Message Queuing Telemetry Transport (MQTT)

Betriebliche Informationssysteme, von Markus Reinhold , Lukas Brüggemann und Dominik Jülg

Gliederung

1. Übersicht über den MQTT Standard
2. Protokoll
3. Skalierbarkeit und Effizienz
4. Sicherheit
5. Use-Case

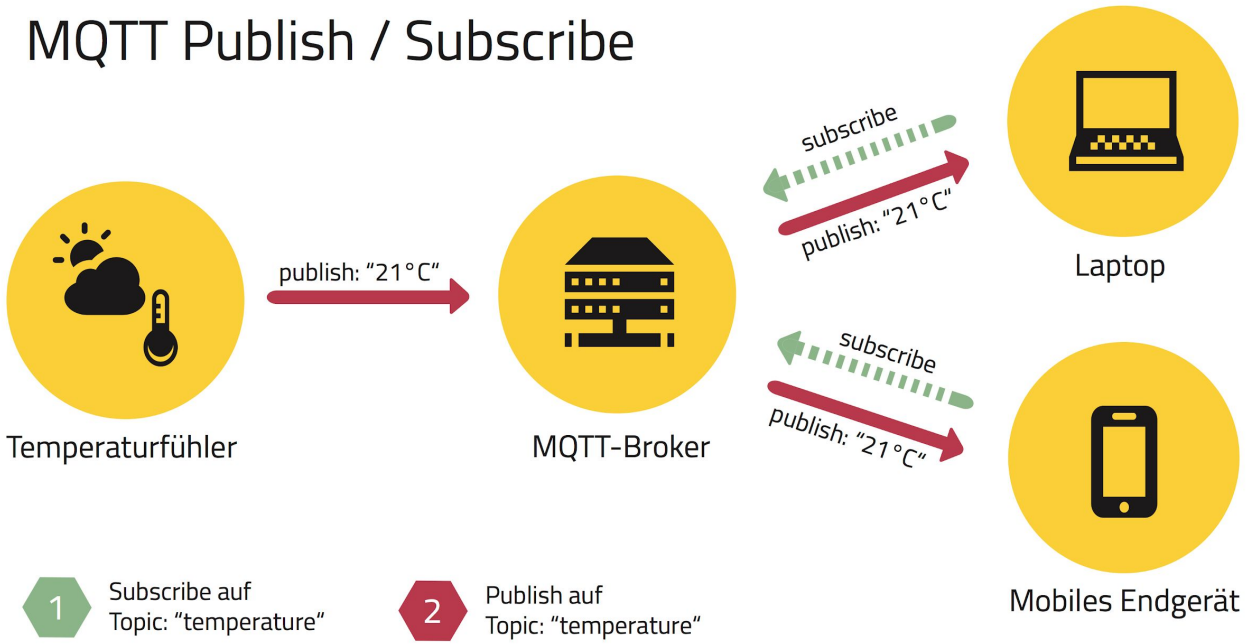
Übersicht über den MQTT Standard

Was ist MQTT?

- MQTT ist ein standardisiertes Messaging-Protokoll von OASIS
- Läuft über TCP
- Vorrangig genutzt für das Internet der Dinge (IoT)
- Extrem leichter Publish- / Subscribe-Messaging-Transport
- Verbinden von Remote-Geräten mit geringem Code-Footprint und minimaler Netzwerkbandbreite
- MQTT wird in vielen Branchen eingesetzt (Automobilindustrie, Fertigung, Telekommunikation, Öl und Gas usw.)

Client/Server

- Es gibt einen Server (genannt Broker)
  - Da MQTT nur ein Standard ist gibt es viele verschiedenen Implementierungen für Broker
- Alles andere sind Clients
  - Clients können entweder publishen
  - Oder auf einer Topic subscriben



Protokoll

Topics

- String zur Verwaltung von Subscriptions
- kann mit Slash (/) in Unter-Topics (Levels) separiert werden
- Beispiel: SmartHome/Erdgeschoss/Wohnzimmer/Temperatur

Wildcards

- einzelnes Level mit +
- Beispiel: SmartHome/Erdgeschoss/+/Temperatur > Temperatur aus allen Räumen im Erdgeschoss
- mehrere Level mit #
- Beispiel: SmartHome/Erdgeschoss/# > alle Nachrichten vom Erdgeschoss

Verschiedene Qualitäts-Levels

Vorteil von MQTT: Nachrichten werden garantiert gesendet, auch, wenn die verwendete Verbindung unzuverlässig ist

sog. Quality of service levels (von oben nach unten: sicherer, aber langsamer)

- *At most once* (0)
- *At least once* (1)
- *Exactly once* (2).

Levels können von den Clients selbst ausgewählt werden

QoS 0 (at most once)

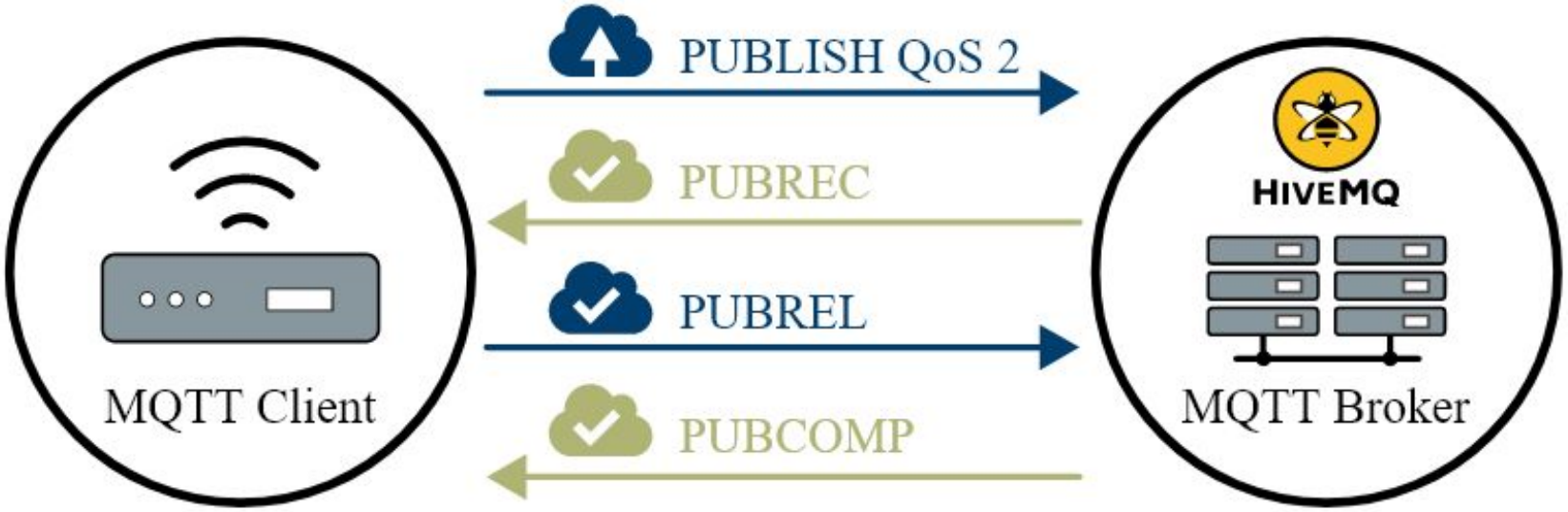
- keine Garantie, dass die Nachricht empfangen wurde
- Client speichert die Nachricht nicht und kann sie somit auch nicht nochmal senden
- Broker sendet keine Empfangsbestätigung

QoS 1 (at least once)

- Garantie, dass Nachricht mindestens einmal empfangen wurde
- Client speichert Nachricht solange, bis Paket *PUBACK* vom Broker zurückkommt
- bis dahin wird die Nachricht periodisch versendet

QoS 2 (exactly once)

- Garantie, dass Nachricht *genau* einmal empfangen wurde
- Client sendet solange, bis vom Broker eine Antwort kommt
- Broker wartet mit Weiterverarbeitung, bis Sendung abgeschlossen ist
- weitere zwei Packages, um den Empfang zu quittieren



weitere Features

persistent sessions	automatischer Neuaufbau der Verbindung bei Abbruch
last will and testament	Nachricht, welche gesendet wird wenn ein Client unerwartet die Verbindung trennt (um mit dieser Information andere Clients darüber zu informieren)

Demo

Skalierbarkeit und Effizienz

Grundlagen

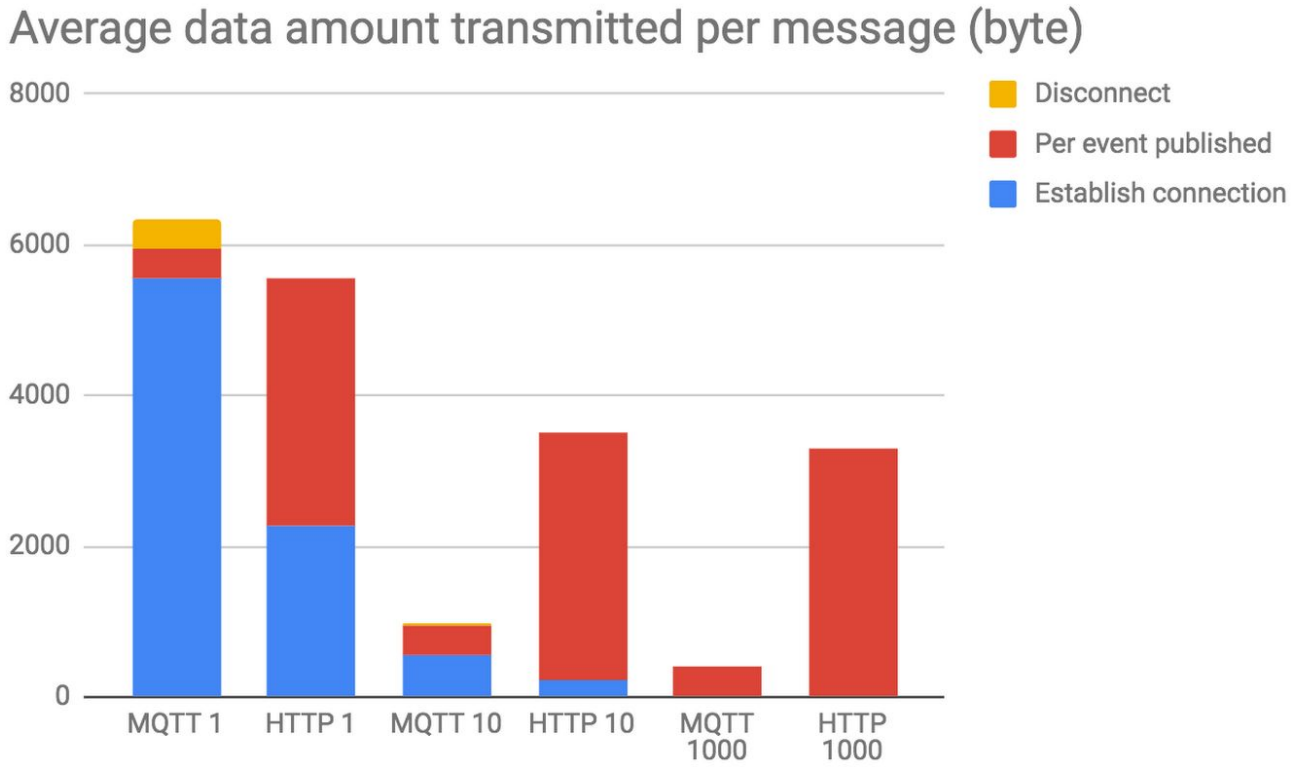
- MQTT kann skaliert werden
- nach Anzahl von Publishern und Subscribern ist stärkere Hardware notwendig (Vertikale Skalierung)
- Horizontale Skalierbarkeit ist auch möglich, abhängig von Broker-Implementierung.

HTTP VS MQTT

- 10% weniger Traffic mit MQTT
- 22% Energie effizienter als HTTP
- 15% Ressourcen effizienter als HTTP

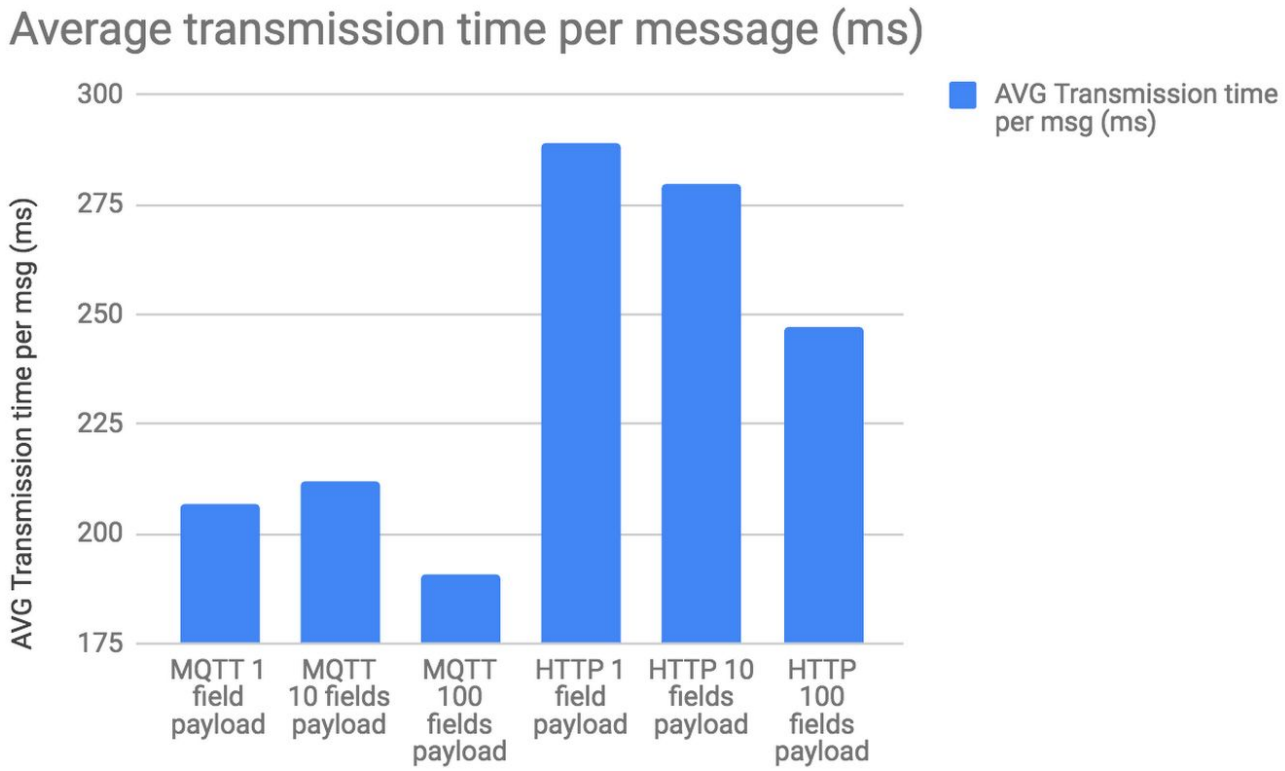
Effizienz

- mehr als 90% des Traffics ist für den Aufbau und Beenden einer Verbindung
- vorteil gegenüber HTTP bei wiederverwendung von bestehender Verbindung



Effizienz

- höchste Effizienz möglich bei hoher dichte der payload message



Sicherheit

Möglichkeiten zur sicheren Kommunikation

- eigener Broker ist Sicherste weg
- initialen Verbindungsaufbau zu einem MQTT-Broker Authentifizieren
  - mit Benutzernamen und Passwort
  - komplexere Authentifizierungsmethoden wie OAuth 2.0
- Authorisierung
  - bösartiger Client Nachrichten von allen Topics abonnieren und damit unberechtigterweise Daten mitlesen
  - Broker entscheidet
  - Einschränkungen konfigurieren (maximale Nachrichtengröße der Clients und die maximale Bandbreite MQTT-Sender)

Dominik Jülg, Informatik Master, Software Engineering
Hochschule für Technik, Wirtschaft und Kultur Leipzig

Verschlüsselung

- Client initialisiert selbst verbindung mit Broker
  - kein Öffnen der Verbindungen von außen möglich
  - Brute-Force- und Denial-of-Service-Attacken auf einzelne Geräte sind schwierig durchzuführen
- TLS einsetzten
  - Man-In-The-Middle-Attacken weitgehend vermeiden
  - Verschlüsselte Verbindung

Use Cases

Matternet

- Über Matternet senden Drohnen und Landestationen Echtzeitdaten über MQTT an HiveMQ

- So kann geprüft werden wo sich eine Drohne befindet und wann eine Lieferung ankommt

Mobility Services

- Carsharing von BMW und Daimler nutzt MQTT um Kommunikation zwischen Auto und App zu ermöglichen

Quellen

- **https://mqtt.org/assets/img/mqtt-logo-ver.jpg 09.01.2021**
- **https://www.informatik-aktuell.de/betrieb/netzwerke/mqtt-leitfaden-zum-protokoll-fuer-das-internet-der-dinge.html 09.01.2021**
- **https://m.heise.de/developer/artikel/Sichere-IoT-Kommunikation-mit-MQTT-Teil-1-Grundlagen-3645209.html?seite=all 09.01.2021**
- **https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/ 09.01.2021**
- **https://flespi.com/blog/http-vs-mqtt-performance-tests 09.01.2021**
- **https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols 09.01.2021**
- **https://internetofbusiness.com/matternet-amazon-city-drone-delivery/ 09.01.2021**
- **https://www.bmw.ca/content/dam/bmw/common/topics/fascination-bmw/BMWi/mobility%20services/drivenow-1.jpg 09.01.2021**