MQTT++

Evelyn G. Coronel

Programación Orientada a Objetos
Instituto Balseiro

(12 de diciembre de 2020)

I. MQTT SIMPLIFICADO EN C++

En este trabajo se basó en la implementación del protocolo de MQTT utilizando los temas de la materia.

A. Librería `mqtt.hpp`

Esta librería están almacenadas las variables reservadas para el protocolo, dentro del `namespace mqtt`. También se implementó una clase que asigna valores de identificación (ID) únicos a los clientes.

B. Librería `mqtt_errors.hpp`

C. Librería `mqtt_message.hpp`

D. Librería `mqtt_client.hpp`

En esta librería están implementadas las clases utilizada para instanciar objetos que se comporten con un cliente. La misma está dentro del `namespace mqtt_client`.

1. Clase `mqtt_client::client_virtual`

En esta clase esta implementada de tal forma que la misma y sus hijos no puedan copiarse. También se declaran funciones virtuales puras para que luego van a ser implementadas en las clases hijas.

2. Clase `mqtt_client::client`

Esta clase es heredada de la clase `mqtt_client::client_virtual`. Se asigna un ID al azar, usando `mqtt_client::get_id()` y `mqtt_client::set_id(<unsigned int>)` cuando se conecta al broker que transmite los mensajes

Un cliente puede estar conectado a un solo server ya que al conectarse/desconectarse la variable `mqtt_client::Connected` cambia `mqtt::CONNECTED` / `mqtt::DISCONNECTED`. El estado del cliente se obtiene mediante la función `mqtt_client::isConnected()`.

Un cliente puede estar suscrito a varios topics al mismo tiempo, la función `mqtt_client::client::subscribe(<string>)` agrega un nuevo topic a una lista miembro de la clase `mqtt_client::client`, la función `mqtt_client::client::get_topic()` devuelve dicha lista.

El cliente solo puede estar conectado a un servidor al mismo tiempo. Tiene la opción de hacer que sus mensajes siempre vayan por defecto, al inicio de la cola de mensajes con `mqtt_client::set_Priority(mqtt::HIGH)`.

La función `mqtt_client::client::reply()` debe ser implementada por cada tipo de cliente, ya que cada uno va a tener una forma de responder distinta.

E. Librería `mqtt_server.hpp`

F. Librería `mqtt_broker.hpp`

La librería esta con el `namespace mqtt_broker`

1. Clase `mqtt_broker`

Esta clase es una clase hija de `mqtt\_server::server` Como atributos la clase tiene una lista con las direcciones de los clientes conectados al broker, un generador de IDs, además de un `mutex` y un `condition_variable` para manejar la lista de clientes y la cola de mensajes.

Los métodos de esta clase incluyen a:

- `sin_subs()`: Devuelve `False` si no hay clientes conectados, `True` caso contrario.
- `publish_from(<cliente *, mensaje*>)`

  Publica un cliente a los demás clientes mediante el servidor.
- `publish(<mensaje*>)`

  Publica el servidor a todos los clientes
- `connect(<cliente *, string>)`

  Conecta un cliente al servidor
- `disconnect(<cliente *>)`

  Desconecta un cliente del servidor.

- `broadcast_message()`

  La función que publica los mensajes a todos los clientes.

G. Ejemplos