VSYS

Client-Server

Generelle Information

Der Server und der Client wurden von uns zusammengelegt, da sich diese von ihren Funktionen Kernfunktionen, ausgenommen das Bannen einer IP und der Authentifizierung am LDAP Server. Beide sind also das ein und das selbe Programm.

Daher werden beide im unterschiedlichen Modus gestartet, bedienen sicher aber großteils der gleichen Header-Dateien und gleicher Methoden, soweit für den Client bzw. Server vorgesehen ist.

Die Programme werden folgendermaßen gestartet:

Server: ./klcp_tool server [IP] [PORT] [DATEIPFAD]
Client: ./klcp_tool client [IP] [PORT] [DATEIPFAD]

Dateipfad steht dabei für den Pfad, in dem Files gedownloadet bzw von dem files geuploaded werden. Dieser kann sowohl absolut, als auch relativ zum Programm sein.

Das Programm wurde in C++ geschrieben.
Für detailliertere Beschreibung einzelner Funktionen sind die Kommentare in den dementsprechenden Dateien anzusehen!

Header-Dateien

Es existieren 8 Header-Dateien:

- connection.hpp
- klcp.hpp
- common.hpp
- server.hpp
- client.hpp
- ldap_connect.hpp
- BanIP.hpp

connection.hpp:

Hier wird der Socket für Client und Server erstellt und eingestellt, sowie die Funktionen für das Versenden einer Datei und einer Nachricht sind hier definiert und deklariert.

klcp.hpp:

KLCP steht für „Klein Lepojic Communication Protocol“. Das Protokoll ist von HTTP inspiriert, und deklariert je nach Typ (FILE, COMMAND oder MESSAGE) entsprechende Header, und sendet, falls nötig (MESSAGE und FILE), danach Daten im Binärformat.

common.hpp:

Eine Header Datei mit oft verwendeten Funktionen, wie umrechnen von Dateigrößen von Files zu „Human Readable“ Formaten, auflisten von Files in einem Verzeichnis, readn und writen, sowie Output von Info bzw Error messages.

server.hpp:

Die Implementation der Server spezifischen Funktionen. Besteht aus einer Server Funktion sowie einer Thread Funktion, welche die eigentliche Logik enthält.

client.hpp:

Es wird ein Socket für den Client erstellt und eine Verbindung zu der in der Kommandozeile angegebenen IP-Adresse und Port erstellt. Dementsprechendes Error-Handling wird gesetzt,falls keine Verbindung erstellt werden kann. Es bestehen drei Arten eines Requests an den Server:

- Message
- File
- Command

Wird die Eingabe mit dem Kommando „LOGIN“ begonnen, so wird eine Maskierung der Passworteingabe vorgenommen. Die Eingabe wird nach Absenden des Passwortes wieder auf den Ursprung zurückgesetzt. Weitere Behandlung per if-else der für den Client vorgesehenen Kommandos. Schlussendlich wird nach getaner Arbeit der Socket geschlossen.

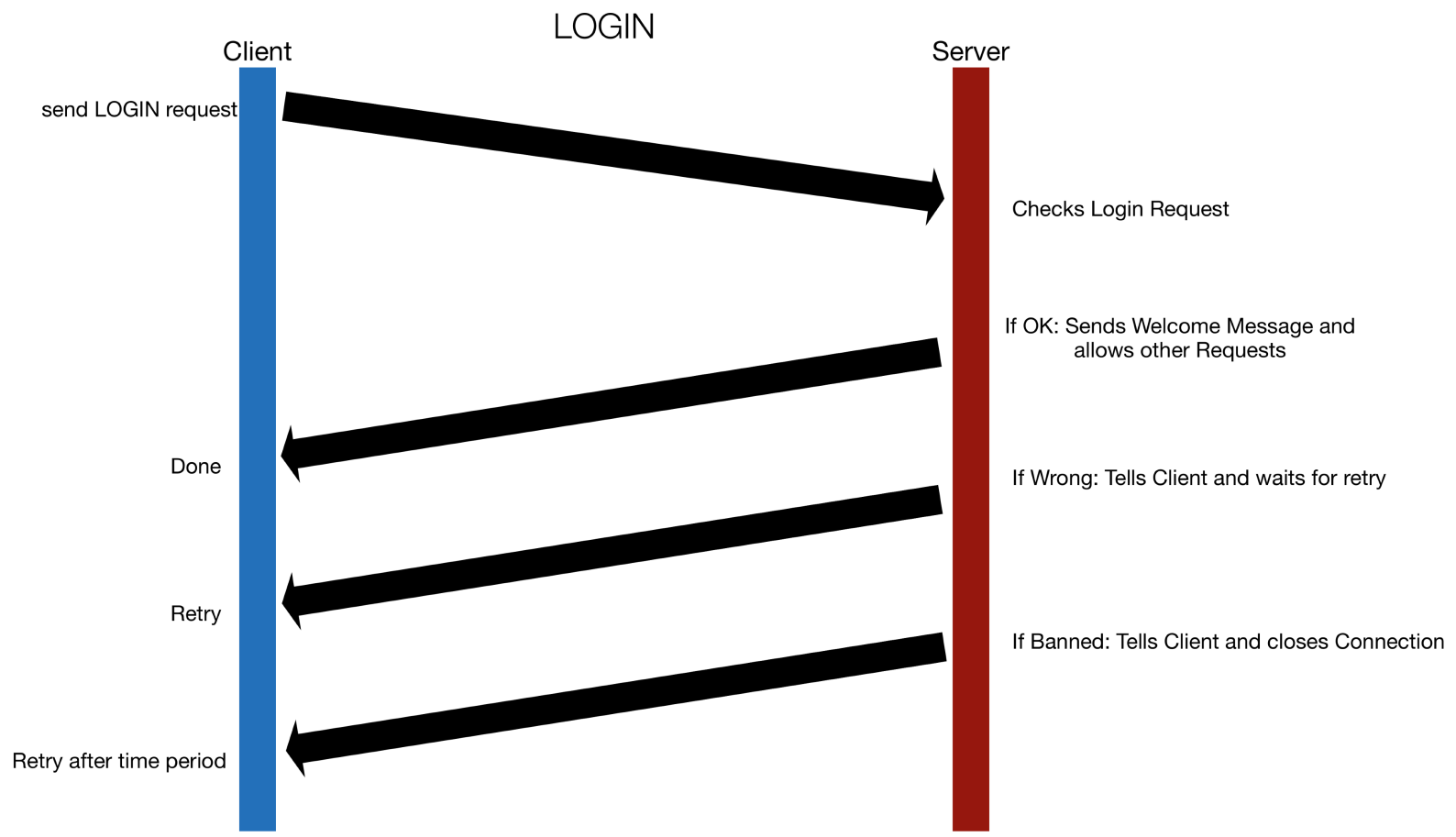
ldap_connection.hpp:

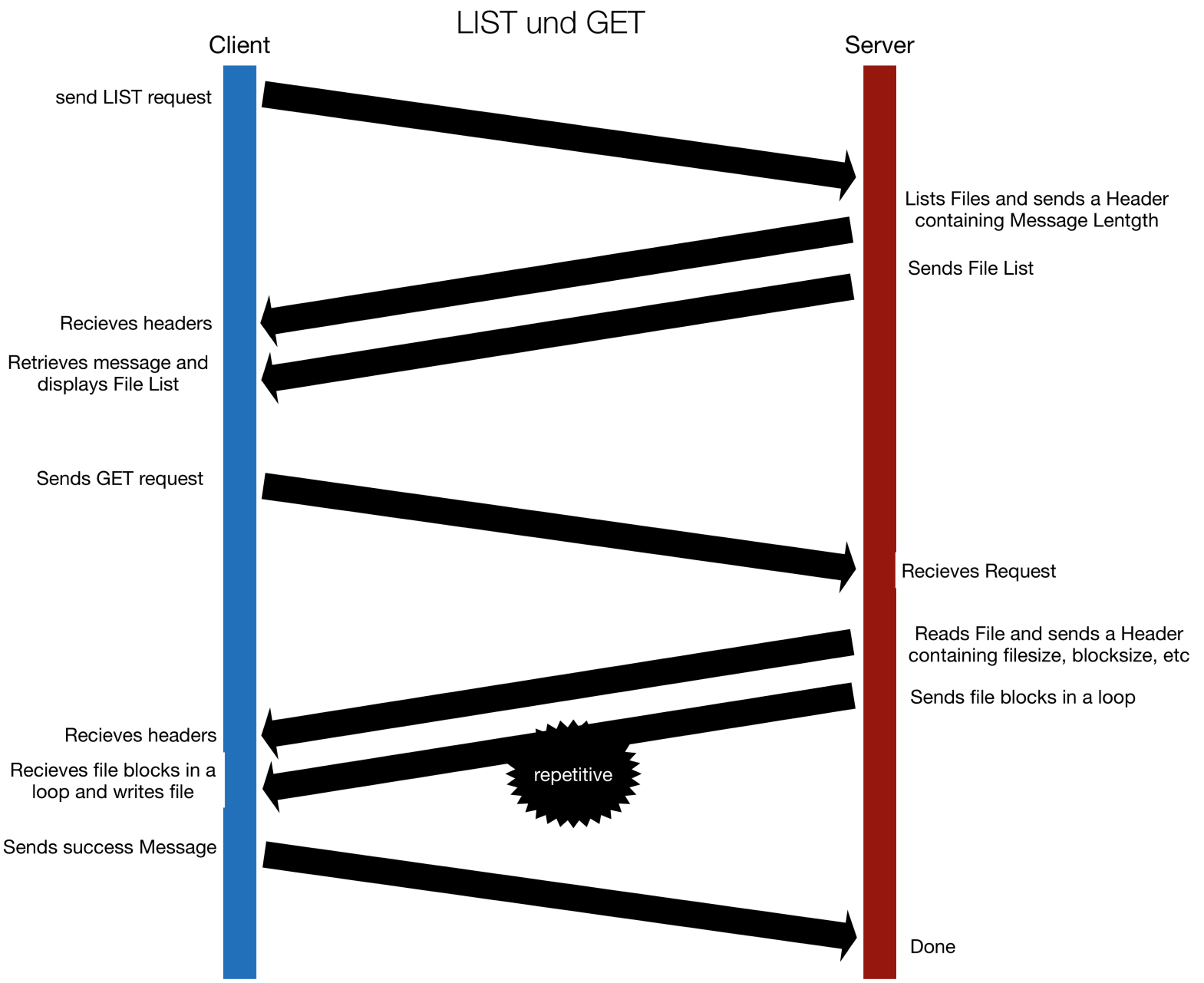
Die Implementierung der LDAP Funktionalität wird möglichst simpel gehalten. Neben den benötigten Deklarationen und Funktionen, die LDAP spezifisch sind, wird hier eine Variable eingeführt, welche die Login-Versuche mitzählt, um diese später für dien eventuellen Ban-Prozess weiterzuleiten.

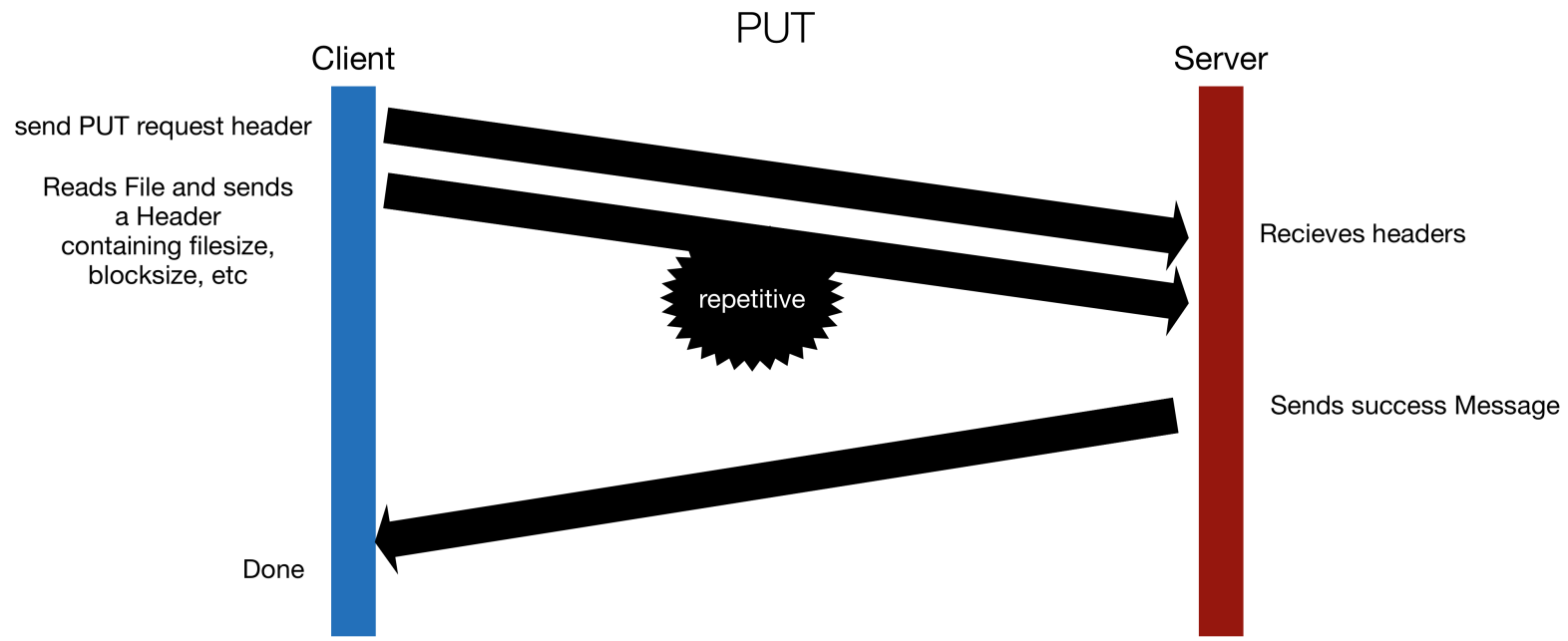
BanIP.hpp:

Es werden Variablen zur Zeitermittlung gesetzt. Berechnet wird die Zeit des Bans indem die Differenz zwischen der momentanen Timestamp und der Timestamp des Zeitpunkts des Bans errechnet wird. Ergibt dieser einen wahren Bool-Wert, so wird die IP gebannt. Die Ban-Dauer beträgt 60 Minuten. Ergibt dieser einen falschen Wert, so ist der Client erfolgreichen eingeloggt und die IP-Adresse wird, falls davor schon gebannt, wieder freigegeben.

Client-Server Kommunikation




LIST und GET
Client
Server
send LIST request
Lists Files and sends a Header containing Message Lentgth
Sends File List
Recieves headers
Retrieves message and displays File List
Sends GET request
Recieves Request
Reads File and sends a Header containing filesize, blocksize, etc
Sends file blocks in a loop
Recieves headers
Recieves file blocks in a loop and writes file
repetitive
Sends success Message
Done


PUT
Client
Server
send PUT request header
Reads File and sends
a Header
containing filesize,
blocksize, etc
repetitive
Recieves headers
Sends success Message
Done