Technische Universität München
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. W. Kellerer
Betreuer - Andreas Blenk, Christian Sieber

Internetkommunikation - Class Project SS2014

Programmieraufgabe 1

Web Server

In dieser Programmieraufgabe sollen Sie einen einfachen Web Server in Python implementieren. Das im Folgenden beschriebene Szenario ist in Abbildung 1 dargestellt. Der Web Server muss nur eine Anfrage gleichzeitig bearbeiten können und die folgenden Mindestanforderungen implementieren:

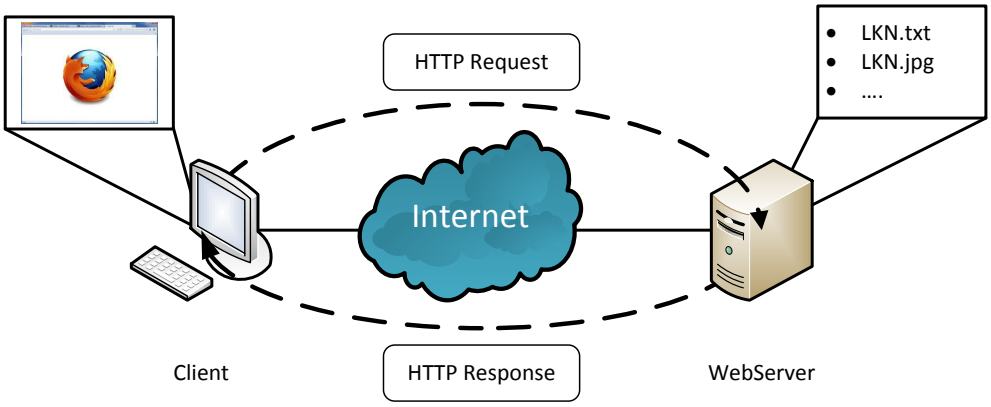

Abbildung 1: Web Server Szenario.

- Eine Socket Verbindung akzeptieren, wenn ihn ein HTTP Client via einer TCP Anfrage kontaktiert.
- Die HTTP Requests vom Clienten annehmen.
- Die Anfrage bearbeiten.
- Den root Ordner nach der angeforderten Datei durchsuchen.
- Eine HTTP Response erzeugen, welche die Datei enthält.
- Die Antwort mit der Datei über die bestehende TCP Verbindung an den anfragenden Clienten schicken.

Der Web Server soll nur `txt` und `jpg` Dateien zur Verfügung stellen. Außerdem sollen sich alle Dateien im root Verzeichnis befinden. D.h. es müssen keine Unterverzeichnisse nach Dateien durchsucht werden. Das zip File enthält die zwei Beispieldateien, die `LKN.txt` Datei und die `LKN.jpg` Datei. Falls die angeforderte Datei vorhanden ist, sollte der Server mit einer `200 OK HTTP Response` antworten und die Datei übertragen. Sollte eine Datei nicht auf dem Server vorhanden sein, so soll der Server eine `404 Not Found HTTP Response` erzeugen und an den Clienten zurückschicken. Die Response soll zusätzlich folgenden HTML Code enthalten:

```
<html>
    <head></head>
    <body>
        <h1>404 Not Found</h1>
    </body>
</html>\r\n
```

Beide Antworten sollen gemäß **RFC 2616** implementiert werden, dementsprechend auch die Status-line sowie die Header-lines **(Hinweis: Achten Sie besonders auf die korrekten Endungen der Zeilen!)**. Die Header-lines sollen mindestens folgende Attribute enthalten:

- Date
- Server
- Content-type

Das heruntergeladene zip File enthält zusätzlich noch eine Python Datei `WebServer.py`. In dieser Python Datei stellen wir ihnen das Codegerüst mit den notwendigen Funktionen zur Verfügung. Die Namen der Funktionen sowie ihre Argumente dürfen nicht verändert werden. Die Funktion `response200_OK()` soll nur den Header zurückliefern, falls die Anfrage korrekt war. Diese Methode sollte von der Funktion `parseMessage()` aufgerufen werden.

```
def response200_OK():
"""
Send a 200 OK message if the requested file exists on the
server.
"""
```

Die Funktion `response404_NotFound()` soll die vollständige Antwort, falls die File nicht gefunden werden konnte, zurückliefern.

```
def response404_NotFound():
"""
Send a 404 Not Found message if the requested file does not
exist on the server.
"""
```

Die Funktion `parseMessage()` soll die empfangen Nachricht verarbeiten und die kompletten Antwortdaten zurückliefern. Diese Funktion sollte die Funktionen `response200_OK`, `response404_NotFound`, `readAndReturnTXT` sowie `readAndReturnJPG` aufrufen und die Nachricht entsprechend zusammensetzen.

```
def parseMessage(message):
"""
Parse the received request.
Check if the request is valid.
Check if the requested file exists.
Create and return the response with or without the requested
file.
"""
```

Die Funktion `readAndReturnTXT()` soll die `LKN.txt` File einlesen und die Rohdaten zurückliefern.

```
def readAndReturnTXT():
"""
Read LKN.txt file and return the data
"""
```

Die Funktion `readAndReturnJPG()` soll die `LKN.jpg` File einlesen und die Rohdaten zurückliefern.

```
def readAndReturnJPG():
"""
Read LKN.jpg file and return the data
"""
```

In dieser Funktion sollen Nachrichten empfangen und an die Funktion `parseMessage()` übergeben werden. Anschließend sollen die Nachrichten von der Funktion `parseMessage()` entgegengenommen und an den Client vollständig zurückgeschickt werden.

```
def main():
   while 1:
        returnMessage = parseMessage(receivedMessage)
        connectionSocket.send(returnMessage)
```

Implementieren Sie die Logik, soweit diese in den Funktionen nicht vollständig ist. Sie dürfen je nach Bedarf den Code um eigene Funktionen erweitern. Nutzen Sie für die Evaluierung **nur** den Firefox Browser (v28). Der WebServer muss nicht für weitere Browser implementiert und getestet werden. Überprüfen sie ihren Server mithilfe des Firefox Browsers. Geben Sie z.B. folgende Zeile in die Adressleiste des Firefox Browsers ein, um eine Datei auf dem WebServer, welcher auf dem lokalen Host auf Port 8080 lauscht, anzufragen.

```
http://localhost:8080/LKN.txt
```

Bonus: Implementieren Sie weitere Antwortnachrichten gemäss dem RFC Standard.