| Walther-Rathenau-Gewerbeschule Freiburg | **mqtt**<br>MQ Telemetry Transport | Fach: ITS | Gruppe: |
|---|---|---|---|
| | | Dat.:5. April 2019 | Seite **1** |
| | | Name: | |
| | | Klasse: E3FI1 | |
| | | Punkte: /20 | Note: |

# 1 Ein paar einleitende Fragen zu MQTT

- Für was steht die Abkürzung und von welcher Firma wurde MQTT entwickelt?
- Haben die Begriffe hinter den ersten beiden Buchstaben der Abkürzung noch technische Bedeutung?

  In vielen Quellen wird *MQTT* als Abkürzung von Message *Message Queuing Telemetry Transport* beschrieben.

  E-Mail (smtp) basiert z.B. auf *Message Queuing*: Eine E-Mail wird zunächst der Sendewarteschlange hinzugefügt. Unterwegs wird sie auf jedem Zwischenknoten wieder in einer Warteschlange gespeichert bis der Knoten bereit ist, sie weiterzuleiten.

  Hätte man keine *Message Queuing*, müsste man vor dem Senden sicherstellen, dass auf dem gesamten Pfad vom Sender bis zum Empfänger alle Knoten die Mail sofort weiterleiten können.

  *Message Queuing* kann man sich nun so vorstellen, als würden Software-Anwendungen mit einer Art E-Mail-Verfahren untereinander kommunizieren.

  Tatsache ist aber, dass MQTT normalerweise **kein** Queuing verwendet. MQTT wurde 1999 bei IBM von Andy Stanford-Clark und Arlen Nipper entwickelt. Die beiden mussten ein Protokoll programmieren mit dem kleine, batteriebetriebene Geräte an einer Öl-Pipeline via Satellit (sehr kleine Bandbreite, d.h. langsame Datenraten) Telemetriedaten (z.B. Messwerte) austauschen können.

  Als Name des Protokolls wurde MQ Telemetry Transport verwendet, wobei das MQ sich auf das Produkt IBM MQ bezieht.

  Seit 2010 ist MQTT ein offener Standard und kann patent- und lizenzfrei verwendet werden.

  Seither wird die Buchstabenfolge MQTT nicht mehr als Abkürzung betrachtet sondern ist schlicht und einfach der Name des Protokolls.

- Worin unterscheidet sich das Kommunikationsmodell von MQTT von einer klassischen Server-Client-Architektur? Beschreiben Sie das Kommunikationsmodell.

  Bei einer klassischen Server-Client-Architektur kommuniziert der Client direkt mit dem *Endpunkt* der Anwendung. Z.B. bei http *wartet* der Web**server** darauf, dass an Port 80 Verbindungsanfragen von Clients (aka Browser) eintreffen und bedient diese dann.

  Bei MQTT gibt es *keine* direkte Verbindung zwischen den beiden Endpunkten (Z.B. Temperaturmessstelle und Temperaturanzeige).

Stattdessen wird mit einem *Publish/Subscribe*-Muster gearbeitet: die beiden Endpunkte werden dabei durch einen *Broker* (Makler) komplett voneinander entkoppelt und müssen sich nicht mehr kennen.

Aufgabe des Brokers ist es, eingehende Meldungen vom Publisher (Herausgeber) zu filtern und zu speichern und sie korrekt an die Subscriber (Abonnenten) zu verteilen.

Die Entkopplung durch den Broker ist *räumlich*: die Endpunkte müssen ihre jeweiligen IP-Adressen nicht kennen und *zeitlich*: veröffenlichen und abonnieren muss nicht gelichzeitig stattfinden.

Da die Hauptlast der Meldungsverarbeitung beim Broker liegt, können die Endpunkte (Publisher/Subscriber) auch auf Mikrocontrollern mit geringen Resourcen betrieben werden.

- Auf welchem OSI-Layer arbeitet das Protokoll?

  Der Austausch der MQTT-Nachrichten erfolgt mit TCP/IP. Das Protokoll selbst gehört also zur Anwendungsschicht (OSI-Layer 7).

- Welche Servicequalitätsstufen gibt es?

  **Höchstens Einmal** geringste Qualität, der Publisher verschickt seine Daten genau einmal, kümmert sich aber nicht darum, ob Nachrichtenverluste auftreten (fire & forget).

  **Mindestens Einmal** hier ist sichergestellt, dass mindestens eine Nachricht ankommt. Es ist jedoch nicht ausgeschlossen, dass eine Nachricht mehrfach übermittelt wird. Das kann u.U. problematisch sein, wenn z.B. ein Steuerbefehl gesendet wird, der nur einmal ausgeführt werden soll.

  **Genau Einmal** das ist die höchste QoS-Stufe. Die Nachricht kommt genau einmal an.

- Wie ist der feste MQTT-Header aufgebaut? Wie lang ist er?

  Der feste MQTT-Header ist supereinfach: er besteht aus exakt 2 Bytes. Das erste Byte enthält die Schlüsselnummer des Pakettyps (4bit, Beschreibung unten) und einige Flags, das zweite Byte die restliche Länge des MQTT-Pakets.

- Welchen Port verwendet MQTT (dezimal und hexadezimal)?

  | dezimal | hexadezimal |
  |---|---|
  | 1883 | 075B |

# 2 Protokollnachrichten

- Wie ist ein MQTT-Control-Paket aufgebaut? Aus welchen drei Teilen besteht es im Allgemeinen?

| |
|---|
| Fester Header; (2Bytes) in allen MQTT-Paketen vorhanden |
| Variabler Header; in manchen MQTT-Paketen vorhanden |
| Nutzdaten; in manchen MQTT-Paketen vorhanden |

Die variablen Header sind deutlich komplexer aufgebaut als der feste Header und enthalten z.B. zusätzliche Flags, die an dieser Stelle nicht beschrieben werden. Wichtig zu wissen ist aber, dass der variable Header den Topic-Namen als UTF-8 String enthält.

- Welches Format hat der feste Header (Fixed Header)?

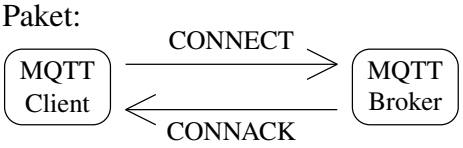
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 1 | Typ des Control Packets | | | | Flags | | | |
| Byte 2 | restliche Länge | | | | | | | |

- Wieviele Protokollnachrichten gibt es?

  Da 4 Bit (Bits 7 bis 4) den Nachrichtentyp codieren, gibt es maximal 16 Möglichkeiten. 2 davon sind reserviert, somit sind es 14 Control-Packet-Typen

- Ein MQTT-Client möchte Messwerte veröffentlichen. Beschreiben Sie den Ablauf der Kommunikation zwischen Client und Broker.

  Wie gesagt: MQTT ist ein einfaches Protokoll: der MQTT-Client (Publisher oder Subscriber), sendet ein CONNECT-Paket, der Broker antwortet mit einem CONNACK-Paket:




# 3 Mitschnitt einer Datenübertragung

Ein mqtt-Paket wurde mitgeschnitten. Die Hexdaten zeigen:

1. den Ethernet-Frame-Header
2. den IP-Header
3. den TCP-Header
4. den MQTT-Header und dessen Daten. Der MQTT-Header beginnt hier mit $1025_{hex}$

```
0x0000:  90e2 ba21 cc8c b827 ebd6 5d9c 0800 4500  ...!...'..]...E.
0x0010:  005b 6136 4000 4006 c450 0a0a 0005 0a0a  .[a6@.@..P......
0x0020:  00fe cbe8 075b bafc 2b5d 76b7 e502 8018  .....[..+]v.....
0x0030:  01c9 b43f 0000 0101 080a bfec 90ee 00aa  ...?............
0x0040:  b411 1025 0006 4d51 4973 6470 0302 003c  ...%..MQIsdp...<
0x0050:  0017 6d6f 7371 7075 622f 3135 3139 382d  ..mosqpub/15198-
0x0060:  7261 7370 6265 7272 79                   raspberry
```

- Kennzeichnen Sie die verschiedenen Header.
- Markieren Sie die beiden verwendeten TCP-Ports.
- Markieren Sie den MQTT-Header

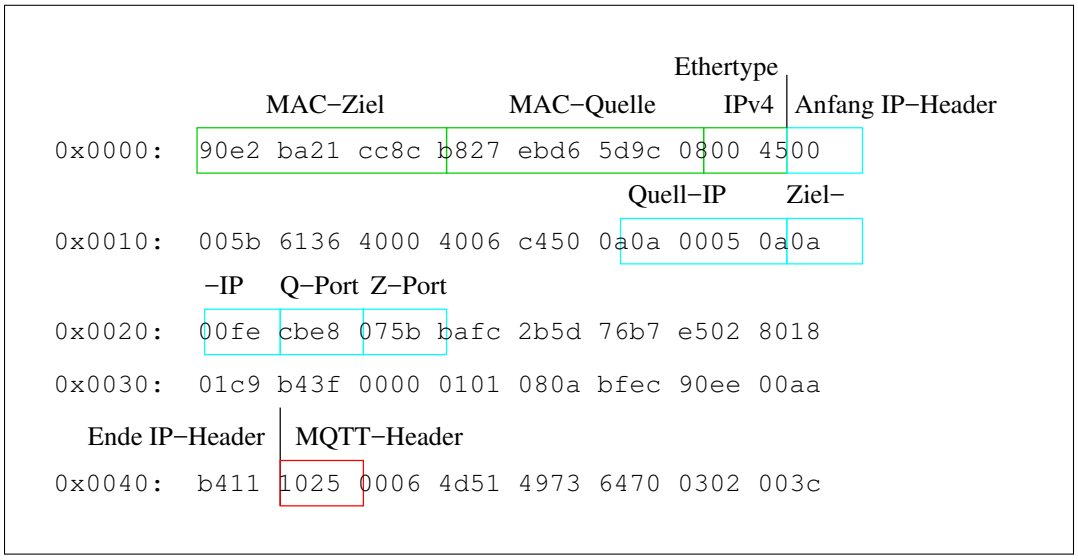



Abbildung 1: Header von Ethernet (grün), IP-Paket (türkis) und MQTT (rot)

Im weiteren Verlauf der Kommunikation werden folgende MQTT-Header identifiziert:

| Hex | Binär |
|---|---|
| 10 25 | 0001 0000 0010 0101 |
| 20 02 | 0010 0000 0000 0010 |
| 30 46 | 0011 0000 0100 0110 |
| 30 41 | 0011 0000 0100 0001 |

- Bestimmen Sie die MQTT Control Packet Types, die Flags und die verbleibende Länge des Pakets.

| Hex | Binär | Control Packet Type | Flags | Restliche Länge/Bytes |
|---|---|---|---|---|
| 10 25 | 0001 0000 0010 0101 | CONNECT | 0000 | 37 |
| 20 02 | 0010 0000 0000 0010 | CONNACK | 0000 | 2 |
| 30 46 | 0011 0000 0100 0110 | PUBLISH | 0000 | 70 |
| 30 41 | 0011 0000 0100 0001 | PUBLISH | 0000 | 65 |

- Welche *Flags* gibt es, welche Werte sind vorgegeben und in welchem Fall werden sie überhaupt ausgewertet?

  **DUP** Duplicate Delivery of Control Packet

  **QoS1 QoS0** Bit 1 und Bit 0 von Quality of Service

  **Retain** Ist dieses Flag gesetzt, speichert der Broker den publizierten Wert als *last known good value*. Ein Subscriber der sich neu anmeldet, muss dann nicht erst auf den nächsten verfügbaren Messwert warten, sondern erhält sofort diesen mit Retain-Flag gesetzten Wert.

  - Bei fast allen Control-Packets sind alle Flags auf 0 gesetzt (Ausnahme:s.u.) und nur beim PUBLISH-Control-Packet, lassen sich die Flags frei wählen.
  - Bei PUBREL (Publish Release), SUBSCRIBE UND UNSUBSCRIBE ist Bit1 gesetzt.

- Welcher *Quality of Service* wird verwendet?

  Da alle Flags im festen Header im mitgeschnittenen Beispiel 0 sind, ist als QoS *höchstens einmal* gesetzt. D.h. Der Publisher verschickt die PUBLISH-Nachricht genau einmal und prüft nicht, ob sie zugestellt wurde.