 time모듈을 이용하여 Elapsed time을 측정 및 비교하였다. 동일 환경을 적용해야 하므로 URL Filter와 Image Filter 모두 적용된 상태에서 측정하였다.

```
4 [Conn: 4/20]
[ 0 ] URL filter | [ 0 ] Image filter

[CLI connected to 127.0.0.1:35608]
[CLI ==> PRX --- SRV]
> GET http://yscec.yonsei.ac.kr/favicon.ico
> Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
[SRV connected to linuxhowtos.org:80]
[CLI --- PRX ==> SRV]
> GET www.linuxhowtos.org
> Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
[CLI --- PRX <== SRV]
> HTTP/1.1 200 OK
> 3893 bytes text/html; charset=utf-8
[CLI <== PRX --- SRV]
> HTTP/1.1 200 OK
> 3893 bytes text/html; charset=utf-8
[CLI disconnected]
[SRV disconnected]
-----
Elapsed Time : 2.496164321899414
```

단일 Thread인 경우 (브라우저로 마지막 패킷을 전송완료한 시점) - (브라우저에서 접속을 요청한 시점)을 기준으로 측정하였고, Multi-Thread인 경우는 (해당 thread가 마지막 패킷을 전송 완료한 시점) - (접속을 요청하여 해당 thread가 시작된 시점)을 기준으로 측정하였다.

http://yscec.yonsei.ac.kr/?image_off 를 이용하여 총 3번 반복 측정하였고, 그 결과는 다음과 같다.



눈에 띄게 Multi-Thread가 빠름을 알 수 있다. text보다 image의 용량이 일반적으로 크다는 점을 고려하면, image_on인 상황에서 image가 많은 http요청을 주고받을 경우 이 시간 효율 차이는 더욱 클 것으로 예상된다.