Architektur

Anforderungen

1. Kommunikation muss sicher sein.
2. Kommunikation muss robust gegenüber hohen Latenzen & Verbindungsabbrüchen sein.
3. Kommunikation muss leichtgewichtig sein.
4. System muss viele Clients behandeln.
5. System muss viele Haltestellen behandeln.
6. System muss viele Autonome Fahrzeuge koordinieren.
7. System muss variierende Anzahl an Teilnehmern zur Laufzeit erlauben.
8. Autonome Fahrzeuge müssen Routenoptimierung anhand Nachfrage durchführen können.

Entwurf

Die Anforderungen an das System können wie folgt erfüllt werden:

1/2/3: Das MQTT-Protokoll erfüllt alle Anforderungen. Das Protokoll ist sehr leichtgewichtig, weswegen es bei IoT-Geräten bevorzugt verwendet wird. Die Kommunikation kann mithilfe von SSL verschlüsselt werden und eine zusätzliche Authentifizierung mit Nutzername und Passwort ist integriert. Zusätzlich unterstützt es den QoS „exactly once“ und hat noch andere Mechanismen, die eine abgebrochene Verbindung z.B. erkennbar machen.

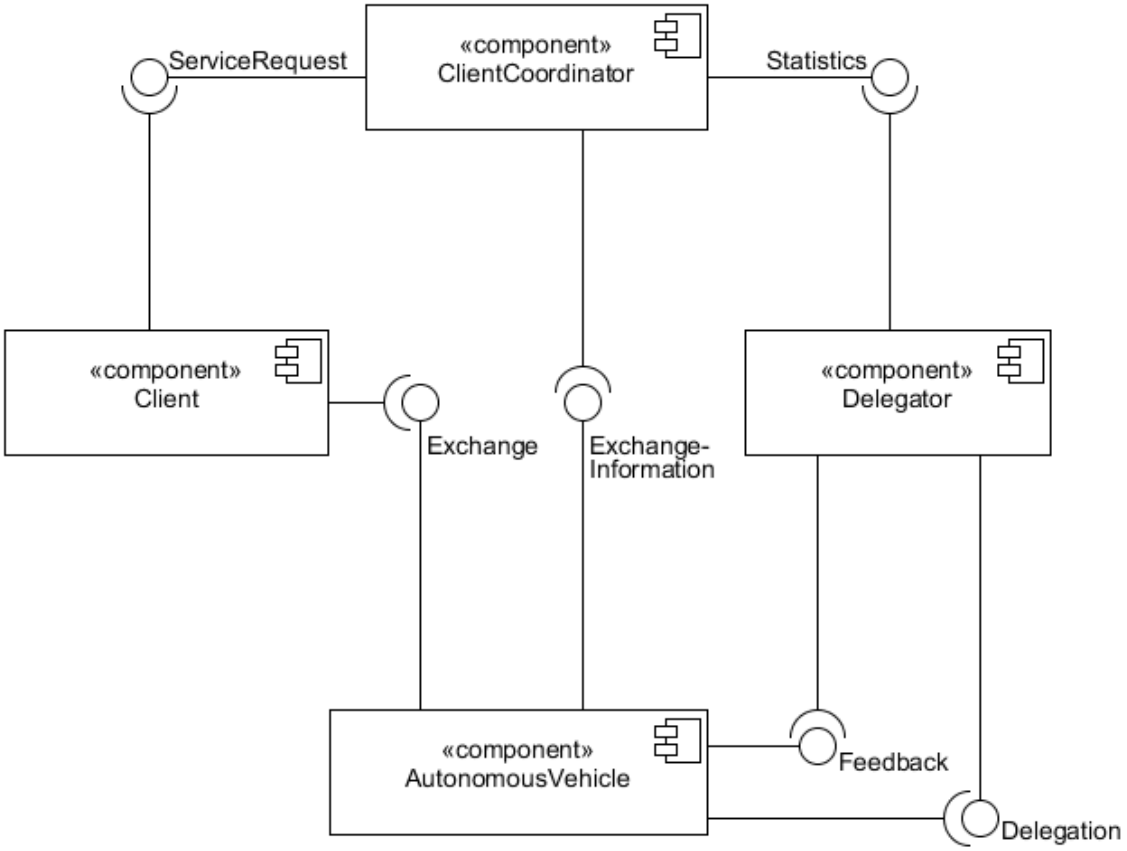
4: Um eine hohe Anzahl an Clients bedienen zu können ließe sich die Kommunikation mit dem Service über eine verteilte Anmelde-Komponente realisieren.

5: Es sollte eine Datenbank geben, auf die die Teilnehmer des Systems eine gemeinsame Sicht auf Routenpunkte haben, da auf die Informationen u.a. von Clients, Autonomen Fahrzeugen und sonstigen Komponenten, die evtl. Aufgaben zur Routenoptimierung übernehmen zugegriffen wird.

6/7/8: Es sollte eine Komponente geben, die das Laufzeitverhalten des Systems in Bezug auf Auslastung überwacht und dementsprechend Systemweite Optimierungen vornimmt.

Komponenten

Aus den vorgegangenen Überlegungen ergibt sich folgende Überlegung zu den Komponenten und ihrer Beziehungen:



Der ClientCoordinator stellt die Haupt-Schnittstelle zur Client-seitigen Nutzung des Service dar. Er regelt die Anfragen der Clients und die Abwicklung. Die Nutzung von MQTT als Kommunikationsprotokoll ermöglicht u.a. die Komponente als Verteiltes System zu implementieren, was eine Abhandlung von vielen Clients erlaubt.

Der Delegator bekommt eine Statistische Sicht von dem ClientCoordinator zur Verfügung gestellt. Mit diesen Informationen zu gewünschten Strecken der Clients und Statusinformationen der AutonomousVehicles führt er komplexe Berechnungen zur Verteilung und Routen-Delegierung durch und delegiert aufgrund der Ergebnisse die AutonomousVehicles.

Der Client hat als einzige Aufgabe, den Service zu nutzen. Hierfür setzt er sich mit dem ClientCoordinator in Verbindung (mithilfe einer leichtgewichtigen Applikation), welcher die Anfragen bearbeitet.

Das AutonomousVehicle fährt eine Route ab und lässt sich von dem Delegator befehligen. An Routenpunkten sammelt es Clients ein und teilt dem ClientCoordinator deren Teilnahme am Dienst mit.

Modellierung

Die Modellierung der Clients und Autonomen Fahrzeuge ist den Automaten, welche zuvor in Uppaal modelliert wurden (s. Verifikationsziele) sehr ähnlich.

Die Server-Seite des Systems hat durch die Anforderungen an die Skalierbarkeit und Rechenleistung weitere Unterteilung in unterschiedliche Aufgaben nötig. Dafür wurde zunächst einmal die ClientCoordinator Komponente unterteilt in ihre internen Aufgaben: