A. Weisbecker, M. Burmester & A. Schmidt (Hrsg.): Mensch und Computer 2015 Workshopband, Stuttgart: Oldenbourg Wissenschaftsverlag, 2015, S. 217-224.

Sichere Vernetzung von Geräten in Smart Factories mit MQTT

Martin Maritsch, Christian Kittl, Thomas Ebner

evolaris next level GmbH

Zusammenfassung
Smart Factories sind Produktionsstandorte, welche in der Lage sind, sich weitestgehend ohne menschliches Zutun zu organisieren. Dies soll durch verstärkte Kommunikation der Maschinen untereinander ermöglicht werden. Dabei spielt das Internet of Things (IoT), also die Vernetzung von Geräten, eine große Rolle um eine zentrale Datenauswertung durchzuführen. Da mitunter sensible Daten übertragen werden, ist es wichtig, auf eine geeignete Sicherung der Übertragung zu achten. Wir präsentieren das MQTT (Message Queue Telemetry Transport) -Protokoll, welches sich hierfür sehr gut eignet. Neben einigen Features, die für IoT-Anwendungen von Relevanz sind, ist es auch mit vergleichsweise geringem Aufwand möglich, die Datenübertragung zu sichern.

1 Einleitung

Das Internet of Things (IoT) basiert auf der Vernetzung von *Dingen* und der Auswertung der generierten Daten. Speziell im Rahmen von sogenannten Smart Factories (vgl. Westkämper & Jendoubi 2003, Kagermann et al. 2013) an Produktionsstandorten ist die Vernetzung von Produktionsgeräten und –maschinen wichtig: Telemetriedaten von Geräten lassen unter anderem Rückschlüsse auf die Auslastung und den korrekten Gebrauch zu. Des Weiteren lassen sich auch potenziell nötige Wartungsarbeiten vorhersehen, um so die Ausfall- bzw. Standzeit von Geräten so gering wie möglich zu halten.

Für die Vernetzung einer Vielzahl an Geräten ist eine sinnvolle Architektur unerlässlich. Diese soll eine hohe Verbindungsverfügbarkeit gewährleisten und jederzeit einfach um neue Geräte erweitert werden können. Auch die Sicherheit spielt eine nicht zu vernachlässigende Rolle, vor allem dann, wenn Daten von Geräte an verschiedenen Standorten zentral aggregiert werden sollen und diese über unsichere Kanäle – wie insbesondere das Internet – übertragen werden sollen. Im Folgenden geben wir einen Überblick über ein IoT-Protokoll, welches von uns im Rahmen eines Projektes als das für diese Anforderungen am besten geeignete ausgewählt und eingesetzt wurde. Inhalt des Projektes war die Vernetzung eines

Messgeräts, welches im Automotive-Bereich von Motorenentwicklern vor Ort eingesetzt wird, mit dem Messgerätehersteller zum Zweck der laufenden Kontrolle des Betriebsstunden, um rechtzeitige Wartungen automatisiert auslösen zu können.

2 Vergleich verschiedener Protokolle

Es gibt unzählige Protokolle, welche für die Datenübertragung zwischen dislozierten Geräten und einer zentralen Stelle in Frage kommen. In Tabelle 1 ist ein Vergleich zwischen MQTT und 2 der gängigsten Netzwerkprotokoll-Standards dargestellt.

	MQTT	**HTTP**	**Email (SMTP)**
Transport-Protokoll	TCP/IP	TCP/IP	TCP/IP
Sicherheit	optional (TLS)	optional (TLS)	optional (TLS)
Unterscheidung zwischen publizierenden und abonnierenden Geräten	Ja	Nein (muss implementiert werden)	Nein
One to Many bzw. Many to One	Ja	Nein (muss implementiert werden)	Ja
Einfache Integration neuer Geräte und Services	Ja	Nein (muss implementiert werden)	Nein (Muss implementiert werden)
Quality of Service	Ja (At most once, At least once, Exactly once)	Nein (muss implementiert werden)	Nein
Protokollspezifischer Nachrichten-Overhead	2 Byte + variabler Header (~ 3 Byte)	200 – 2000 Byte (typisch 750 Byte)	~ 1000 Byte

Tabelle 1: Vergleich verschiedener Protokolle

3 MQTT

Message Queue Telemetry Transport (MQTT) ist ein Übertragungsprotokoll, welches speziell für Machine-to-Machine Kommunikation (M2M) entwickelt wurde. Aufgrund seines Aufbaus, welcher einen stark reduzierten Overhead verglichen mit anderen Protokollen aufweist, eignet es sich vor allem für die Kommunikation in Bereichen mit eingeschränkten Netzwerken und limitierten Bandbreiten. Des Weiteren unterscheidet sich MQTT von anderen Protokollen durch seine Infrastruktur: Basierend auf einem Publish-Subscribe-Modell (siehe Abbildung 1) werden Nachrichten zwischen Sendern und Empfängern über eine zentrale Einheit geleitet: den Message Broker.

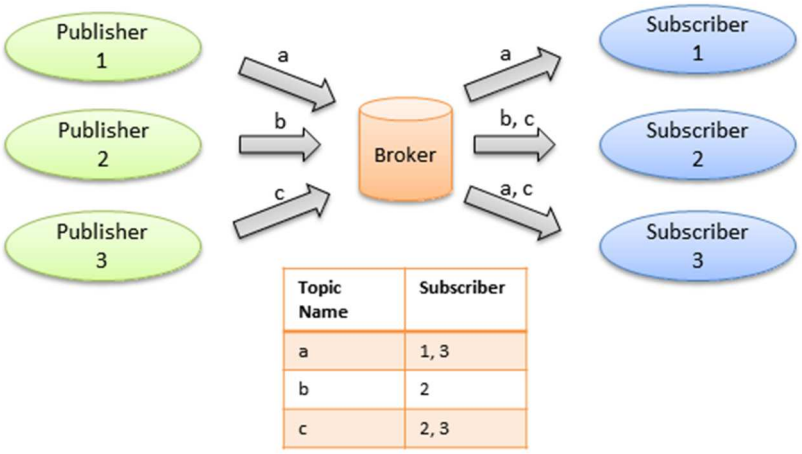


Abbildung 1: Publish-Subscribe Modell (eigene Darstellung)

Dabei ist es auch möglich eine Hierarchie an Message Brokern aufzubauen (Bridging). So kann beispielsweise an jedem Produktionsstandort ein Broker installiert werden, welcher als Bridge zwischen Geräten und zentralem Broker dient.

In den folgenden Kapiteln werden die für unsere Anwendung relevantesten Konzepte von MQTT beschrieben.

3.1 Topics

Die Kategorisierung der Nachrichten wird durch das Konzept von Topics ermöglicht. Diese bieten die Möglichkeit, Nachrichten in Hierarchien (ähnlich wie in einem Forum) abzulegen. So bietet sich für Empfänger die Möglichkeit, auszusuchen, welche Nachrichten er erhalten möchte. Dabei sind auch Wildcards in den Topic-Namen erlaubt (+: Single Level, #: Multi Level)

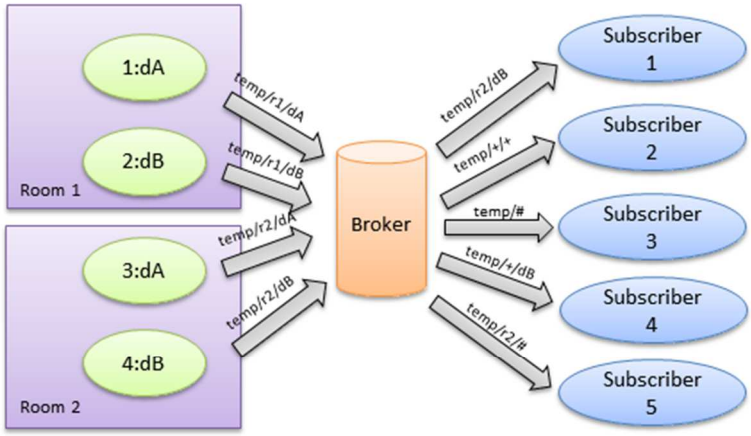
Die Topic-Hierarchie kann beispielsweise benutzt werden, um Publisher in verschiedene Kategorien einzuteilen.

In unserem Beispiel gibt es 2 Räume mit jeweils 2 verschiedenen Geräten, die ihre Temperatur im MQTT-Netzwerk in verschiedene Topics publizieren:

- *temperature/room1/deviceA*
- *temperature/room1/deviceB*
- *temperature/room2/deviceA*
- *temperature/room2/device*

Die Topics *roomX* bzw. *deviceX* bieten nun umfassende Möglichkeiten, die gewünschten Daten zu erhalten (siehe auch Abbildung 2):

- *temperature/room1/deviceB* Daten von Gerät B aus Raum 1
- *temperature/+/+* bzw. *temperature/#* Daten von allen Geräten aus allen Räumen
- *temperature/room2/+* Daten von allen Geräten aus Raum 2
- *temperature/+/deviceA* Daten von allen Geräten mit dem Namen A



Subscriber	Topic Filter	Nachrichten von Geräte ID
1	temp/r2/dB	4
2	temp/+/+	1, 2, 3, 4
3	temp/#	1, 2, 3, 4
4	temp/+/dB	2, 4
5	temp/room2/+	3, 4

Abbildung 2: Verschiedene Subscriber-Möglichkeiten (eigene Darstellung)

3.2 Quality of Service

Ein weiteres Feature von MQTT ist Quality of Service (QoS). Jede Nachricht kann mit einem QoS versehen werden. Je nach QoS ist eine unterschiedliche Behandlung der Nachricht betreffend der Zustellung vorgesehen (siehe Tabelle 2).

QoS Level	Zustellung	Zustellgarantie
0	max. 1x	Keine Garantie; „Best Effort“
1	min. 1x	Garantierte Zustellung; Duplikate möglich
2	genau 1x	Garantierte Zustellung; keine Duplikate möglich

Tabelle 2: Quality of Service

QoS 0 entspricht dem *Fire & Forget*-Prinzip: Sobald eine Nachricht das Gerät verlassen hat, wird sie verworfen. Im Gegensatz dazu wird bei QoS 1 & 2 auf eine Bestätigung gewartet,

dass die Nachricht (mindestens) ein Mal zugestellt wurde. Erst danach darf die Nachricht vom Sender verworfen werden. Höhere QoS-Levels bedeuten mehr Netzwerk-Traffic und höhere Latenzzeiten, da pro Nachricht mehrere Kontroll- und Bestätigungs-pakete zwischen den Entitäten in der Architektur ausgetauscht werden müssen. Nachdem MQTT als Leichtgewichts-Protokoll spezifiziert ist und nicht alle Anwendungen eine garantierte Zustellung benötigen, hat man die Wahl des QoS dem Anwender überlassen.

3.3 Last Will

Ein weiteres für IoT relevantes Konzept, welches von MQTT angeboten wird, ist *Last Will*. Ein Subscriber kann dem MQTT Broker beim Herstellen der Verbindung seinen *Letzten Willen* mitteilen. Dabei handelt es sich um eine MQTT-Nachricht und ein Topic, welche vom MQTT-Broker bei Abbruch der Verbindung mit dem Gerät veröffentlicht wird. Der Abbruch der Verbindung wird durch ein Ausbleiben der Pings eines Gerätes über einen definierten Zeitraum detektiert. Somit kann zum Beispiel ein Ausfall von Geräten in der Infrastruktur in Echtzeit überwacht werden.

4 Sicherheit

Die Sicherheit in Netzwerken spielt in vielen Anwendungen eine entscheidende Rolle und muss bereits bei der Planung des Systems berücksichtigt werden. Vor allem wenn Daten über unsichere Kanäle wie das Internet übertragen werden, ist eine angemessene Verschlüsselung, etwa mittels TLS, unumgänglich.

4.1 TLS

Transport Layer Security (TLS) bietet eine gute und einfache Möglichkeit, die Übertragung von MQTT-Nachrichten zwischen Geräten und dem zentralen Broker zu sichern.

Da TLS nichts mit MQTT direkt zu tun hat sondern im OSI-Modell eine Stufe darunter liegt (Transportebene) wird an dieser Stelle nur kurz darauf eingegangen.

TLS sollte aus folgenden Gründen verwendet werden:

1. Es bietet eine sichere Übertragung über potenziell unsichere Kanäle (z.B. Internet)
2. Wenn Client-Authentifizierung mittels Zertifikaten verwendet wird, kann der Broker gegenüber den Geräten seine Authentizität bezeugen. Umgekehrt kann der Broker die Authentizität der Geräte überprüfen.
3. Client-Zertifikate können auch dazu verwendet werden, die Integrität von Geräten sicherzustellen. Aufgrund des Zertifikates kann der Zugriff von Geräten auf bestimmte Topics vom Broker verweigert werden (z.B. welches Gerät darf in welches Topic publizieren)

5 Praxis

Lightweight-Protokolle wie MQTT gewinnen – mit dem Boom des IoT – zunehmend an Bedeutung, da es gilt, eine Vielzahl von Geräten einfach und schnell zu vernetzen. Auch Facebook hat zur energiesparenden und schnellen Nachrichtenübertragung für den *Facebook Messenger* MQTT verwendet (Zhang 2011).

Folgende Vorteile wurden von uns bei Anwendungen mit MQTT im Vergleich zu den am meist verbreiteten Netzwerkprotokollen (HTTP, SMTP) beobachtet:

- Strom-/Batterieeinsparungen auf Client-Geräten (vgl. Nicholas 2012)
- Bandbreiteneinsparungen
- Viele gleichzeitige Verbindungen auf einem Broker möglich (in Größenordnungen von mehreren 100.000, zum Vergleich schafft ein Apache2-Server standardmäßig nur 150 gleichzeitige Verbindungen[1])
- Kürzere Latenzzeiten

In Tests konnten wir auch über relativ unstabile Verbindungen (Gerät in den USA mit lokaler Mobilfunkanbindung über das Internet zu unserem MQTT-Broker in Graz verbunden) durchaus akzeptable Latenzzeiten von unter 500 Millisekunden erreichen. Die durchschnittliche lokale Latenz hat für Nachrichten mit verschiedenen Payload-Größen in unseren Tests im Schnitt 58 Millisekunden betragen. Dabei wurde der Weg vom Publisher über den Broker bis zum Subscriber inklusive TLS Verschlüsselung gemessen. Das MQTT-Protokoll spezifiziert auch eine Authentifizierung mittels Benutzername und Passwort, deren Durchführung dem jeweiligen MQTT-Broker obliegt. Diese ist natürlich nur über einen sicheren Kanal (z.B. Intranet) bzw. gesicherten Kanal (z.B. TLS über Internet) sinnvoll. Da unsere Architektur generell mit TLS-geschützten Verbindungen arbeitet, haben wir uns entschlossen, die Clients mit TLS-Zertifikaten auszustatten, welche deren Authentizität gegenüber dem Broker gewährleistet. Somit wäre eine zusätzliche Benutzerkennung mittels Benutzername-Passwort-Paar redundant.

Für MQTT sind bereits über 40 Client-Bibliotheken verfügbar, welche eine einfache Integration in bestehende Lösungen in verschiedensten Programmiersprachen gewährleisten.

In unserer Architektur verwenden wir einen Java-Service, welcher am Broker ein Topic abonniert, die dort publizierten Telemetriedaten parst und anschließend in einer Datenbank abgelegt. Je nach Inhalt der Telemetriedaten werden Workflows getriggert, welche z.B. bei kritischen Werten automatisch Alarm-SMS versenden bzw. Mitarbeiter zur Nachforschung auffordern (siehe Abbildung 3).

[1] vgl. http://stackoverflow.com/a/13373244 (abgerufen am 19.06.2015)

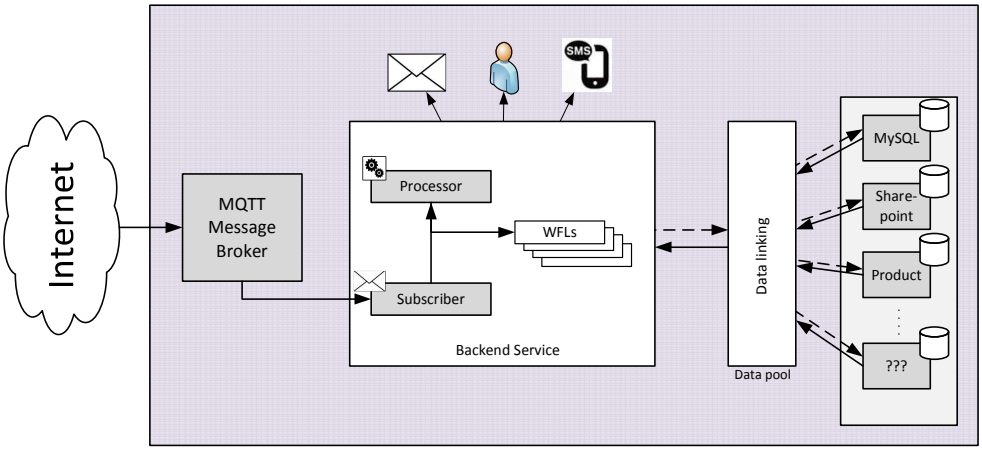


Abbildung 3: Architektur eines Telemetriesystems mit automatischer Datenverarbeitung mittels MQTT Message Broker und nachgelagerten Workflows (WFLs)

6 Schlussteil

MQTT bietet eine einfache Möglichkeit, eine große Anzahl an Geräten für das IoT miteinander zu vernetzen. Auf Grund seiner Eigenschaften ist es auch für Geräte in eingeschränkten Umgebungen, sei es nun die Stromversorgung oder Internetanbindung, sinnvoll zu verwenden. Das Protokoll eignet sich für verschiedenste Einsatzszenarien und bietet auf Grund seiner Features speziell für das IoT klare Vorteile gegenüber den weit verbreiteten Netzwerkprotokollen wie HTTP oder SMTP und wird aufgrund der rasch voranschreitenden Verbreitung von IoT-Vernetzung und der gennannten Vorteile künftig eine sehr weite Verbreitung finden..

Literaturverzeichnis

Kagermann, H., Walster, W. & Helbig, J. (Hrsg.) (2013). *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0. Deutschlands Zukunft als Produktionsstandort sichern; Abschlussbericht des Arbeitskreises Industrie 4.0*, Forschungsunion Wirtschaft - Wissenschaft, Promotorengruppe Kommunikation; Deutsche Akademie der Technikwissenschaften e.V. –acatech, http://forschungsunion.de/pdf/industrie_4_0_abschlussbericht.pdf, abgerufen am 17.6.2015

Nicholas, S. (2012). Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android, http://stephendnicholas.com/archives/1217, abgerufen am 16.6.2015

Westkämper, E. & Jendoubi, L. (2003). Smart Factories - Manufacturing Environments and Systems of the Future. In Bley, H. (Hrsg.): *Progress in Virtual Manufacturing Systems*. Proceedings. 36th CIRP International Seminar on Manufacturing Systems, June 03-05, 2003, Saarbrücken: S. 13-16

Zhang, L. (2011) *Building Facebook Messenger*. https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920, abgerufen am 17.6.2015

Kontaktinformationen

evolaris next level GmbH
Hugo-Wolf-Gasse 8-8a, A-8010 Graz
Martin.Maritsch@evolaris.net
www.evolaris.net