Diane RICTER

Damien ISSANCHOU

CEI – CS : Choix des protocoles et leurs implémentations

Documentation

1. Choix du Protocole

Les deux protocoles retenus sont AMQP et MQTT.

Nous avons d'emblée **éliminé les protocoles WMI et TAXII** des solutions possibles puisqu'ils présentent respectivement les inconvénients suivants :

- WMI et son implémentation sont propriétaires et soumis à la License Windows
- La documentation de TAXII étant très incomplète, tout un mode de fonctionnement de TAXII est simplement introduit mais non détaillé. Or ce mode de fonctionnement (*publish/subscribe*) est celui qui présente le plus d'intérêt pour notre cas d'étude. Il nous a donc semblé improbable de pouvoir créer un standard basé sur un protocole avec une documentation lacunaire.

Dans un second temps, nous avons décidé de **conserver MQTT**. En effet, ce protocole remplit les critères de comparaison de notre tableau tout en étant orienté IoTs. Il est donc plus léger que les AMQP et Kafka ce qui le rend propice aux utilisations avec les IoTs. Néanmoins, de part sa légèreté, MQTT ne permet pas des fonctionnalités telles que l'utilisation de queues, l'utilisation de Kerberos ou le multiplexage qui sont utiles sur les sondes classiques. Nous avons donc décidé d'utiliser AMQP ou Kafka sur celles-ci.

Enfin, une étude produite par les Nokia Bell Labs comparant AMQP et Kafka nous a conduit à privilégier AMQP pour notre cas d'utilisation.

- AMQP présente de meilleures performances dans la situation « At Least Once » i.e. lorsque l'on veut être sûre que le message est reçu par chacun des consommateurs concernés.
- AMQP garantit que l'ordre d'envoi des messages est conservé
- AMQP permet un routage complexe des messages

Nous avons donc fait le choix d'utiliser les protocoles AMQP et MQTT.

2. Choix de l'Implémentation

- ### Implémentation AMQP

Nous avons trouvé, dans le domaine libre, deux implémentations majeures du protocole AMQP : **RabbitMQ** et **ActiveMQ**. Elles proposent toutes les deux un *broker* et des librairies clientes.

RabbitMQ se révèle être l'implémentation de référence, car la plus utilisée à ce jour. Elle est disponible sous licence permissive (*Mozilla Public Licence*), et pour diverses plateformes (*Debian, Ubuntu, CentOS, RHEL, Fedora, Mac OS, UNIX génériques, Windows*) sous forme de paquets et/ou de code source. Un de ses aspects intéressants est qu'elle possède des performances supérieures aux autres implémentations d'AMQP, ainsi que de nombreuses extensions. La librairie client est bien documentée et intuitive ; nous nous tournerons vers le *binding Python* (2.7 et 3.4) proposé par RabbitMQ. De plus, cette implémentation gère aussi le protocole MQTT.

Apache propose son implémentation concurrente ActiveMQ, sous licence permissive également (*Apache Licence 2.0*). Cependant, elle s'avère moins utilisée, et présente quelques aspects peu attirants. Effectivement, son *broker* est développé en *Java*, offrant donc de moins bonnes performances, et sa librairie cliente semble moins simple d'utilisation.

- ### Implémentation MQTT

Un premier candidat implémentant le protocole MQTT est *RabbitMQ*. L'avantage principal est la simplicité de raisonnement et de maintenance qui en découle. Toutefois, dans la mesure où MQTT vise les IoT, les performances fournies par la librairie cliente *Python* peuvent s'avérer insuffisantes. Le problème ne se pose pas quant à la réutilisation de cette implémentation comme *broker* MQTT. ActiveMQ supporte également MQTT, mais nous y retrouvons les défauts mentionnés dans la partie précédente.

Ainsi, pour ce qui est des libraires clientes, nous nous tournons vers le projet Eclipse Paho, sous licence permissive *Eclipse Public Licence 1.0*). Le projet est relativement récent (première version sortie en 2014), mais brille par son activité. Il offre des librairies clientes entre autres en C avec librairies POSIX et/ou Windows, et C "embarqué", c'est-à-dire strictement conforme à la norme ANSI pour une meilleure portabilité. Une libraire *Python* est aussi disponible, envisageable pour les clients non embarquées (Manager, etc...)