Verteilte Systeme

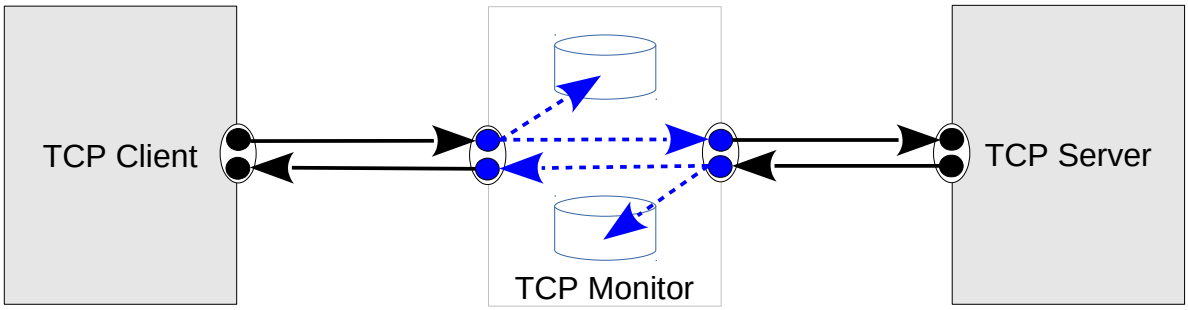
Übung B2

Bearbeitungszeit 2 Wochen

Die Übung adressiert den Umgang mit TCP Socket-Verbindungen, TCP Ports, und Streams. Weiterführende Informationen zum TCP-Protokoll gibt es u.A. in der englischen WIkipedia.

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 zu einem TCP Monitor Server 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.



Kopiert dazu die Klasse **`TcpMonitorServerSkeleton`** nach **`TcpMonitorServer`**. Diese Klasse repräsentiert den Rahmen für einen TCP-Monitor im *Acceptor/Service* Multithreading-Entwurfsmuster. Implementiert die `run()`-Methode des inneren Connection-Handlers nach den dort unter TODO hinterlegten Anweisungen.

Verwendet zum Starten des TCP-Monitors die Klasse **`TcpMonitorApplication`** welche die Bedienung mittels GUI vereinfacht – dazu verwendet sie intern die Klasse TcpMonitorServer. Startet dazu die App-Klasse ohne Laufzeit-Parameter, gebt die folgende Information in die zugehörigen Eingabefelder ein, und drückt dann den Start-Knopf:

- Service-Port: 8010
- Redirect-Host: <hostname>
- Redirect-Port: 80 (HTTP), 443 (HTTPS), 21 (FTP)

Verwendet zum Testen im Web- bzw. FTP-Client folgende Adress-Daten, je nachdem zu welcher Art Server euer TCP-Monitor umleiten soll:

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

Falls alles funktioniert sollte Euer TCP-Monitor in der Lage sein im Prinzip jedwede TCP-Nachrichten abzuhören die vom Client über den lokalen Monitor-Port an den Zielserver weitergereicht werden, und umgekehrt. Allerdings gilt dies nur eingeschränkt für HTTP:

- Ein HTTP-Sicherheitsfeature (der seit HTTP 1.1 notwendige Host-Eintrag im HTTP Request Header) unterbindet zudem die Funktion des TCP-Monitors auf vielen populären Web-Servern. Ersetzt daher bei den zum Web-Server übertragenen Daten den Wert des Host-Headers durch den richtigen Hostnamen.
- Es kann einige Zeit dauern bis ein HTTP-Server eine TCP-Verbindung schließt, gerade wenn HTTPS verwendet wird. Verwendet daher die Future.get(10, TimeUnit.SECONDS) Methode zur Resynchronisation mit Timeout-Unterstützung.
- Zudem übertragen Web-Server ihre Inhalte heute fast ausschließlich GZIP-komprimiert, wodurch man in der Log-Ausgabe bei HTTP leider immer weniger lesbare Dokumente sehen bekommt, außer man dekodiert die Daten entsprechend - machen wir aus Aufwandsgründen nicht.

Beachtet dass ihr den Browser-Cache leeren müsst wenn ihr einen neuen HTTP-Server über einen bereits vorher verwendeten lokalen Port beobachten wollt; andernfalls findet u.U. 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.

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.