Computer Networks

CS 421

Can Kocagil

21602218

Programming Assignment I

Department of Electric & Electronics Engineering

Bilkent University

Ankara, Turkey

5.11.2021

Table of Contents

List of Figures

1. FILE DOWNLOADER

Let's recall the assignment specifications as follows. In this programming assignment, we are asked to implement a program in either Java or Python, that is supposed to download an index file to obtain a list of text file URLs and download some of these files depending on their sizes.

Command-line interface is required for the application, with two parameters.

- **index file**: [Required] The URL of the index that includes a list of text file URLs.
- **lower end point - upper end point**: [Optional] If this argument is not given, a file in the index is downloaded if it is found in the index. Otherwise, the bytes between lower end point and upper end point inclusively are to be downloaded.

When a user enters the command above, our program will send an HTTP GET request to the server in order to download the index file with URL index file. If the index file is not found, the response is a message other than 200 OK. In this case, our program will print an error message to the command-line and exits. If the index file is found, the response is a 200 OK message. When this is the case, our program will print the number of file URLs in the index file and send an HTTP HEAD request for each file URL in the index file.

When requested file is not found, the response is a message other than 200 OK. In this case, your program will print a message to the command-line indicating that the file is not found. Then, an HTTP HEAD request is sent for the next file

When requested file is found in the server, the response is a 200 OK message which includes the size of the file in bytes in the header. When this is the case, there are three possibilities:

- If the user does not give a range as a command-line argument, your program should send an HTTP GET message to obtain the content of the whole file
- If a range is given as a command-line argument and the size of the file is smaller than lower end point, the file will not be downloaded and your program will print a message to the command-line indicating that the file is not requested. Then, an HTTP HEAD request is sent for the next file.
- If a range is given as a command-line argument and the size of the file is not smaller than lower end point, the range is satisfiable. Then, your program should send an HTTP GET message with the range lower end point-upper end point and obtain a part of the file content from the HTTP 206 Partial Content response.

Then, if the program successfully obtains the file or a part of the file, it saves the content under the directory in which your program runs. The name of the saved file should be the same as the downloaded file and a message indicating that the file is successfully downloaded is printed to the command-line. Then an HTTP HEAD request is sent for the next file.

That was the *FileDownloader* application specifications. Additionally, the assignment asks to provide a brief explanation as to how the GET requests and corresponding responses operate. The structure of this paper will be as follows. I briefly provide the Python code snippets for central operations for HTTP communication, include the outputs and their explanations. Then, I will finalize this paper by providing brief explanations on HTTP GET requests and responses. Let's start. The Python codes are just representative, as not whole codes are posted.

Given the fact that index file is broken or non-structured, I created a class for parsing URL to get host, directory and file name as follows.

```python
class URLParser:
    def __init__(self, url):
        self.url = url
        self.parse()

    def parse(self):
        """ Web URL parser """
        splitted_url = self.url.split('/')

        if splitted_url[0].startswith('http:'):
            splitted_url = splitted_url[2:]

        host_name = splitted_url[0]
        directory_name = "/".join(splitted_url[1:-1])
        file_name = splitted_url[-1]

        self.parsed_data = {
            'host': host_name,
            'directory_name' : '/' + directory_name,
            'file_name':file_name
        }

        for k, v in self.parsed_data.items():
            setattr(self, k, v)
```

Then, I created a HTTP client class for functionalizing the GET and Partial GET methods as follows.

```python
class HTTPClient:
    def __init__(
        self,
        url: URLParser,
        port = 80,
        verbose = 1,
    ):
        self.url = url
        self.verbose = verbose

        self.url_parsed = URLParser(url)
        self.port = port

        self.host = self.url_parsed.host
        self.filename = os.path.join(
            self.url_parsed.directory_name,
            self.url_parsed.file_name
        )

        self.data = []
```

Since we are implementing HTTP client to make requests, what we need is the URL of the content. From URL, we can extract host name, file name etc., as in the case of previous Python box. Then, let's see the HTTP GET request and other functionalities as follows. This will be a longer Python Code.

```python
    def get(
        self,
        bytes_to_read = 4096,
        lower_endpoint = 0,
        upper_endpoint = 1e30000,
        is_index_file = False,
    ):

        """ Sends HTTP GET request or Partial GET request if endpoints are specified """

        if is_index_file:
            print(f"URL of the index file: {self.url}")

        if lower_endpoint > upper_endpoint:
            print(f"Lower endpoint: {lower_endpoint} cannot be higher than upper endpoint: {upper_endpoint} !")

        is_range_given = any(
            [
                lower_endpoint != 0,
                upper_endpoint != 1e30000,
            ]
        )

        if self.verbose:
            if not is_range_given:
                if is_index_file:
                    print(f"No range is given")
            else:
                if is_range_given:
                    if is_index_file:
                        print(f"Lower endpoint = {lower_endpoint}\nUpper endpoint = {upper_endpoint}")

        if any(
            [
                lower_endpoint < 0,
                lower_endpoint < 0,
                lower_endpoint > 1e30000,
                upper_endpoint > 1e30000
            ]
        ):
            print(f"Lower Endpoint or Upper endpoint cannot be lower than 0 or higher than Python infinity!")

        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((self.host, self.port))
```

```python

            if not is_range_given:
                get_message = f"GET {self.filename} HTTP/1.0\r\nHost: {self.host}\r\n\r\n"
                print(f"HTTP GET request is sending")

            else:
                get_message = f"GET {self.filename} HTTP/1.0\r\nHost: {self.host}\r\n\r\nUser-Agent:Mozilla 5.0\rRange: bytes={lower_endpoint}-{upper_endpoint}\n\n"
                print(f"HTTP Partial GET request is sending")

            s.sendall(
                get_message.encode('utf-8')
            )

            while 1:
                try:
                    buf = s.recv(bytes_to_read)
                    self.data.append(buf)

                except socket.error as e:
                    print(f"Error receiving data: {e}" )

                    sys.exit(1)

                if not len(buf):
                    break

        response = HTTPResponse(
                b"".join([
                    data for data in self.data if data != b''
                ])
        )

        succesfully_downloadad = len(response.body.split()) != 0

        if not is_index_file and succesfully_downloadad:
            if is_range_given:
                print(f"{self.url} (range = {lower_endpoint}-{upper_endpoint}) is downloaded")
            else:
                print(f"{self.url} (size = {sys.getsizeof(response.body)}) is downloaded")

        if not is_index_file and not succesfully_downloadad:
            print(f"{self.url} is not found")

        return response
```

From line 1-43, I checked endpoint conditions and print some message to inform the command-line reader. Starting from line 43, I created a socket with TCP connection mechanism, and connect the host with specified port. Then, to make a HTTP GET request message I created a variable called *get_message* in line 47. This GET requests is made when no endpoints are specified, that is controlled by a boolean variable *is_range_given*. If endpoints are given, we want to make partial GET request with the given bytes range, and is provided in line 51. Then, we convert GET request from string to byte, then make the actual request. From that point, I opened a infinite loop, to receive content from the server I requested. Then, I created a HTTP message parser as *HTTPResponse* class to parse, as in line 71. Then, I print some command-line reader messages. Here is the HTTP response message parser.

```
1  class HTTPResponse:
2
3      status_codes = {
4          200: 'OK',
5          404: 'Not Found',
6          501: 'Not Implemented',
7      }
8
9      def __init__(self, data):
10         self.response_line = None
11         self.body = None
12         self.blank_line = False
13         self.header = None
14         self.status = None
15         self.parsed_data = {}
16         self.status_code = None
17
18         self.parse(data)
19
20     def parse(self, data):
21         """ HTTP response parsing """
22         response_list = data.decode('utf-8').split('\r\n')
23
24         if response_list[0].startswith('HTTP'):
25             self.response_line = response_list[0]
26
27             if '200' in self.response_line.split():
28                 self.status = 'HTTP/1.1 200 OK'
29                 self.status_code = 200
30
31             elif '404' in self.response_line.split():
32                 self.status = 'HTTP/1.1 404 Not Found'
33                 self.status_code = 404
34
35             else:
36                 self.status = 'HTTP/1.1 501 Not Implemented'
37                 self.status_code = 501
38
39         header = []
40
41         for data_line in response_list[1:-1]:
```

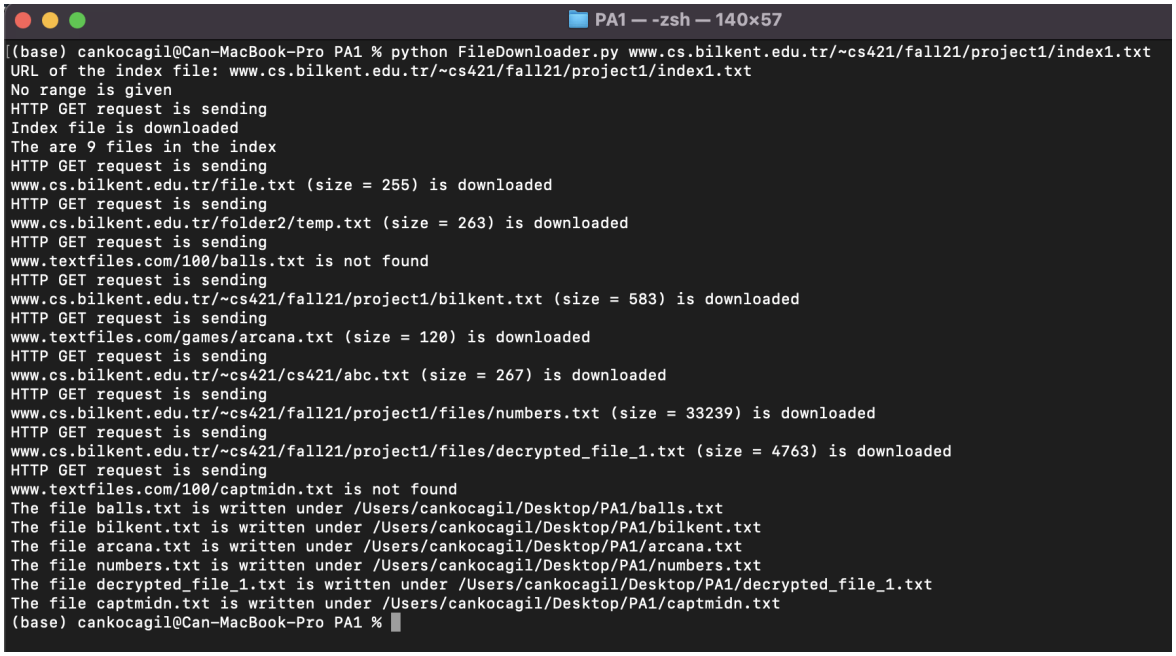
```
            if data_line == " ":
                self.blank_line = True
            else:
                header.append(data_line)

        self.header = "\n".join(header)

        self.body = response_list[-1]#.split()

        self.parsed_data['response_line'] = self.response_line
        self.parsed_data['status'] = self.status
        self.parsed_data['status_code'] = self.status_code
        self.parsed_data['header'] = self.header
        self.parsed_data['body'] = self.body

        return self
```

Here, I created my own HTTP response parser to parse the responses. All the code I provided up to now, covers most of the functionalities of our application, and finally I provide the main script as follows.

```
def main(args):
    """ Main script for FileDownloader """
    index_file = args.index_file

    client =  HTTPClient(
        url = index_file
    )
    response = client.get(
        is_index_file = True
    )

    print(f"Index file is downloaded")
    print(f"The are {len(response.body.split())} files in the index")

    responses = []

    if args.endpoints:
        lower_endpoint, upper_endpoint = [int(endpoint) for endpoint in
        ↪  args.endpoints.split('-')]

    else:
        lower_endpoint, upper_endpoint = 0, 1e30000

    urls = response.body.split()

    for text_url in urls:

        client =  HTTPClient(
            url = text_url,
            verbose=1
    )

        response = client.get(
```

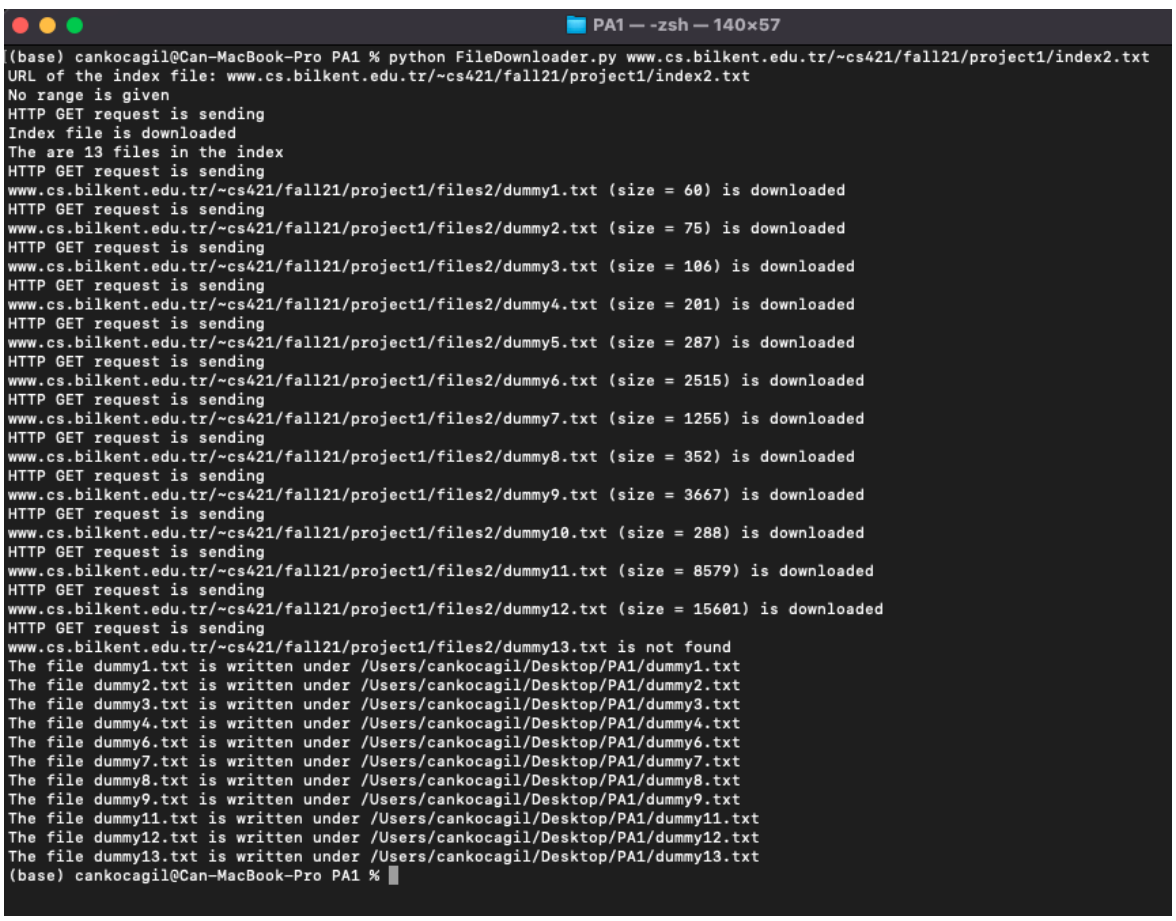
```
            lower_endpoint=lower_endpoint,
            upper_endpoint=upper_endpoint,
            is_index_file = False,
        )

        responses.append(
            response
        )

    status_codes = [response.status_code for response in responses]
    bodies = [response.body for response in responses]
    headers = [response.header for response in responses]

    for idx, (file_name, body) in enumerate(zip(urls, bodies)):

        if status_codes[idx] == 200:
            write_txt(
                file_name.split('/')[-1],
                body
            )

        else:
            pass
```

Now, let's provide sample results as follows. The snapshots of the terminal are self-explained and formatted according to the structure asked in the assignment.



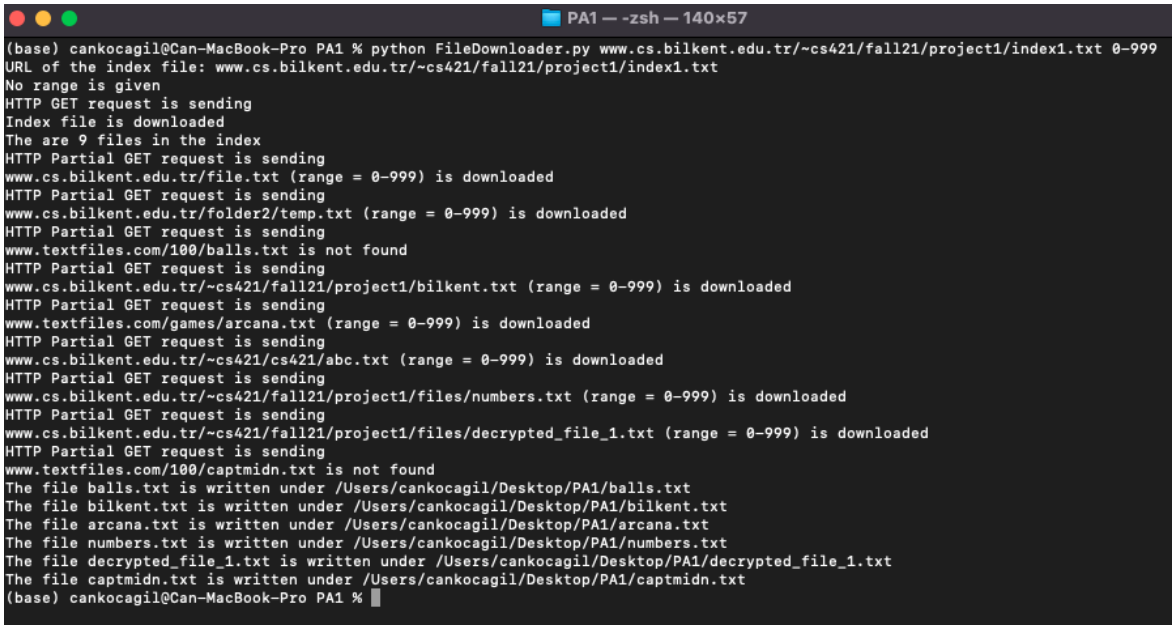
```
PA1 — -zsh — 140×57
(base) cankocagil@Can-MacBook-Pro PA1 % python FileDownloader.py www.cs.bilkent.edu.tr/~cs421/fall21/project1/index1.txt
URL of the index file: www.cs.bilkent.edu.tr/~cs421/fall21/project1/index1.txt
No range is given
HTTP GET request is sending
Index file is downloaded
The are 9 files in the index
HTTP GET request is sending
www.cs.bilkent.edu.tr/file.txt (size = 255) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/folder2/temp.txt (size = 263) is downloaded
HTTP GET request is sending
www.textfiles.com/100/balls.txt is not found
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/bilkent.txt (size = 583) is downloaded
HTTP GET request is sending
www.textfiles.com/games/arcana.txt (size = 120) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/cs421/abc.txt (size = 267) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files/numbers.txt (size = 33239) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files/decrypted_file_1.txt (size = 4763) is downloaded
HTTP GET request is sending
www.textfiles.com/100/captmidn.txt is not found
The file balls.txt is written under /Users/cankocagil/Desktop/PA1/balls.txt
The file bilkent.txt is written under /Users/cankocagil/Desktop/PA1/bilkent.txt
The file arcana.txt is written under /Users/cankocagil/Desktop/PA1/arcana.txt
The file numbers.txt is written under /Users/cankocagil/Desktop/PA1/numbers.txt
The file decrypted_file_1.txt is written under /Users/cankocagil/Desktop/PA1/decrypted_file_1.txt
The file captmidn.txt is written under /Users/cankocagil/Desktop/PA1/captmidn.txt
(base) cankocagil@Can-MacBook-Pro PA1 %
```

FIGURE 1. Sample Command-line Screenshot-I



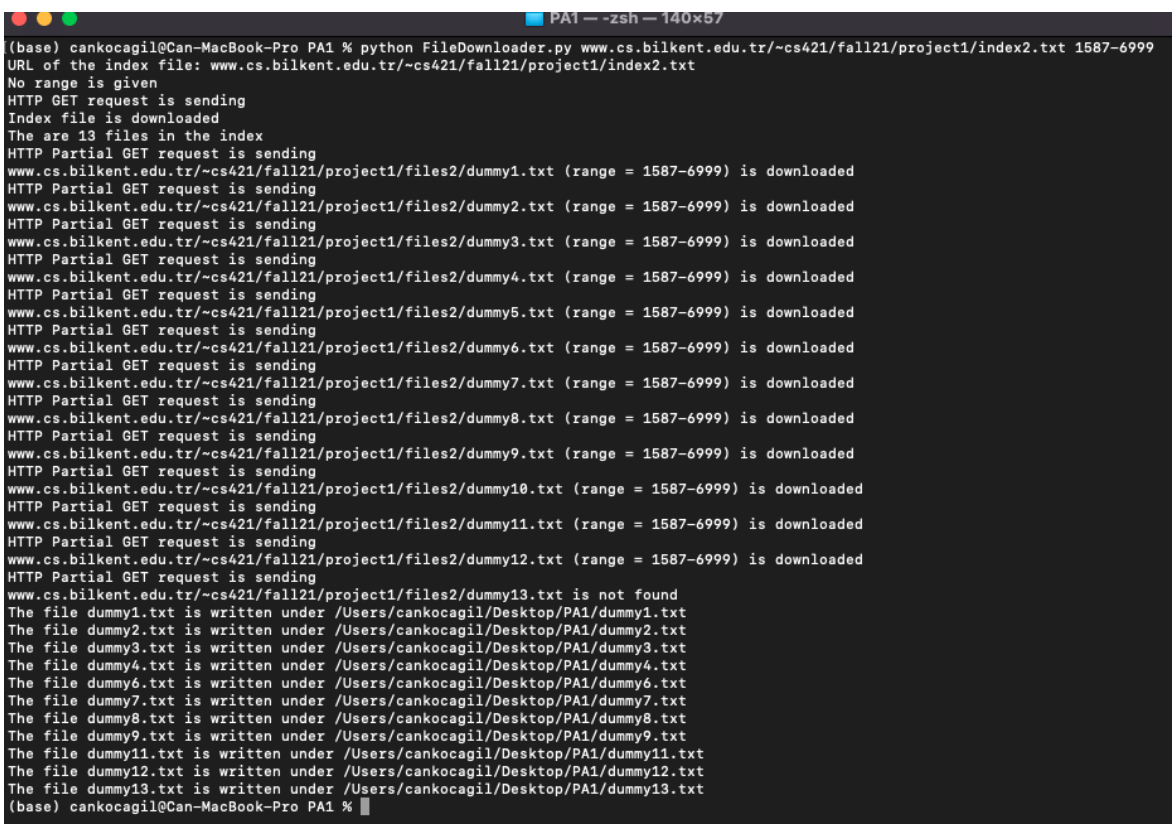
```
PA1 — -zsh — 140×57
(base) cankocagil@Can-MacBook-Pro PA1 % python FileDownloader.py www.cs.bilkent.edu.tr/~cs421/fall21/project1/index2.txt
URL of the index file: www.cs.bilkent.edu.tr/~cs421/fall21/project1/index2.txt
No range is given
HTTP GET request is sending
Index file is downloaded
The are 13 files in the index
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy1.txt (size = 60) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy2.txt (size = 75) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy3.txt (size = 106) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy4.txt (size = 201) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy5.txt (size = 287) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy6.txt (size = 2515) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy7.txt (size = 1255) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy8.txt (size = 352) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy9.txt (size = 3667) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy10.txt (size = 288) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy11.txt (size = 8579) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy12.txt (size = 15601) is downloaded
HTTP GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy13.txt is not found
The file dummy1.txt is written under /Users/cankocagil/Desktop/PA1/dummy1.txt
The file dummy2.txt is written under /Users/cankocagil/Desktop/PA1/dummy2.txt
The file dummy3.txt is written under /Users/cankocagil/Desktop/PA1/dummy3.txt
The file dummy4.txt is written under /Users/cankocagil/Desktop/PA1/dummy4.txt
The file dummy6.txt is written under /Users/cankocagil/Desktop/PA1/dummy6.txt
The file dummy7.txt is written under /Users/cankocagil/Desktop/PA1/dummy7.txt
The file dummy8.txt is written under /Users/cankocagil/Desktop/PA1/dummy8.txt
The file dummy9.txt is written under /Users/cankocagil/Desktop/PA1/dummy9.txt
The file dummy11.txt is written under /Users/cankocagil/Desktop/PA1/dummy11.txt
The file dummy12.txt is written under /Users/cankocagil/Desktop/PA1/dummy12.txt
The file dummy13.txt is written under /Users/cankocagil/Desktop/PA1/dummy13.txt
(base) cankocagil@Can-MacBook-Pro PA1 %
```

FIGURE 2. Sample Command-line Screenshot-II



```
PA1 — -zsh — 140×57
(base) cankocagil@Can-MacBook-Pro PA1 % python FileDownloader.py www.cs.bilkent.edu.tr/~cs421/fall21/project1/index1.txt 0-999
URL of the index file: www.cs.bilkent.edu.tr/~cs421/fall21/project1/index1.txt
No range is given
HTTP GET request is sending
Index file is downloaded
The are 9 files in the index
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/file.txt (range = 0-999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/folder2/temp.txt (range = 0-999) is downloaded
HTTP Partial GET request is sending
www.textfiles.com/100/balls.txt is not found
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/bilkent.txt (range = 0-999) is downloaded
HTTP Partial GET request is sending
www.textfiles.com/games/arcana.txt (range = 0-999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/cs421/abc.txt (range = 0-999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files/numbers.txt (range = 0-999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files/decrypted_file_1.txt (range = 0-999) is downloaded
HTTP Partial GET request is sending
www.textfiles.com/100/captmidn.txt is not found
The file balls.txt is written under /Users/cankocagil/Desktop/PA1/balls.txt
The file bilkent.txt is written under /Users/cankocagil/Desktop/PA1/bilkent.txt
The file arcana.txt is written under /Users/cankocagil/Desktop/PA1/arcana.txt
The file numbers.txt is written under /Users/cankocagil/Desktop/PA1/numbers.txt
The file decrypted_file_1.txt is written under /Users/cankocagil/Desktop/PA1/decrypted_file_1.txt
The file captmidn.txt is written under /Users/cankocagil/Desktop/PA1/captmidn.txt
(base) cankocagil@Can-MacBook-Pro PA1 %
```

FIGURE 3. Sample Command-line Screenshot-III



```
PA1 — -zsh — 140×57
(base) cankocagil@Can-MacBook-Pro PA1 % python FileDownloader.py www.cs.bilkent.edu.tr/~cs421/fall21/project1/index2.txt 1587-6999
URL of the index file: www.cs.bilkent.edu.tr/~cs421/fall21/project1/index2.txt
No range is given
HTTP GET request is sending
Index file is downloaded
The are 13 files in the index
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy1.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy2.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy3.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy4.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy5.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy6.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy7.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy8.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy9.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy10.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy11.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy12.txt (range = 1587-6999) is downloaded
HTTP Partial GET request is sending
www.cs.bilkent.edu.tr/~cs421/fall21/project1/files2/dummy13.txt is not found
The file dummy1.txt is written under /Users/cankocagil/Desktop/PA1/dummy1.txt
The file dummy2.txt is written under /Users/cankocagil/Desktop/PA1/dummy2.txt
The file dummy3.txt is written under /Users/cankocagil/Desktop/PA1/dummy3.txt
The file dummy4.txt is written under /Users/cankocagil/Desktop/PA1/dummy4.txt
The file dummy6.txt is written under /Users/cankocagil/Desktop/PA1/dummy6.txt
The file dummy7.txt is written under /Users/cankocagil/Desktop/PA1/dummy7.txt
The file dummy8.txt is written under /Users/cankocagil/Desktop/PA1/dummy8.txt
The file dummy9.txt is written under /Users/cankocagil/Desktop/PA1/dummy9.txt
The file dummy11.txt is written under /Users/cankocagil/Desktop/PA1/dummy11.txt
The file dummy12.txt is written under /Users/cankocagil/Desktop/PA1/dummy12.txt
The file dummy13.txt is written under /Users/cankocagil/Desktop/PA1/dummy13.txt
(base) cankocagil@Can-MacBook-Pro PA1 %
```

FIGURE 4. Sample Command-line Screenshot-IV

1.1. **Brief explanation on GET requests and responses.** I explained the context on previous sections, that are "in-action" explanations. Here, I will be providing more theoretical information on HTTP GET and responses.

As HTTP is built on top of TCP, we need to create a TCP agent, that then learns to make requests. Hence, the initial model is to introducing the TCP. Then, to make actual HTTP, we taught the protocol to make HTTP GET requests. Here is the sample HTTP GET request format.

```
GET /index.html HTTP/1.1
Host: example.com
Connection: keep-alive
User-Agent: Mozilla/5.0
```

FIGURE 5. HTTP Request Example

Hence, the format is as follows.

- The request line (it's the first line)
- Request headers (optional)
- A blank line
- Request body (optional)

GET is used to request data from a specified resource and is one of the most common HTTP methods. Let's itemize the basic functionalities of HTTP GET request.

- GET requests can be cached [1]
- GET requests remain in the browser history [1]
- GET requests can be bookmarked [1]
- GET requests have length restrictions [1]
- GET requests are only used to request data (not modify) [1]

Let's move on to HTTP GET responses. In follows, I provided a sample HTTP response parsed data in json format.

```
    print(json.dumps(response.parsed_data, indent = 4))
[15] ✓ 0.1s                                                                    Python
... {
        "response_line": "HTTP/1.1 200 OK",
        "status": "HTTP/1.1 200 OK",
        "status_code": 200,
        "header": "Date: Wed, 10 Nov 2021 17:14:14 GMT\nServer: Apache/2.4.25 (FreeBSD) OpenSSL/1.0.2u-freebsd PHP/7.4.15\nLast-Modified: Sat, 06 Nov
    2021 11:11:25 GMT\nETag: \"197-5d01cd2175dc5\"\nAccept-Ranges: bytes\nContent-Length: 407\nConnection: close\nContent-Type: text/plain\n",
        "body":
    "www.cs.bilkent.edu.tr/file.txt\nwww.cs.bilkent.edu.tr/folder2/temp.txt\nwww.textfiles.com/100/balls.txt\nwww.cs.bilkent.edu.tr/~cs421/fall21/proje
    ct1/bilkent.txt\nwww.textfiles.com/games/arcana.txt\nwww.cs.bilkent.edu.tr/~cs421/cs421/abc.txt\nwww.cs.bilkent.edu.tr/~cs421/fall21/project1/files
    /numbers.txt\nwww.cs.bilkent.edu.tr/~cs421/fall21/project1/files/decrypted_file_1.txt\nwww.textfiles.com/100/captmidn.txt\n"
    }
```

FIGURE 6. Example HTTP Response Message in JSON Format

Let's see the HTTP GET response deeper as follows. Recall that I created a HTTPResponse Python class to parse to response. In the following, I will print to response as strings.

```
    print(response.response_line)
    print(response.header)
    print(response.body)
[21]  ✓ 0.2s                                                    Python
...  HTTP/1.1 200 OK
     Date: Wed, 10 Nov 2021 17:14:14 GMT
     Server: Apache/2.4.25 (FreeBSD) OpenSSL/1.0.2u-freebsd PHP/7.4.15
     Last-Modified: Sat, 06 Nov 2021 11:11:25 GMT
     ETag: "197-5d01cd2175dc5"
     Accept-Ranges: bytes
     Content-Length: 407
     Connection: close
     Content-Type: text/plain

     www.cs.bilkent.edu.tr/file.txt
     www.cs.bilkent.edu.tr/folder2/temp.txt
     www.textfiles.com/100/balls.txt
     www.cs.bilkent.edu.tr/~cs421/fall21/project1/bilkent.txt
     www.textfiles.com/games/arcana.txt
     www.cs.bilkent.edu.tr/~cs421/cs421/abc.txt
     www.cs.bilkent.edu.tr/~cs421/fall21/project1/files/numbers.txt
     www.cs.bilkent.edu.tr/~cs421/fall21/project1/files/decrypted_file_1.txt
     www.textfiles.com/100/captmidn.txt
```

FIGURE 7. Example HTTP Response Message in String Format

As we can see from the above figure, HTTP GET response fundamentally consist of the following structures

- **Response Line**: Response line indicates what version of HTTP request is send, and the response status
- **Response Header**: Response header contain more information about the resource to be fetched, and about the client requesting the resource. Additionally, response header contains date, accept-ranges, connection, content-type, location and some other supplementary information.
- **Response Blank Line**: Response blank line crucial for discrimination of body, the data we want, from the header.
- **Response Body**: Response body is the data we want to fetch from the server.

1.2. **Conclusion.** This will be non-technical conclusion.*FileDownloader* was a playful and simple networking application to enlighten our understanding of HTTP GET requests and responses. Given the technical specifications, our command-line application serves specific purpose in action, that is downloading some text files from the web with the given initial URL index file. Additionally, *FileDownloader* is able to make Partial GET requests to fetch bytes within the given endpoints. By doing so, we play with TCP agents, HTTP communication, HTTP response message parsing in action, that gives me practical opportunity to apply bunch of networking operations in high-level programming languages. Nevertheless, the programming assignment was quite fruitful and playful for me.

APPENDIX A. CODE

```python
# To add a new cell, type '# %%'
# To add a new markdown cell, type '# %% [markdown]'
# %%
import argparse
import logging
import socket
import json
import sys
import os
import re
# %%
class URLParser:
    def __init__(self, url):
        self.url = url
        self.parse()

    def parse(self):
        """ Web URL parser """
        splitted_url = self.url.split('/')

        if splitted_url[0].startswith('http:'):
            splitted_url = splitted_url[2:]

        host_name = splitted_url[0]
        directory_name = "/".join(splitted_url[1:-1])
        file_name = splitted_url[-1]

        self.parsed_data = {
            'host': host_name,
            'directory_name' : '/' + directory_name,
            'file_name':file_name
        }

        for k, v in self.parsed_data.items():
            setattr(self, k, v)

class HTTPResponse:

    status_codes = {
        200: 'OK',
        404: 'Not Found',
        501: 'Not Implemented',
    }

    def __init__(self, data):
        self.response_line = None
        self.body = None
        self.blank_line = False
        self.header = None
        self.status = None
        self.parsed_data = {}
        self.status_code = None
```

```python

        self.parse(data)

    def parse(self, data):
        """ HTTP response parsing """
        response_list = data.decode('utf-8').split('\r\n')

        if response_list[0].startswith('HTTP'):
            self.response_line = response_list[0]

            if '200' in self.response_line.split():
                self.status = 'HTTP/1.1 200 OK'
                self.status_code = 200

            elif '404' in self.response_line.split():
                self.status = 'HTTP/1.1 404 Not Found'
                self.status_code = 404

            else:
                self.status = 'HTTP/1.1 501 Not Implemented'
                self.status_code = 501

        header = []

        for data_line in response_list[1:-1]:
            if data_line == " ":
                self.blank_line = True
            else:
                header.append(data_line)

        self.header = "\n".join(header)

        self.body = response_list[-1]#.split()

        self.parsed_data['response_line'] = self.response_line
        self.parsed_data['status'] = self.status
        self.parsed_data['status_code'] = self.status_code
        self.parsed_data['header'] = self.header
        self.parsed_data['body'] = self.body

        return self

class HTTPClient:
    def __init__(
        self,
        url: URLParser,
        port = 80,
        verbose = 1,
    ):
        self.url = url
        self.verbose = verbose

        self.url_parsed = URLParser(url)
        self.port = port

```

```python
        self.host = self.url_parsed.host
        self.filename = os.path.join(
            self.url_parsed.directory_name,
            self.url_parsed.file_name
        )

        self.data = []

    def get(
        self,
        bytes_to_read = 4096,
        lower_endpoint = 0,
        upper_endpoint = 1e30000,
        is_index_file = False,
    ):

        """ Sends HTTP GET request or Partial GET request if endpoints are specified """

        if is_index_file:
            print(f"URL of the index file: {self.url}")

        if lower_endpoint > upper_endpoint:
            print(f"Lower endpoint: {lower_endpoint} cannot be higher than upper endpoint: {upper_endpoint} !")

        is_range_given = any(
            [
                lower_endpoint != 0,
                upper_endpoint != 1e30000,
            ]
        )

        if self.verbose:
            if not is_range_given:
                if is_index_file:
                    print(f"No range is given")
            else:
                if is_range_given:
                    if is_index_file:
                        print(f"Lower endpoint = {lower_endpoint}\nUpper endpoint = {upper_endpoint}")

        if any(
            [
                lower_endpoint < 0,
                lower_endpoint < 0,
                lower_endpoint > 1e30000,
                upper_endpoint > 1e30000
            ]
        ):
            print(f"Lower Endpoint or Upper endpoint cannot be lower than 0 or higher than Python infinity!")

```

```python
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((self.host, self.port))

            if not is_range_given:
                get_message = f"GET {self.filename} HTTP/1.0\r\nHost: {self.host}\r\n\r\n"
                print(f"HTTP GET request is sending")

            else:
                get_message = f"GET {self.filename} HTTP/1.0\r\nHost: {self.host}\r\n\r\nUser-Agent:Mozilla 5.0\rRange: bytes={lower_endpoint}-{upper_endpoint}\n\n"
                print(f"HTTP Partial GET request is sending")

            s.sendall(
                get_message.encode('utf-8')
            )

            while 1:
                try:
                    buf = s.recv(bytes_to_read)
                    self.data.append(buf)

                except socket.error as e:
                    print(f"Error receiving data: {e}" )

                    sys.exit(1)

                if not len(buf):
                    break

        response = HTTPResponse(
                b"".join([
                    data for data in self.data if data != b''
                ])
        )

        succesfully_downloadad = len(response.body.split()) != 0

        if not is_index_file and succesfully_downloadad:
            if is_range_given:
                print(f"{self.url} (range = {lower_endpoint}-{upper_endpoint}) is downloaded")
            else:
                print(f"{self.url} (size = {sys.getsizeof(response.body)}) is downloaded")

        if not is_index_file and not succesfully_downloadad:
            print(f"{self.url} is not found")

        return response

def write_txt(file_name, body):
    """ Writes strings to file in current directory """
```

```python
    current_dir = os.getcwd()
    write_dir = os.path.join(current_dir, file_name)

    with open(f"{write_dir}", 'a') as f:
        f.write(body)

    print(f"The file {file_name} is written under {write_dir}")

def header2dict(header):
    """ Converts HTML string header to dictionary """
    header_dict = {}

    for text in header:
        index = text.find(':')

        if index != -1:
            key = text[:index]
            value = text[index + 1:]
            header_dict[key] = value.strip()

    return header_dict

def main(args):
    """ Main script for FileDownloader """
    index_file = args.index_file

    client =  HTTPClient(
        url = index_file
    )
    response = client.get(
        is_index_file = True
    )

    print(f"Index file is downloaded")
    print(f"The are {len(response.body.split())} files in the index")

    responses = []

    if args.endpoints:
        lower_endpoint, upper_endpoint = [int(endpoint) for endpoint in
        ↪  args.endpoints.split('-')]

    else:
        lower_endpoint, upper_endpoint = 0, 1e30000

    urls = response.body.split()

    for text_url in urls:

        client =  HTTPClient(
            url = text_url,
            verbose=1
    )

        response = client.get(
```

```
            lower_endpoint=lower_endpoint,
            upper_endpoint=upper_endpoint,
            is_index_file = False,
        )

        responses.append(
            response
        )

    status_codes = [response.status_code for response in responses]
    bodies = [response.body for response in responses]
    headers = [response.header for response in responses]

    for idx, (file_name, body) in enumerate(zip(urls, bodies)):

        if status_codes[idx] == 200:
            write_txt(
                file_name.split('/')[-1],
                body
            )

        else:
            pass
# %%
if __name__ == '__main__':
    assert len(sys.argv) != 1, f"Please provide index file, it is required!"

    index_file = sys.argv[1]

    if 1 <= len(sys.argv) <= 2:
        endpoints = None
    else:
        endpoints = sys.argv[2]

    class Args:
        index_file = index_file
        endpoints = endpoints

    main(
       Args()
    )
```

REFERENCES

[1] *HTTP request methods*. URL: https://www.w3schools.com/tags/ref_httpmethods.asp.