CS 421 - Computer Networks
Fall 2021 Programming Assignment 1
FileDownloader

Emrecan Kutay 21702500 Mehmet Berk Şahin 21703190

1. Introduction

In this assignment, the purpose was to study, implement, and observe the HTTP protocol over a TCP socket. For this purpose, a file downloader app was implemented which downloads the base HTML file, extracts and reads the possible URLs in it, and downloads the content in them if possible. During this process, the availability of the content was also checked with the status messages like **200 OK, 404 NOT FOUND, 206 PARTIAL CONTENT**. Furthermore, the user is able to have partial content instead of the whole by selecting the range. This was done by using the partial requests in HTTP. Having the communication implementation, obtained contents was saved in a .txt file which has names same as the original.

2. Implementation

In the implementation, TCP connection and Python Socket library were investigated first since a TCP connection is required which HTTP will communicate over. After the TCP Socket connection, an HTTP GET message should be sent. To find the format of the HTTP GET message, RFC2616 was investigated which was provided in the assignment. It was found that the message should be in the format,

“GET “ + requestURL + " HTTP/1.1\r\nHost: " + serverURL + "\r\n\r\n"

For example, to have the base HTML file, the sent message is,

“ GET /~cs421/fall21/project1/index1.txt HTTP/1.1\r\nHost: www.cs.bilkent.edu.tr \r\n\r\n”

Sending the get message, a response message was obtained. In the obtained message, there is an HTTP header and a content part. In order to control the received response is accurate. **200 OK** message was searched through the message. If it was not found, it means that there is an inaccuracy, so the program does not continue to the rest. Having the **200 OK** message, we have received a correct response message, so the URLs were extracted from the message. For the extraction, the content part of the message was separated and lines containing “.txt” extent were taken as new destination URLs. Having new destination URLs, head requests were sent to each of them. However, since each may have a different server, the old TCP socket closed and a new one was created for each. The response of the head message also indicated whether there is available content. It also stated the length of the available message. However, during the implementation, it was found that for the head response of content with 0 length, the head message does not include a content-length part. Since we should get the

content length before sending a GET message to it for range request, it was implemented in a way that if the content-length part does not exist, the size was taken as 0.
Also, it was asked for range requests in the assignment. To understand whether a user enters a range or not, try-except was used as can be seen in the code. If the range was given, HTTP Get message with a range was sent which is in the format,

"GET /~cs421/fall21/project1/index1.txt HTTP/1.1\r\nHost: www.cs.bilkent.edu.tr\r\n Range: bytes= 0-100 \r\n\r\n"

For an example of a get request for interval 0-100. The most critical part is getting large files over HTTP. This case was also observed in the Wireshark Assignment and also encountered here. During the implementation, it was found that the system was getting only the first response, not the possible continuing ones for text files having large sizes. Since there are multiple segments in the response, they should be considered. For the solution, a loop was implemented in which it controls the received messages at each iteration. If the received message is empty, it exits the loop. Having this implementation, continuing segments were also captured and the message was completely received properly. The interval was also checked and compared with the size of the content and requirements in the assignment was implemented. Also, to get the original file names of ".txt" files, the part containing ".txt" between the '/' was taken in the URL was taken.

3. Conclusion

To sum, in this programming assignment, we get familiar with the internals of the HTTP protocol by using the Socket package of Python distribution. Our program takes, as an input, the URL of the html file. Then, it downloads the content for the given range of each html file and saves those files to the local drive. Furthermore, to understand the content of the HTTP GET and HEAD messages and to verify our results, we used Wireshark. We learned how to implement HTTP protocol in Python, we saw the practical application of range request and we observed the importance of the headers in HTTP messages.