BILKENT UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CS 421 – Computer Networks

Programming Assignment 1

11.11.2021

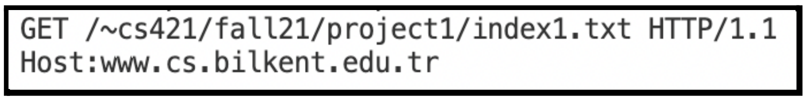
Burak Yetiştiren 21802608

Işık Özsoy 21703160

What is HTTP?

The *Hypertext Transfer Protocol (HTTP)* is an application-level protocol for data communication for the *World Wide Web (WWW)*. It is a *TCP* based communication protocol which is used to transfer data such as *HTML* and image files, query results. In *HTTP* the default port number is 80 for a *TCP socket*, which we also used in this assignment.

In this assignment, we were required to send an HTTP GET request to the server to request and download the index file. The structure of the HTTP GET message was as follows:



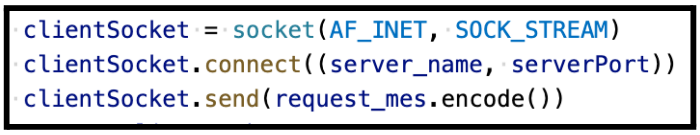
```
GET /~cs421/fall21/project1/index1.txt HTTP/1.1
Host:www.cs.bilkent.edu.tr
```

Figure 1: *HTTP GET* message structure

Here, the "/~cs421/fall21/project1/index.txt" part specifies the path of the file in the server, where "www.cs.bilkent.edu.tr" specifies the name of the *server host*. Name of the *server host* is specified in the *HTTP GET* request message, under the *Host header*. For our type of HTTP connection, we chose *HTTP 1.1*, which is known as *HTTP with persistent connection*, and differs from *HTTP 1.0*, which is a *non-persistent connection* by leaving the connection open after *TCP handshake*, and receiving the first file.

1. TCP operations using socket

After the message was created a *client socket*. After creation, we connected to the *server* using the *client socket*, which would initiate the *three-way handshake* between the *client* and the *server*. As we explained previously, we used the default *port number* for a *TCP socket*, that is *80* to connect to the *server*. After the connection establishment, we first *encoded* the message from string to bytes to be able to send it using *TCP*. Then we sent the message to the *TCP send buffer* through the socket
In Figure 2, the explained procedure could be seen in Python code. In Figure 3, the procedure can be seen visually by the illustration.



```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((server_name, serverPort))
clientSocket.send(request_mes.encode())
```

Figure 2: *TCP* operations using socket in Python code

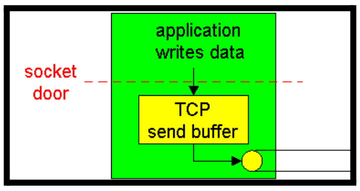
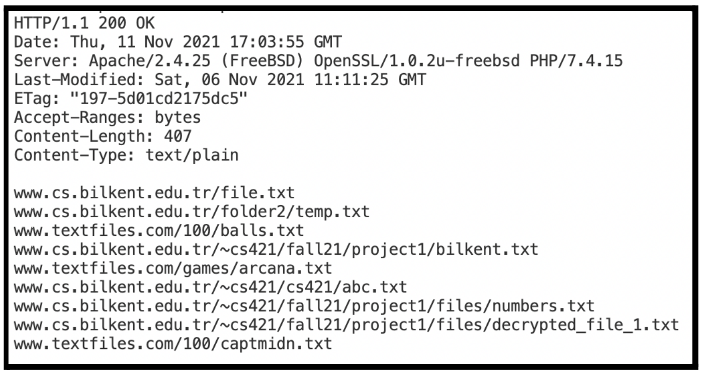


Figure 3: *TCP* operations using socket illustrated [1]

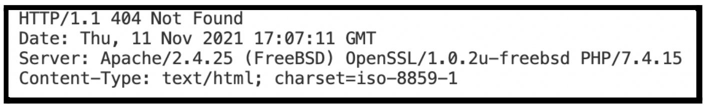
2. HTTP Response Status Codes

After these procedures, we were ready to receive the *HTTP status message* back from the *server* it would send to the *client*. If the *server* could successfully send the requested file, it would return a message containing *HTTP 200 OK response status code*. If the *server* could not find the file that we have requested, it would return the *HTTP 404 Not Found response status code* contained in the message. Additionally, we have observed *HTTP 400 Bad Request*, and *HTTP 206 Partial Content response status codes* in the scope of this assignment. The first *response status code* indicated a syntax error in the *request message*, which was directed to the server, whereas the latter *response status code* indicated the successful delivery of a file *partially*, which is contained in the *server*. In accordance with the *request* sent to the *server*, which contains some arguments in the *range header*, indicating a *partial request* to the *server* for a specific file. Of course, there are much more *response status codes* other than the four we have stated, but for the scope of this assignment, we wanted to specify the ones we could observe while implementing our system.



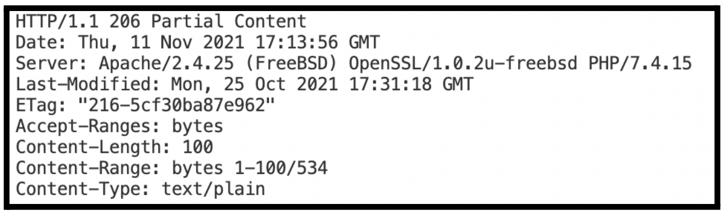
```
HTTP/1.1 200 OK
Date: Thu, 11 Nov 2021 17:03:55 GMT
Server: Apache/2.4.25 (FreeBSD) OpenSSL/1.0.2u-freebsd PHP/7.4.15
Last-Modified: Sat, 06 Nov 2021 11:11:25 GMT
ETag: "197-5d01cd2175dc5"
Accept-Ranges: bytes
Content-Length: 407
Content-Type: text/plain

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 4: Sample server response with *HTTP 200 OK response status code*



```
HTTP/1.1 404 Not Found
Date: Thu, 11 Nov 2021 17:07:11 GMT
Server: Apache/2.4.25 (FreeBSD) OpenSSL/1.0.2u-freebsd PHP/7.4.15
Content-Type: text/html; charset=iso-8859-1
```

Figure 5: Sample server response with *HTTP 404 Not Found response status code*

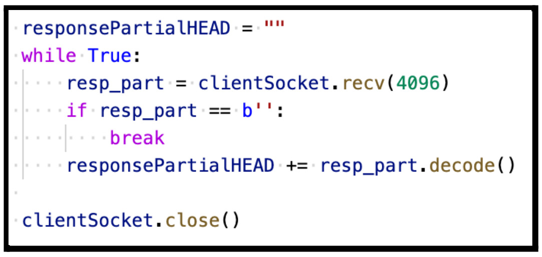


```
HTTP/1.1 206 Partial Content
Date: Thu, 11 Nov 2021 17:13:56 GMT
Server: Apache/2.4.25 (FreeBSD) OpenSSL/1.0.2u-freebsd PHP/7.4.15
Last-Modified: Mon, 25 Oct 2021 17:31:18 GMT
ETag: "216-5cf30ba87e962"
Accept-Ranges: bytes
Content-Length: 100
Content-Range: bytes 1-100/534
Content-Type: text/plain
```

Figure 6: Sample server response with *HTTP 206 Partial Content response status code*

3. *Receiving the messages*

After the server sends the requested message to the client, the response message is buffered in the TCP receive buffer, such that a program should receive the data from the buffer through client socket. We used loops to receive all the data transferred by the server to our program. In Figure 7, an example Python code from our program can be seen. In Figure 8, the illustration of the procedure could be seen.



```python
responsePartialHEAD = ""
while True:
    resp_part = clientSocket.recv(4096)
    if resp_part == b'':
        break
    responsePartialHEAD += resp_part.decode()

clientSocket.close()
```

Figure 7: Receiving the file to the program stored in client *receive buffer* in Python code

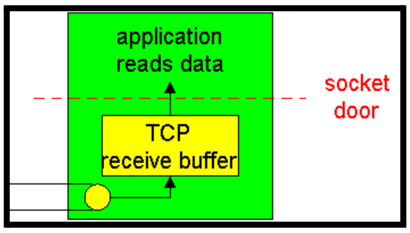


Figure 8: Receiving the file to the program stored in client *receive buffer* illustrated [1]

NOTE: Whilst running our application, the structure of the command should be the following:

<u>If a range is not given:</u>

- python3 FileDownloader.py `<index_file>`

Example:

python3 FileDownloader.py www.cs.bilkent.edu.tr/~cs421/fall21/project1/index1.txt

<u>If a range is given:</u>

- python3 FileDownloader.py `<index_file> <lower_endpoint>-<upper_endpoint>`

Example:

python3 FileDownloader.py www.cs.bilkent.edu.tr/~cs421/fall21/project1/index1.txt 1-100

References

[1] “Transmission control protocol.” http://www2.ic.uff.br/~michael/kr1999/3-transport/3_05-segment.html (accessed Nov. 11, 2021).