Verteilte Systeme

Übung B2 - Einzelarbeit

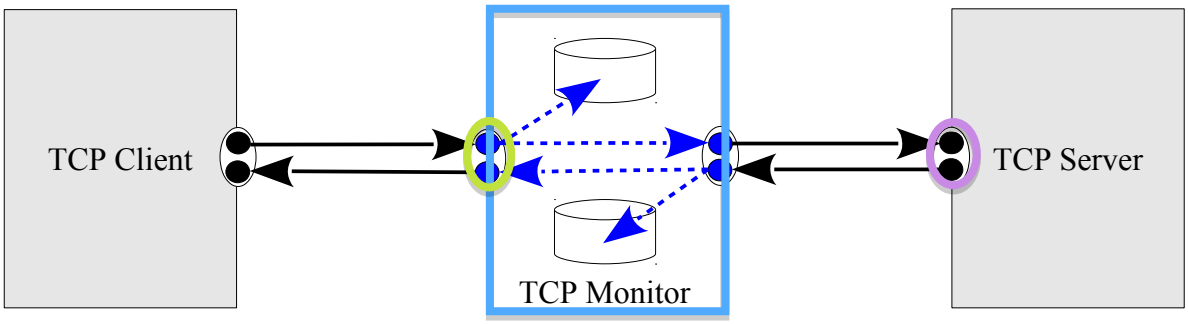
Bearbeitungszeit 2 Wochen

Die Übung adressiert den Umgang mit TCP Socket-Verbindungen, TCP Ports, und Streams. Weiterführende Informationen zum TCP-Protokoll gibt es unter http://en.wikipedia.org/wiki/Tcp_protocol.

Aufgabe: TCP-Monitor

Ein TCP-Monitor ist ein Werkzeug mit dessen Hilfe sich der Nachrichtenaustausch zwischen einem TCP Server und einem TCP Client protokollieren lässt. Die Idee ist dass der Client sich zum TCP Monitor statt direkt zum Server verbindet, und dieser wiederum eine Verbindung zum TCP Server aufbaut. Damit wird es möglich alle Daten die zwischen TCP Client und TCP Server, sowie umgekehrt versendet werden zu protokollieren.

Empfehlung: verwender BinaryTransporter.java & MultiOutputStream.java



Kopiert dazu die Klasse TcpMonitorSkeleton nach TcpMonitor. Diese Klasse repräsentiert den Rahmen für einen TCP-Server im (dynamischen) Acceptor/Service Multithreading-Entwurfsmuster. Implementiert die run()-Methode des Connection-Handlers nach den dort unter TODO hinterlegten Anweisungen. Startet Euren Tcp-Monitor mit den folgenden Parametern:

- 8010 (Service-Port)
- c:/temp (Context-Verzeichnis)
- www.cnn.com:80 (HTTP-Server) oder uranus.f4.htw-berlin.de:21 (FTP-Server)

Falls alles funktioniert sollte Euer Monitor in der Lage sein jedwede TCP-Nachrichten zu überwachen die vom Client über den lokalen Monitor-Port an den Zielserver weitergereicht werden, und umgekehrt. Allerdings gilt dies nur eingeschränkt für HTTP, da dort ein Sicherheitsfeature die Funktion des Monitors auf vielen populären Sites unterbindet (dazu ist der seit HTTP 1.1 notwendige Host-Eintrag im HTTP Request Header da); www.cnn.com sollte jedoch funktionieren. Um dieses zu umgehen müssten HTTP-Anfragen vom Monitor geparst, und der Host-Eintrag vor der Weitergabe abgeändert werden, was nicht Teil dieser Übung ist.

Beachtet auch dass ihr den Browser-Cache leeren müsst wenn ihr einen neuen HTTP-Server über einen bereits vorher verwendeten lokalen Port beobachten wollt; andernfalls findet keine TCP-Kommunikation statt weil die Inhalte aus dem lokalen Browser-Cache bezogen werden. Beachtet zudem bei Verwendung eines FTP-Clients dass das **passivate**-Kommando „PASV“ verwendet werden sollte um Firewall-Probleme beim Transfer von Verzeichnisinhalten und Dateien zu umgehen – siehe auch Aufgabe B1.

Zum Testen verwendet folgende Adressen, je nachdem mit welcher Art TCP-Server euer Monitor verbunden werden soll. In Eurem Context-Verzeichnis sollten dann bei der Bedienung des Web-Browsers oder FTP-Clients Dateien erscheinen die die jeweils kommunizierten HTTP/FTP Request- und Response Daten enthalten:

- **Web-Browser**: Eingabe von http://localhost:8010/ in die Adresszeile
- **FTP-Client:** Starten mittels Terminal-Aufruf „ftp localhost 8010“ (Linux & Windows)

Setzt Unterbrechungspunkte um den Ablauf und den Informationsfluss mittels Debugger genauer zu verstehen. Achtet speziell darauf dass Browser nach Beantwortung einer Anfrage für eine HTML-Seite als nächstes eine Vielzahl von Dateien (z.B. css und jpg) abfragen, dass solche Server eine TCP-Verbindung unter Umständen länger als für die Dauer einer Anfrage aufrecht erhalten, und dass solche Verbindungen daher für mehr als eine Anfrage genutzt werden können.