Verteilte Systeme

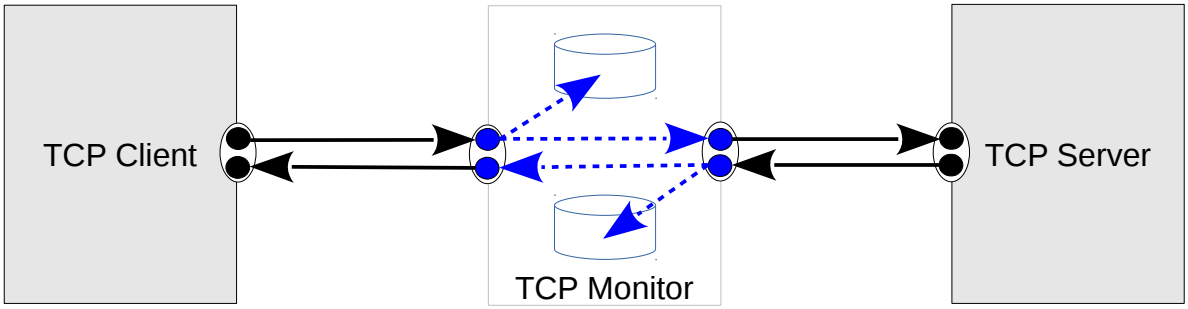
Übung C2

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.



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.

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

- Service-Port: 8010
- Forward-Host: www.sueddeutsche.de (HTTP) bzw. uranus.f4.htw-berlin.de:21 (FTP)
- Forward-Port: 80 (HTTP) bzw. 21 (FTP)

Ihr könnt alternativ auch die Klasse **`TcpMonitor`** selbst starten, dann als mit folgenden Laufzeit-Argumenten zu fütternder Kommandozeilen-Client:

- 8010 (Service-Port)
- c:/temp (oder anderes beschreibbares Verzeichnis für die Log-Datei Ausgabe)
- www.sueddeutsche.de:80 (HTTP) bzw. uranus.f4.htw-berlin.de:21 (FTP)

Verwendet zum Testen im Web- bzw. FTP-Client folgende Adress-Daten, je nachdem mit welcher Art TCP-Server euer Monitor verbunden werden 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:

- Es kann einige Minuten dauern bis ein HTTP-Server eine TCP-Verbindung schließt, und die Daten damit vom TCP-Monitor sichtbar gemacht werden – beim Testen ist dann Geduld gefragt. Wenn einige Minuten später immer noch nichts erscheint, hilft oft eine nachfolgende HTTP-Anfrage gegen die gleiche Site.
- 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; www.sueddeutsche.de sollte jedoch z.B. funktionieren. Um diese Einschränkung zu umgehen müssten HTTP-Anfragen vom TCP-Monitor geparst, und der Host-Eintrag vor der Weitergabe abgeändert werden, was (ein wenig) Aufwand verursacht und nicht Übungsziel ist.
- Zudem übertragen immer mehr Web-Server ihre Inhalte GZIP-Komprimiert, wodurch man in der Log-Ausgabe sowieso immer weniger lesbares zu sehen bekommt.

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.