Dokumentation

Matthias Vonend Jan Grübener Patrick Mischka
Michael Angermeier Troy Keßler Aaron Schweig

28. März 2020

Inhaltsverzeichnis

1 Basisanforderungen

1.1 Chatfunktionalität

Chatnutzer meldet sich auf einem Server an und zu genau einem anderen Nutzer einen Chat führen.

Textnachricht soll angezeigt werden sobald sich angemeldet

Textnachrichten in korrekter Reihenfolge

Chats beider verschiedenen Nutzer zeitnah verarbeiten (Non-Persistent Server)

Zwei identische Server mit kompletten Chat-Verlauf

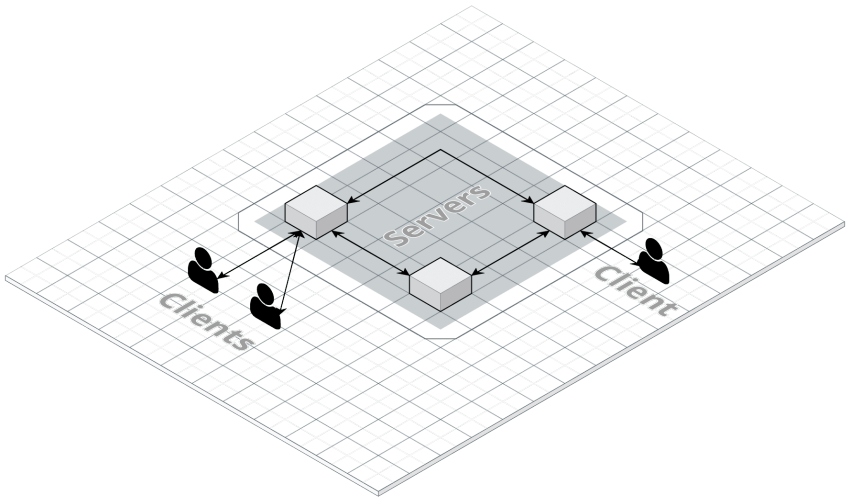

Abbildung 1: Architektur

1.2 Fehlerbehandlung

Aus den Anforderungen geht hervor, dass es mindestens zwei Server geben muss, die sämtliche Informationen des Chatsystems besitzen müssen. Bricht eine Node zusammen muss dementsprechend eine andere Node dessen Aufgabe übernehmen.

1.2.1 Client

//TODO Troy

Im Fehlerfall scheitert das Senden einer Nachricht an den Server. In diesem Fall versucht sich der Client mit einer anderen Node zu verbinden und sendet die Nachricht erneut.

1.2.2 Server

Serverseitig können verschiedene Fehler auftretten. VIele Fehler werden bereits durch das TCP-Protocol verhindert. Dennoch können grundsätzlich die folgende Fehlerfälle eintretten:

1. Nachricht des Clients kann nicht korrekt empfangen/gesendet werden
   In diesem Fall muss der Server davon ausgehen, dass die Verbindung zusammengebrochen ist und er beendet seine Verbindung. Der Server verlässt sich darauf, dass der Client erneut eine Verbindung aufbaut. Alle für den Client relevanten Nachrichten werden dann zu diesem übertragen und der Client muss neue Informationen zurück übertragen
2. Nachrichten einer Nachbarnode können nicht korrekt empfangen/gesendet werden
   Ähnlich zur Clientverbindung muss der Server davon ausgehen dass die Verbindung zusammengebrochen ist. Allerdings ist der Server hier selbst

dafür zuständig sich erneut zu verbinden. Sämtliche Nachrichten, die an eine Node gesendet werden müssen werden in einer Queue aufbewahrt und nacheinander gesendet. Scheitert die Verbindung, so bleibt die Queue unverändert und wird nach einem erneuten Verbinden weiter abgearbeitet. Es wird solange versucht zu verbinden, bis eine Verbindung zustande gekommen ist. Sobald eine Verbindung wieder aufgebaut wurde synchronisieren sich die Nodes um wieder einen vollständigen Informationsstand zu besitzen. Sind keine Nachrichten zu senden hat die Node keine Möglichkeit festzustellen ob eine Verbindung noch existiert. Zu diesem Zweck existiert ein Heartbeat, der periodisch die Nachbarnodes anpingt und so prüft ob die Verbindung noch existiert.

Wie gerade beschrieben führen alle beteiligten stehts eine synchronisation durch wodurch diese immer den kompletten Informationsbestand besitzen.

2 Erweiterungen

2.1 Grafische Oberfläche bei den Nutzern

2.2 Verwendung von Emojis

2.3 Mehrere Chatverläufe pro Nutzer

2.4 Persistentes Speichern der Chatverläufe

Sämtliche der Node bekannten Informationen (Nutzer, Chats, Nachrichten) werden in einem Warehouse verwaltet. Um diese Informationen zwischen Node-Neustarts zu persistieren muss demnach das Warehouse als Datei gespeichert werden und bei Node-Start wieder geladen werden. Existiert noch kein Speicherstand wird die Node mit einem leeren Warehouse initialisiert. Während die Node ausgefallen ist, können andere Nodes gleichzeitig neue Informationen erhalten haben. Sobald die Node wieder verfügbar ist müssen andere Nodes dies bemerken und die ausgefallene Node mit Informationen versorgen. Um den Ausfall festzustellen ist ein Heart-Beat vorgesehen. Dieser pingt jede Sekunde alle benachbarten Nodes an um sicherzustellen, dass die Verbindung noch existiert. Sollte eine Unterbrechung festgestellt, versucht die Node die Verbindung wieder aufzubauen. Gelingt dies, so wird das Warehouse übersendet. Jede Node prüft ob sie alle Informationen des empfangenen Warehouses bereits besitzt und fügt neue Informationen hinzu. Sofern die Node neue Informationen erhalten haben, broadcastet diese ihren neuen Stand an alle benachbarten Nodes um diese auch auf den neuesten Stand zu bringen. Der Speichervorgang kann je nach System und Warehousegröße längere Zeiten in anspruch nehmen. In dieser Zeit können in der Regel keine weiteren Anfragen verarbeitet werden. Um diese Zeit zu minimieren kümmert sich ein eigener Thread um die Speicherung und speichert das Warehouse in Intervallen. Zu beachten ist der Fall, dass eine Node zusammenbricht während der Speichervorgang in Arbeit ist. In diesem Fall würde die node sämtliche Informationen verlieren, da die Speicherdatei korrupt ist. Um dies zu verhindern wird der Speicherstand zunächst in eine Tempdatei geschrieben und nach erfolgreicher speicherung an den Zielort verschoben.

2.5 Verschlüsselte serverseitige Speicherung der Chats

2.6 Gruppenchats

2.7 Verschlüsselte Übertragung der Chat-Nachrichten

3 Anhang

```
  @Override
  public void run() {
40  Thread.currentThread().setName("Connection Handler");
    while (!this.context.isCloseRequested().get()) {
      System.out.println("Handling");
      try {
        var object = inputStream.readObject();
45      var packet = (Packet) object;
        this.handlePacket(packet);
      } catch (IOException | ClassNotFoundException e) {
        break;
      }
50  }
    this.close();
  }

  private void handlePacket(final Packet packet) throws IOException
     {
55  for (var listener : this.context.getListeners()) {
      try {
        var methods = listener.getClass().getDeclaredMethods();
        for (var method : methods) {
          if (method.getName().equals("next") && !method.
    isSynthetic()) {
60          var packetType = method.getParameters()[0];
            if (packetType.getType().isAssignableFrom(packet.
    getClass())) {
              var retu = (Packet) method.invoke(listener, packet,
    this.context, this);
              this.pushTo(retu);
            }
65        }
        }

      } catch (SecurityException | IllegalArgumentException |
    IllegalAccessException
          | InvocationTargetException e) {
70      e.printStackTrace();
      }
    }
    this.context.getWarehouse().print();
  }
75
  public void pushTo(final Packet packet) {
```

../src/main/java/vs/chat/server/ConnectionHandler.java

Test