WS 25/26 Verteilte Systeme

Praktikumsaufgabe 3

Koordination und Registry-Service für die Steuerung von Roboterarmen

1 Ziel der Aufgabe

Es soll ein verteiltes System zur Steuerung von bis zu vier Roboterarmen entwickelt werden. Teil der Aufgabe ist dabei sowohl die Implementierung eines Namendienstes als auch die Koordination zwischen mehreren Clients.

Sie entwickeln einen zentralen Registry-Service, bei dem sich Roboterarme registrieren können. Ein Client fragt diesen Dienst nach verfügbaren Armen und kommuniziert anschließend direkt mit den ausgewählten Armen. Dabei ist auszuschließen, dass zwei Clients gleichzeitig den selben Roboterarm benutzen. Im Fokus stehen hier die Themen Naming (Name, Adresse, Dienst) und die Koordination zwischen den Clients, nicht die Robotik.

---

2 Aufgabenstellung

2.1 Registry-Service (Registry-Server)

Entwickeln Sie einen zentralen Dienst, der alle Clients und die verfügbaren Roboterarme verwaltet.

- Entwerfen Sie ein schlankes Anwendungsprotokoll, basierend auf JSON, zwischen Registry und Clients.
- Speichern Sie pro Client und Roboterarm mindestens eine ID, Name, IP-Adresse, Port und den Typ (Client oder Roboter).
- Implementieren Sie den Registry-Server in der Programmiersprache Ihrer Wahl. Er muss mehrere parallele Verbindungen bedienen können.
- Stellen Sie mindestens folgende Operationen bereit:
    - register(name, ip, port)
    - unregister(name)
    - list(type)
- Persistenz darf in-memory erfolgen; dauerhafte Speicherung ist nicht erforderlich.
- Verhalten bei doppelten Namen: Registrierungen mit einem bereits existierenden *name* sind vom Registry-Service abzulehnen; der Registry-Service liefert eine aussagekräftige Fehlermeldung. Clients und Robot-Nodes müssen auf diesen Fehler reagieren (z. B. neuen Namen wählen oder abbrechen).

Die konkrete Nebenläufigkeitsstrategie (z. B. Thread pro Verbindung, Thread-Pool, NIO) wählen und begründen Sie im Bericht.

2.2 Roboter-Steuerung (Robot-Node)

Hinweis: Für die Kommunikation zwischen Robot-Node und Roboter verwenden Sie die von der Arbeitsgruppe „CaDS“ (Prof. Martin Becke) bereitgestellte Bibliothek (Repository: https://github.com/Transport-Protocol/CaDSPracticalExamVS). Dort finden sich unter anderem die Klassen *CaDSRoboticArmReal* und *CaDSRoboticArmSimulation*.

Entwickeln Sie ein Java-Programm (Robot-Node), das einen Roboterarm im Registry-Service registriert und diesen steuern kann.

Der Robot-Node bildet den logischen Server, den die Terminal-Clients ansprechen; die eigentlichen Roboter ist für den Client nicht direkt sichtbar.

- Registrieren Sie den Arm beim Start mit Name und Verbindungsparametern.
- Halten Sie den Eintrag zur Laufzeit aktuell (z. B. durch periodische Meldungen oder einen eigenen Mechanismus).
- Entfernen oder invalidieren Sie den Eintrag beim Beenden.
- Der Robot-Node soll abhängig von der Konfiguration entweder die reale Implementierung *java ICaDSRoboticArm roboticArm = new CaDSRoboticArmReal(roboticArmHostAddress, roboticArmHostPort);* oder die Simulation *java ICaDSRoboticArm roboticArm = new CaDSRoboticArmSimulation();* verwenden.

Überlegen Sie selbst, wie Sie Verbindungsabbrüche, doppelte Namen und eine nicht erreichbare Registry behandeln.

2.3 Terminal-Clients

Implementieren Sie einen Client (z. B. Terminal-Anwendung), der den Registry-Service nutzt und danach Steuerbefehle an einen oder mehrere vom Benutzer ausgewählten Robot-Nodes sendet.

- Fragen Sie periodisch oder auf Nachfrage die Liste der bekannten Roboterarme ab und zeigen Sie Name und Adresse/Status an.
- Ermöglichen Sie die Auswahl eines Roboterarms (z. B. per Eingabe von Name oder ID).
- Stellen Sie danach eine Verbindung zu den ausgewählten Robot-Nodes her und übertragen Sie die Steuerbefehle entweder
    - über Remote Procedure Calls (vgl. Praktikumsaufgabe 1) oder
    - über einen von Ihnen entworfenen Request/Response-Mechanismus.
- Die Terminal-Clients können in der Programmiersprache Ihrer Wahl implementiert werden
- Damit nicht mehrere Clients gleichzeitig auf den selben Roboter zugreifen können soll ein Token Ring Algorithmus implementiert werden. Jeder Client soll von der Registry eine eindeutige fortlaufende ID bekommen. Die Clients können Vorgänger und Nachfolger in der Ring Topologie basierend auf den IDs von der Registry abfragen.
  Diese Abfrage muss regelmäßig wiederholt werden um auf Änderungen in der Topologie

(z.B. beendete Clients) reagieren zu können. Der Client mit der ersten ID generiert einmalig beim Start des Systems das Token.

Die Kommunikation zwischen den Clients und von den Clients mit dem Roboterarmen darf nicht über den Registry-Service laufen, sondern ausschließlich direkt über IP und Port. Für die Abnahme genügt es, einfache Positionen einzelner Motoren zu setzen (keine komplexen Bewegungsabläufe erforderlich). Es ist ausdrücklich erlaubt für diese Aufgabe Programmcode aus den Lösungen aus Aufgabe wiederzuverwenden. Als Bibliotheken dürfen nur die JAVA-CaDS Robotersteuerung und eine Bibliothek für die Serialisierung von Nachrichten benutzt werden.

Ihr System sollte mindestens folgende Szenarien demonstrieren:

- erfolgreicher Ablauf von Registrierung, Auflistung und Steuerung eines Roboters mit mehreren Clients,
- ein Client beendet sich → der Eintrag im Registry-Service reagiert entsprechend (z. B. verschwindet nach Ihrem gewählten Mechanismus) und die Ring Topologie wird entsprechend von den Clients angepasst,
- der Registry-Service ist nicht erreichbar → der Terminal-Client meldet eine nachvollziehbare Fehlersituation.

Für die Abnahme können entweder die realen Roboterarme im Labor oder die Simulationen (CaDSRoboticArmSimulation) genutzt werden.

---

3 Dokumentation und Abgabe

Erstellen Sie eine Dokumentation, die folgende Punkte beschreibt:

1. **Architekturübersicht**
    - Übersicht über Registry-Service, Robot-Node und Terminal-Client sowie deren Kommunikationsbeziehungen.
    - Mindestens eine Ablaufdarstellung (z. B. Sequenzdiagramm) für Registrierung und Nutzung eines Roboters.
2. **Protokollbeschreibung**
    - Beschreibung der definierten Operationen (register, unregister, list) mit Syntax und Semantik.
    - Beschreibung der Token-Übergabe
    - Mindestens ein vollständiges Beispiel (Request und Response) pro Operation.
    - Beschreibung des Verhaltens bei Fehlerfällen (z. B. doppelter Name, ungültige Anfragen).
3. **Wesentliche Designentscheidungen**
    - Auswahl Ihrer Nebenläufigkeitsstrategie (z.B. Thread pro Verbindung, Thread-Pool, NIO).
    - Mechanismus zur Aktualität der Einträge (z.B. periodische Meldungen, Timeouts).

Spätester Abgabezeitpunkt ist am Anfang des vierten Praktikum Termins.

Hinweis zur Simulation

Die Klasse *CaDSRoboticArmSimulation* benötigt ***JavaFX***. Für die Ausführung können Sie das JavaFX SDK lokal herunterladen und beim Start Ihrer Anwendung per VM-Argumenten hinzufügen.

Beispiel (Pfad anpassen):

--module-path /path/to/javafx-sdk-21.0.2/lib \

--add-modules=javafx.controls,javafx.fxml

Stellen Sie sicher, dass diese Optionen beim Start des Robot-Node gesetzt werden, wenn Sie die Simulation nutzen.