MQTT : MQ Telemetry/Transport
Un protocole pour l'IoT

Ph. Truillet

Octobre 2019 – v. 1.4

1. MQTT : comment ça marche ?

MQTT est un protocole ouvert, simple, léger et facile à mettre en œuvre. Ce protocole est idéal pour répondre aux besoins suivants :

- l'utilisation d'une très faible bande passante,
- l'utilisation sur les réseaux sans fils,
- la consommation en énergie,
- la rapidité avec un temps de réponse supérieur aux autres standards du web actuel,
- la fiabilité,
- et l'usage de faibles ressources processeurs et de mémoire.

MQTT est un standard ISO (**ISO/IEC PRF 20922 :2016**[1]). Au départ, **TT** (premier nom du protocole) a été développé par Andy Stanford-Clark (IBM) et Arlen Nipper (Eurotech, actuellement Cirrus Link) en 1999 pour le contrôle d'un pipeline dans le désert. L'objectif était d'avoir un protocole de bande passante efficace utilisant peu d'énergie à un moment où les périphériques ne pouvaient être connectés qu'au travers de satellites.

Le protocole MQTT utilise une architecture « *publish/subscribe* » en contraste avec le protocole HTTP et son architecture « *request/response* » (cf. Figure 1).

Le point central de la communication est le broker MQTT (qui par défaut utilise le port 1883 pour les connexions non chiffrées et le port 8883 pour celles chiffrées en SSL/TLS), en charge de relayer les messages des émetteurs vers les clients. Chaque client s'abonne via un message vers le broker : le « topic » (sorte d'information de routage pour le broker) qui permettra au broker de réémettre les messages reçus des producteurs de données vers les clients. Les clients et les producteurs n'ont ainsi pas à se connaître, ne communiquant qu'au travers des topics. Cette architecture permet des solutions multi-échelles.

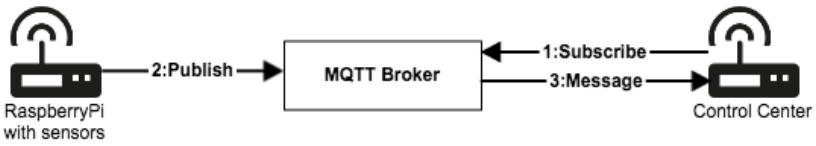


Figure 1 : Architecture MQTT "Publish/Subscribe"

Chaque client MQTT a une connexion ouverte en permanence avec le broker. Si la connexion s'arrête, le broker bufférise les messages et les émet dès que la reconnexion avec le client est effectuée.

Un « **topic MQTT** » est une chaîne de caractères qui peut posséder une hiérarchie de niveaux séparés par le caractère « / ». Par exemple, une information de température du salon pourrait être envoyée sur le topic « **maison/salon/temperature** » et la température de la cuisine sur « **maison/cuisine/temperature** ».

[1] https://www.iso.org/standard/69466.html

Nota : Les topics doivent au moins contenir un caractère (ne pas commencer par le caractère « $ », réservé à un usage interne de MQTT), sont sensibles à la casse et le topic « / » est valide !

Wildcards

- Le signe « **+** » est un caractère « *wildcard* » qui permet des valeurs arbitraires pour un niveau en particulier

Ex : « **maison/+/temperature** » permet de s'abonner aux températures de la cuisine et du salon

- le signe « **#** » pour plus d'un niveau (ce signe doit se trouver à la fin). Les messages envoyés peuvent être de toutes sortes mais ne peuvent excéder une taille de 256 Mo.

Ex : si un *publisher* s'abonne au *topic* « **maison/salon/#** », il recevra toutes les données provenant du salon. De même, s'il s'abonne au *topic* « **maison/#** », il collectera toutes les données des sondes de la maison.

2. Sécurité

Les données IoT échangées peuvent s'avérer très critiques, c'est pourquoi il est aussi possible de sécuriser les échanges à plusieurs niveaux :

- Transport en SSL/TLS,
- Authentification par certificats SSL/TLS,
- Authentification par login/mot de passe.

3. QoS -Qualité de Service

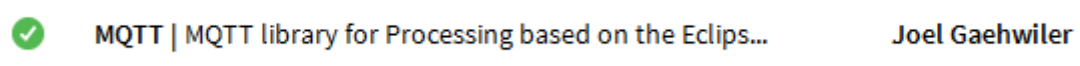
MQTT intègre en natif la notion de QoS. En effet le *publisher* à la possibilité de définir la qualité de son message. Trois niveaux sont possibles :

- Un message de **QoS niveau 0** « *At most once* » sera délivré tout au plus une fois. Ce qui signifie que le message est envoyé sans garantie de réception, (le broker n'informe pas l'expéditeur qu'il l'a reçu et le message)
- Un message de **QoS niveau 1** « *At least once* » sera livré au moins une fois. Le client transmettra plusieurs fois s'il le faut jusqu'à ce que le Broker lui confirme qu'il a était transmis sur le réseau.
- Un message de **QoS niveau 2** « *exactly once* » sera obligatoirement sauvegardé par l'émetteur et le transmettra toujours tant que le récepteur ne confirme pas son envoi sur le réseau. La principale différence étant que l'émetteur utilise une phase de reconnaissance plus sophistiquée avec le broker pour éviter une duplication des messages (plus lent mais plus sûr).

4. Exercices de chauffe

4.1 Un premier exemple avec Processing.org

Ouvrez Processing, installez la librairie MQTT (Menu « **Outils** » | « **Ajouter un outil** », onglet « **Libraries** » MQTT.



Ouvrez l'exemple installé avec la librairie MQTT (Menu « **Fichier** » | « **Exemples…** »,| « **Contributed Libraries** » | « **MQTT** », exemple *PublishSubscribe*.

Ouvrez maintenant dans un navigateur le site **shiftr.io** à l'adresse **https://shiftr.io/try** (cf. Figure 2)

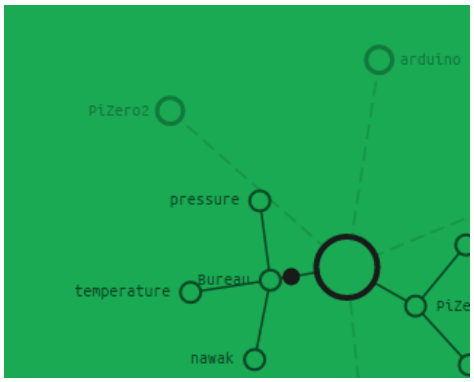


Figure 2 : Architecture MQTT "Publish/Subscribe"

Ecrivez tout d'abord un programme Processing qui génère des valeurs aléatoires de température toutes les secondes et les envoie au broker MQTT (**vous devez déterminer le topic**). Chaque instance lancée sera considérée comme une « *pièce* » de la maison)

Ecrivez ensuite un programme qui devra s'abonner à l'ensemble des températures émises par chacune des instances et afficher la moyenne générale des températures de la « maison » (générée donc par chacune des pièces) et les moyennes par « pièce ».

4.2 Python (Paho)

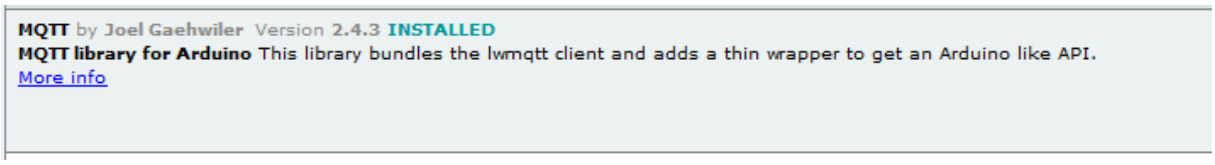
MQTT est multi-langage et on peut utiliser python comme support. Pour utiliser MQTT sous Python, allez voir la documentation ici : **https://pypi.python.org/pypi/paho-mqtt**

Ecrivez le même programme émetteur de données en Python.

4.3 Arduino

De la même manière, il est possible de générer des messages MQTT depuis un périphérique compatible arduino connecté à TCP/IP (réseau filaire ou WiFi)

Il faut dans un premier temps récupérer la librairie MQTT en faisant « **Croquis** | inclure une bibliothèque | Gérer les **bibliothèques** » et chercher la bibliothèque **MQTT** (Arduino 1.8.7 et ultérieur)



Vous trouverez dans la section « *Exemples* » du code permettant d'émettre des données depuis arduino directement sur MQTT.

Nota : *Par manque de périphériques arduino de type ESP8266, vous pouvez sauter cette étape.*

4.4 Un broker MQTT

Le broker MQTT le plus connu reste **Mosquitto** (géré par la fondation **eclipse**)

Vous pouvez télécharger **Mosquitto** ici : **https://www.eclipse.org/mosquitto/download**

Pour cause de dépendances multiples, téléchargez la version sous Windows disponible ici : **https://github.com/truillet/upssitech/blob/master/SRI/3A/ID/TP/Outils/mosquitto_portable.zip**

Il faut ensuite extraire le contenu du fichier dans un répertoire et lancer **mosquitto.exe**. Votre broker est désormais disponible et peut être utilisé directement en local (**mqtt://localhost**)

5. Exercice

Il s'agit maintenant de créer et de gérer un réseau de capteurs à partir de de données « **réelles** » locales et à distance (en utilisant par exemple des données récupérées via une **API REST** comme **OpenWeatherMap**). Pour ce faire :

1. créez à partir d'**arduinos** et de capteurs de luminosité, température des sondes … et générez des valeurs vers le broker MQTT.
2. écrivez un programme tiers capable de lire des données du broker et agréger les valeurs des différents capteurs positionnés dans l'environnement.
3. proposez une visualisation (type évolution des valeurs dans le temps) de vos données