Technische Universität München
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. W. Kellerer
Betreuer - Christian Sieber, Raphael Durner, Benedikt Hess

Internetkommunikation - Class Project

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:

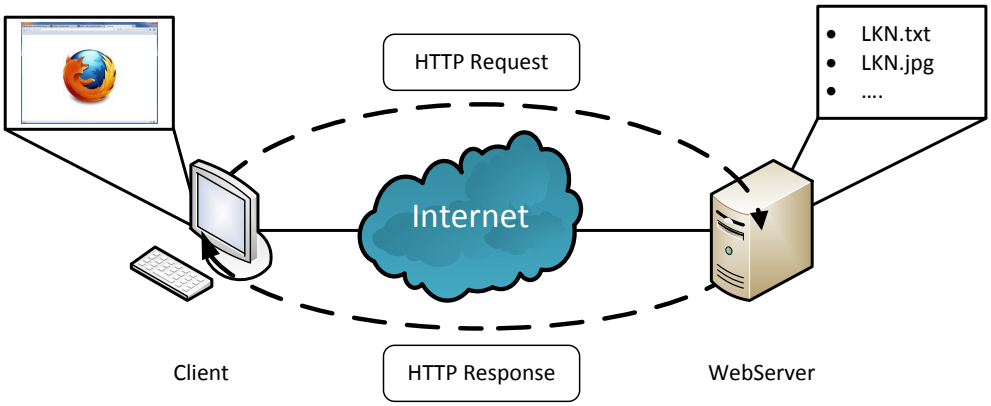

Abbildung 1: Web Server Szenario.

- Eine Socket Verbindung akzeptieren, wenn ihn ein HTTP Client via einer TCP Anfrage kontaktiert.
- Die HTTP Requests vom Client 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, `LKN.txt` und `LKN.jpg`. Falls die angeforderte Datei vorhanden ist, soll 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 Client 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 den response-header **(Hinweis: Achten Sie besonders auf die korrekten Endungen der Zeilen!**). Der response-header sollen mindestens folgende Felder enthalten:

- Date
- Server
- Content-type

Achten Sie bei den Attributen auf RFC-Konformität und nutzen Sie `Python Tutorial WebServer` als Server response-header field.

Das heruntergeladene zip File enthält die Python Datei `www/serve.py`. In dieser Python Datei stellen wir das Codegerüst mit den notwendigen Funktionen zur Verfügung. Die Namen der Funktionen sowie ihre Argumente dürfen mit Ausnahme der Funktion `serve(address,port)` verändert werden. Ein Teil der Funktionalität, wie zum Beispiel die `404 Not Found HTTP Response`, ist bereits vorgegeben.

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 den Firefox Browser. 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
```

Das zip File enthält außerdem ein Testskript `TestWebServer.py`, das ausgeführt werden muss, **während** Ihr WebServer läuft und die Anwendung auf mögliche Fehler überprüft.

Bonus: Implementieren Sie weitere response-header Felder gemäss dem RFC Standard.