Modul 63211– Verteilte Systeme
Einsendeaufgaben 1 – WS 2025/2026
FernUniversität Hagen, Lehrgebiet Kooperative Systeme

Aufgabe 1.1: Transparenz **6 + 6 + 6 Punkte**

1. Eine Ressource im Internet wird durch einen *Universal Resource Locator* (URL) identifiziert. Erklären Sie alle Informationen, die in

   `https://www.fernuni-hagen.de/mi/studium/`

   enthalten sind.

2. Inwiefern stellt die Verwendung von URLs eine Möglichkeit dar, *Ortstransparenz* zu erreichen? Warum bieten URLs mit Hilfe des *DNS* auch *Replikationstransparenz*?

3. Betrachten wir das E-Mail-System im Internet, wobei die Prozesse *Mail User Agent* und *Message Transfer Agent* (*Mail-Server*) daran beteiligt sind.

   Welche Prozesse gehören zur Anwendungsschicht und welche zur Middleware-Schicht?

   Welche Prozesse sind *zeitlich entkoppelt* (*temporally decoupled*), aber *referentiell gekoppelt* (*referentially coupled*)?

Aufgabe 1.2: Architektur **6 + 6 Punkte**

Erklären Sie kurz, wie Publish-Subscribe-Systeme sich von (klassischen) eng gekoppelten Systemen unterscheiden.

Aufgabe 1.3: Peer-to-Peer **6 + 6 + 6 + 6 + 6 Punkte**

1. Was ist der Unterschied zwischen *vertikaler Verteilung* (engl. vertical distribution) und *horizontaler Verteilung* (engl. horizontal distribution)?

2. In einem *Peer-to-Peer-System* werden Ressourcen von Hosts im Internet angeboten. Wozu benötigt man ein *Overlay-Netzwerk* für ein *Peer-to-Peer-System*?

3. Was ist ein strukturiertes *Overlay-Netzwerk*?

4. Beim Routing in einem strukturierten *Overlay-Netzwerk* werden die Nachrichten gemäß der logischen Verbindung von Peers gesendet. Ist der kürzeste Weg zwischen zwei Peers im *Overlay-Netzwerk* immer auch der kürzeste Weg zwischen ihnen im physischen Netzwerk? Begründen Sie Ihre Antwort.

5. Wir betrachten ein System aus *super peers* und *weak peers*, das Anfragen zu Dateien beantworten kann.

(a) Welche Informationen soll der Index von *super peers* mindestens speichern?

(b) Welche Informationen muss ein *weak peer* beim Eintreten in das System seinem *super peer* mitteilen?

(c) Was muss ein *super peer* tun, wenn sich ein *weak peer* bei ihm abmeldet?

Aufgabe 1.4: Client/Server 25 Punkte

Betrachten Sie das Python-Programm zur Client/Server-Kommunikation im Buch in Fig 2.3 (Seite 59, Distributed Systems, Third edition, Version 3.03 (2020)).

Entwickeln Sie daraus ein Client/Server-Programm in Python über TCP, bei dem der Client durch `"send time"` nach dem Datum und der Uhrzeit fragt, der Server dies beantwortet, und der Client die Antwort zur Kontrolle ausgibt. Bringen Sie Ihr Programm auf Ihrem Computer zum Laufen!

Hinweis: Server- und Client-Prozess können auf demselben Computer laufen, die für diesen Testzweck über die IP-Nummer 127.0.0.1 (localhost) kommunizieren dürfen. Wählen Sie selbst feste Portnummern.

Python ist eine Programmiersprache, die immer mehr Bedeutung gewinnt und in der jeder Informatiker Erfahrungen gesammelt haben sollte. Diskutieren Sie gerne Ihre Lösungen in der Newsgruppe.

Aufgabe 1.5: Anwendung von Threads 5 + 5 + 5 Punkte

Wir betrachten einen Server mit einem Cache-Bereich in folgendem vereinfachten Modell: Der Server bekommt laufend Anfragen, die in 80 % der Fälle mit Hilfe des Cache in 5 ms beantwortet werden können. Im ungünstigeren Fall (20 %) ist ein langsamer Festplattenzugriff und deshalb zusätzlich 25 ms nötig, hier also 30 ms pro Anfrage.

- Wie viele Anfragen pro Sekunde kann der Server maximal beantworten, wenn er dafür einen einzigen Prozess ohne weitere Threads einsetzt?

Wir überlegen nun, ob es vorteilhaft ist, die Anfragen auf eine feste Zahl von mehreren parallelen Threads zu verteilen. Die Zeit zur Erzeugung, Umschaltung und beim Scheduling von Threads soll vernachlässigt werden.

- Wie viele Anfragen pro Sekunde kann der Server höchstens mit Hilfe von mehreren Threads beantworten?
- Wie viele Threads sollen sinnvollerweise hier vorgesehen werden?