1 MQTT

MQTT als einfaches, stark datenkonzentriertes Protokoll.

	MQTT (MESSAGE QUEUING TELEMETRY TRANSPORT)
ZIEL / NUTZEN	Einfaches und sparsames Protokoll; geringe Anforderung an Netzwerkbandbreite
ARCHITEKTUR	Pub/Sub mit zentralem Broker
NETZWERK	TCP
QOS	at most once, at least once, exactly once
SICHERHEIT	SSL / TLS
STANDARD	ISO / OASIS
FEATURES	Themenbasiert, Datenkonzentriert, Keep-Alive-Messages, Letzter Wille, Message-Retention

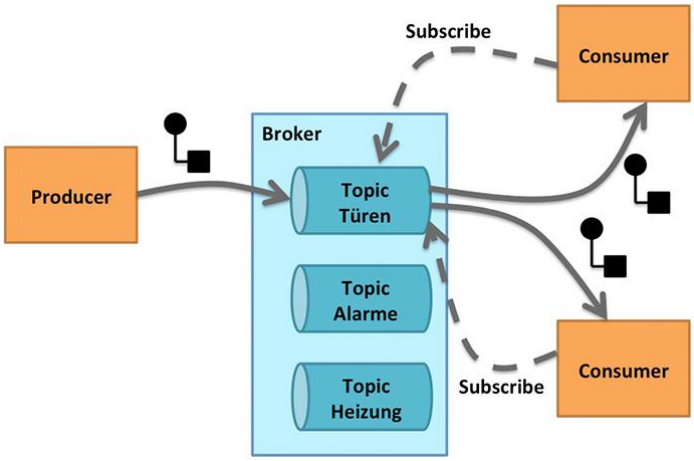
Entwickelt wurde MQTT 1999 von Stanford-Clark und Nipper, um eine Öl Pipeline in der Wüste zu überwachen. Neben Zuverlässigkeit war der sparsame Umgang mit Bandbreite und Ressourcen ein Ziel der Entwicklung.

MQTT eignet sich für die Kommunikation mit Komponenten des Internet of Things, bei denen Bandbreite und Energie knapp sind. Die Übertragung von Nachrichten erfolgt dank des kompakten Binärprotokolls annähernd in Echtzeit.

2 MQTT

2.1 Publish/Subscribe

MQTT setzt vom Konzept das Publish/Subscribe Muster um. Dabei kommuniziert der Sender einer Nachricht nicht direkt mit den Empfängern, sondern schickt die Nachricht an einen Broker, der die Zustellung der Nachricht übernimmt. Der Broker entkoppelt den Sender oder Producer von den Empfängern, den Consumern. Producer können immer senden, sie wissen nicht, ob es überhaupt Consumer gibt. Consumer und Producer sind Clients des Brokers.



Ein Producer kann pro Thema ein Topic einrichten, für das sich Consumer anmelden (subscriben) können. In der Abbildung links haben zwei Consumer eine Subscription auf das Topic Türen. Sendet jetzt ein Producer eine Nachricht z.B. über das Öffnen einer Türe an das Topic Türen, so übermittelt der Broker jeweils eine Kopie der Nachricht an die beiden Subscriber

Ein Subscriber kann sich für eine beliebige Anzahl Topic einschreiben. In der

Abbildung unten ist der Consumer an allen Topics interessiert, die Ereignisse vom Erdgeschoss (eg) liefern. Das Zeichen # ist eine Wildcard, die zu allen Topics im Erdgeschoss (eg) passt.

Im Gegensatz zu Client/Server Protokollen wie HTTP arbeitet Publish/Subscribe ereignisorientiert. Ein Consumer muss nicht ständig beim Server anfragen, ob neue Daten vorliegen. Der Broker sendet von sich aus neue Daten.

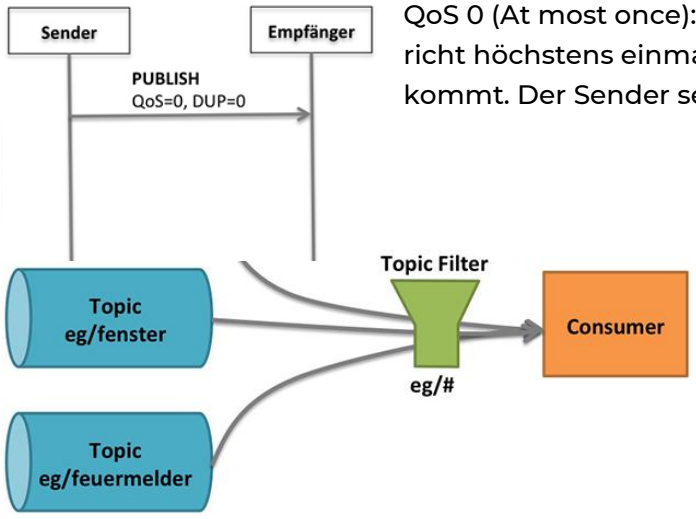
MQTT Broker können in zwei Gruppen aufgeteilt werden: Broker, die speziell für MQTT entwickelt wurden und Broker, die MQTT über einen Konnektor unterstützen.

Native Broker sind wesentlich kleiner als Broker, bei denen MQTT nur ein Protokoll von vielen ist. Einige diese Broker wie z.B. mosquitto eignen für den Betrieb auf einem Mikrokontroller oder Microcomputer.

2.2 Features

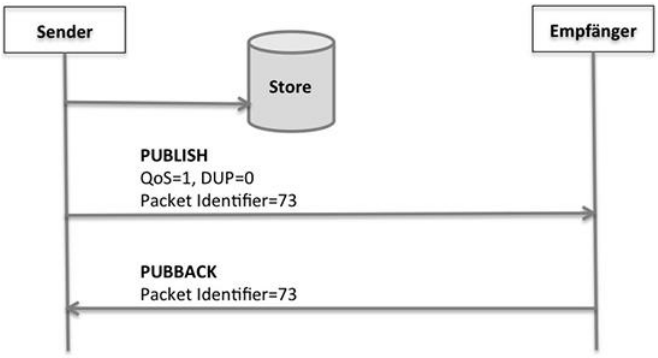
2.2.1 Quality of Service

Quality of Service (QoS) steht für den Grad der Zuverlässigkeit, mit der Nachrichten zugestellt werden. MQTT kennt die drei Stufen 0, 1 und 2, auf deren Einhaltung man sich verlassen kann. Jede Stufe ist ein Kompromiss zwischen Zuverlässigkeit und Ressourcenverbrauch. Der Sender kann bei jeder Nachricht die QoS Stufe angeben.

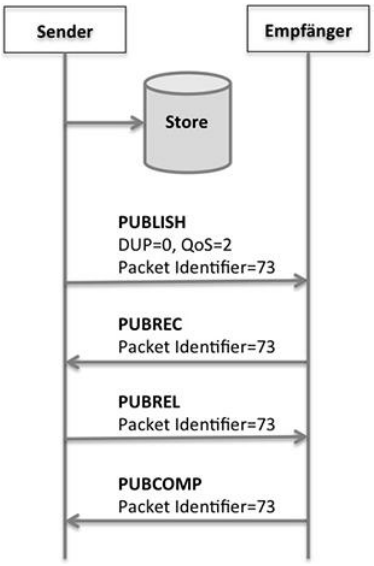


QoS 0 (At most once): Die 0 garantiert lediglich, dass eine Nachricht höchstens einmal ankommt, nicht, dass sie überhaupt ankommt. Der Sender sendet eine PUBLISH Nachricht mit der QoS Stufe 0 und kümmert sich nicht weiter darum. Daher wird dieses Verfahren auch Fire and Forget genannt: schnell und Ressourcen schonend.

QoS 1 (At least once): QoS 1 garantiert, dass eine Nachricht mindestens einmal beim Empfänger eintrifft. Eine Nachricht kann aber auch mehrmals beim Empfänger eintreffen. Der Empfänger muss damit umgehen können, dass Duplikate mehrfach ankommen können. Dafür stehen dem Empfänger Packet Identifier zur Verfügung, die einer Nachricht hinzugefügt werden.



Vor dem Versand einer Nachricht kann der Sender eine Nachricht permanent speichern. Die Nachricht steht dann auch nach einem Ausfall für eine erneute Übertragung zur Verfügung. Der Empfänger quittiert daraufhin den Empfang mit einem PUBACK, das den selben Packet Identifier enthält. Vor dem Empfang des PUBACK Paketes darf der Sender eine Nachricht mit dem Packet Identifier und dem ursprünglichen Inhalt beliebig oft an den Empfänger senden.



QoS 2 (Exactly once): Die höchste Stufe 2 garantiert, dass keine Nachrichten verloren gehen und dass keine Duplikate entstehen. Eine Nachricht kommt exakt einmal beim Empfänger an. Ein zweistufiger Bestätigungsablauf sorgt dafür, dass die Garantien eingehalten werden. Durch die zusätzlichen Nachrichten verzögert sich die Zustellung und wertvolle Bandbreite wird beansprucht.

Der Empfänger antwortet auf den Empfang einer Nachricht mit einem PUBREC. Darauf sendet der Sender ein PUBREL, auf welches der Empfänger mit einem PUBCOMP antwortet.

Die Tabelle zeigt alle QoS Stufen in der Übersicht.

QoS	Name	Beschreibung	Verlust möglich	Duplikate möglich
0	At most once	Nachricht wird genau einmal verschickt. Es gibt keine Antwort und keine Bestätigung vom Server. Ein Retry wird nicht durchgeführt. Schnellste Art des Nachrichtenversandes	ja	nein
1	At least once	Nachricht kann mehrfach verschickt werden. Server antwortet mit einer Bestätigung.	nein	ja
2	Exactly once	Erhöhter Overhead	nein	nein

2.2.2 Last Will Testament (LWT):

Wird die Verbindung vom Publisher zum Broker unterbrochen, kann der Broker stellvertretend für einen Publisher eine voreingestellte Nachricht quasi als dessen letzten Willen versen-

den. Ein Publisher kann so bestimmen, welche Nachricht seine Subscriber nach seiner Verbindungstrennung bekommen. Dabei kann Verbindungstrennung sowohl Verlust der Leitung als auch Zerstörung des Publishers bedeuten.

Im folgenden Beispiel legt der Publisher fest, dass die Nachricht „Ich bin ausgefallen“ an das Topic „Heizung“ gesendet wird, wenn der Broker die Verbindung zum Client verliert.

```
var client = mqtt.connect('mqtt://localhost', {
  clientId : "me",
  will : { topic : "Heizung", payload : "Ich bin ausgefallen!" }
});
```

2.2.3 Message Persistence

Wird die Verbindung zu einem Client unterbrochen, so kann der Server neue Nachrichten für diesen Client für eine spätere Zustellung zwischenspeichern.

2.2.4 Retained Messages

Meldet sich ein Consumer zum ersten Mal für ein Topic an, so bekommt er normalerweise erst eine Nachricht, wenn ein Producer danach eine Nachricht zu dem Topic sendet. Eine Anzeige für die Temperatur würde nach der Verbindung mit einem Topic also so lange keinen Wert anzeigen, bis ein Temperaturfühler einen neuen Wert sendet. Mit Retained Messages bekommt ein Consumer einen letzten Wert (oder mehrere frühere), der vom Broker zwischengespeichert wurden, sofort zugestellt. Die Temperaturanzeige könnte also anstatt keinem Wert zumindest einen älteren Wert anzeigen. Dieses Verfahren wird auch Last Value Queue genannt.

2.2.5 Persistent Sessions

Zwischen Client und Broker kann eine Sitzung so aufgebaut werden, dass ein Consumer abgeschaltet und neu gestartet werden kann und der Consumer seine alte Session wiederaufnimmt. Der Broker kann daraufhin Nachrichten, die der Consumer in der Zwischenzeit verpasst hat, an ihn ausliefern. Eine Session bekommt einen eindeutigen Identifier.

2.2.6 Heartbeats & Keep Alive

Clients können dem Server über die Keep Alive Zeit mitteilen, wie lange eine inaktive Verbindung gehalten werden soll. Wird die Zeit überschritten, kann der Server die Verbindung trennen. Sendet der Client ein PINGREQ Paket, beginnt die Zeitmessung für den Keep Alive von vorne. PINGREQ dient auch dazu zu prüfen, ob der Server und die Netzverbindung noch aktiv sind.

2.2.7 Web Sockets Transport

Web Browser können über MQTT over Web Sockets direkt mit einem Broker eine Verbindung aufbauen. Über diese Verbindung können Push Nachrichten an den Browser geschickt werden.

2.2.8 Performance

MQTT wurde für Netzwerke mit großer Verzögerungszeit und kleiner Bandbreite entwickelt. Besonders kleinste Nachrichten können in kürzester Zeit und großer Zahl ausgetauscht werden. Die notwendigen Header Felder wie QoS oder Flags werden kompakt als Binärwerte gespeichert. Damit eignet sich MQTT besonders für kleinste Nachrichten wie z.B. einzelne Messwerte eines Sensors. Bei größeren Nachrichten ab wenigen Kilobytes kann ein Transport über HTTP mit Komprimierung effizienter sein.

2.2.9 Sicherheit

MQTT selbst unterstützt nur eine Absicherung über Benutzername und Passwort. Die Kommunikation kann mit SSL bzw. TLS auf Transportebene verschlüsselt werden. Broker können zusätzlich Client Zertifikate für die Authentifizierung verwenden oder den Zugriff über Access Control Lists z.B. durch die Überprüfung der IP einschränken.