**Verteilte Systeme – Anforderungsanalyse:**

**1. Glossar**

Definition **Sockets**:

Ein Socket ist eine Schnittstelle zwischen einem Anwendungsprozess und den vom Betriebssystem angebotenen Kommunikationsprotokollen der Transportschicht.

Man kann sich Sockets als eine Art Stecker vorstellen, mit denen man eine Verbindung zwischen zwei auf verschiedenen Rechnern / demselben Rechner laufenden Prozessen herstellen kann. Über diese Verbindung können dann Daten zwischen den beiden Prozessen ausgetauscht werden.

Definition **HTTP**:

HTTP ist das Akronym für Hypertext Transfer Protocol.

Es regelt die Kommunikation zwischen Servern und Clients (Web Browser) bzw. Suchmaschinen-Trackern noch oberhalb des TCP/IP-Protokolls.

Definition **UDP**:

UDP ist ein verbindungsloses Transport-Protokol.

Definiton **TCP**:

TCP ist ein verbindungsorientiertes Transport-Protokol.

Definition **IoT**:

Internet of Things ist die globale Infrastruktur in der virtuelle und physische Geräte vernetzt sind.

Definition **MQTT**:

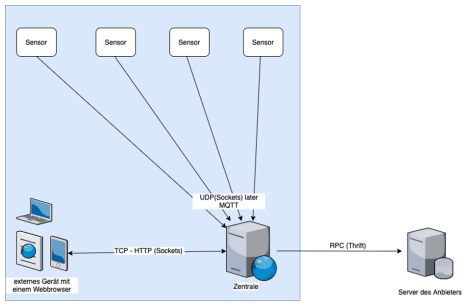
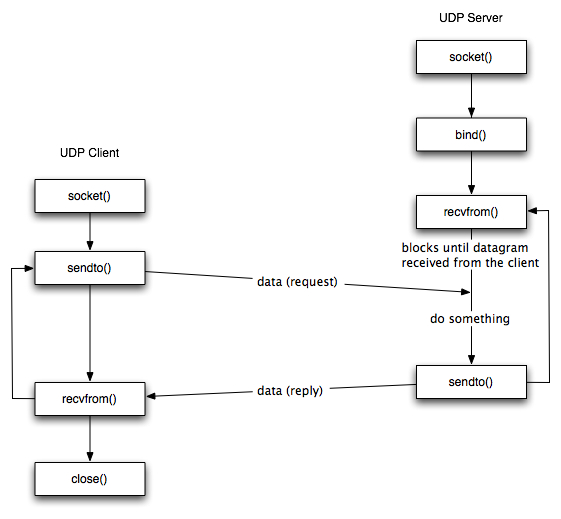
Message Queue Telemetry Transport.MQTT ist in erster Linie ein Messaging Protokoll für das IoT. Anders als Request/Response Protokolle wie HTTP oder CoAP entkoppelt es alle Kommunikationsteilnehmer und der Datenaustausch wird über das Senden von Nachrichten über einen zentralen Verteiler an interessierte Teilnehmer implementiert, weshalb kein intrinsisches Wissen über andere Applikationen und Endpunkte vorhanden sein muss.

Definition **RPC**:

Remote Procedure Call ist ein für das Network File System (NFS) entwickeltes Protokoll für die Sitzungsschicht und ermöglicht einen entfernteren Aufruf einer Prozedur. Mit Remote Procedure Call können Clients auf einem entfernten Server Prozeduren aufrufen.

**Kommunikation**

Die Kommunikation zwischen den Sensoren und dem Zentralrechner wird mittels **UDP-Sockets** realisiert, da sie performanter und schneller als TCP-Sockets sind und ein gewisser Paketverlust in dieser Kommunikation sich nicht als problematisch darstellt.

Die Kommunikation zwischen dem Zentralrechner und der Clients, also die externen Geräte werden verbindungsorientiert mit **TCP-Sockets** realisiert.  


**Sensoren**

Es werden mindestens 2 Sensoren benötigt, die zu dem Zentralrechner übermittelt werden.

* Temperatur
* Helligkeit
* Geräusche (dB)
* Bewegungen

Jeder Sensor erfasst jeweils einen dieser Werte und wird als eigenständiger Prozess simuliert. Die Konfiguration der Sensoren soll mit Hilfe einer Konfigurationsdatei geschehen. Der Sendeintervall soll Variabel sein, um so verschiedene Werte zu testen und daraus eine Evaluation schließen zu können. Mit jedem Sensordatenpaket soll die ID, ein Zeitstempel, IP und Port übertragen werden.

Da die Sensoren von einander abhängen werden sie in einem Thread laufen.

Die Sensordaten werden mittels einem String übertragen. Dieser sieht wie folgt aus:

**<Adresse>:<Port> <Sensorname> <Daten> <Einheit> <Zeitstempel>**

Beispiel:

(**Localhost:5000) Geräuschssensor 30dB 25.11.19**

Dieser String wird in jedem Sensor aufgebaut und mittels UDP Paketen versendet. Im Zentralrechner wird dieser verarbeitet und gespeichert.

**Zentralrechner (Smart Home)**

Ziel des Zentralrechners ist das Empfangen der Sensordaten und die Verarbeitung der Daten: Parsen, Speichern und Übertragung über HTTP. Die Zentralrechner werden als HTTP-Server realisiert. Die Daten werden mit einem HTTP GET Befehl verarbeitet. Alle Sensordaten (auch Alte) müssen gespeichert werden. Dies soll mit einer REST-API ermöglicht, indem man Zugriff auf einzelne Sensordaten ermöglicht.

REST-API = Representational State Transfer - Application Programming Interface.

Ist eine Programmierschnittstelle (Architekturansatz) die beschreibt wie kommuniziert wird in

Verteilten Systemen. Mögliche standardisierte Verfahren sind HTTP/S, URI, JSON oder XML.

In der Aufgabenstellung ist URI als Identifikator gegeben. URI = Uniform Resource Identifier.

Die HTTP-Response soll unbedingt korrekt sein, z.B. Content-length.

Pro Sensor soll eine URL verwendet werden.

Damit der User diese Daten abrufen kann, ruft er folgende URL auf:

[**http://localhost:8080/sensor=<Sensorname**](http://localhost:8080/sensor=<Sensorname)**>**

Jedem Sensor wird beim erstellen eine ID zugewiesen. Diese Überschneidet sich mit den anderen Sensoren und sind zugehörig zu einem Auto. Diese ID wird in den Programmparametern übergeben.

**RPC**

Die Werte der Sensoren soll nun von den Zentralrechnern an die Herstellerserver weitergegeben werden. TCP soll als Protokoll zwischen HTTP-Server und Browser genutzt werden. Der Herstellerserver soll die übermittelten Daten persistent speichern

Datenpakete seperieren, limitieren

**Hochverfügbarkeit und Konsistenz**

* Thrift als RPC
* Master Slave
* Speichert alle Sensordaten von Autos redundant auf mehreren Rechnern
* Anmeldung an Server mit IP und Port
* Rückmeldung falls erfolgreich gespeichert werden konnte
* Schreiben in eine Datenbank oder .txt file
* ggf. Serversynchronisierung

**MoM mittels MQTT**

* Sammelt Daten von den Herstellerservern mittels MQTT
* Bereitet alle Daten auf und stellt ein Gesamtbild der Verkehrslage für Zeitpunkte in der Vergangenheit, textuell dar.

**Nicht-funktionale Anforderungen**

* Ausfallsicherheit der Server
* Datenverlust auf das Minimum reduzieren
* Performance
* Bedienbarkeit