Site officiel : http://mqtt.org/ **Crédits images :** The Things Network, Wikipedia, hivemq

Le protocole **MQTT (Message Queuing Telemetry Transport)** est un protocole de messagerie de type **publication/souscription** s'appuyant sur le protocole TCP/IP.
MQTT permet à deux équipements distants de communiquer via des messages avec une faible bande passante, donc une faible charge en données et une faible consommation.
En raison de cela, ce protocole est spécialement dédié au monde du **M2M (machine to machine) et aux objets connectés**

MQTT est porté sur Arduino, STM32, les nano-ordinateurs LINUX de type Raspberry pi ou sur PC

Les clients sont connectés au monde physique (capteurs / actionneurs)
Le **Broker** ou **serveur MQTT** concentre les données issues des clients.

Des librairies MQTT sont disponibles pour la conception de clients MQTT dans les langages comme **C, C++, Java, C#, Python etc..**

Avantages de MQTT

Flexibilité

MQTT est sur la couche "session" au-dessus de la couche réseau TCP/IP sur le modèle OSI, utilisée par les protocoles comme HTTP. MQTT est donc routable sur Internet. Il est possible de transmettre n'importe quel message sur les topics (sujets), de l'ASCII, du binaire ou du JSON.

Légèreté

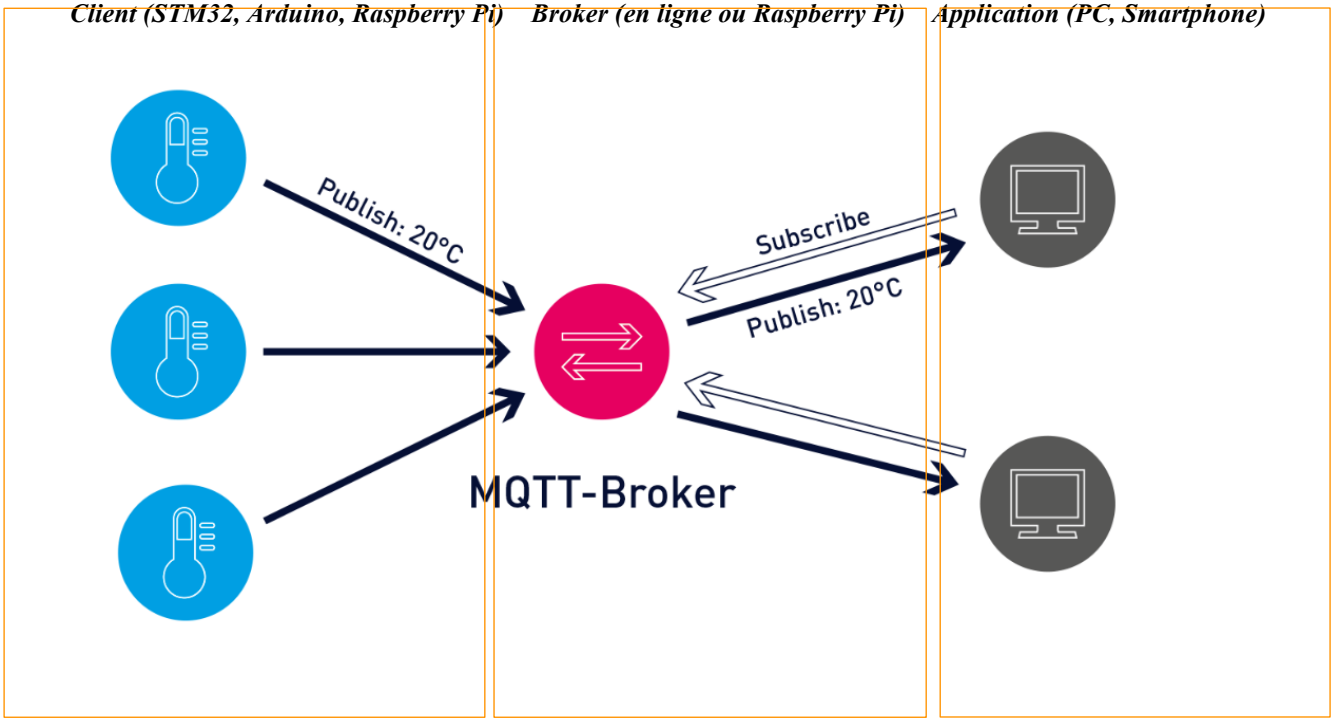
Les échanges MQTT sont beaucoup plus légers que HTTP.

Sécurité

La sécurisation est possible avec un chiffrement SSL/TLS, mais cela a un coût non négligeable en termes de performances sur de l'embarqué.
MQTT sécurisé ne peut être déployé sans consommation excessive qu'avec du matériel puissant ce qui exclut les Arduino 8bits par exemple.

Intégrité des données

MQTT introduit la notion de *qualité de service* (QOS) qui permet à un client de s'assurer qu'un message a bien été transmis, avec différents niveaux de fiabilité.

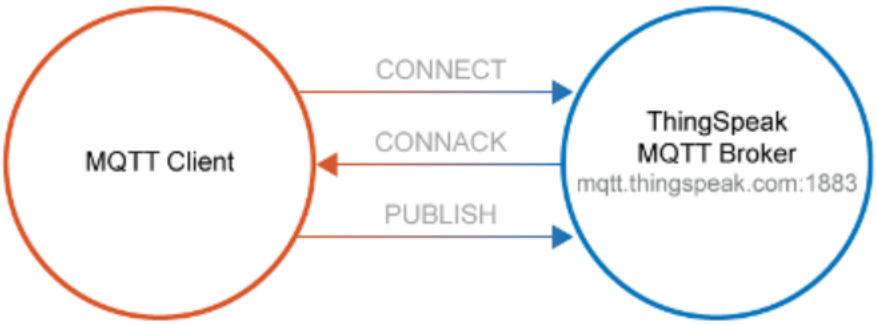


Échanges MQTT

Une session MQTT est divisée en quatre étapes : connexion, authentification, communication et terminaison.

Un client commence par créer une connexion TCP/IP vers le broker.
Les ports standards sont : 1883 pour la communication non chiffrée et 8883 pour la communication chiffrée utilisant SSL/TLS.

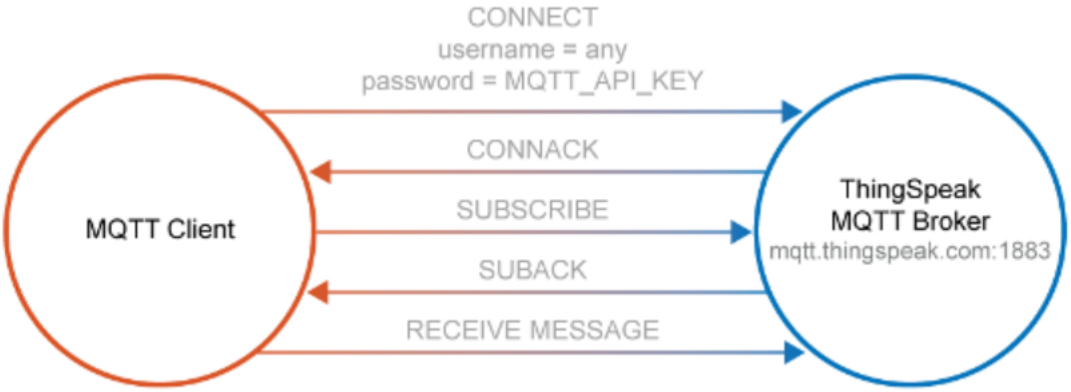
Publication MQTT



Ici le client possède des capteurs et transmets leurs valeurs
Le client se connecte au broker avec son identifiant et mot de passe (CONNECT)
Le broker accepte la connexion (CONNACK)
Le client peut publier des données (PUBLISH)

- Les données sont regroupées dans des TOPICS, (ex température, pression, humidité , GPS, etc...)

Souscription MQTT



Ici le client va recevoir les messages transmis au broker par les capteurs afin de les exploiter (prise de décision, affichage, etc...)
Le client se connecte au broker avec son identifiant et mot de passe (CONNECT)
Le broker accepte la connexion (CONNACK)
Le client souscrit à des TOPICS (ex température, pression, humidité , GPS ... (SUBSCRIBE)
A chaque publication le broker transmet les données des TOPICS au client. (SUBACK)
Un acquittement peut être transmis (RECEIVE MESSAGE)

Quality Of Service (QOS)

Envoyer et oublier (QoS 0)

Le client ne recevra aucune confirmation du broker à la réception. De même, un message remis à un client par le broker ne sera pas acquitté. C'est le moyen le plus rapide de publier et de recevoir des messages, mais aussi celui où la perte de messages est la plus susceptible de se produire.

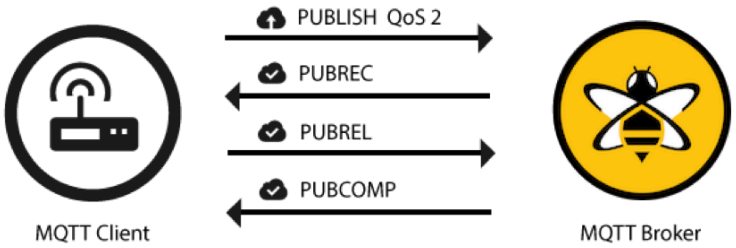
Au moins une fois (QoS 1)

Chaque message du client vers le broker ou du broker vers le client doit être acquitté. Si l'acquittement n'arrive pas dans le temps imparti, le message est réexpédié.

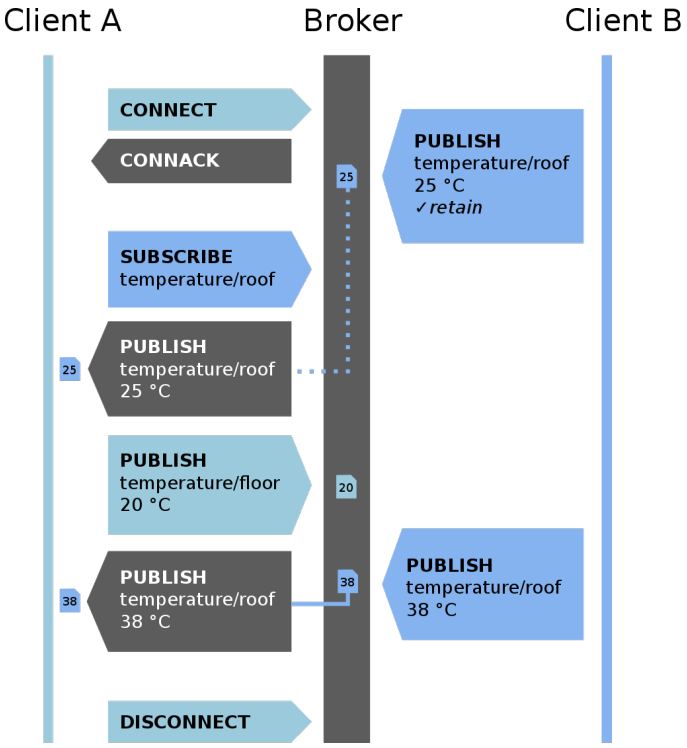
Exactement une fois (QoS 2)

QoS 2 est le niveau de service le plus élevé de MQTT. Ce niveau garantit que chaque message n'est reçu qu'une seule fois par les destinataires prévus. QoS 2 est le niveau de qualité de service le plus sûr et le plus lent. La garantie est fournie par au moins deux flux de demande / réponse (une négociation en quatre parties) entre l'expéditeur et le récepteur. L'expéditeur et le destinataire utilisent l'identifiant de paquet du message PUBLISH d'origine pour coordonner la remise du message.

Lorsqu'un récepteur obtient un paquet QoS 2 PUBLISH d'un expéditeur, il traite le message de publication en conséquence et répond à l'expéditeur avec un paquet PUBREC qui reconnaît le paquet PUBLISH. Si l'expéditeur n'obtient pas de paquet PUBREC du récepteur, il envoie à nouveau le paquet PUBLISH avec un drapeau dupliqué (DUP) jusqu'à ce qu'il reçoive un accusé de réception.



Exemple de séquence d'échanges



MQTT et modèle OSI

MQTT se situe sur la couche 5 du modèle OSI.

Dans la couche 6 le message est codé. (binaire, JSON, ASCII ou autre) et transmis à la couche 5 (MQTT) qui se charge du protocole. Les données MQTT sont alors encapsulées dans une trama TCP-IP (couche 4)

Layer	Dénomination	Protocole	Description
7	Application		Acquisition de données/contrôle processus
6	présentation	Binaire, JSON, ASCII	Codage du message
5	Session	MQTT	Message IOT
4	Transport	TCP	Transmission Control Protocol
3	Réseau	IP	Internet Protocol
2	Liaison	802.11	Adressage MAC
1	Physique	802.11	Ondes radio (Wi-Fi)

Exemple de message MQTT

Pour une description exhaustive du protocole MQTT : https://openlabpro.com/guide/mqtt-packet-format/

Control Header	Packet Lenght	Variable length header	Payload
Type de message	Longueur du message	Dépend du type de message	données
1 octet	1 à 4 octets	0 à x octets	0 à 256 MOctets

La taille du message dépend du type de message :

2 octets : Control header + Lenght : par exemple "CONNACK" pour un acquittement de connexion
3 octets : Control header + Lenght + Variable header : par exemple "PUBACK" pour un acquittement de publication
n octets : Control header + Lenght + Variable header+ Payload (données) : par exemple "CONNECT" avec ID et pass.

La taille minimum d'un paquet est 127 octets, Paket length contient alors un octet codé sur 7bits.
La taille maximum d'un paquet est de 256 Moctets.

Exemple de message MQTT : Ouverture de session (connexion) avec l'ID : PYTHON1

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Meaning	Header	Remaining Length	Length of prtocol name		Protocol Name +Version					Connect Flags	Keep Alive		Length								
Hex	0x10	0x13	0x0	0x4	0x4d	0x51	0x54	0x54	0x4	0x2	0x0	0x3c	0x0	0x7	0x70	0x79	0x74	0x68	0x6F	6E	0x31
Ascii		19		4	M	Q	T	T	4			60		7	P	Y	T	H	O	N	1

0x10 : CONNECT
Le message contient 0x13 soit 19 octets
0x00 puis 0x04 puis le nom du protocole qui est MQTT puis 0x04
0x02 pour Connect flag
0x3C durée de validité du message en secondes
0x07 est la taille du payload (message)

Connect flag:D7 : nom utilisateurD6 : mot de passeD5 : RETAIND4-D3 QoSD2 : FlafD1 : réinit sessionD0 : réservé

Crédits images : The Things Network, Wikipedia, hivemq