SDLE 2022/2023 - 1st Semester
Assignment 1 -- Reliable Pub/Sub Service

1. Introduction

In this project, you shall design and implement a reliable publish-subscribe service.

2. Specification

Essentially, this service offers two simple operations:

`put()`
: to publish a message on a topic

`get()`
: to consume a message from a topic

Messages are arbitrary byte sequences.

In addition, the service should support the following operations:

`subscribe()`
: to subscribe a topic

`unsubscribe()`
: to unsubscribe a topic

Topics are identified by an arbitrary string.

Topics are created implicitly when a subscriber subscribes to a topic that does not exist yet.

All subscriptions are **durable**, according to the Java Message Service's (Jakarta Messaging) terminology. I.e. a topic's subscriber should get all messages put on a topic, as long as it calls `get()` enough times, until it explicitly unsubscribes the topic. This should be guaranteed even if the subscriber runs intermittently. I.e. a subscription is independent of the lifetime of the process that makes that subscription.

The publish-subscribe service should guarantee "exactly-once" delivery, in the presence of communication failures or process crashes, except in "rare circumstances", i.e. only rarely and only under some failure conditions. Note that processes may crash and recover. Furthermore, processes may be unreachable for some time because of network problems, but eventually the network heals. I.e. the time intervals during which a process is unreachable have a finite length.

By "exactly-once" delivery guarantees we mean the following:

1. on successful return from `put()` on a topic, the service guarantees that the message will eventually be delivered "to all subscribers of that topic", as long as the subscribers keep calling `get()`
2. on successful return from `get()` on a topic, the service guarantees that the same message will not be returned again on a later call to `get()` by that subscriber

In order to be able to satisfy these requirements subscribers must have an id, which can be an arbitrary string.

Concurrent invocations of `put()` and `subscribe()/unsubscribe()` make it hard to define "all subscribers of that topic" in the first guarantee. We will not try to specify rigorously what these subscribers should be. However, the idea is that in a single server implementation of the service, if the server has successfully processed a subscription request on a topic by a subscriber before it processes a put request on the same topic by a publisher, and afterwards that subscriber calls `get()` enough times and does not unsubscribe that topic, one of the `get()` calls will return the message published as an argument of that `put()`.

3. Constraints

Your service must be implemented on top of libzmq, a minimalist message oriented library. You can use any language for which libzmq has a binding.

This project must be carried out in groups of 4, exceptionally 3 students. All group members must be registered in the same lab section (turma).

4. What and how to submit?

By October 21, at 20:00, you must submit your code and a plain ASCII file named `README` with instructions for compiling and running your application via FEUP's Git Service (GitLab@FEUP) (on a project we will create and assign you).

Furthermore, you must submit a report, a PDF file, also via Gitlab@FEUP. The report shall cover the design as well implementation aspects, and possible tradeoffs, with a focus on the algorithms you designed to ensure the "exactly-once" guarantees above under the failure model described above. In particular, you must describe the "rare circumstances" under which you implementation does not satisfy these guarantees.

5. Demo

You will have to demo your project in a lab class.