Praktikum Verteilte Systeme

Sommersemster 2019

Prof. Dr. Michael von Rüden
Prof. Dr. Lars-Olof Burchard
Prof. Dr. Ronald Moore

Übersicht

Im Rahmen des Praktikums *Verteilte Systeme* soll eine Anwendung aus dem Bereich der Wetterdienste entwickelt werden. Dazu sollen die Technologien *Sockets*, *RPC* (Apache Thrift*)* sowie *Message-Oriented-Middleware* (MQTT) verwendet werden.

Rahmenbedienungen

- Das Bearbeiten der Praktikumsaufgaben findet in Zweier-Teams statt.
- Die Praktikumsaufgaben müssen zuhause vor- und nachbereitet werden, da die Bearbeitungszeit während des Praktikumstermins nicht ausreicht um eine vollständige Lösung zu implementieren.
- **Jede Aufgabe muss** spätestens im auf darauffolgenden Praktikumstermin **testiert sein**. Alle Aufgaben müssen spätestens zum letzten Gruppentermin testiert sein. Andernfalls gilt die Prüfungsvorleistung als „nicht bestanden“ und eine Zulassung zur Klausur im folgenden Prüfungszeitraum ist nicht möglich.
- Die Lösungen müssen im **GitLab** der H-DA (https://code.fbi.h-da.de) zur Verfügung gestellt werden.
- Es ist ein **Build Tool** (Make, Maven, etc.) zu verwenden.
- **Jede Lösung muss getestet werden**. Schreiben Sie zu jeder ihrer Lösungen **mindestens einen funktionalen Test** sowie einen **Performance-Test**.
- Die Aufgaben werden im Rahmen einer Abnahme mündlich mit dem Dozenten durchgesprochen. **Hierbei müssen alle Teammitglieder die Lösung erklären können**.
- **Nach Abschluss jeder Aufgabe muss ein Protokoll angefertigt werden**, welches die wesentlichen Ergebnisse beinhaltet. Weitere Informationen befinden sich in den jeweiligen Aufgabenstellungen.
- Die Aufgaben sind mittels **Java oder C/C++** zu lösen. Eine Kombination beider Sprachen ist erlaubt. Andere Programmiersprachen sind nur nach Absprache mit dem Dozenten zugelassen. Dasselbe gilt auch für andere Middlewarekomponenten als die oben erwähnten Sockets, REST, Apache Thrift, ProtoBuf und MQTT. Sockets und das HTTP Protokoll müssen nativ implementiert werden. Darüber hinaus ist das Einbinden weiterer Bibliotheken, z.B. von Loggern oder zur Konfiguration, erlaubt.

Aufgabenstellung

Im Rahmen des Praktikums sollen unterschiedliche Systeme rund um das Thema Wetterdienst simuliert werden. Dazu ist in mehreren Phasen jeweils ein Teil des Gesamtsystems zu erstellen, wie in Abbildung 1 dargestellt. Am Ende müssen mehrere Wetterstationen (mind. 3) mit verschiedenen Wetterdiensten kommunizieren. Die Server der Wetterdienste wiederum sollen aus Gründen der Performance und der Ausfallsicherheit redundant ausgelegt werden.

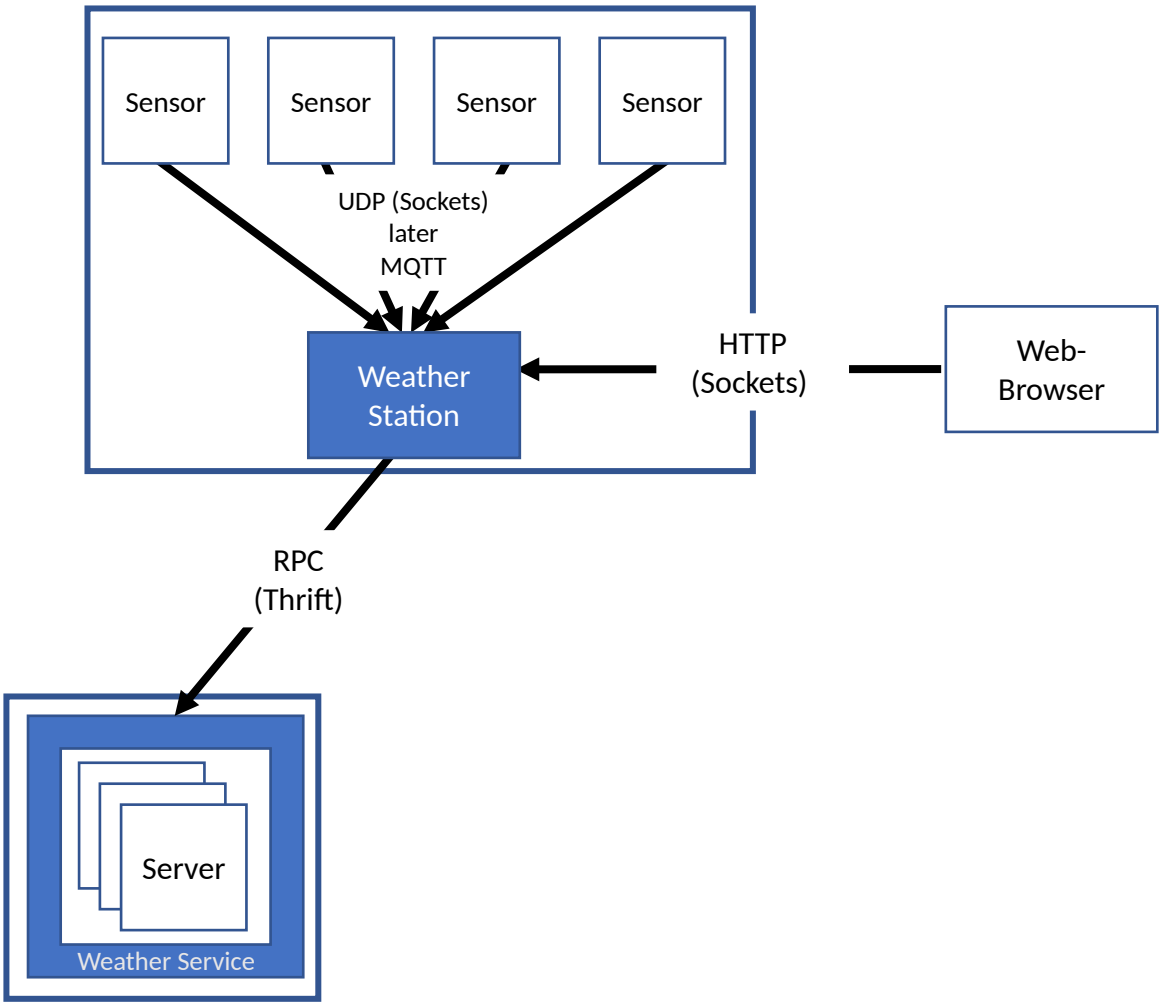

Abbildung 1: Gesamtsystem mit (nur) einer Wetterstation, inklusive verschiedener Sensoren, die mit einem Server des Wetterdienstes kommuniziert. Die Server des Wetterdienstes sind redundant ausgelegt.

Beachten Sie: Es kommt in diesem Praktikum **nicht** darauf an ein möglichst realistisches Sensorverhalten zu simulieren. Der Fokus soll auf der Kommunikation der verteilten Komponenten untereinander sowie dem Software-Design, dem Testen und dem Ausrollen (Deployen) der verteilten Anwendungen liegen.

Aufgabe 0 (Projektplan)

Zunächst soll eine Analyse der Anforderungen erstellt werden, die von der Gruppe in den Aufgaben 1 - 4 implementiert werden. Schauen Sie sich hierfür die Aufgabenstellungen 1 - 4 an und erstellen Sie die Anforderungen für jeden Termin **schriftlich**. Beachten Sie, dass das Bestehen der weiteren Termine davon abhängt, ob die erstellten Anforderungen umgesetzt wurden. Die Anforderungen sollen neben der zu verwendenden Programmiersprache, den Kommunikationstechnologien auch ein Systemdesign inkl. der Übertragungsformate sowie ein Konzept der Test- und Simulationsumgebung für jeden Termin enthalten.

Änderungen am Konzept sind im Laufe des Praktikums erlaubt, müssen jedoch dokumentiert werden. Am Ende des Termins müssen dem Dozenten die Anforderungen vorgestellt und ggf. ergänzt werden. Schließlich wird ein Anforderungsdokument als Protokoll abgegeben. Dieses dient als Checkliste für die folgenden Termine.

In a Nutshell

- Erstellen Sie eine Anforderungsanalyse (funktionale und nicht-funktionale Anforderungen) für das Gesamtsystem und fixieren Sie die Anforderungen schriftlich.
- Leiten Sie aus Ihren Anforderungen ein erstes grobes Systemdesign ab und überlegen Sie, wie die verschiedenen Komponenten miteinander interagieren.
- Leiten Sie aus den Anforderungen und dem Systemdesign funktionale, nicht-funktionale sowie Performance-Tests ab.
- Leiten Sie aus den Anforderungen und dem Systemdesign einen **Projektplan** ab. Geben Sie Tasks an, die für die weiteren Aufgaben bearbeitet werden müssen.
- Überlegen Sie, wie Sie die unterschiedlichen Systeme effizient operativ ausrollen (deployen) können.

Die (harte) Deadline für diese Aufgabe ist der **3. Praktikumstermin**. Zu diesem Zeitpunkt muss eine konsistente, testierfähige schriftliche Ausarbeitung zu Aufgabe 0 vorliegen. Andernfalls gilt die Prüfungsvorleistung als „nicht bestanden“. Die weiteren Aufgaben werden – unabhängig von ihrer Güte – erst testiert, wenn Aufgabe 0 erfolgreich bestanden ist.

Lernziele

Die erste Aufgabe hat unter anderem folgende Lehr- und Lernziele:

- Selbstständiges Arbeiten
- Durchführen einer Anforderungsanalyse
- Herausarbeiten sinnvoller funktionaler und nicht-funktionaler Tests
- Erstellen eines ersten, groben Software- bzw. Systemdesigns
- Aufstellen eines validen Projektplans
- Anfertigen eines Protokolls

Aufgabe 1 (UDP und TCP Sockets)

Im ersten Schritt sollen Sensoren einer Wetterstation Informationen liefern. Dazu sollen Temperatur, Luftfeuchtigkeit, Windgeschwindigkeit und Regen von jeweils einem Sensor erfasst bzw. simuliert werden. Jeder Sensor soll als eigenständiger Prozess laufen. Die Informationen sollen sich ständig ändern und in einem geeigneten Nachrichtenformat mittels UDP an die Wetterstations übermittelt werden. Dort sollen die Nachrichten unter Angabe von IP, Port und Typ des Sensors auf der Standardausgabe ausgegeben werden.

Darüber hinau muss in der Wetterstation ein einfacher HTTP-Server implementiert werden, der mindestens den HTTP GET Befehl korrekt und vollständig verarbeiten kann. Die HTTP-Schnittstelle soll über eine REST-API den Zugriff auf einzelne Sensordaten, alle Sensordaten sowie die Historie der Sensordaten (jeweils mit einer eigenen URI) ermöglichen. Dazu müssen auch die Daten aus der Vergangenheit in der Wetterstation gespeichert werden.

Der HTTP Server soll ohne Hilfsmittel (d.h. ohne vorhandene Bibliotheken) implementiert werden und mindestens HTTP GET unterstützen. Es ist hierbei erforderlich, dass eingehenden HTTP GET Anfragen komplett und korrekt eingelesen und verarbeitet werden. Das bedeutet u.a., dass es nicht ausreichend ist, die erste Zeile eine HTTP Nachricht zu lesen. Zudem müssen die Sensoren weiter laufen. Das bedeutet, dass die Wetterstation gleichzeitig mit den Sensoren als auch mit HTTP Klienten in Kontakt bleiben soll.

Lernziele

Die zweite Aufgabe hat unter anderem folgende Lehr- und Lernziele:

- Selbstständiges Arbeiten
- Software Engineering und Design
- Kommunikation mittels UDP- und TCP-Sockets
- Implementierung und Verwendung von HTTP und REST
- Effizientes Deployment unterschiedlicher Anwendungen, z.B. mittels Skript

Aufgabe 2 (RPC)

Für die zweite Aufgabe sollen die zuvor implementierten Wetterstationen ihren Status, d.h. die aktuellen Werte der Sensoren, über Thrift an die Wetterdienste übermitteln. Hierzu muss die standardisierte und per Thrift-Datei zur Verfügung gestellte API sowohl am Server (Wetterdienst) als auch am Client (Wetterstation) implementiert werden. Der Wetterdienst soll die so übermittelten Daten aller Wetterstationen persistent speichern.

Lernziele

Die dritte Aufgabe hat unter anderem folgende Lehr- und Lernziele:

- Selbstständiges Arbeiten
- Software Engineering und Design
- Einbinden und Anwenden externer Software-Bibliotheken
- Einbinden und Anwenden von RPCs am Beispiel von Thrift
- Implementieren einer vorgegebenen bestehenden API

Aufgabe 3 (Hochverfügbarkeit und Konsistenz)

Für die dritte Aufgabe sollen die Server des Wetterdienstes aus Gründen der Ausfallsicherheit redundant ausgelegt werden. Der Wetterdienst betreibt daher mindestens drei Server parallel. Mit jedem Server ist mindestens eine Wetterstation verbunden. Die Server tauschen unter Verwendung eines RPCs (Thrift oder Protobuf) untereinander die empfangenen Daten aus. Dabei muss sichergestellt werden, dass alle Daten auf allen Servern in der gleichen Reihenfolge vorliegen.

Um die Ausfallsicherheit des Gesamtsystems zu testen, soll es während des Betriebs immer wieder zu zufälligen (simulierten) Ausfällen einzelner Server kommen.

Lernziele

Die vierte Aufgabe hat unter anderem folgende Lehr- und Lernziele:

- Selbstständiges Arbeiten
- Software Engineering und Design
- Auswahl, Design und Implementierung von Hoch-Verfügbarkeits (HA) und Konsitenz-Modellen

Aufgabe 4 (MoM mittels MQTT)

Ihr Kunde stellt nun fest, dass das Anbinden der Sensoren an die Wetterstation mittels UDP keine gute Design-Entscheidung war. Um das System besser skalieren zu können, sollen die Sensoren nun mit MQTT an die Wetterstationen angeschlossen werden.

Überarbeiten Sie Ihre in Aufgabe 2 implementierten Sensoren und die Wetterstation so, dass die Daten nun mit MQTT übertragen werden.

Lernziele

Die fünfte Aufgabe hat unter anderem folgende Lehr- und Lernziele:

- Selbstständiges Arbeiten
- Software Engineering und Design
- Einbinden und Anwenden externer Software-Bibliotheken
- Einbinden und Anwenden einer Message-oriented Middleware am Beispiel von MQTT

Bonusregelung

Mit Hilfe des Praktikums kann ein **0.3-Noten-Bonus** für die Klausur erworben werden. Der Bonus ist nur einmal gültig – nämlich exakt für die in diesem Semester anstehende Klausur. Der Bonus kann dann erteilt werden, wenn Sie eine herausragende Gesamtlösung (Aufgaben 0-4) präsentieren die deutlich über den Anforderungen und den Erwartungen liegt. Der Bonus wird zudem nur dann wirksam, wenn Sie die Klausur auch ohne Bonus mit mindestens 4.0 bestehen.