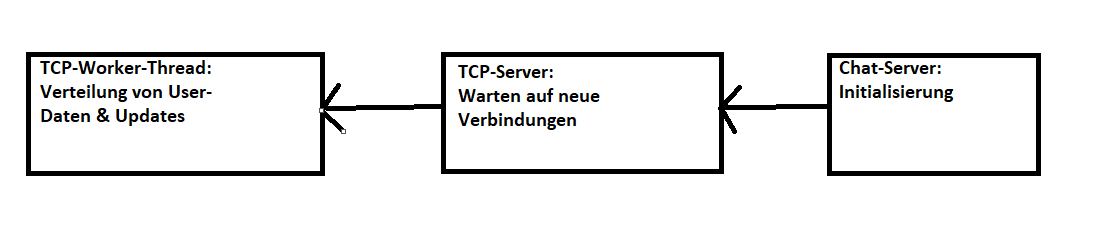
Chat:

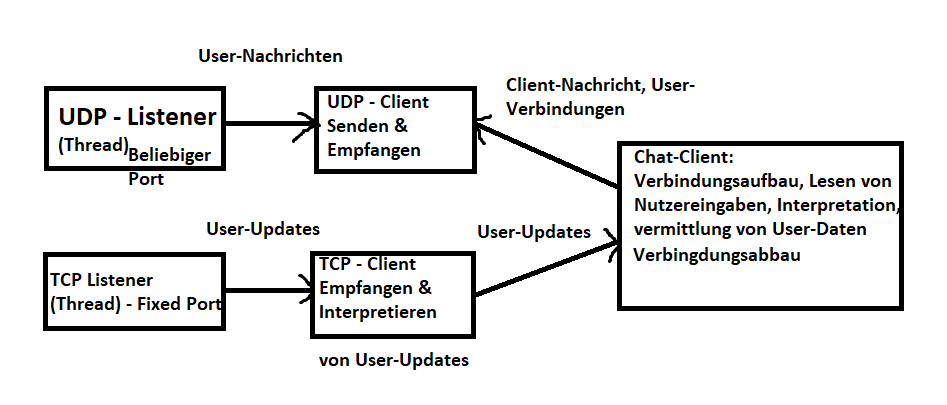
* Einloggen in den Chat-Server:
  + TCP, da hier keine Informationen verloren gehen dürfen und die Verbindung gehalten wird wodurch wir sagen können wer aktiv ist.
* Chat zwischen mehreren Benutzern:
  + UDP, da es schneller ist und hier datenverlust nicht so schlimm ist.

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 – Zeilentrennzeichen – „\r\n“

DC – double colon - Interzeilentrennzeichen – „::“

Protokollname – „myprotocol“

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

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

HEADER – Kopfzeile von durch den Server gesendeter Nachrichten enthält Protokollname und den jeweiligen Server-Befehl, getrennt durch DC

Payload – Inhalt der Übertragenen Nachricht ohne Identifizierungsmerkmale und Steueranweisungen

3. Text-Codierung

Sämtliche Kommunikation erfolgt über US-ASCII. Die Trennung Zeilen erfolgt über CRLF am Ende. Das Ende einer TCP-Nachricht wird durch eine Leerzeile angezeigt (CRLF). Jede TCP – Nachricht vom Server hat eine Headerzeile bestehend aus Protokollnamen und Befehlsname, getrennt durch DC. Die folgenden Zeilen beinhalten die Payload.

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. Nachrichten vom Server werden mit dem Protokollnamen und dem Befehl getrennt durch das DC versandt.

Formatierung vom Server gesendeter Befehle:  
[Protokollname]::[Befehl]CRLF

[Payload1]CRLF

[Payloadn]CRLF

CRLF

Vom Server gesendete Befehle:

|  |  |
| --- | --- |
| 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. |

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 keine Kopfzeile.

Beispiel-Anmeldung: „Meinname::12356“

Als Antwort wird vom Server wird ein HEADER gesandt mit FULLTABLE als Befehl. Die Payload enthält die angemeldeten Chatuser als USER formatiert. Die erste Zeile der Payload enthält das USER-Objekt vom Clienten selbst. Diese Zeile 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 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 informiert. Die Nachricht enthält den getrennten Nutzer als USER-Objekt in der 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.

6. Fehlermeldungen

Die einzige Kommunikation von Client zu Server beläuft sich auf die Anmeldung und das Trennen der Verbindung. Eine Fehlermeldung wird bei inkorrekter Anmelde-Syntax nicht gegeben. Die Verbindungtrennung erfolgt durch schließen der Socket, hier ist keine Fehlermeldung notwendig.

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.