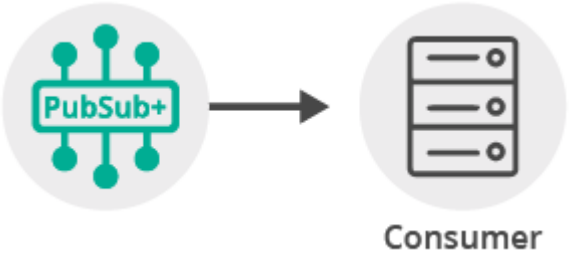
Publish/Subscribe

[Mise à jour le 20/12/2021]

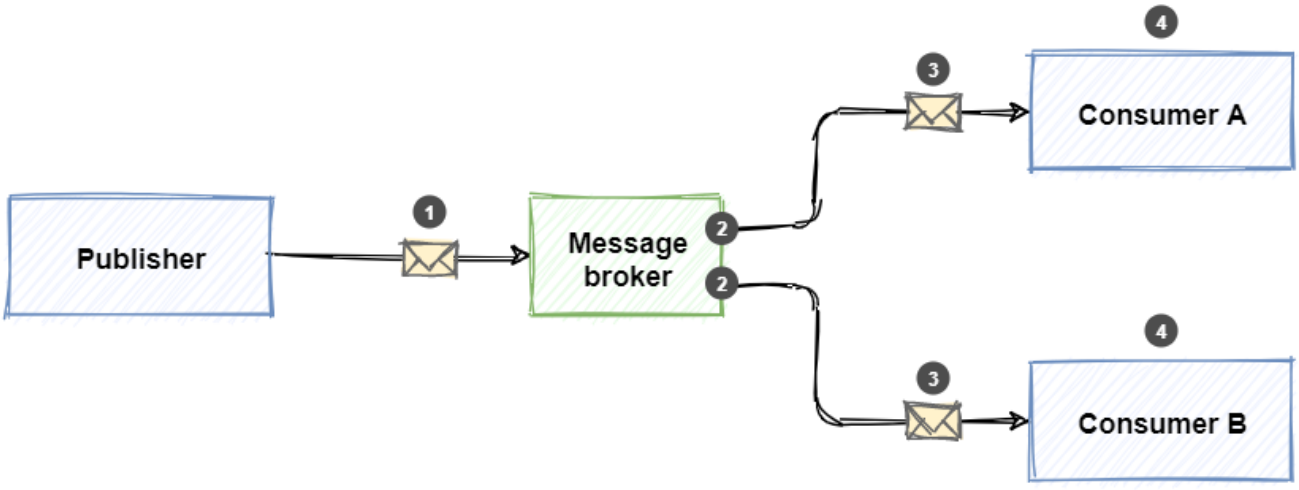


- **Source** : Mooc Fun “Programmer l'internet des objets”
- **Lecture connexe** : Raspberry Pi - Installer un broker MQTT Mosquitto

1. Présentation

Il existe d’autres formalismes que REST. Un autre formalisme, très populaire, est orienté “diffusion” en utilisant le principe ”**publier/abonner**” ou ”**publish/subscribe**”. Comme nous allons le montrer dans la suite, même si les fonctionnalités entre ces deux modes peuvent sembler similaires, la philosophie de conception est très différente : publish/subscribe vise des applications intégrées tandis que REST vise l’interopérabilité globale.

Le modèle publish/subscribe fait le découplage entre l’expéditeur d’un message et son destinataire. Dans ce paradigme (cf. figure ci-dessous), il existe des ”**Publishers**” qui produisent des données ou des messages et envoient le message à une entité généralement appelée ”**Broker**”. En outre, les messages peuvent être classés en ”**Topics**”, contenus ou types, etc. Ensuite, il existe des abonnés qui souscrivent au broker, par exemple à un topic donné, afin de recevoir les messages qui les intéressent, comme montré dans le schéma.

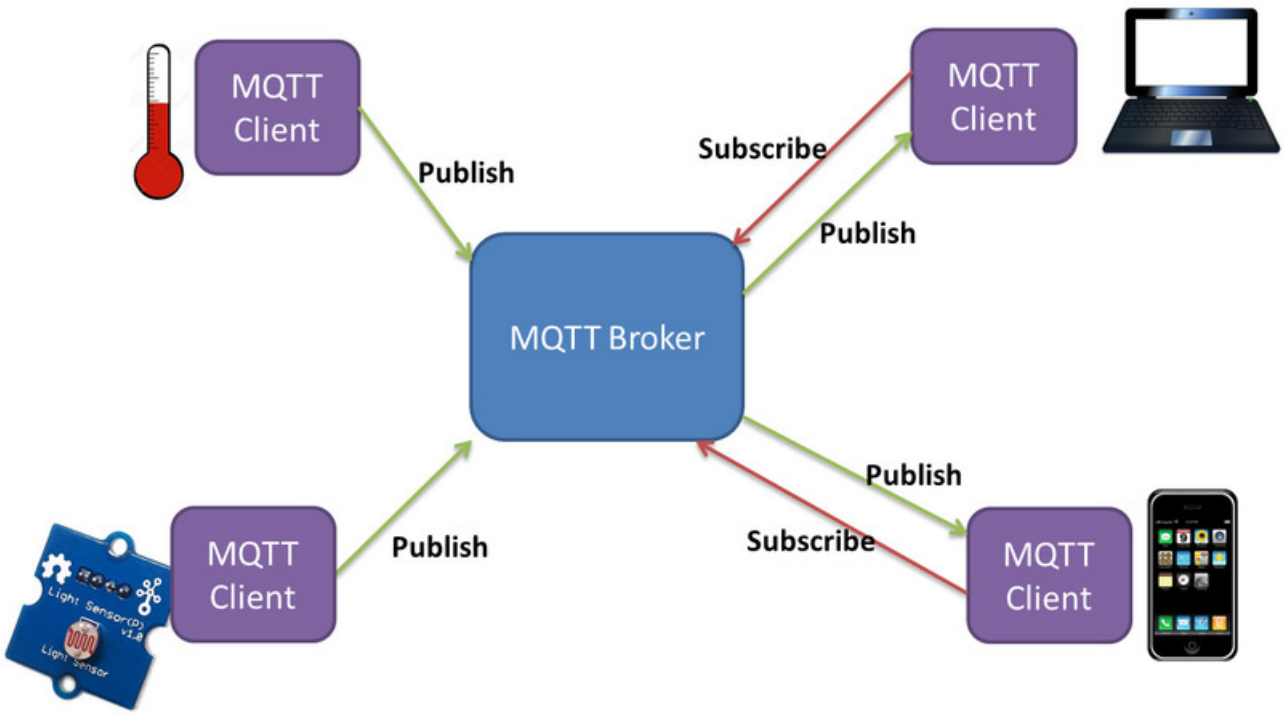


1. Les éditeurs envoient des messages au courtier de messages.
2. Les abonnés sont liés à un abonnement sur le courtier de messages.

3. Le courtier de messages transfère une copie du message aux abonnements intéressés.
4. Les abonnés utilisent des messages de leurs abonnements.

Le broker peut alors utiliser des filtres pour envoyer uniquement ces messages aux abonnés du topic concerné. Il existe plusieurs protocoles Publish-Subscribe tels que **MQTT** (**Message Queuing Telemetry Transport**), AMQP (Advanced Message Queuing Protocol), JMS (Java Messaging Service) ou XMPP (Extensible Messaging Protocol et Présence).

Imaginons par exemple que plusieurs capteurs soient installés dans deux bâtiments A et B. Certains capteurs collectent des informations sur la température et d'autres collectent des informations sur l'humidité. Ces capteurs peuvent envoyer les données régulièrement à un broker central.



Les données peuvent être classées en différentes rubriques qui peuvent également être organisées de manière hiérarchique. Par exemple, le **topic** /sensor signifie "toutes les données de capteurs", /sensor/buildingA/ signifie "des données de capteurs uniquement installées dans le bâtiment A". En plus, /sensor/buildingA/temperature pourrait signifier "des données de capteurs de température installés uniquement dans le bâtiment A".

Certains abonnés peuvent s'abonner aux messages en fonction de leur intérêt. Ainsi, un abonné intéressé uniquement par les données d'humidité du bâtiment B peut s'abonner au sujet /sensor/buildingB/humidity et le broker n'enverra que ces données à cet abonné.

2. Client/Serveur versus Publish/Subscribe

Les principaux avantages du paradigme publish-subscribe par rapport au paradigme client-serveur, tels qu'inclus en REST, sont les suivants :

- **faible couplage** entre émetteur et récepteur, le broker sert d'intermédiaire et stocke les informations ;
- **passage à l'échelle**. Les données provenant d'une source ne sont émises qu'une fois par la

source. Le broker les recopie vers tous les abonnés. Dans un mode client/serveur, les données doivent être émises par le serveur autant que fois que les clients le demandent.

L'absence de couplage entre l'expéditeur et le destinataire se fait en termes d'espace, de temps et de synchronisation. Celui qui publie les données a une tâche simplifiée. Il n'a pas à gérer ou connaître ceux qui les consomment, il n'a qu'à les envoyer au broker.

MQTT est très léger et conçu pour les périphériques de faibles puissances. Il a une très petite empreinte logicielle et est optimisé pour fonctionner dans les environnements à faible bande passante. Cela rend **MQTT idéal pour les applications IoT**. Malgré tout, l'usage de TCP et des très nombreux acquittements peut s'avérer lourds pour les équipements ou les réseaux très contraints. Une version plus légère basée sur UDP existe pour ces cas d'usage, mais elle est peu utilisée.

S'ils se ressemblent, les principes de nommage des topics MQTT et des URI REST sont complètement différents. Par rapport à MQTT, le chemin dans l'URI n'a pas de sémantique. Il a juste vocation à être unique. Il ne peut pas être utilisé pour agréger plusieurs sources d'information. Si deux capteurs publient respectivement sur les topics /sensor/buildingA/temperature et /sensor/buildingB/temperature, un subscriber peut s'abonner au topic /sensor/*/temperature pour recevoir toutes les mesures ; ce qui est impossible avec REST : il faudra autant de requêtes que de capteurs pour récupérer l'ensemble des mesures.

Les URI sont simplement uniques au monde par leur construction alors que les topics du MQTT sont spécifiques à une application. Un topic MQTT peut être interprété différemment par deux applications différentes. Cela ne permet pas une interopérabilité sémantique. Les abonnés doivent être construits avec une connaissance des topics utilisés par les publieurs.

2. Client/Serveur versus Publish/Subscribe