Socket Programmierung in Python

Suman Kafle

13.06.2024

Lab-Anleitung

Bevor Sie mit dem Labor beginnen, stellen Sie sicher, dass Sie die folgenden Schritte ausgeführt haben:

1. Klonen Sie das Git-Repository mit dem Laborcode auf Ihren lokalen Computer.
2. Stellen Sie sicher, dass Sie Python auf Ihrem Computer installiert haben. Dieses Labor erfordert Python-Kenntnisse.

1 Entwicklung eines Webservers in Python

Aufgabenstellung

In dieser Übung werden Sie die Grundlagen der Socket-Programmierung für TCP-Verbindungen in Python kennenlernen. Ihr Ziel ist es, einen Webserver zu erstellen, der HTTP-Anfragen empfängt, verarbeitet und entsprechende Antworten zurücksendet. Diese Aufgabe umfasst das Parsen von HTTP-Anfragen, das Lesen von Dateien aus dem Dateisystem und das Erstellen von HTTP-Antwortnachrichten.

Zielsetzung

Entwickeln Sie einen Webserver, der folgende Aufgaben erfüllt:

1. Empfang und Parsing von HTTP-Anfragen.
2. Abrufen der angeforderten Datei aus dem Dateisystem des Servers.
3. Erstellen und Zurücksenden einer HTTP-Antwortnachricht, die den Inhalt der angeforderten Datei oder eine Fehlermeldung enthält, falls die Datei nicht existiert.

Anforderungen

1. Verarbeitung der HTTP-Anfrage:
    - Der Server soll HTTP-Anfragen über einen bestimmten Port empfangen und die Anfrage parsen, um den angeforderten Dateinamen zu ermitteln.
2. Dateiabruf und Antworterstellung:
    - Der Server soll die angeforderte Datei aus dem Dateisystem lesen.

- Wenn die Datei vorhanden ist, soll der Server eine HTTP 200 OK-Antwort zusammen mit dem Dateiinhalten zurücksenden.
- Wenn die Datei nicht vorhanden ist, soll der Server eine HTTP 404 Not Found-Antwort zurücksenden.

3. Fehlerbehandlung:
    - Der Server soll in der Lage sein, Fehler zu erkennen und entsprechend zu reagieren, beispielsweise bei ungültigen Anfragen oder nicht vorhandenen Dateien.

Hinweise

- Verwenden Sie Python und die Socket-Bibliothek zur Implementierung des Servers.
- Platzieren Sie eine HTML-Datei im gleichen Verzeichnis wie den Servercode.

Einzureichende Materialien

- Den vollständigen, gut dokumentierten Quellcode Ihres Webservers.

2 Entwicklung eines Multi-Connection Servers mit mehreren Clients

Aufgabenstellung

Das Ziel dieser Aufgabe ist es, einen Server in Python zu erstellen, der gleichzeitig mehrere Client-Verbindungen handhaben kann. Dies ermöglicht, dass mehrere Clients gleichzeitig Nachrichten an den Server senden und entsprechende Antworten (Echo) erhalten können.

Zielsetzung

Entwickeln Sie einen Webserver, der folgende Aufgaben erfüllt:

- Der Server zeigt an, dass er gestartet wurde und auf Verbindungen wartet.
- Mehrere Clients verbinden sich mit dem Server, senden Nachrichten und erhalten die Echo-Nachrichten zurück.
- Der Server verarbeitet jede Verbindung in einem separaten Thread, sodass mehrere Clients gleichzeitig bedient werden können.

Anforderungen

1. Erstellung der Serverfunktion
2. Erstellung der Clientfunktion
3. Fehlerbehandlung:
    - Der Server/Client soll in der Lage sein, Fehler zu erkennen und entsprechend zu reagieren.

Hinweise

- Verwenden Sie Python und die Socket-Bibliothek zur Implementierung des Servers.
- Platzieren Sie eine HTML-Datei im gleichen Verzeichnis wie den Servercode.

Einzureichende Materialien

- Screenshot.

3 UDP

Einführung

In diesem Labor werden Sie die Grundlagen der Socket-Programmierung für UDP in Python kennenlernen. Sie werden lernen, wie man Datagram-Pakete mit UDP-Sockets sendet und empfängt und wie man eine ordnungsgemäße Socket-Timeout-Einstellung vornimmt. Im Verlauf des Labors werden Sie mit einer Ping-Anwendung vertraut gemacht und deren Nützlichkeit zur Berechnung von Statistiken wie der Paketverlustquote verstehen.

Aufgabenstellung

Der vollständige Code für den Ping-Server ist im git angegeben. Sie müssen diesen Code kompilieren und ausführen.

In Servercode werden 30% der Pakete des Clients als verloren simuliert. Der Server befindet sich in einer Endlosschleife und wartet auf eingehende UDP-Pakete. Wenn ein Paket eingeht und eine randomisierte Ganzzahl größer oder gleich 4 ist, kapitalisiert der Server einfach die enthaltenen Daten und sendet sie an den Client zurück.

Paketverlust

UDP bietet Anwendungen einen unzuverlässigen Transportdienst. Nachrichten können im Netzwerk aufgrund von Router-Überlauf, fehlerhafter Hardware oder aus anderen Gründen verloren gehen. Da Paketverlust in typischen Campusnetzwerken selten oder sogar nicht existent ist, injiziert der Server in diesem Labor künstlichen Verlust, um die Auswirkungen von Netzwerk-Paketverlust zu simulieren. Der Server erstellt eine variable randomisierte Ganzzahl, die bestimmt, ob ein bestimmtes eingehendes Paket verloren geht oder nicht.

Client-Code

Sie müssen das folgende Client-Programm implementieren.

Der Client sollte 10 Pings an den Server senden. Da UDP ein unzuverlässiges Protokoll ist, kann ein vom Client an den Server gesendetes Paket im Netzwerk verloren gehen oder umgekehrt. Aus diesem Grund kann der Client nicht unbegrenzt auf eine Antwort auf eine Ping-Nachricht warten. Der Client sollte bis zu eine Sekunde auf eine Antwort warten; wenn innerhalb einer Sekunde keine Antwort empfangen wird, sollte Ihr Client-Programm davon ausgehen, dass das Paket während der Übertragung im Netzwerk verloren ging. Sie müssen die Python-Dokumentation konsultieren, um herauszufinden, wie man den Timeout-Wert auf einem Datagram-Socket einstellt.

Insbesondere sollte Ihr Client-Programm:

1. die Ping-Nachricht mit UDP senden (Hinweis: Im Gegensatz zu TCP müssen Sie keine Verbindung herstellen, da UDP ein verbindungsloses Protokoll ist.)

2. die Antwortnachricht des Servers, falls vorhanden, ausgeben

3. die Round-Trip-Zeit (RTT) in Sekunden für jedes Paket berechnen und ausgeben, falls der Server antwortet

4. andernfalls "Request timed outäusgeben

Während der Entwicklung sollten Sie das Programm `UDPPingerServer.py` auf Ihrem Computer ausführen und Ihren Client testen, indem Sie Pakete an `localhost` (oder 127.0.0.1) senden. Nachdem Sie Ihren Code vollständig debuggt haben, sollten Sie sehen, wie Ihre Anwendung über das Netzwerk kommuniziert, indem Sie den Ping-Server und den Ping-Client auf verschiedenen Maschinen ausführen.

Derzeit berechnet das Programm die Round-Trip-Zeit für jedes Paket und gibt sie einzeln aus. Ändern Sie dies so, dass es der Funktionsweise des Standard-Ping-Programms entspricht. Sie müssen die minimale, maximale und durchschnittliche RTT am Ende aller Pings vom Client berichten. Darüber hinaus berechnen Sie die Paketverlustquote (in Prozent).