MQTT

Die Sprache im Internet der Dinge (IoT)

MQTT

Message-Queue-Telemetry-Transport

MQTT

Geschichte

- 1999 von IBM entwickelt
- seit 2013 standardisiert über Organization of Structured Information Standards (OASIS) als IoT-Protokoll
- Machine-to-Machine-Kommunikation (M2M)
- setzt auf TCP/IP-Protokoll auf (Port 1883 und 8883 reserviert durch Internet Assigned Numbers Authority (IANA))

MQTT

Geschichte

- wird von einer Community weiter entwickelt
- aktuell in der Version 5 (Stand April 2019)

MQTT

Eigenschaften

- es ist einfach zu implementieren
- Leichtgewichtig und minimaler Protokoll-Overhead
- Push-Messaging
- Einstellbare Stufen (Quality of Service) im Protokoll
  Unabhängig von Dateninhalten (Kopf und Inhalt)
- Nachrichten können zwischengespeichert werden
- eine Verbindung zwischen Client/Server besteht ständig

MQTT

Einsatzgebiete

- Eigentlich überall da, wo Daten zwischen Geräten unterschiedlichster Bauart ausgetauscht werden sollen (M2M)
- auf ressourcenarmen Geräten (IoT, Esp, u.s.w.)
- sehr beliebt bei Hausautomatisierung o.ä.
- Messenger-Dienste basieren auf MQTT

MQTT

Sicherheit

- Authentifizierung gegenüber Broker via User/Passwort einstellbar
- SSL/TLS-Support zwischen Clients und Broker konfigurierbar

MQTT

Wie funktioniert es?

- Publisher
- Broker
- Subscriber

MQTT

Wie funktioniert es?

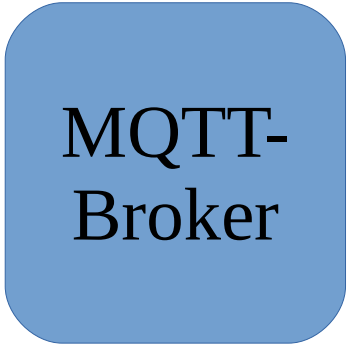
- Publisher
  Client (Temperatur-Sensor z. Bsp. Wohnzimmer)

MQTT-Client

MQTT

Wie funktioniert es?

- Broker
  Server (Software)



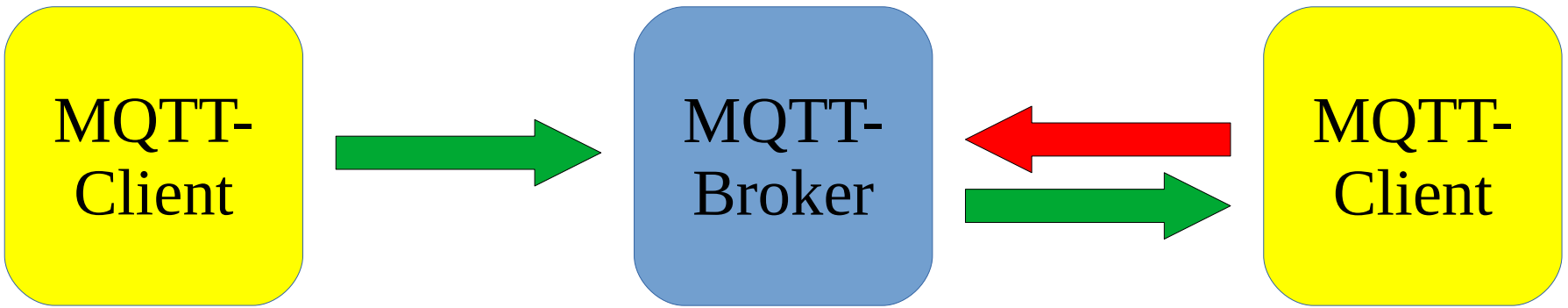
MQTT

Wie funktioniert es?

- Subscriber
  Client (Abonnenten)

MQTT

Wie funktioniert es?



MQTT

Broker

- Moquette
- Mosquitto
- MQTTRoute
- Emqttd
- HiveMQ
- HBMQTT

https://github.com/mqtt/mqtt.github.io/wiki/servers

MQTT

Topic-Filter

- Wildcard
  - Single-Level Wildcard

  sensors/+/temperature
    - sensors/sensor1/temperature
    - sensors/sensor2/temperature
    - sensors/sensor3/temperature

MQTT

Topic-Filter

- Wildcard
  - Multi-Level Wildcard

  sensors/sensor1/#
    - sensors/sensor1/temperature
    - sensors/sensor1/pressure
    - sensors/sensor1/humidity/in
    - sensors/sensor1/humitity/out

MQTT

Retained-Flag

Normalerweise bekommen Subscriber erst dann Nachrichten zu einem abonnierten Topic zugestellt, wenn ein Publisher aktuell eine Nachricht zum Broker sendet.
… mit dem Retained-Flag gekennzeichnete Nachrichten werden vom Broker zwischengespeichert
… und werden vom Broker sofort ausgeliefert, wenn sich ein Subscriber entsprechend beim Broker anmeldet

z. Bsp. Messenger-Dienste

MQTT

Last Will and Testament (LWT)

- Was soll passieren, wenn die Verbindung zwischen einem Client (Publisher/Subscriber) und Broker abbricht?
- Es kann eine Nachricht vom Client definiert werden, die bei Verbindungsabbruch, vom Broker publiziert wird

z. Bsp. Strassenbeleuchtung, Notausgänge

MQTT

Quality of Service (QoS)

- QoS definiert/gewährleistet Garantien bezüglich der Zustellung von Nachrichten

- MQTT-Protokoll definiert 3 Stufen:
  - 0 → höchstens einmal (...aber auch keinmal!)
  - 1 → mindestens einmal (...aber auch mehrmals!)
  - 2 → exakt einmal

MQTT

Quality of Service (QoS)

- QoS gilt in beide Richtungen:
  Client - Broker
  Broker - Client
- Der Client gibt den QoS-Level vor

MQTT

Quality of Service (QoS)

- QoS 0
  verlorene Nachrichten sind tolerierbar, (...kommen aber selten vor, da Netzwerk stabil…)
- QoS 1
  Nachricht muss ankommen Netzwerk-Overhead von QoS 2 ist zu hoch Duplikate sind tolerierbar
- QoS 2
  Nachricht muss ankommen, Duplikate sind nicht tolerierbar

MQTT

für Softwareentwickler

- MQTT-Bibliotheken, -Erweiterungen, -Module, -Firmware
  https://github.com/mqtt/mqtt.github.io/wiki/libraries
- U.a. für:
  C/C++, Java, Javascript, Node.js, PHP, Python, Nodemcu
- MQTT – Docs
  http://docs.oasis-open.org/mqtt/mqtt/

MQTT

Weiterführende Informationen

- http://mqtt.org/
- http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
- https://github.com/mqtt/mqtt.github.io/wiki