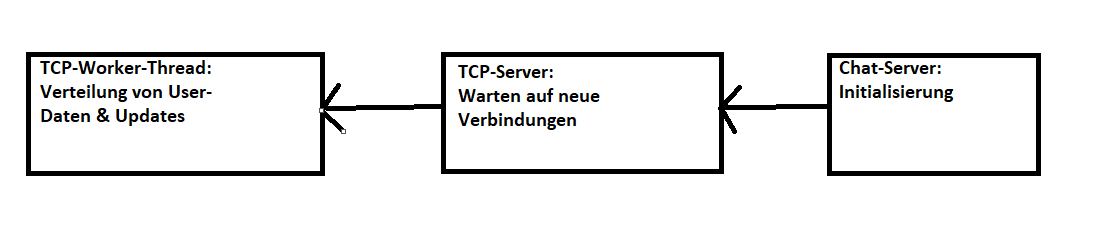
Chat:

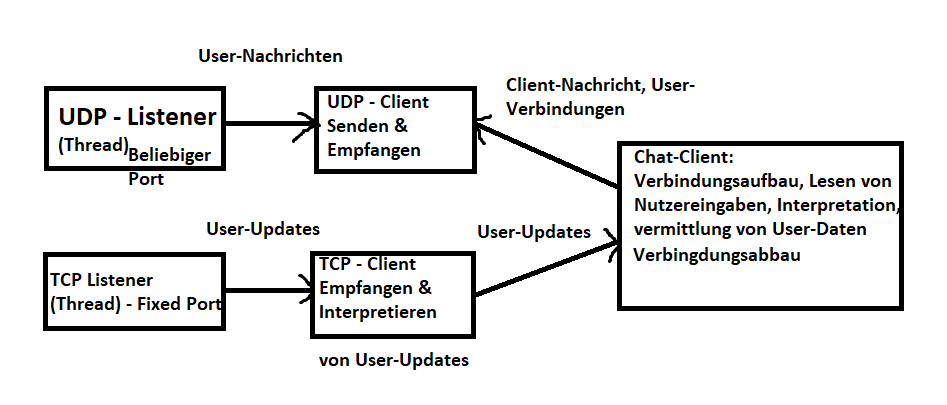
* Einloggen in den Chat-Server:
  + TCP, da hier keine Informationen verloren gehen dürfen und die Verbindung gehalten wird wodurch wir sagen können wer aktiv ist.
* Chat zwischen mehreren Benutzern:
  + UDP, da es schneller ist und hier datenverlust nicht so schlimm ist.

Architektur:

Server:



Client:



TCP Server Portnummer: 56789

UDP Client Portnummern: Beliebig

TCP Client Portnummern: Beliebig

UDP Buffer Size: 1024

~~Message split Operator: “\r\n”~~

~~Inline Message Seperator: „::“~~

Get all active Users Command: “getUsers!”

Styleguide: https://tools.ietf.org/html/rfc7322

|  |  |
| --- | --- |
| HAW-Praktikum Rechnernetze | S. Kuhnt |
| Request for Comments: XXXXX | individual Contributor |
| Category: Experimental | C. Schulz |
|  | individual Contributor |
|  | November 2019 |

**Kommunikation mit unserem Chat-Server**

Status of this Memo

Dieser Entwurf beschreibt das Kommunikationsprotokoll unseres Chatservers und der Clienten. Distribution of this memo is unlimited.

Copyright Notice

Dieses Dokument unterliegt keiner Urheberrechtsbeschränkung.

Abstract

This document describes the necessary properties of a client to connect to the chat server and send and receive messages from it and between clients.

Table of contents

1. introduction
2. Glossar
3. Text-Codierung
4. Kommunikation
   1. Server – Client Kommunikation
      1. Anmeldung
      2. Neue Verbindungen
      3. Verbindungstrennung
   2. Client – Client Kommunikation
5. Server-Befehle
6. Fehlermeldungen
7. Security Considerations

1. Introduction

Dieses Dokument geht aus dem Praktikum 2 der Veranstaltung Rechnernetze hervor. Es soll beschrieben werden, wie unser Entwurf einer Client-Server chat-applikation miteinander kommuniziert. Dieses Dokument stellt insofern ein Protokoll zur Kommunikation dar.

2. Glossar

CRLF – carriage return line feed – End-Zeichen einer TCP Nachricht – „\r\n“

HS - HEADSPLIT – „###“ – Trennung von Protokollnamen und Befehl im Header

HBS - HEADBODYSPLIT – „~~~“ – Trennung von Header und Body in TCP Nachrichten

PS - PAYLOADSPLIT – „;;;“ – Trennung von Payload-teilen

IPS - INTERPAYLOADSPLIT – „:::“ – Trennung von mehreren Bestandteilen in einem Payload-teil

Protokollname – „myprotocol“

USER – Serialisiertes Nutzer Objekt eines angemeldeten Chat-Users nach dem Muster:  
 [id]IPS[name]IPS[inetadress(ip)]IPS[UDP-Listen-Port]

Beispiel: 12:::Mustermann:::192.168.0.88:::57999

Header – Kopfzeile von TCP Nachrichten enthält Protokollname und den jeweiligen Server-Befehl, getrennt durch HEADSPLIT

Payload – Inhalt der Übertragenen Nachricht ohne Identifizierungsmerkmale und Steueranweisungen

Server – hier der Verbindungsknotenpunkt auf TCP Basis zwischen den angemeldeten Chatnutzern

Client – hier die lokale Implementierung des Kommunikationsprotokolls des Chatnutzers

3. Text-Codierung

Sämtliche Kommunikation erfolgt über UTF-8.

4. Kommunikation

Es gibt 2 Kommunikationswege. Die Kommunikation mit dem Server einerseits und die Kommunikation unter den Clienten andererseits. Der Server leitet informationen über angemeldete Nutzer weiter. Die Textnachrichten der Clienten untereinander werden direkt zwischen den Clienten ausgetauscht.

a. Server-Kommunikation

Während der gesamten Nutzungsdauer besteht eine TCP Verbindung zum Server unter Server-Port 56789. Jede TCP – Nachricht hat eine Headerzeile bestehend aus Protokollnamen und Befehlsname. Die folgenden Zeilen beinhalten die Payload. Das Ende einer TCP-Nachricht wird durch eine Leerzeile angezeigt (CRLF).

Formatierung vom Server gesendeter Befehle:  
[Protokollname]HS[Befehl]HBS

[Payload1]PS

[Payloadn]PS

CRLF

Vom Server gesendete Befehle: (auch header für anmeldung, quit)

|  |  |
| --- | --- |
| Name | Payload |
| FULLTABLE | Liste von angemeldeten Nutzerobjekten, jede Zeile der Payload enthält einen Nutzer als USER formatiert. Das die erste Zeile ist der angemeldete Client selbst, diese Zeile kann sich in den folgenden Zeilen wiederholen. |
| DISCONNECTED | Nutzer Objekt als USER formatiert. Der übergebene Nutzer hat sich vom Server getrennt und soll keine UDP-Textnachrichten mehr empfangen. Da Namen doppelt vergeben sein können sollten Nutzer aus der eigenen Cache-Liste mittels ID gelöscht werden. |
| CONNECTED | Nutzer Objekt als USER formatiert. Der übergebene Nutzer hat sich im Chat angemeldet. |
| ERROR | Wird an den Clienten bei fehlerhafter Anfrage gesendet |

Vom Server erkannte Befehle:

|  |  |
| --- | --- |
| AUTH | Anmeldung am Server |
| QUIT | Schließen der Verbindung, hierdurch werden die Clientdaten auch bei allen anderen Clienten ausgetragen |

Der Client sollte eine Liste mit Nutzern halten um empfangene Nachrichten einem Namen zuzuordnen und eine Nachricht an alle angemeldeten Nutzer zu senden. Diese Liste sollte durch Updates der o.g. Serverbefehle aktuell gehalten werden.

i. Anmeldung

Clienten melden sich beim Server mit einem selbst gewählten Nutzernamen und einem vom Clienten gewählten Port an welchem die UDP-Nachrichten durch den Clienten empfangen werden. Die Anmeldungsnachricht des Clienten enthält zunächst den HEADER mit dem Befehl AUTH.

Beispiel-Anmeldung: „myprotocol###AUTH~~~PeterPan:::12356[CRLF]“

Die Antwort vom Server enthält im HEADER den Befehl FULLTABLE. Die Payload enthält einen oder mehrere Chat-User als USER-Objekt formatiert. Das erste USER-Objekt ist der anmeldende Client selbst. Dieses kann sich in der übrigen Payload wiederholen.

ii. Neue Verbindungen

Bei Neuanmeldungen am Server wird der Client durch eine TCP-Nachricht mit dem Befehl CONNECTED im HEADER informiert. Die Nachricht enthält den neuen Nutzer als USER-Objekt in der Payload.

iii. Verbindungstrennung

Bei Verbindungstrennungen eines anderen Clienten wird der Client durch eine TCP-Nachricht mit dem Befehl DISCONNECTED im HEADER informiert. Die Nachricht enthält den getrennten Nutzer als USER-Objekt in der Payload.

Die eigene Trennung vom Chat-Service erfolgt durch eine TCP-Nachricht mit QUIT im Header, ohne Payload.

b. Client-Client Kommunikation

Der Client sendet eine Textnachricht an alle Chatuser (inklusive ihm selbst). Nachrichten werden als UDP – Paket gesendet. Die Adressen (IP & Port) werden bei Anmeldung durch den Server mitgeteilt und über die Updates aktuell gehalten.

Nachrichten enthalten in der ersten Zeile die ID des sendenden Clienten und in der zweiten Zeile beginnt die Nachricht. Die Zeilen werden durch CRLF getrennt.

Die gesendeten UDP-Pakete sind maximal 1024 Byte groß. Nach Abzug der ID(19\*4) und CRLF(2\*4) bleiben 940 Byte für die Payload übrig (235 Zeichen).

6. Fehlermeldungen

Die einzige Kommunikation von Client zu Server beläuft sich auf die Anmeldung und das Trennen der Verbindung. Eine Fehlermeldung wird bei inkorrekter Request-Syntax gegeben. Im HEADER wird dann der Befehl ERROR gegeben. Ein Schließen der Verbindung ohne QUIT Befehl wird nur Serverseitig geloggt. Hierdurch werden jedoch wie bei einem ordnungsgemäßen QUIT Befehl die Client-Daten bei allen Clienten ausgetragen.

7. Security Considerations

UserId kann erraten werden wodurch sich Clients als andere User ausgeben können.

Die Nachrichten werden unverschlüsselt übermittelt. Aus den Empfangenen Userdaten kann die USERID eines anderen Nutzers gelesen werden um eigene Nachrichten im Namen eines anderen Nutzers zu senden.