MQTT vs. WebSub in Tapas

Group 3[1]

[1]Advanced Software and Systems Engineering

Universität St.Gallen

MQTT

Publish/subscribe protocol running mostly over TCP/IP, which consists of message brokers and clients. Brokers receive messages from the clients and route them to appropriate destination clients. Publishers send new data to the broker, which then distributes the data to any clients subscribed to that topic. Usually, there is a central broker, where topics are published.

Advantages

- Each client can both publish and subscribe (bi-directional)
- Allows for persistent sessions
- Very scalable
- Manages and tracks all client connection states
- Reduced impact on network (less bandwidth consumption)
- Works well for constrained devices
- Different QoS measures leading to high reliability:
  - At most once
  - At least once
  - Exactly once
- Loose coupling
- Workflow decoupling
- Allows for central control

Disadvantages

- No inherent security layer (self-defined if necessary)
- Central brokerage possibly introduces a single point of failure (if no cluster is used)
- Content format not predictive

WebSub

Protocol to implement a publish/subscribe pattern over HTTP callbacks (webhooks). No ongoing connection or session is necessary to receive any publishes. The ecosystem consists of publishers, subscribers and hubs. Subscribers running a web accessible server subscribe to topics and get directly notified whenever subscribed topics have updated. Publishers updated their topics on the hubs, which subsequently call the subscribers.

Advantages

- Includes a verification of intent mechanism
- Includes a validation mechanism
- Loose coupling - Subscribers don't need to know anything about the architecture or configurations of the hub or the publisher
- Very scalable
- Workflow decoupling
- Discovery and subscriptions at runtime
- Decentralized distribution of hubs

Disadvantages

- Content format not predictive - implies agreement
- Topic discovery more complex due to decentralized nature

Context

There currently is no need to care about security issues concerning the tapas application. Thus, any kind of verification of intent or validation mechanism does not seem to be a priority as of now. Furthermore, the number of publishers and subscribers is limited to the number of groups in class, making any kind of thoughts on scalability redundant. MQTT with its central broker, where topics are stored, should potentially serve as a easier alternative to implement concerning topic discovery, as there is only one topic everyone subscribes to on a central broker everyone knows. However, this implies some coupling of services as the accessing of the central broker is hardcoded. Having a central broker would potentially introduce a single point of failure. However, as HiveMQ provides clusters this should not be considered a problem for our applications.

In the case of WebSub, the topic discovery is done by distributing two URLs, namely for the hub itself and the topic. Thus, each group has to define its hub and topic URL, which is then provided to all groups. Using either crawling mechanisms and information sharing across groups or directly publishing one's hub and topics, the discovery of new hubs and topics can happen at run time. This greatly benefits the decentralization and scaling of the network. This benefit although comes with the cost of increased complexity.

In the case of the tapas application MQTT provides an easy-to-use option, which is setup rather fast. While this seems enticing, it comes at the cost of reduced flexibility compared to WebSub. Hub and topic discovery at run time further loosens coupling of services and topic discovery can be freely steered by all groups. Consequently, there is for example an advantage of being free to only subscribe to relevant auctions, which benefits all groups. Further, new agencies or topics - if they would exist - could be discovered at run time. Mimicking such a behavior in MQTT to the full extent is not possible. Thus, in terms of efficiency, extensibility and scaling we would favor WebSub as an implementation.