Project3 2016147556 김성하

1.Introductin/Reference

기본 소켓 프로그래밍으로 프록시 서버를 구현하였으며, http request 가 들어올 때마다 thread 를 부여해 처리하는 방식으로 multithreading 이 가능하게 했습니다. 브라우저 http request setting 은 127.0.0.1 port:9001 로 했습니다.

Software environment: Ubuntu 20.04.1 LTS

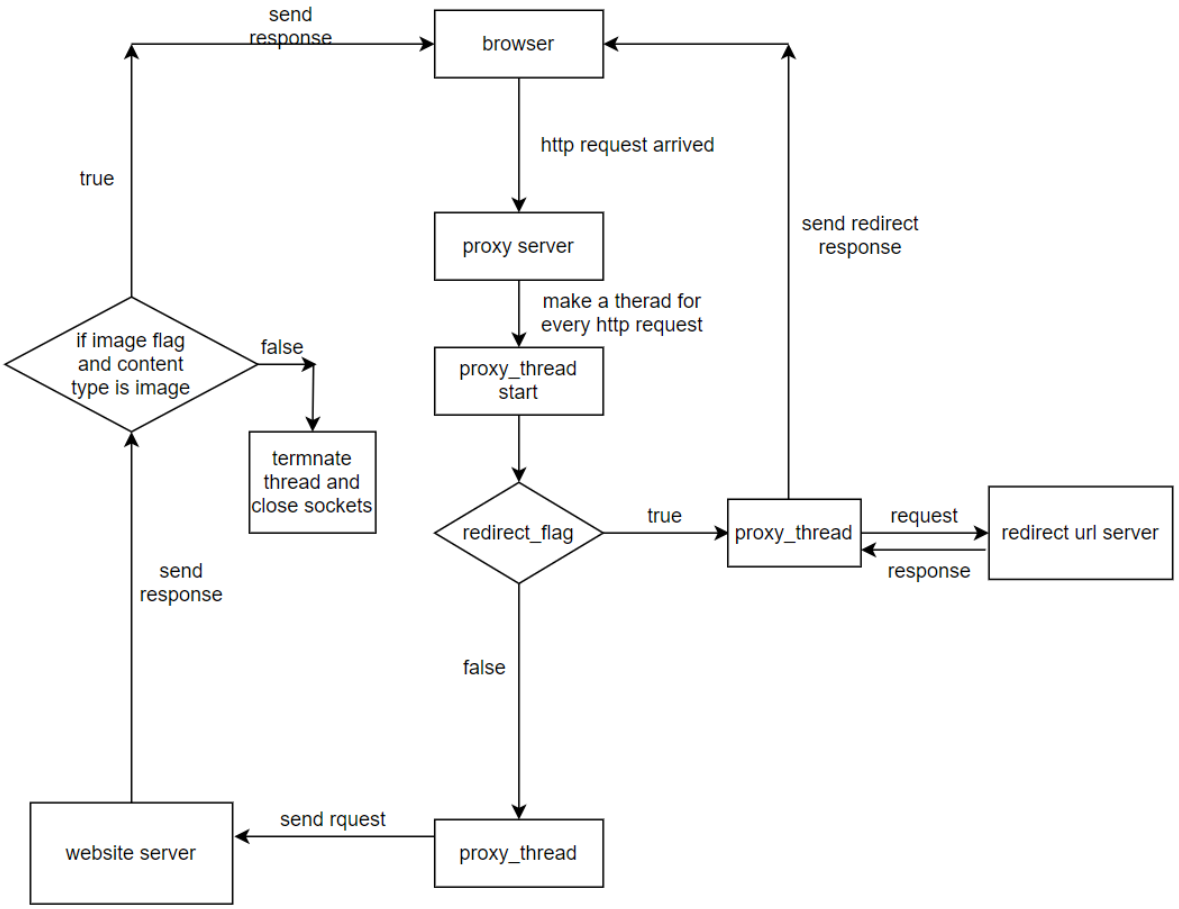
Language: python3 python 3.8.5

Reference:

https://docs.python.org/ko/3/howto/sockets.html

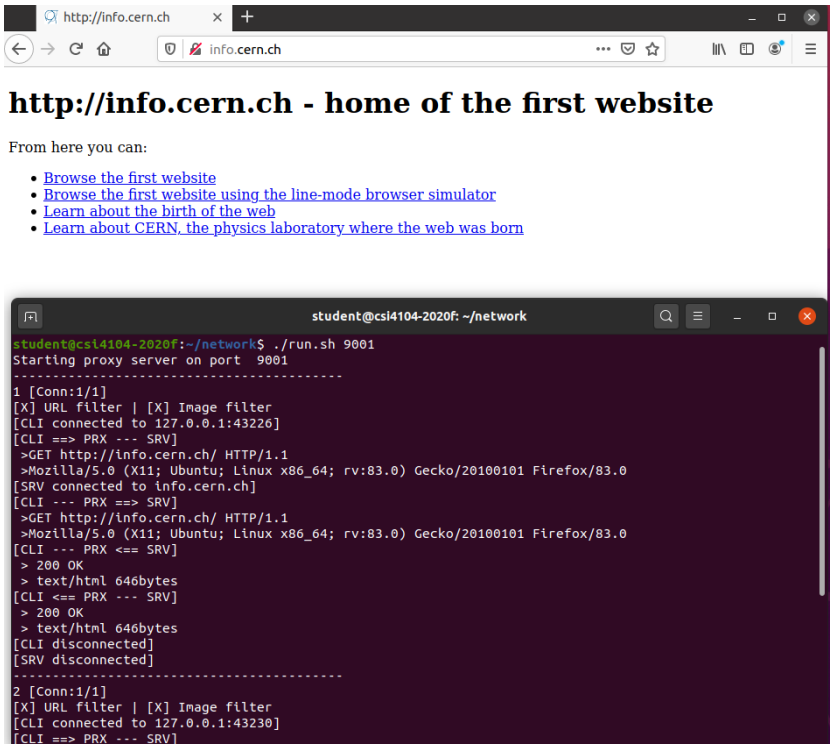
http://pythonstudy.xyz/python/article/24-%EC%93%B0%EB%A0%88%EB%93%9C-Thread

2. Flow chart

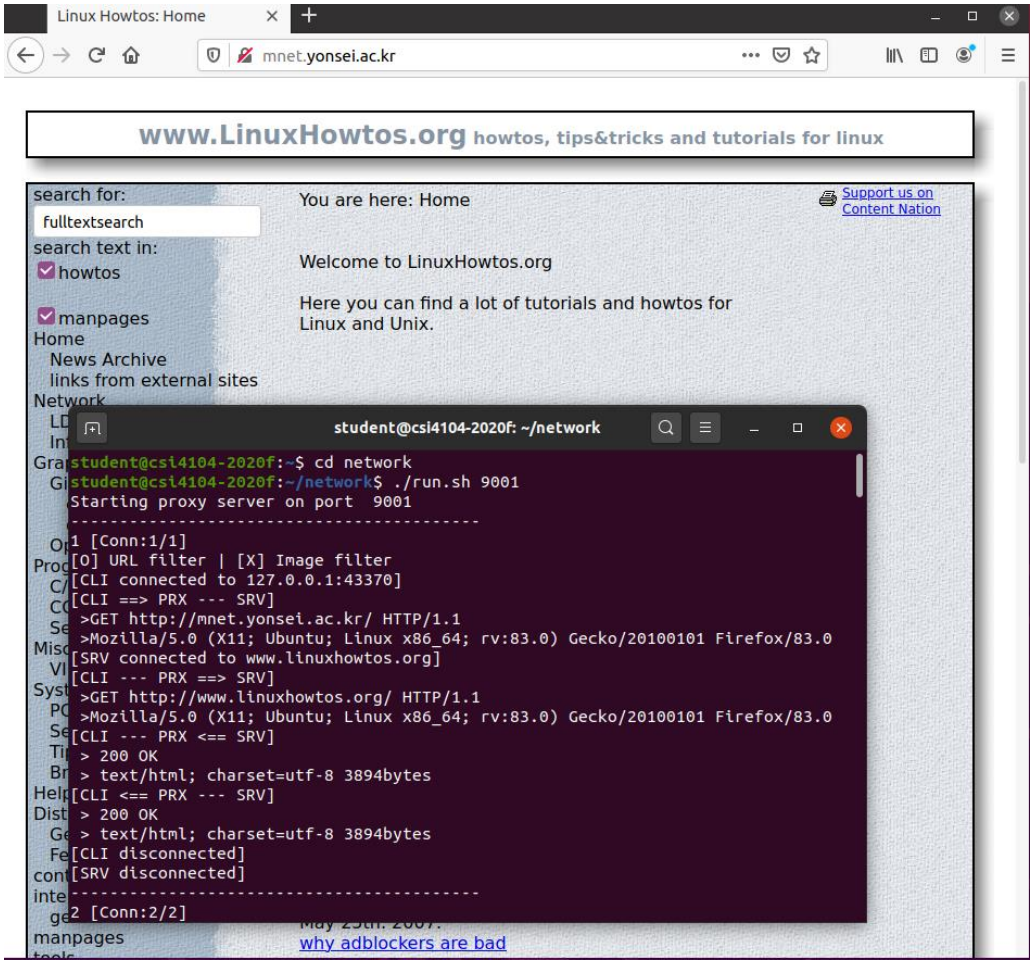


브라우저 에서 http request 를 프록시 서버에 전달하고 나면, 프록시 서버에서는 http request 하나당 thread 하나를 부여해 처리를 시작합니다. 여기서 thread 가 시작될 때 URL 분석을 통해 redirect, image filter 를 위한 flag 를 정의합니다. 만약 redirect 을 해야 한다면, thread는 정의된 사이트로 접속을 하고 response 를 받은 다음 이것을 browser 에 전달합니다. 만약 redirect 를 안 해도 된다면, thread 는 원래 주소에 request 를 보내고, response 를 받아서 이것을 browser 에 전해줍니다. 여기서 만약 image filtering 을 해야 한다면, thread 는 이 response 를 browser 에 전달하지 않고, browser socket, server socket 을 닫은 다음 thread 를 종료합니다.

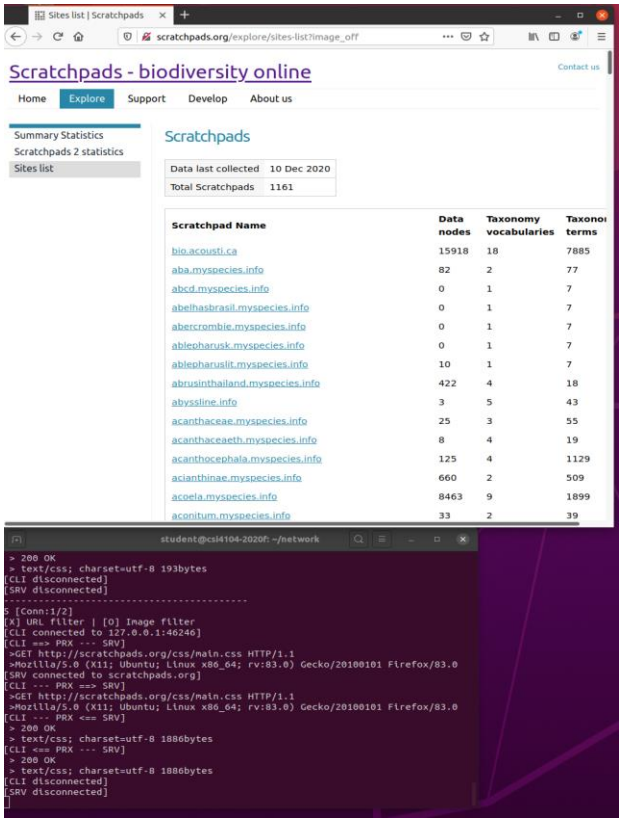
3. Snapshot of each function



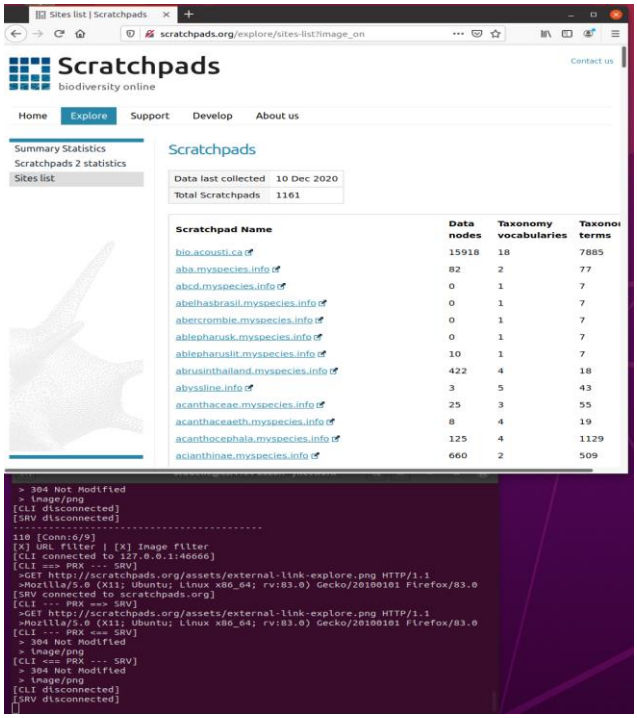
http://info.cern.ch/로 접속을 하였을 때 log 가 올바르게 출력되고, 두개의 소켓을 사용하여 request/response 의 forwarding 이 올바르게 작동하는 것을 확인했습니다.



URL 에 Yonsei 가 포함되면, www.linuxhowtos.org 로 가게 flag 를 통해서 redirect 를 구현했고, 실제로 올바르게 작동을 하고, log 상으로도 proxy server 에서 request 를 변경해 response 를 받아오는 것과 URL filter 가 켜져 있는 것을 확인했습니다.



만약 URL 맨 끝에 ? image_off 가 있으면,이미지가 표시되지 않은 것을 확인하였고, log 상으로도 image filter 가 켜져 있으며, image 파일이 browser 안 가는 것을 확인했습니다.



반대로 그 이후에 ?image_on 을 URL 끝에 표시를 하면, 이미지가 다시 표시되는 것을 확인하였고, log 상으로도 image filter 가 꺼져 있고, image 파일 이 가는 것을 확인했습니다.

4. Logical explanation in blocks

```
import os,sys,threading,socket
import signal
import requests

port=int(sys.argv[1])
host='127.0.0.1'
proxy_socket=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
proxy_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
proxy_socket.bind((host,port))
proxy_socket.listen()
print("Starting proxy server on port ",port)
num=0
image_flag=False
lock=threading.Lock()
shared_num=0
thr_list=[False for i in range(10000)]

def get_least():
        for i in range(10000):
                if(thr_list[i]==False):
                        return i


def control_c(sig,frame):
   print('exit')
   proxy_socket.close()
   sys.exit(0)

signal.signal(signal.SIGINT,control_c)
```

Proxy server 구현의 시작부분으로, 브라우저 상으로 넘겨주는 포트번호를 받고, 소켓을 설정합니다. get_least 함수는 thread 번호를 할당하는 함수로, 그때 비어 있는 가장 작은 번호로 thread 번호를 할당해줍니다. 여기서 lock 과 shared_num 의 변수는 thread 별로 출력이 겹치지 않게 하도록 하는 설정입니다. control_c 함수는 ctrl c exception 을 처리해주는 함수입니다. thr_list는 threadnum을 관리합니다.

```
while True:
        conn,client_addr=proxy_socket.accept()
        num=num+1
        thr_num=get_least()
        thr_list[thr_num]=True
        temp_thr=threading.Thread(target=proxy_threading,args=(conn,client_addr,thr_num,num))
        temp_thr.daemon=True
        temp_thr.start()


signal.pause()
```

그후에 browser 로부터 http request 를 받고, 해당 thread 를 실행시킵니다. 여기서 num 은 몇번째로 thread 가 생성되었는지를 나타냅니다.

```
33 def proxy_threading(conn,client_addr,thrr_num,index):
34         global shared_num
35         global image_flag
36         print_list=[]
37         request=conn.recv(999999)
38         #length is 0, break
39         if(len(request)==0):
40                 thr_list[thrr_num]=False;
41                 return
42         request_list=request.split(b'\r\n')
43         first_line=request_list[0]
44         second_line=request_list[1]
45         third_line=request_list[2]
46         url=first_line.split()[1].decode('utf-8')
47         if(len(url)>=10 and url[-10:]=="?image_off"):
48                 image_flag=True
49
50         elif(len(url)>=9 and url[-9:]=="?image_on"):
51                 image_flag=False
52
53         host_name=second_line.decode('utf-8')[6:]
54         user_agent=third_line.decode('utf-8')[12:]
55         #if  method connect, break
56         method=first_line.split()[0].decode('utf-8')
57         if(method=="CONNECT"):
58                 thr_list[thrr_num]=False;
59                 conn.close()
60
61         redirect_flag=False
62         if('yonsei' in url):
63                 redirect_flag=True
```

Thread 를 시작하는 부분으로, request 로부터 필요한 정보를 추출하고, 정보를 바탕으로 flag 를 설정해주고, connect_method 에 대해서 thread를 종료시킵니다.

```
66         #clit-->prx---srv
67         print_list.append("----------------------------------------")
68         print_list.append(str(index)+" [Conn:"+str(thrr_num+1)+"/"+str(threading.active_count()-1)+"]")
69         print_list.append("["+("O" if redirect_flag else "X")+"] URL filter | ["+("O" if image_flag else "X")+"] Image filter")
70         print_list.append("[CLI connected to 127.0.0.1:"+str(client_addr[1])+"]")
71         print_list.append("[CLI ==> PRX --- SRV]")
72         print_list.append(" >"+first_line.decode('utf-8'))
73         print_list.append(" >"+user_agent)
74
75
76         #iterate request
77         if redirect_flag and method=="GET":
78                 host_name="www.linuxhowtos.org"
79                 request_list[0]=b"GET http://www.linuxhowtos.org/ HTTP/1.1"
80                 for i in range(len(request_list)):
81                         if len(request_list[i])>=8 and request_list[i][:7]==b"Referer":
82                                 request_list[i]=b"Referer: http://www.linuxhowtos.org"
83
84                         if len(request_list[i])>=5 and request_list[i][:4]==b"Host":
85                                 request_list[i]=b"Host: www.linuxhowtos.org"
86                 request=b"\r\n".join(request_list)
87
```

Client request 를 받아오면 해당 정보를 저장하고, 만약 redirect 해야 한다면, request 를 변경합니다

```
95          #cli---prx-->srv
96          s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
97          s.connect((host_name,80))
98          print_list.append("[SRV connected to "+host_name+"]")
99          s.send(request)
00          print_list.append("[CLI --- PRX ==> SRV]")
01          first_line=request.split(b'\r\n')[0]
02          third_line=request.split(b'\r\n')[2]
03          host_name=second_line.decode('utf-8')[6:]
04          print_list.append(" >"+first_line.decode('utf-8'))
05          user_agent=third_line.decode('utf-8')[12:]
06          print_list.append(" >"+user_agent)
07
```

Proxy 에서 server 로 보내는 부분으로, 해당 정보를 저장합니다. s.send(request) 로 정보를 서버에 전달합니다

```
111         #cli<--prx<--srv
112         while True:
113                 try:
114                         data=s.recv(99999)
115                 except:
116                         a=1
117                 if(len(data)>0):
118                         data_list=data.split(b'\r\n')
119                         #for the first data
120                         if(data_list[0][:4]==b"HTTP"):
121                                 status_pos=data_list[0].find(b" ")
122                                 print_list.append("[CLI --- PRX <== SRV]")
123                                 print_list.append(" > "+data_list[0][status_pos+1:].decode('utf-8'))
124                                 flag=False
125                                 for i in data_list:
126                                         if(len(i)>=13 and i[:12]==b"Content-Type"):
127                                                 flag=True
128                                                 print_list.append(" > "+i[14:].decode('utf-8'))
129                                                 if(i[14:19]==b'image' and image_flag ):
130                                                         s.close()
131                                                         conn.close()
132                                                         thr_list[thrr_num]=False
133                                                         return
134                                                 for j in data_list:
135                                                         if(len(j)>=15 and j[:14]==b"Content-Length"):
136                                                                 print_list[-1]+=" "
137                                                                 print_list[-1]+=j[16:].decode('utf-8')
138                                                                 print_list[-1]+="bytes"
139                                                                 break
140                                                 break
141                                 if(not flag):
142                                         print_list.append(" > ")
143
144                                 print_list.append("[CLI <== PRX --- SRV]")
145                                 print_list.append(print_list[-3])
146                                 print_list.append(print_list[-3])
147
148                         #send data
149
150
151
152                         conn.send(data)
153                 else:
154                         break
```

서버에서 proxy 가 정보가 오면, 크기가 클 겨우 나뉘어서 정보가 오는데, 올때마다 client 로 보내주고, 첫번쨔 header 에 대한 정보가 들어올 경우, content-type 과 content-length 를 받아주고, 이미지를 필터링 해야 한다면, 소켓을 닫고, thread를 종료합니다.

```
157         #close connection
158         s.close()
159         conn.close()
160         thr_list[thrr_num]=False;
161
162         lock.acquire()
163         shared_num+=1
164         for i in print_list:
165                 print(i)
166         print("[CLI disconnected]")
167         print("[SRV disconnected]")
168         lock.release()
169
```

Client 로 정보를 다 보내주고, 소켓을 닫은 다음, lock 과 shared_num 변수를 이용해서 한 번에 출력을 해줍니다.

5. Comparison between applying multithreading and not

Multithreading을 사용하지 않은 결과, 사용한 결과에 비해서 시간이 현저히 증가했으며, 비록 multihreading 을 사용한 코드상에서 출력을 위해서, lock 을 사용했음에도 불구하고, 시간이 현저히 빨리 걸렸습니다.