Anforderungsanalyse Kraftwerk

Aufgabestellung

Im Rahmen des Praktikums Verteilte Systeme soll eine Anwendung aus dem Bereich "Stromerzeugung/Kraftwerke" entwickelt werden. Dazu sollen die Technologien Sockets, RPC (Apache Thrift, gRPC/Protobuf) sowie Message-Oriented-Middleware (MQTT) verwendet werden.

- **Aufgabe 1** : Projektplan
- **Aufgabe 2** : UDP und TCP Sockets
- **Aufgabe 3** : RPC
- **Aufgabe 4** : MoM mittels MQTT
- **Aufgabe 5** : Hochverfügbarkeit und Konsistenz

Funktionale Anforderungen

Erzeuger/Verbraucher-Service

Als Erzeuger/Verbraucher möchte ich meine aktuelle Stromproduktion/meinen aktuellen Stromverbrauch an die Zentrale schicken können.

1. Anfragen werden per UDP an die Zentrale, Port 50000 gesendet.
2. Die Strommenge soll variieren.
3. Die statischen Informationen zu einem Teilnehmer werden aus Umgebungsvariablen gelesen.
4. Die gesendeten Daten sind in einem gültigem JSON-Format mit den Attributen für die Paketnummer, ID des Teilnehmers, Art des Teilnehmers und die aktuelle Strommenge.

Zentrale-Service

Als Zentrale möchte ich, dass die Daten über die verbrauchte Strommenge von den Verbrauchern erhalten und gespeichert werden können.

1. Anfragen per UDP auf Port 50000 werden angenommen.
2. Daten, die auf dem UDP-Sockel ankommen, werden in Dateien zugehörig zu der ID des Verbrauchers gespeichert.

3. Die Daten sollen im JSON-Format sein.
4. Die Daten enthalten mindestens die Folgenden Daten:
   - Art (Verbraucher)
   - eindeutige ID des Verbrauchers
   - Betrag der aktuellen Strommenge
5. Der Zentralserver stellt Port 80 für HTTP-Anfragen zur Verfügung.
6. Der Zentralserver erlaubt Dateien nur aus bestimmten Verzeichnissen abzurufen (REST-API):
   - Es sollte eine Liste von erzeugten Teilnehmern auf einer Webseite dargestellt sein (<IP>/list).
   - Es sollte die Information über jeden einzigen Teilnehmer über seine eindeutige ID im URL bereitstehen (<IP>/<ID>).
7. Wird der Zentralserver ohne REST-Parameter angesprochen, liefert er eine Test-Seite aus.
8. Der Zentralserver kann GET-Anfragen komplett auswerten.
9. Der Zentralserver loggt für HTTP-Anfragen die folgenden Inhalte:
   - Timestamp in UTC YYYY-MM-DD HH:MM:SS [+0000]
   - HTTP-Statuscode der Anfrage
   - Übertragene Bytes
   - IP des Clients
   - User-Agent des Nutzers (Browser-Informationen)

Steuerung

Die Zentrale kann einzelne Erzeuger/Verbraucher abschalten, um einen Energieausgleich zu schaffen.

1. Die Steuerung der Erzeuger/Verbraucher erfolgt via RPC.
   - Der Zentralserver besitzt eine Apache Thrift oder gRPC/protobuf-Instanz.
   - Die Teilnehmer haben eine RPC-Schnittstelle.

Datenabfrage

Externe Clients sollen die Möglichkeit haben den aktuellen Status und die komplette Historie zu erhalten, um Auswertungen durchführen zu können.

1. Ein externer Client kann den aktuellen Status des Zentralservers und der angebundenen Erzeuger und Verbraucher auslesen.

2. Ein externer Client kann die komplette Historie ausgegeben bekommen.
3. Die Abfragen erfolgen per RPC.

Nicht-Funktionale Anforderungen

Initialisierung

Als Entwickler möchte ich, dass das ganze System mithilfe von Docker Compose initialisiert wird.

1. Einem Container wird genau ein Service zugeteilt.
2. Im docker-compose.yml sollte es möglich sein, Erzeuger/Verbraucher zu definieren (Anzahl, Art, Id).
3. Alle Erzeuger/Verbraucher Container nutzen das gleiche Docker-Image.
4. Der Build-Prozess der Docker-Images kann ausserhalb von Docker-Compose mittels eines Skripts erfolgen.

Performanz

Die verschiedenen Prozesse des Zentralservers sollen sich nicht gegenseitig blockieren und Ausfälle sollen verkraftet werden können.

1. Es können gleichzeitig Anfragen auf dem TCP- und UDP-Port gestellt werden.
2. Die Zentralen sind redundant ausgelegt und der Ausfall einzelner Zentralen hat keine nennenswerten Datenausfälle zur Folge.

Usability

1. Es soll mindestens einen funktionalen Test sowie einen Performance-Test für jede Aufgabe geschrieben werden.

Dokumentation

1. Der Code soll ausführlich dokumentiert sein.
2. Es soll eine README.md Datei erstellt werden, die eine detaillierte Anleitung beinhaltet wie die Software kompiliert und mittels Docker und Docker-Compose gestartet und getestet wird.

Lizenzen

1. Das gitlab-Repository soll über ein Lizenz-File verfügen, welches die Lizenz der Software ausweist.

Technische Anforderungen

1. Es ist ein Build Tool (Make, Maven, etc.) zu verwenden.
2. Die Lösungen müssen containerisiert sein und mittels Skripten, Docker (v19.03) und Docker-Compose(v1.24) zu testen sein.
3. Programmiersprache: C++
4. Sockets
5. REST
6. MQTT
7. gRPC/Apache Thrift
8. HTTP Protokoll