

Verteilte Systeme

Praktikum

Lars-Olof Burchard, Ronald Moore, Michael von Rüden

Hochschule Darmstadt
Fachbereich Informatik
Sommersemester 2022

1 Übersicht

Im Rahmen des Praktikums Verteilte Systeme soll eine Anwendung aus dem Bereich des *Industrial IoT (IIoT)* entwickelt werden. Dazu sollen die Technologien *Sockets*, *Remote Procedure Calls (RPCs)* sowie *Message-Oriented-Middleware (MOM)* verwendet werden.

2 Grundregeln

Folgende Regeln gelten für das gesamten Praktikum und für alle Aufgaben:

- Das Bearbeiten der Praktikumsaufgaben findet soweit wie möglich in Zweier-Teams statt. Suchen Sie sich zu Beginn der Lehrveranstaltung einen verlässlichen Partner. Klären Sie im Vorfeld Ihre gegenseitigen Erwartungen.
- Die Praktikumsaufgaben müssen zuhause vor- und nachbereitet werden, d.h. außerhalb der Praktikumstermine.
- Alle Aufgaben müssen spätestens zum letzten Gruppentermin abgenommen sein. Andernfalls gilt die Prüfungsvorleistung (PVL) als *nicht bestanden* und eine Zulassung zur Klausur im folgenden Prüfungszeitraum ist nicht möglich.
- Jeder Gruppe wird ein Repository im GitLab der H-DA (<https://code.fbi.h-da.de>) zur Verfügung gestellt. Dieses Repository ist für das Praktikum zu verwenden.
- Die Lösungen müssen im Master-Branch Ihres Repositorys zur Verfügung gestellt werden. Eigene Repositories sind nicht zulässig.
- Es ist ein Build Tool (Make, cMake, Maven, Gradle, etc.) zu verwenden.
- Die Lösungen müssen containerisiert und mittels Docker und Docker-Compose zu testen sein.
- Die Aufgaben werden im Rahmen der Praktikumstermine mit dem Dozenten durchgesprochen. Hierbei müssen alle Teammitglieder anwesend sein und die Lösung erklären können.
- Es sind keine Programmiersprachen vorgegeben. Eine Kombination mehrerer Sprachen ist erlaubt.

- Sockets und HTTP müssen nativ, d.h. ohne externe Bibliotheken, implementiert werden. Alle weiteren Komponenten und Bibliotheken, z.B für Thrift, gRPC, MQTT, aber auch für Logging oder die Konfiguration, dürfen frei gewählt werden. Bei Fragen besprechen Sie Ihre Pläne mit Ihrem Dozenten.
- Jede Lösung muss getestet werden. Schreiben Sie zu jeder ihrer Lösungen mindestens einen funktionale Test sowie mindestens einen nicht funktionale Test.
- Die Leistung des Systems soll unter Last bemessen werden. Sie sollen für jede Aufgabe entscheiden, wie das Verhalten Ihres Systems am besten bemessen werden kann, um die Grenzen Ihres Systems am Besten festzustellen.

3 Aufgabenstellung

Im Rahmen des Praktikums sollen unterschiedliche Systeme rund um das Thema *Industrial IoT (IIoT)* simuliert werden. Dazu ist in mehreren Phasen jeweils ein Teil des in Abbildung 1 dargestellten Gesamtsystems zu erstellen. Am Ende müssen mehrere Machinensensoren über einen IoT Gateway mit dem Server eines Dienstanbieters kommunizieren. Der Server des Dienstanbieters wiederum sollen die Daten aus Gründen der Ausfallsicherheit redundant speichern.

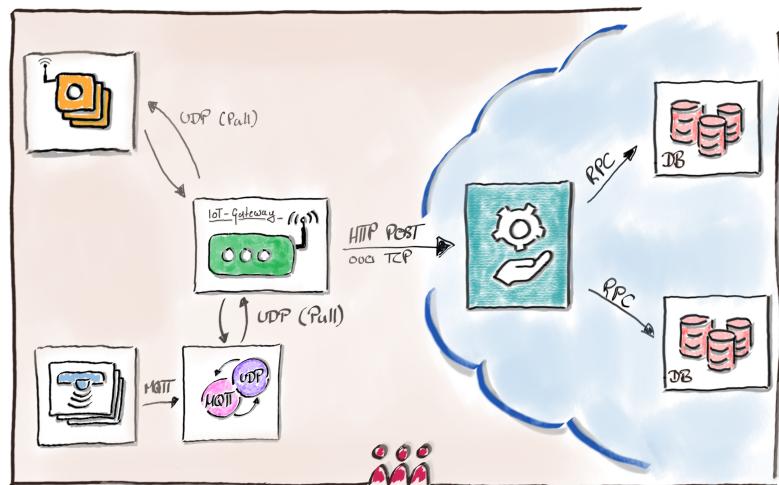


Abb. 1. Das Gesamtsystem mit zwei Arten von Sensoren, einem MQTT-to-UDP Adaper, einem IoT-Gateway und dem Server des Service Providers mit zwei angeschlossenen Datenbanken in der Cloud.

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

3.1 Aufgabe 1 - Vorbereitung

Der erste Praktikumstermin dient der Vorbereitung und Planung des gesamten weiteren Praktikums. Finden Sie sich in Ihrem Team zusammen. Klären Sie unbedingt gegenseitige Erwartungshaltungen. Bereiten Sie Ihre System (Entwicklungsumgebung, Docker, Kommunikationskanäle, etc.) vor. Testen Sie Ihren GitLab-Zugang und den Zugang zum zur Verfügung gestellten Repository.

In der Vergangenheit hat es sich als überaus nützlich erwiesen, wenn die Projektgruppen einen Projektplan und Zeitrahmen erstellt haben. Überlegen Sie gemeinsam, wann wer welche Teile der Softwareentwicklung abgeschlossen haben soll. Nutzen Sie z.B. GitLab Issues und GitLab Milestones für das Projektmanagement.

3.2 Aufgabe 2 - UDP Sockets

Im ersten Schritt sollen die Machinensensoren Informationen liefern. Dazu sollen z.B. Temperatur, Luftfeuchtigkeit, Helligkeit, etc. von jeweils einem Sensor erfasst bzw. simuliert werden. Die Sensorinformationen sollen sich ständig ändern. Weiterhin soll der IoT-Gateway die Sensor-Informationen in regelmäßigen Zeitabständen mittels eines zum implementierenden Request-Reply-Protokolls über UDP abfragen und auf der Standarausgabe ausgeben. Jeder Sensor als auch der IoT-Gateway sollen als eigenständiger Prozess (bzw. als eigenständiger Docker Container) laufen.

Testen Sie die Performance Ihrer Implementierung indem Sie die Round-Trip-Time (RTT) Ihres Request-Reply-Protokolls messen. Sammeln Sie RTT Werten unter verschiedene Konditionen (z.B. unterschiedliche Anzahl Sensoren). Werten Sie den Test statistisch valide aus und erstellen Sie ein (sehr kurzes) Messprotokoll das den Test und die Ergebnisse darstellt.

Hinweis: Sie werden diese Tests in Zukunft wiederholen. Also lohnt es sich, den Test und die Auswertung weitestgehend zu automatisieren.

3.3 Aufgabe 3 - TCP Sockets und HTTP

Im nächsten Schritt soll dem Gesamtsystem zunächst der Server eines Service-Anbieters in der Cloud hinzugefügt werden. Dieser Server kann über eine HTTP-Schnittstelle, d.h. mittels HTTP POST, Daten entgegennehmen und diese in einer internen Datenstruktur abspeichern. Darüber hinaus soll der IoT-Gateway erweitert werden, sodass er die Daten, die er bei den Sensoren abgefragt hat, über HTTP an den Server des Service-Anbieters schickt.

Sowohl der HTTP Server als auch der Client sollen ohne Hilfsmittel (d.h. ohne externe Bibliotheken) implementiert werden. Der Server und Client sollen HTTP POST unterstützen. Der Server soll zusätzlich HTTP GET Anfragen von Ihrem Browser (Firefox oder Chrome oder Ähnliches) akzeptieren, sodass die empfangene Daten angeschaut werden können. Es ist hierbei erforderlich, dass eingehende HTTP Anfragen komplett und korrekt eingelesen und verarbeitet werden. Das bedeutet u.a., dass GET und POST unterschieden werden müssen und, dass es nicht ausreichend ist, nur die erste Zeile einer HTTP Nachricht zu lesen.

Testen Sie die Performance Ihrer Implementierung indem Sie auf der Client-Seite die Round-Trip-Time (RTT) Ihrer HTTP POST Anfrage messen. Werten Sie den Test statistisch valide aus und erstellen Sie ein (sehr kurzes) Messprotokoll das den Test und die Ergebnisse darstellt.

Auch diese Tests werden Sie in Zukunft wiederholen. Also lohnt es sich nach wie vor, die Tests und die Auswertung weitestgehend zu automatisieren.

Wiederholen Sie den Performance-Test aus Aufgabe 2 und vergleichen Sie die Testergebnisse, um fest zu stellen, wie die Performance der UDP Schnittstelle leidet, wenn gleichzeitig die HTTP Schnittstellen unter Last stehen. Dokumentieren Sie die dabei gewonnene Erkenntnisse in Ihrem Messprotokoll und geben Sie es im Moodle ab.

3.4 Aufgabe 4 - Remote Procedure Calls (RPC)

Für die nächste Aufgabe soll das Cloud-System des Dienst-Anbieters erweitert werden. Der Server soll die Daten nicht mehr nur lokal in einer Datenstruktur speichern, sondern in eine zu implementierenden externen In-Memory Datenbank auslagern. Der Server soll über einen Remote Procedure Call, z.B. Thrift, ProtoBuf oder gRPC, mit der neuen Datenbank kommunizieren können. Dabei sollen grundsätzlich die CRUD (Create, Read, Update, Delete) Methoden in der Datenbank implementiert werden.

Testen Sie die Performance Ihrer Implementierung indem Sie auf der Client-Seite die *Response Time* Ihrer *Remote Procedure Calls* messen. Werten Sie den Test statistisch valide aus und erstellen Sie ein (sehr kurzes) Messprotokoll das den Test und die Ergebnisse darstellt.

Wiederholen Sie den Performance-Tests aus Aufgabe 2 und Aufgabe 3 und vergleichen Sie die Testergebnisse, um fest zu stellen, wie die Performance aller Schnittstellen leidet, wenn gleichzeitig die RPC Schnittstelle unter Last steht. Dokumentieren Sie die dabei gewonnene Erkenntnisse in Ihrem Messprotokoll und geben Sie es im Moodle ab.

3.5 Aufgabe 5 - Message-oriented Middleware (MoM)

Das System soll nun um eine neue Art von Sensoren erweitert werden. Diese Sensoren können nicht mehr direkt über das UDP Request-Reply Protokoll abgefragt werden, sondern stellen Ihre Information mittels eines MoM-Protokolls,

z.B. MQTT oder AMQP, zur Verfügung. Um die Sensoren dennoch an den bestehenden IoT-Gateway anbinden zu können, muss zudem ein Adapter geschaffen werden, der Nachrichten über MQTT annimmt, und per UDP-Abfrage Seitens des IoT-Gateways zur Verfügung stellt.

Testen Sie die Performance Ihrer Implementierung indem Sie den Durchsatz ihres MOM-Infrakstruktur messen, z.B. in Nachrichten pro Minute oder Megabyte pro Sekunde oder etwas in der Art. Werten Sie den Test statistisch valide aus und erstellen Sie ein (sehr kurzes) Messprotokoll das den Test und die Ergebnisse darstellt.

Wiederholen Sie den Performance-Tests aus Aufgabe 2, 3 und 4 und vergleichen Sie die Testergebnisse, um fest zu stellen, wie die Performance aller Schnittstellen leidet, wenn gleichzeitig die MOM Schnittstelle unter Last steht. Dokumentieren Sie die dabei gewonnene Erkenntnisse in Ihrem Messprotokoll und geben Sie es im Moodle ab.

3.6 Aufgabe 6 - Hochverfügbarkeit und Konsistenz

Um auf Seiten des Service-Anbieters die Ausfallsicherheit weiter zu erhöhen, sollen die Daten redundant in zwei Datenbanken gespeichert werden. Im die Konsistenz der beiden Datenbanken sicherzustellen soll dazu ein Zwei-Phasen-Commit-Protokoll implementiert werden.

Wiederholen Sie den Performance-Tests aus Aufgabe 2 bis Aufgabe 5 und vergleichen Sie die Testergebnisse.

4 Pi-Lab

In diesem Semester steht Ihnen erstmals das Pi-Lab (<https://pi-lab.h-da.io/>) zur Verfügung um Ihre Anwendung in einem echten verteilten System zu deployen.

Das Pi-Lab besteht aus 8 Pi-Lab *Cubes*. Jeder Cube besteht wiederum aus 7 Raspberry Pi 4B *Knoten* mit jeweils 8 GB Arbeitsspeicher. Der Head-Knoten verfügt über ein Display. Die 6 Worker-Knoten sind Headless. Die Raspberry Pis werden mittels PXE gebootet und verfügen über keinen persistenten Speicher. Ein Neustart eines Knoten setzt den Pi somit wieder in seinen Ausgangszustand zurück. Alle Daten oder Änderungen gehen verloren.

Die Cubes wie auch einzelne Knoten lassen sich aus dem Netz der H-DA über eine Web-GUI starten und stoppen. Standardmäßig booten die Knoten eine modifizierte Version von Ubuntu 20.04 LTS Server, d.h. ohne GUI. Wenn ein Knoten erfolgreich gestartet ist, können Sie sich über SSH mit Ihrem H-DA-Account auf dem Knoten einloggen.

Auf allen Knoten sind Git, Docker und Docker-Compose vorinstalliert. Sie können Ihre Anwendung somit relativ problemlos auf dem Pi-Lab deployen. Beachten Sie aber, dass es sich bei den Raspberry Pis um eine ARM-Architektur handelt.

Eventuell müssen Sie Ihre Software (und die verwendeten Docker Images) an die Architektur anpassen bzw. auf dem Pi selbst kompilieren lassen.,

Das Pi-Lab ist derzeit noch in der Beta-Phase. Wir freuen uns daher über eine rege Nutzung und konstruktives Feedback. Die effiziente Nutzung des Pi-Labs im Rahmen des Labors qualifiziert Sie für die Bonuspunkte (siehe Abschnitt. 5). Sollten Sie darüber hinaus Interesse haben, z.B. im Rahmen einer Hiwi-Tätigkeit, einer Praxisphase oder einer Bachelorarbeit, an der Weiterentwicklung des Pi-Labs mitzuwirken, so sprechen Sie uns gerne direkt an.

5 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 1-5) 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.