

2022 컴퓨터 네트워크__중간 대체 과제__README

국민대학교 소프트웨어학부

20213091 최지원

1. Description

1.1 기본 구현

- 소켓 통신을 활용하여 TCP 기반의 Server, Client 프로그램을 작성한다.
- CLIENT에서는 HTTP 프로토콜의 GET/HEAD/POST/PUT Request 요청을 할 수 있게 한다.
- SERVER에서는 CLIENT 측의 Request에 따라 응답 메시지를 구성하여 Response 할 수 있게 한다.

1.2 추가 구현

- CLIENT 측에서 HTTP 프로토콜의 Request, DELETE와 OPTIONS 추가로 요청할 수 있도록 한다.
- HTTP 명령어 수행 시 CLIENT가 요청한 파일을 실제로 생성 및 수정, 삭제하도록 구현한다.
- SERVER에서는 CLIENT로부터 받은 Request에 따라 응답 메시지를 구성하여 RESPONSE 한다.

2. Environment

2.1 사용 언어 및 개발 환경

- Python 3
- Mac OS, Visual Studio Code

2.2 사용 모듈 (Python3)

- socket : Python에서 소켓 프로그래밍을 구현하기 위하여 사용 (기본 지원)
- datetime: SERVER 측에서 전달하는 응답 메시지 중 HTTP 헤더 Date를 구현하기 위하여 사용
- os: HTTP 명령어 가운데, DELETE(요청 파일 삭제)를 구현하기 위하여 사용

3. Additional Applications [가산점]

HTTP 명령어 수행 결과를 WireShark 로 캡처하여 README 에 제출함	0
HTTP 명령어 수행시에 Server에서 Client가 요청하는 파일을 실제로 생성하거나 update함	0
CLIENT 측에서 Request 할 수 있는 HTTP 명령어를 추가 구현함 (DELETE, OPTIONS)	0

Files

1. TCP_CLIENT

1.1 TCP_CLIENT 소켓 생성 및 전송 함수

```
1  import socket
2
3  serverPort = 3091
4  serverName = '127.0.0.1'
5
6  def create_socket_and_send_message(request_message):
7      # create client socket
8      clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9      clientSocket.connect((serverName, serverPort))
10     clientSocket.send(request_message.encode('utf-8'))
11
12     # checking response
13     receive_message = clientSocket.recv(65535)
14     print(receive_message.decode())
15
16     # closing client socket
17     clientSocket.close()
```

코드 1 TCP_CLIENT.py (Line 1 ~ 17)

Code Line 1: Python으로 socket 프로그래밍을 구현할 것이므로 모듈을 import 해준다.

Code Line 3, 4: Client가 연결할 Sever측 port 번호와 ip를 지정해준다.

port 번호는 0 ~ 1023(well-known), 1024 ~ 49151(registered), 49152 ~ 65535(dynamic)로 나뉨.
해당 프로젝트에서는 port번호로 3091을 사용하고 ip로는 localhost의 ip를 부여하였음.

Code Line 8: AF_INET은 해당 socket(clientSocket)을 IP version 4용으로 사용함을 의미한다.
SOCK_STREAM은 해당 socket(clientSocket)에서 TCP packet을 받음을 의미한다.

Code Line 9: 위에서 설정한 Server의 ip와 port 번호를 tuple 형태로 받은 뒤, 연결을 시도한다.

Code Line 10: Server측으로 request_message를 utf-8 형태로 encoding해준 뒤 전달해준다.

Code Line 13, 14: Server측에서 Response해준 메시지를 받아온 뒤, decoding한 것을 print한다.

Code Line 17: 데이터 송수신이 끝났으므로 socket 연결을 끊어준다.

1.2 Request message (GET) – 200 OK / 404 Not Found

```
1 # GET method – 200 OK
2 request_message = 'GET /meow.html HTTP/1.1\r\n'
3 request_message += 'Host: 127.0.0.1:12000\r\n'
4 request_message += 'User-Agent: Safari/537.36\r\n'
5 request_message += 'Connection: Keep-Alive\r\n\r\n'
6
7 create_socket_and_send_message(request_message)
8
9 # GET method – 404 Not Found
10 request_message = 'GET /mmmmmeow.html HTTP/1.1\r\n'
11 request_message += 'Host: 127.0.0.1:12000\r\n'
12 request_message += 'User-Agent: Safari/537.36\r\n'
13 request_message += 'Connection: Keep-Alive\r\n'
14
15 create_socket_and_send_message(request_message)
```

코드 2 TCP_CLIENT.py (Line 19 ~ 33)

그림 캡션 옆에 있는 Line은 GitHub에 업로드 되어 있는 Code가 기준입니다

1.3 Request message (PUT) – 201 Create

```
1 # PUT method – 201 Create
2 put_content = 'Hello my name is Choi Jiwon'
3 request_message = 'PUT /introduce-myself.txt HTTP/1.1\r\n'
4 request_message += 'Host: 127.0.0.1:12000\r\n'
5 request_message += 'Content-type: text/plain\r\n'
6 request_message += 'Content-length: ' + str(len(put_content)) + '\r\n\r\n'
7 request_message += put_content + '\r\n\r\n'
8
9 create_socket_and_send_message(request_message)
```

코드 3 TCP_CLIENT.py (Line 35 ~ 43)

1.4 Request message (POST) – 201 Create / 200 OK

```
1  # POST method - 201 Create or 200 OK
2  post_content = 'Cat is meow meow, Dog is bow bow'
3  request_message = 'POST /bowbow.txt HTTP/1.1\r\n'
4  request_message += 'Host: 127.0.0.1:12000\r\n'
5  request_message += 'Content-type: text/plain\r\n'
6  request_message += 'Content-length: ' + str(len(post_content)) + '\r\n\r\n'
7  request_message += post_content + '\r\n\r\n'
8
9  create_socket_and_send_message(request_message)
10
11 # POST method - 200 OK
12 post_content = 'Kookmin university computer science'
13 request_message = 'POST /introduce-myself.txt HTTP/1.1\r\n'
14 request_message += 'Host: 127.0.0.1:12000\r\n'
15 request_message += 'Content-type: text/plain\r\n'
16 request_message += 'Content-length: ' + str(len(post_content)) + '\r\n\r\n'
17 request_message += post_content + '\r\n\r\n'
18
19 create_socket_and_send_message(request_message)
```

코드 4 TCP_CLIENT.py (Line 45 ~ 63)

1.5 Request message (OPTIONS) – 200 OK

```
1  # OPTIONS method - 200 OK
2  request_message = 'OPTIONS * HTTP/1.1\r\n'
3  request_message += 'Host: www.localhost:' + str(serverPort) + '\r\n'
4  request_message += 'Accept: *\r\n\r\n'
5
6  create_socket_and_send_message(request_message)
```

코드 5 TCP_CLIENT.py (Line 65 ~ 70)

1.6 Request message (DELETE) – 200 OK

```
1 DELETE method - 200 OK
2 request_message = 'DELETE /introduce-myself.txt HTTP/1.1\r\n'
3 request_message += 'Host: 127.0.0.1:12000\r\n\r\n'
4
5 create_socket_and_send_message(request_message)
```

코드 6 TCP_CLIENT.py (Line 72 ~ 76)

1.7 Request message (HEAD) – 200 OK

```
1 # HEAD method - 200 OK
2 request_message = 'HEAD /meow.html HTTP/1.1\r\n'
3 request_message += 'Host: 127.0.0.1:12000\r\n'
4 request_message += 'User-Agent: Safari/537.36\r\n'
5 request_message += 'Connection: Keep-Alive\r\n\r\n'
6
7 create_socket_and_send_message(request_message)
```

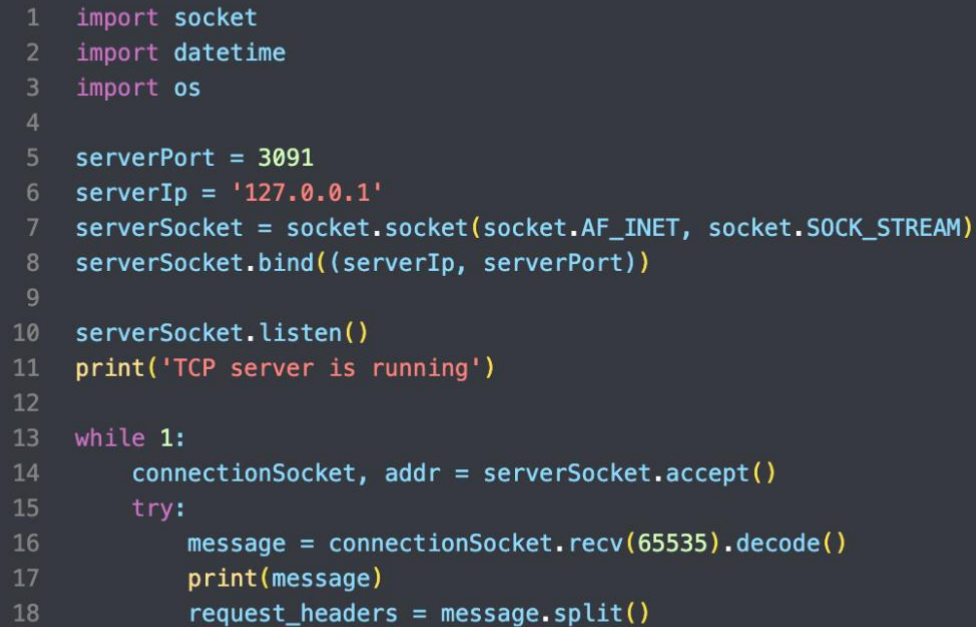
코드 7 TCP_CLIENT.py (Line 78 ~ 84)

- Client가 Server로 메시지를 전송할 때 HTTP request message의 양식에 맞추어 전송하도록 구현
- method마다 request line과 header lines의 내용을 조금씩 다르게 구성했으며 정리하면 아래와 같음

method	Request Line	Header Lines
GET	method URL version	Host, User-Agent, Connection
HEAD		
PUT		Host, Content-type, Content-length
POST		
OPTIONS		Host, Accept
DELETE		Host

2. TCP_SERVER

2.1. TCP_SERVER

A screenshot of a code editor showing Python code for a TCP server. The code is numbered from 1 to 18. It imports socket, datetime, and os. It sets serverPort to 3091 and serverIp to '127.0.0.1'. It creates a serverSocket and binds it to (serverIp, serverPort). It calls serverSocket.listen(). It prints 'TCP server is running'. It enters a while loop that accepts connections. Inside the loop, it tries to receive a message (65535 bytes), prints it, and splits it into request_headers.

```
1  import socket
2  import datetime
3  import os
4
5  serverPort = 3091
6  serverIp = '127.0.0.1'
7  serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8  serverSocket.bind((serverIp, serverPort))
9
10 serverSocket.listen()
11 print('TCP server is running')
12
13 while 1:
14     connectionSocket, addr = serverSocket.accept()
15     try:
16         message = connectionSocket.recv(65535).decode()
17         print(message)
18         request_headers = message.split()
```

코드 8 TCP_SERVER.py (Line 1 ~ 18)

Code Line 1~3: Server를 구현하기 위하여 필요한 module들을 import 해준다.

Code Line 5~8: socket을 만드는 과정은 Client와 동일하다. localhost를 이용할 것이므로 ip는 127.0.0.1

Code Line 10: Client측의 요청을 기다린다.

Code Line 16~18: 연결된 Client으로부터 수신한 메시지를 decoding하여 message 변수에 저장.
message에 저장되어 있는 데이터를 split하여 header 별로 분리 <- 배열로 관리함

Code Line 18~: Client에서 요청한 method마다 다른 header들이 들어오므로 request header의 정보에 맞추어 분기하여 상황에 맞는 response Message을 다시 Client에 전달해주도록 함


```

1 while 1:
2     connectionSocket, addr = serverSocket.accept()
3     try:
4         message = connectionSocket.recv(65535).decode()
5         print(message)
6         request_headers = message.split()
7         if request_headers[2] == 'HTTP/1.1' and request_headers[0] in ['GET', 'HEAD']:
8             try: # 존재하는 주소로 접근하는 경우 / html 파일에 접근할 수 있는 경우
9                 html_file = open(request_headers[1][1:], "r")
10                html_file_content = html_file.read()
11                html_file.close()
12
13                response_message = 'HTTP/1.1 200 OK\r\n'
14                response_message += 'Content-Type: text/html\r\n'
15                response_message += 'Content-Length: ' + str(len(html_file_content)) + '\r\n'
16                response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
17
18                if request_headers[0] == 'GET': # GET 명령어는 HTML 본문을 읽어옴 -> HEAD는 상태 확인 용도
19                    response_message += html_file_content + '\r\n\r\n'
20                else: # HEAD의 경우에는 html 본문을 return 하지 않으므로 생략
21                    pass
22
23            except: # 존재하지 않는 주소에 접근하려는 경우
24                response_message = 'HTTP/1.1 404 Not Found\r\n'
25                response_message += 'Content-Type: text/html\r\n'
26                response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
27
28        elif request_headers[2] == 'HTTP/1.1' and request_headers[0] in ['PUT', 'POST']:
29            if request_headers[0] == 'PUT':
30                html_file_length = 0
31                html_file = open(request_headers[1][1:], "w")
32                for i in range(9, len(request_headers)):
33                    data = request_headers[i] + ' '
34                    html_file.write(data)
35                html_file.close()
36                response_message = 'HTTP/1.1 201 Created\r\n'
37                response_message += 'Location: http://localhost:' + str(serverPort) + request_headers[1] + '\r\n'
38                response_message += 'Content-Type: text/plain\r\n'
39
40                response_message += 'Content-Length: ' + str(html_file_length) + '\r\n'
41                response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
42                response_message += 'http://localhost:' + str(serverPort) + request_headers[1] + '\r\n\r\n'
43            elif request_headers[0] == 'POST':
44                try:
45                    html_file = open(request_headers[1][1:], "r")
46                    html_file.close()
47                    html_file = open(request_headers[1][1:], "a") # mode a는 파일의 마지막에 새로운 내용을 추가할 때 사용
48                    html_file_updated_content = ""
49
50                    for i in range(9, len(request_headers)):
51                        html_file.write(request_headers[i] + ' ')
52                        html_file_updated_content += request_headers[i] + ' '
53                    html_file.close()
54
55                    response_message = 'HTTP/1.1 200 OK\r\n'
56                    response_message += 'Location: http://localhost:' + str(serverPort) + request_headers[1] + '\r\n'
57                    response_message += 'Content-Type: text/plain\r\n'
58                    response_message += 'Content-Length: ' + str(len(html_file_updated_content)) + '\r\n'
59                    response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
60                    response_message += html_file_updated_content + '\r\n\r\n'
61
62                except FileNotFoundError: # 존재하지 않는 경우에는 PUT처럼 생성해줌.
63                    response_message = 'HTTP/1.1 201 Created\r\n'
64                    response_message += 'Location: http://localhost:' + str(serverPort) + request_headers[1] + '\r\n'
65                    response_message += 'Content-Type: text/plain\r\n'
66                    response_message += 'Content-Length: ' + str(html_file_length) + '\r\n'
67                    response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
68                    response_message += 'http://localhost:' + str(serverPort) + request_headers[1] + '\r\n\r\n'
69
70        elif request_headers[2] == 'HTTP/1.1' and request_headers[0] == 'DELETE':
71            os.remove(request_headers[1][1:])
72            response_message = 'HTTP/1.1 200 OK\r\n'
73            response_message += 'Content-Type: text/plain\r\n'
74            response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
75            response_message += 'I have your delete request, will take time to process\r\n\r\n'
76
77        elif request_headers[2] == 'HTTP/1.1' and request_headers[0] == 'OPTIONS':
78            response_message = 'HTTP/1.1 200 OK\r\n'
79            response_message += 'Allow: GET, HEAD, POST, PUT, OPTIONS, DELETE\r\n'
80            response_message += 'Content-Length: 0\r\n'
81            response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
82    except:
83        pass
84
85    connectionSocket.send(response_message.encode())

```

코드 9 TCP_SERVER.py (Line 13 ~ 97)

2.2 Response message (HEAD, GET)

```
19         if request_headers[2] == 'HTTP/1.1' and request_headers[0] in ['GET', 'HEAD']:
20             try: # 존재하는 주소로 접근하는 경우 / html 파일에 접근할 수 있는 경우
21                 html_file = open(request_headers[1][1:], "r")
22                 html_file_content = html_file.read()
23                 html_file.close()
24
25                 response_message = 'HTTP/1.1 200 OK\r\n'
26                 response_message += 'Content-Type: text/html\r\n'
27                 response_message += 'Content-Length: ' + str(len(html_file_content)) + '\r\n'
28                 response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
29
30                 if request_headers[0] == 'GET': # GET 명령어는 HTML 본문을 읽어옴 -> HEAD는 상태 확인 용도
31                     response_message += html_file_content + '\r\n\r\n'
32                 else: # HEAD의 경우에는 html 본문을 return 하지 않으므로 생략
33                     pass
34
35             except: # 존재하지 않는 주소에 접근하려는 경우
36                 response_message = 'HTTP/1.1 404 Not Found\r\n'
37                 response_message += 'Content-Type: text/html\r\n'
38                 response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
```

코드 10 TCP_SERVER.py (Line 19 ~ 38)

2.3 Response message (PUT, POST)

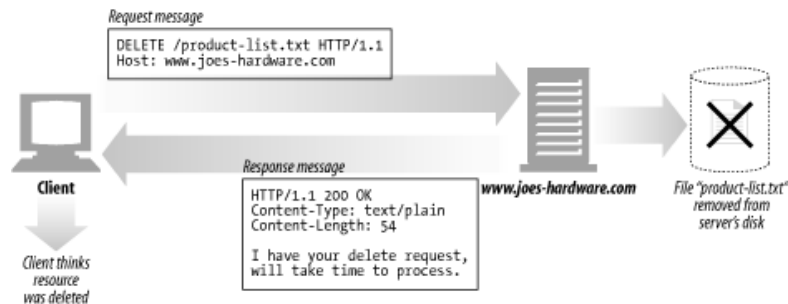
```
40         elif request_headers[2] == 'HTTP/1.1' and request_headers[0] in ['PUT', 'POST']:
41             if request_headers[0] == 'PUT':
42                 html_file_length = 0
43                 html_file = open(request_headers[1][1:], "w")
44                 for i in range(9, len(request_headers)):
45                     data = request_headers[i] + ' '
46                     html_file.write(data)
47                 html_file.close()
48                 response_message = 'HTTP/1.1 201 Created\r\n'
49                 response_message += 'Location: http://localhost:' + str(serverPort) + request_headers[1] + '\r\n'
50                 response_message += 'Content-Type: text/plain\r\n'
51
52                 response_message += 'Content-Length: ' + str(html_file_length) + '\r\n'
53                 response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
54                 response_message += 'http://localhost:' + str(serverPort) + request_headers[1] + '\r\n\r\n'
55             elif request_headers[0] == 'POST':
56                 try:
57                     html_file = open(request_headers[1][1:], "r")
58                     html_file.close()
59                     html_file = open(request_headers[1][1:], "a") # mode a는 파일의 마지막에 새로운 내용을 추가할 때 사용
60                     html_file_updated_content = ""
61
62                     for i in range(9, len(request_headers)):
63                         html_file.write(request_headers[i] + ' ')
64                         html_file_updated_content += request_headers[i] + ' '
65                     html_file.close()
66
67                     response_message = 'HTTP/1.1 200 OK\r\n'
68                     response_message += 'Location: http://localhost:' + str(serverPort) + request_headers[1] + '\r\n'
69                     response_message += 'Content-Type: text/plain\r\n'
70                     response_message += 'Content-Length: ' + str(len(html_file_updated_content)) + '\r\n'
71                     response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
72                     response_message += html_file_updated_content + '\r\n\r\n'
73
74                 except FileNotFoundError: # 존재하지 않는 경우에는 PUT처럼 생성해줌.
75                     response_message = 'HTTP/1.1 201 Created\r\n'
76                     response_message += 'Location: http://localhost:' + str(serverPort) + request_headers[1] + '\r\n'
77                     response_message += 'Content-Type: text/plain\r\n'
78                     response_message += 'Content-Length: ' + str(html_file_length) + '\r\n'
79                     response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
80                     response_message += 'http://localhost:' + str(serverPort) + request_headers[1] + '\r\n\r\n'
```

코드 11 TCP_SERVER.py (Line 40 ~ 80)

2.4 Response message (DELETE) [추가 구현]

```
82 elif request_headers[2] == 'HTTP/1.1' and request_headers[0] == 'DELETE':  
83     os.remove(request_headers[1][1:])  
84     response_message = 'HTTP/1.1 200 OK\r\n'  
85     response_message += 'Content-Type: text/plain\r\n'  
86     response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'  
87     response_message += 'I have your delete request, will take time to process\r\n\r\n'
```

코드 12 TCP_SERVER.py (Line 82 ~ 87)

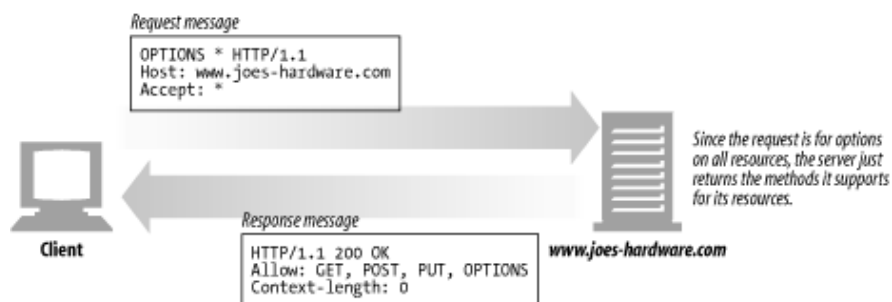


request_header[1]에는 URL이 담겨 있으므로 os 모듈의 remove 함수를 이용하여 Client에서 요청한 파일의 경로로 접근하여 해당 파일을 지울 수 있도록 구현하였음.

2.5 Response message (OPTIONS) [추가 구현]

```
89 elif request_headers[2] == 'HTTP/1.1' and request_headers[0] == 'OPTIONS':  
90     response_message = 'HTTP/1.1 200 OK\r\n'  
91     response_message += 'Allow: GET, HEAD, POST, PUT, OPTIONS, DELETE\r\n'  
92     response_message += 'Content-Length: 0\r\n'  
93     response_message += 'Date: ' + datetime.datetime.now().strftime("%a, %d %b %Y %H:%M") + ' KST\r\n\r\n'
```

코드 13 TCP_SERVER.py (Line 89 ~ 93)



Client 측에서 request할 수 있는 HTML 요청들의 종류를 알려줌 (GET, HEAD, POST, PUT, OPTIONS)

실행 결과

1. TCP_CLIENT.py 터미널 실행 결과

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

ConnectionRefusedError: [Errno 61] Connection refused
(base) choijiwon@choejiwon-ui-MacBookAir ~ % python -u "/Users/choijiwon/Desktop/GitHub-Desktop/2022-ComputerNetwork/TCP_CLIENT.py"
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 165
Date: Tue, 03 May 2022 19:59 KST

<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h1> 2022 CPP </h1>
    <p> KMU CS 20213091 </p>
    <p> Choi-Jiwon </p>
  </body>
</html>

HTTP/1.1 404 Not Found
Content-Type: text/html
Date: Tue, 03 May 2022 19:59 KST

HTTP/1.1 201 Created
Location: http://localhost:3091/introduce-myself.txt
Content-Type: text/plain
Content-Length: 0
Date: Tue, 03 May 2022 19:59 KST

http://localhost:3091/introduce-myself.txt

HTTP/1.1 201 Created
Location: http://localhost:3091/bowbow.txt
Content-Type: text/plain
Content-Length: 0
Date: Tue, 03 May 2022 19:59 KST

http://localhost:3091/bowbow.txt

HTTP/1.1 200 OK
Location: http://localhost:3091/introduce-myself.txt
Content-Type: text/plain

HTTP/1.1 200 OK
Allow: GET, HEAD, POST, PUT, OPTIONS, DELETE
Content-Length: 0
Date: Tue, 03 May 2022 19:59 KST

HTTP/1.1 200 OK
Content-Type: text/plain
Date: Tue, 03 May 2022 19:59 KST

I have your delete request, will take time to process
```

2. TCP_SERVER.py 터미널 실행 결과

```
(base) choijiwon@choejiwon-ui-MacBookAir 2022-ComputerNetwork % python -u "/Users/choijiwon/Desktop/GitHub-Desktop/2022-ComputerNetwork/TCP_SERVER.py"
TCP server is running
GET /meow.html HTTP/1.1
Host: 127.0.0.1:12000
User-Agent: Safari/537.36
Connection: Keep-Alive

GET /mmmmmeow.html HTTP/1.1
Host: 127.0.0.1:12000
User-Agent: Safari/537.36
Connection: Keep-Alive

PUT /introduce-myself.txt HTTP/1.1
Host: 127.0.0.1:12000
Content-type: text/plain
Content-length: 27

Hello my name is Choi Jiwon

POST /bowbow.txt HTTP/1.1
Host: 127.0.0.1:12000
Content-type: text/plain
Content-length: 32

Cat is meow meow, Dog is bow bow

POST /introduce-myself.txt HTTP/1.1
Host: 127.0.0.1:12000
Content-type: text/plain
Content-length: 35

Kookmin university computer science

OPTIONS * HTTP/1.1
Host: www.localhost:3091
Accept: *

DELETE /introduce-myself.txt HTTP/1.1
Host: 127.0.0.1:12000

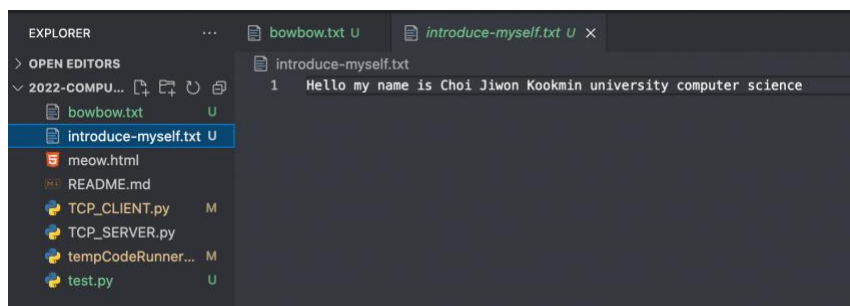
HEAD /meow.html HTTP/1.1
Host: 127.0.0.1:12000
User-Agent: Safari/537.36
Connection: Keep-Alive
```

3. 생성 및 수정된 파일



Bowbow.txt

POST 명령어를 수행 결과 – 여러 번 요청한 결과 -> 계속하여 Cat is ~~가 추가됨을 볼 수 있음



Introduce-myself.txt

PUT 명령어를 수행한 결과 – 여러 번 요청한 결과 (POST와의 차이점을 확인할 수 있음)

4. Client <-> Server 통신 모식도

```
(base) choijiwon@choejiwon-ui-MacBookAir 2022-ComputerNetwork
TCP server is running
GET /meow.html HTTP/1.1
Host: 127.0.0.1:12000
User-Agent: Safari/537.36
Connection: Keep-Alive

GET /mmmmmeow.html HTTP/1.1
Host: 127.0.0.1:12000
User-Agent: Safari/537.36
Connection: Keep-Alive

PUT /introduce-myself.txt HTTP/1.1
Host: 127.0.0.1:12000
Content-type: text/plain
Content-length: 27
Hello my name is Choi Jiwon

POST /bowbow.txt HTTP/1.1
Host: 127.0.0.1:12000
Content-type: text/plain
Content-length: 32
Cat is meow meow, Dog is bow bow

POST /introduce-myself.txt HTTP/1.1
Host: 127.0.0.1:12000
Content-type: text/plain
Content-length: 35
Kookmin university computer science

OPTIONS * HTTP/1.1
Host: www.localhost:3091
Accept: *

DELETE /introduce-myself.txt HTTP/1.1
Host: 127.0.0.1:12000

HEAD /meow.html HTTP/1.1
Host: 127.0.0.1:12000
User-Agent: Safari/537.36
Connection: Keep-Alive

ConnectionRefusedError: [Errno 61] Connection refused
(base) choijiwon@choejiwon-ui-MacBookAir ~ % python -u "/
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 165
Date: Tue, 03 May 2022 19:59 KST

<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1> 2022 CPP </h1>
<p> KMU CS 20213091 </p>
<p> Choi-Jiwon </p>
</body>
</html>

HTTP/1.1 404 Not Found
Content-Type: text/html
Date: Tue, 03 May 2022 19:59 KST

HTTP/1.1 201 Created
Location: http://localhost:3091/introduce-myself.txt
Content-Type: text/plain
Content-Length: 0
Date: Tue, 03 May 2022 19:59 KST
http://localhost:3091/introduce-myself.txt

HTTP/1.1 201 Created
Location: http://localhost:3091/bowbow.txt
Content-Type: text/plain
Content-Length: 0
Date: Tue, 03 May 2022 19:59 KST
http://localhost:3091/bowbow.txt

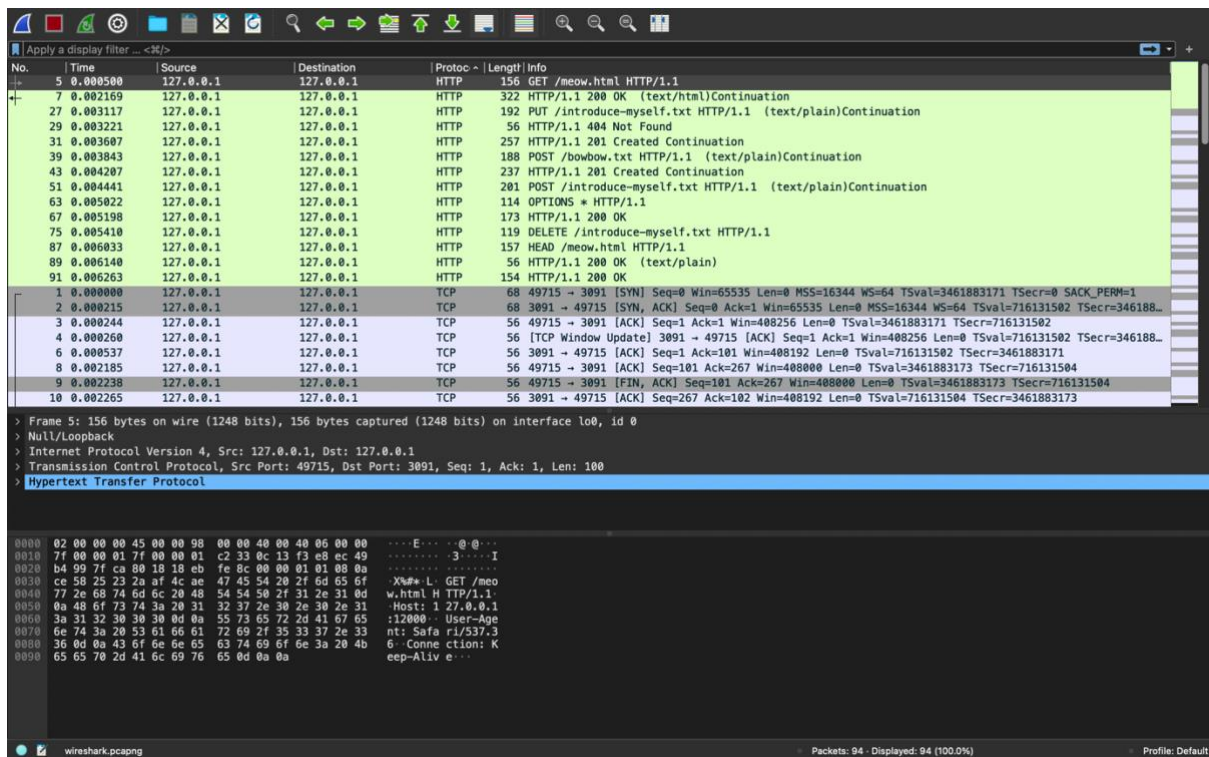
HTTP/1.1 200 OK
Location: http://localhost:3091/introduce-myself.txt
Content-Type: text/plain

HTTP/1.1 200 OK
Allow: GET, HEAD, POST, PUT, OPTIONS, DELETE
Content-Length: 0
Date: Tue, 03 May 2022 19:59 KST

HTTP/1.1 200 OK
Content-Type: text/plain
Date: Tue, 03 May 2022 19:59 KST
I have your delete request, will take time to process

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 165
Date: Tue, 03 May 2022 20:13 KST
```

5. Wireshark 캡처 결과



- Capture한 파일 (.pcapng)는 개인 GitHub에 같이 업로드 하였음
- HTTP Protocol가 상단에 위치하도록 정렬한 결과
- 업로드 되어있는 GitHub 주소 : <https://github.com/Choi-Jiwon-38/2022-ComputerNetwork>