**Les protocoles populaires du modèles publish/subscribes**

1. **Introduction** :

Le monde des applications communicants à travers un réseau est en plein mutation et tend vers des écosystèmes reposant sur l’Open source car leur implémentation pénètre de plus en plus les entreprises vu les solutions propriétaires couteuses. Cette communication consiste en une transmission de message entre divers systèmes avec utilisation d’un protocole de messagerie. Les protocoles de messageries utilisables définies les accords entres les points de terminaison (nœuds), la qualité des services et les formats de données.

Cependant, certain modèle de communication tel que publish/subscribe offrent une topologie de réseau dynamique, une communication orienté contenu et prennent en compte les contraintes (fréquentes de connexions, nombres de système consommateur) ce qui le permet d’être utilisables par les applications telles que Internet des Objets (IoT), etc. Les protocoles les plus populaires utilisant ce modèle sont : AMQP, MQTT, PSync, NDN-Lite.

1. **Le protocole AMQP**
2. **Définition**:

AMQP (Advanced Message Queuing Protocol) est un protocole réseau ouvert (open source) pour la messagerie. Il agit au niveau de la couche application autorisant un échange de message entre deux parties avec utilisation d’une file d’attente (Queuing). Il est standard et peut être utilisé pour la mise en place d’une topologie d’échange de type publication/abonnement.

1. **Les objectifs** :

L’objectif de ce protocole est de mettre en place un serveur de messagerie orienté message en utilisant la fille d’attente et l’interopérabilité pour permettre une production/consommation sans un problème de langage de programmation voire de système d’exploitation

1. **Entités de communications** :

Une communication AMQP est un transfert de message pair-à-pair entre les nœuds d’un réseau. Elle est constituée d’un ensemble de conteneur qui peut contenir plusieurs nœuds. Un conteneur peut être un broker ou application cliente. Un nœud peut être un producteur ou un consommateur responsable d’un stockage ou d’une livraison de message.

Une communication entre deux conteneurs nécessite un établissement de connexion. Les messages envoyés lors d’une connexion sont fragmentés en un ou plusieurs trames. Une connexion est alors constituée de l’envoie d’une séquence de trame ordonnée car la réception de la dernière trame implique que les première trames sont déjà reçues.

Une communication AMQP est basé sur :

1. **Trame** :

Une trame est l’unité de base échangé de AMQP et chaque trame est composée de trois parties distinctes :

Header

Body

Extended header

* Frame header (8 bits) : il est de taille fixe et contient la version protocole AMQP utilisé.
* Extended header (taille variable) : Le traitement de cette zone dépend du type de trame car il contient des données supplémentaires.
* Frame body (taille variable) : c’est le corps de la trame, il contient les données à transmettre et peut avoir différentes formes telles que : *open/close* (la trame d’établissement ou de fin de connexion), *begin/end* (établissement ou finalisation d’une session), *attach/dettach* (), *transfert* (pour envoyer un message), *disposition* (utilisé pour informer le récepteur des changements d’état de livraison). Lors de la livraison d’une trame, elle est reçue dans l’un des cinq états différents qui sont :
* Accepted : indique le succès du traitement du récepteur.
* Rejected : indique l’échec du traitement du message
* Released : indique message non traité
* Modified : un message non traité et non modifié

1. **Source ou producteur** : est un nœud qui crée et envoie les messages.
2. **Cible ou consommateur** : est un nœud qui récupère et traite les messages.
3. **File d’attente (queue)** : est la destination ciblée par broker pour le stockage des messages. Elle peut être plusieurs, chacune ayant un identifiant unique qui la différencie des autres et est réalisée à l’aide d’une mémoire.
4. **Message broker** : il permet de distribuer les messages aux différentes files d’attentes après leurs créations. Un message est stocké dans une file d’attente à condition que les règles de comparaison entre l’identifiant de la file d’attente de celui du message soient respecté. Ces règles appelés ***exchanges*** sont aux nombres de quatre qui sont : ***Direct exchange***, ***Fanout exchange***, ***Topic exchange*** et ***Headers exchange***.

Queue1

**Exchange**

Broker

Queue2

Queue3

1. **Fonctionnement** :

AMQP est un protocole de communication entre les applications à travers un réseau. Il permet une communication asynchrone car l’émetteur et le destinateur n’ont pas à agir au même rythme et le destinateur n’est pas non plus obligé de traiter l’information et d’en accuser la réception. Il s’appuie sur le message broker pour la distribution des messages aux différentes destinations. Les messages étant stockés dans la file d’attente, le producteur peut ainsi travailler sans avoir de période d’inactivité.

Pour établir une connexion l’une des pairs doit envoyer une frame header contenant sa version de protocole préféré à l’autre pair et l’autre pair (cas d’un serveur il doit répondre avec une version inférieure ou égale à celle du client) doit répondre avec sa version de protocole supporté.

Les messages sont distribués par le message *broker* dans les différentes files d’attentes en utilisant les identifiant de ces files et les *bounding* des messages.

Exchange

Consommateur

Queue

Producteur

Broker

Queue

Queue

**Exemple d’application du protocole AMQP** :

Cet exemple met en place un producteur, un consommateur et un serveur RabbitMQ avec le langage Java en localhost.

**RabbitMQ** : est un server (ici broker et gestionnaire de queues) localhost de simulation du protocole AMQP qui écoute par défaut sur le port 15672 pour les versions 3.x et pour la manipulation de ce serveur à travers une interface utilisateur, nous avons activés les plugins de gestions en saisissant dans le terminal RabbitMQ : « rabbitmq-plugins enable rabbitmq\_management ».

Publie Consomme

**Producteur**

**Consommateur**

**RabbitMQ**

Vérifie

Dans cet exemple nous avons utilisés rabbitmq-server-3.8.16.exe, Erlang (otp\_win64\_23.3.exe) et Eclipse (nécessite l’importation des deux librairies amqp-client.jar et slf4j-simple.jar).

Apres avoir lancer le programme producteur (**Prod**), il produit un message et ce message sera stocké dans la queue définie et visible dans **RabbitMQ**. Le programme consommateur (**Cons**) va ensuite vérifier la présence d’un message dans la même queue et le consomme (affiche le message trouvé) s’il y en a.

**NB** : Une exécution continue de **Prod** produit plusieurs messages qui seront stockés dans la même queue.

1. **Qualité de service** :

AMPQ est efficace, portable, multiport et sécurisé qui propose l'authentification et le chiffrement via SASL ou TLS sur les trames TCP. Il fonctionne bien dans les environnements multi client et obéit à des règles strictes, garantissant l'interopérabilité des clients de différents fournisseurs. Ce protocole livre le message de trois manière grâce au principe de réglage automatique des deux entités de communication. Ces trois manières sont :

* ***at-most-once*** : un seul envoi au plus car l’expéditeur peut ne pas envoyer à cause de l’oublie de livraison du massage et l’envoie peut être manqué;
* ***at-least-once*** : l’expéditeur peut envoyer le message et se stabilise (oublie l'état associé à l’étiquette de livraison), il peut oublier qu’il avait envoyé ce message et le renvoie ce qui implique la possibilité de duplication chez le destinateur ;
* ***exactly-once*** : les deux entités se sont stabilisés avant l’expédition/réception ce qui garantie une livraison unique.

1. **Avantage** :

AMQP est un protocole qui échange des messages chiffrés et assure l’interopérabilités entre les applications car il n’impose pas aux producteurs et consommateurs de comprendre le même langage de programmation. Il utilise l’architecture publication/abonnement utilisé dans les systèmes de messageries ainsi que pour la gestion des périphérique IoT**.**

1. **Inconvénient :**

Ce protocole n’est pas compatible à ses versions précédentes et utilise beaucoup de bande passante.

1. **Sécurité**:

* AMQP est protocole d’encodage binaire qui utilise deux protocoles de sécurisation (TLS, SASL) des couches, utilisable après la sélection. Chacun de ces deux protocoles consiste en quelque échange de frame header suivies des négociations de TLS ou de SASL. Cette négociation est un échange de certificat de sécurité pour les échanges sur réseau informatique.

1. **Le protocole MQTT**
2. **Définition**

MQTT est un protocole léger, ouvert et simple pour transport de messagerie de publication / abonnement Client Server. Il est conçu pour être facile à mettre en œuvre et est utilisable dans des environnements contraints tels que l’internet des objets. Il est basé sur TCP/IP, idéal pour la transmission des données utilisant une faible bande passante et est à faible consommation énergétique.

1. **Objectif** :
2. **Fonctionnement** :
3. **Qualité de service** :
4. **Sécurité** :

[**https://www.lemagit.fr/definition/Advanced-Message-Queuing-Protocol-AMQP**](https://www.lemagit.fr/definition/Advanced-Message-Queuing-Protocol-AMQP)

[**https://www.ionos.fr/digitalguide/sites-internet/developpement-web/advanced-message-queuing-protocol-amqp/**](https://www.ionos.fr/digitalguide/sites-internet/developpement-web/advanced-message-queuing-protocol-amqp/)

[**https://blog.eleven-labs.com/fr/rabbitmq-partie-1-les-bases/**](https://blog.eleven-labs.com/fr/rabbitmq-partie-1-les-bases/)