

1. Übung zur Vorlesung „Verteilte Betriebssysteme“

Universität Ulm · Fakultät für Informatik · Verteilte Systeme
Prof. Dr.-Ing. Franz J. Hauck · Andreas I. Schmied
{hauck,schmied}@informatik.uni-ulm.de

Abgabetermin: bis spätestens Freitag 31.10.2003 10:00

Hinweise zur Bearbeitung und Abgabe der VBS-Übungen:

- Die Aufgaben dieser Übung sind unter Java als Konsolenprogramme (textuell ohne GUI) und wenn nicht anders angegeben mit den Klassen des `java.io`-Pakets zu lösen.
- Es wird vorausgesetzt, dass Sie sich ggf. selbstständig das Basiswissen aus der angegebenen Literatur aneignen [1, 2].
- Bei Problemen, die Sie nicht mit Kollegen lösen können, wenden Sie sich bitte an vbs@vs.informatik.uni-ulm.de.
- Achten Sie auf lesbaren Quelltext und saubere objektorientierte Programmierung ohne überflüssigen oder „over-engineered“ Code.
- Halten Sie sich an die Java Code Conventions [6] und dokumentieren Sie Ihre Quellen ausreichend und nur wo nötig. Selbsterklärendes wie „setTitle“ o. ä. soll nicht dokumentiert werden.
- Der Code soll jeweils im Paket-Namensraum `vbs.u<übungsnr>.<login>...`, also z. B. `vbs.u1.xy98.Server` liegen.
- Schicken Sie Ihre Quelldateien (**keine temporären bzw. class-Files!**) inkl. Beschreibung (Textdatei README) und Matrikelnummer in einem Zip/Tar-Archiv an vbs@vs.informatik.uni-ulm.de. Das abgegebene Programm muss fehlerfrei übersetzbar sein (Hauptklasse angeben bzw. Makefile, ANT- oder Bash-Skript beilegen).

Die Arbeiten werden stichprobenartig mit einem Analysewerkzeug geprüft und Plagiate bei allen daran Beteiligten mit null Punkten bewertet! Neben den bewerteten Aufgaben werden weitere unbenotete Zusatzaufgaben gestellt, die Sie zu einer umfassenden Klausurvorbereitung ebenfalls lösen können sollten.

Aufgabe 1 Aufruf-Semantik (10 Punkte)

Entwerfen Sie einen Dienst zur Bewertung von Lehrveranstaltungen: In einer Vorlesung seien an alle Studenten Lose mit One-Time-Passwords verteilt worden, die einerseits die Anonymität sicherstellen und andererseits pro Person nur eine Wertung (Noten von 1 bis 5) zulassen. Die Studenten können mit Hilfe eines Client-Programms mit einem Wahl-Server kommunizieren, wobei eine *At-Most-Once*-Semantik für die Abgabe einer Bewertung eingehalten wird: ein Student kann mit seinem Los maximal eine Bewertung abgeben.

- a) Programmieren Sie einen Server, der UDP-Pakete mit Bewertungen entgegennimmt und an den Client nach dem Empfang eine kurze Erfolgsmeldung zurückmeldet [3]. Bedenken Sie bis zur nächsten Übung folgende Fragen:
 - Welche Port-Nummern sind für die jeweiligen Sockets zulässig?
 - Inwiefern unterscheiden diese sich von TCP-Portnummern?
 - Wie groß dürfen UDP-Pakete werden?
- b) Vervollständigen Sie ein fiktives Ablaufszenario durch ein Clientprogramm, das die parallele Wahl einiger Benutzer simuliert und die jeweilige Anfrage wiederholt, falls der Server nach einer festgelegten Zeitspanne nicht geantwortet hat.
- c) Testen Sie die Einhaltung der Semantik, indem Sie zufällige Verzögerungen und Nachrichtenverluste in das Antwortverhalten des Servers einbauen oder Verdopplung von Client-Nachrichten simulieren. Konstruieren Sie dafür im Client und im Server eine Zwischeninstanz zum Versenden bzw. Empfangen von Paketen anstatt diese direkt an einen Socket zu übergeben. In dieser Zwischeninstanz sollen sie Pakete beliebig manipulieren (z. B. verzögern) können [4].

Aufgabe 2 Gruppenkommunikation (0 Punkte, optional)

Die Veranstalter der Vorlesung wollen einen „Ticker“ auf ihrem Desktop mit den aktuellen Notenwerten sehen. Dazu benötigen Sie ein kleines Anzeigeprogramm, das sich am Server einmalig anmeldet und danach nach jeder Bewertung über die neue Notenverteilung informiert wird.

- a) Erweitern Sie den Server um diese An- und Abmeldeschnittstelle. Verwenden Sie nun TCP-Sockets zur Kommunikation.
 - Entwerfen Sie ein simples Protokoll, indem Sie kleine Nachrichtenobjekte über Objekt-Ströme (classes `ObjectInput/OutputStream`) [5] schicken.
 - Was muss man beachten, wenn zwar wenige Objekte serialisiert werden, diese aber (evtl. indirekt) Referenzen auf sehr viele abhängige Objekte halten?
 - Der Server soll maximal 2 Client-Verbindungen gleichzeitig zulassen.
 - Achten Sie darauf die TCP-Verbindung ordnungsgemäß zu beenden.
 - Bedenken Sie bis zur nächsten Übung, warum sich mehrere Client-Sockets des selben Prozesses simultan an die selbe Server-Portnummer verbinden lassen können?
- b) Gehen Sie davon aus, dass neben dem Professor auch noch die Assistenten gespannt auf die Ergebnisse sind und daher mehrere Ticker am Server anmeldbar sind. Nutzen Sie deshalb Multicast Nachrichten zur Übermittlung der Daten zu den Anzeigeprogrammen.

Literatur

- [1] Java2 SDK Documentation, v.a. API Specification — <http://java.sun.com/j2se/1.4.2/docs/index.html>
- [2] Java Tutorial — <http://java.sun.com/docs/books/tutorial/index.html>
- [3] Java Tutorial, Custom Networking/Broadcasting — <http://java.sun.com/docs/books/tutorial/networking/index.html>
- [4] Java Tutorial, Threads — <http://java.sun.com/docs/books/tutorial/essential/threads/index.html>
- [5] Java Tutorial, Object Serialization — <http://java.sun.com/docs/books/tutorial/essential/io/serialization.html>
- [6] Code Conventions for the Java Programming Language — <http://java.sun.com/docs/codeconv/index.html>