Verteilte Systeme
Praktikum

Tim Fehr, Moritz von Keiser

Hochschule Darmstadt
Fachbereich Informatik
Verteilte Systeme SoSe 2020

i Übersicht

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

ii Rahmenbedingungen

- Das Bearbeiten der Praktikumsaufgaben findet in Zweier-Teams statt.
- Die Praktikumsaufgaben müssen zuhause vor- und nachbereitet werden, da ein Präsenzstermin leider nicht realisierbar ist
- 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 einen funktionale Test sowie einen nicht funktionale Test.
- Die Aufgaben werden im Rahmen eins Webinars mit dem Dozenten durchgesprochen. Hierbei müssen alle Teammitglieder anwesend sein und die Lösung erklären können.
- Nach Abschluss jeder Aufgabe muss eine Testbeschreibung angefertigt werden, welches eine Beschreibung des Tests, der Durchführung, der erwarteten Ergebnisse sowie der tatsächlichen Ergebnisse enthält. Weitere Informationen befinden sich in den jeweiligen Aufgabenstellungen.
- Es sind keine Programmiersprachen vorgegeben. Eine Kombination mehrerer Sprachen ist erlaubt und gewünscht. 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.

1 Aufgabenstellung

Im Rahmen des Praktikums sollen unterschiedliche Systeme rund um das Thema „Smart-Home“ simuliert werden. Dazu ist in mehreren Phasen jeweils ein Teil des Gesamtsystems zu erstellen, wie in Abbildung 1 dargestellt. Am Ende müssen mehrere Sensoren (mind. 3) mit einer Zentrale und Servern des Anbieters kommunizieren.

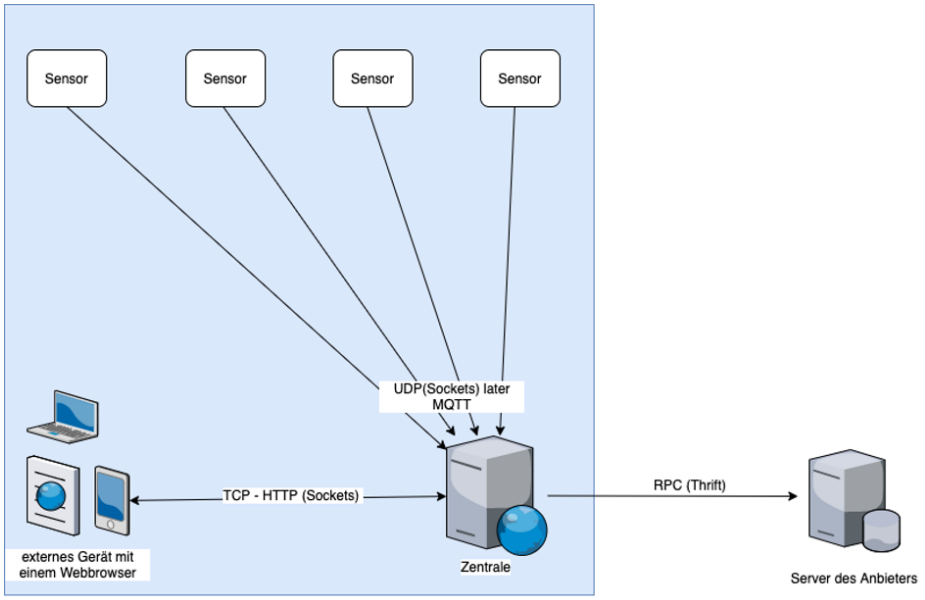

Abb. 1. Gesamtsystem, inklusive verschiedener Sensoren

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.

1.1 Aufgabe 0 - Projektplan/Anforderungsanalyse (1. Termin)

Zunächst soll eine Analyse der Anforderungen erstellt werden, die von der Gruppe in den Aufgaben 1 - 3 (Kapitel 1.1 bis 1.5) implementiert werden. Schauen Sie sich hierfür die Aufgabenstellungen 1 – 3 (Kapitel 1.1 bis 1.5) 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 und nicht-funktionale 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 2. 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:

- 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

1.2 Aufgabe 1a - UDP Sockets (2. Termin)

Im ersten Schritt sollen Sensoren einer Wohnung Informationen liefern. Dazu sollen verschiedene Werte, wie z.B. Temperatur, Helligkeit, usw. 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 Zentrale übermittelt werden. Dort sollen die Nachrichten unter Angabe von IP, Port und Typ des Sensors auf der Standardausgabe ausgegeben werden.

1.3 Aufgabe 1b - TCP Sockets (3. Termin)

Darüber hinaus muss in der Zentrale 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 Zentrale 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:

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

1.4 Aufgabe 2 - Remote Procedure Calls (RPC) (4. Termin)

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

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

1.5 Aufgabe 3 - MoM mittels MQTT (5. Termin)

Ihr Kunde stellt nun fest, dass das Anbinden der Sensoren an die Zentrale mittels *UDP* keine gute Design-Entscheidung war. Um das System besser skalieren zu können, sollen die Sensoren nun mit *MQTT* an die Zentrale angeschlossen werden. Überarbeiten Sie Ihre in Aufgabe 1 (Kapitel 1.2) implementierten Sensoren und die Zentrale so, dass die Daten nun mit *MQTT* übertragen werden.

Lernziele:

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

1.6 Bonusaufgabe - Hochverfügbarkeit und Konsistenz

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 wird zudem nur dann wirksam, wenn Sie die Klausur auch ohne Bonus mit mindestens 4.0 bestehen. Diese Aufgabe ist nicht Teil der PVL und kann bearbeitet werden um den Notenbonus für die Klausur zu erhalten.

Für die Bonus-Aufgabe sollen die Server des Anbieters aus Aufgabe 2 aus Gründen der Ausfallsicherheit redundant ausgelegt werden. Der Anbieter betreibt daher mindestens drei Server parallel. Mit jedem Server ist mindestens eine Zentrale 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:

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