RabbitMQ Utilisation

Lorgnier Théo / Demoulin Raphaël

https://github.com/745th/RabbitMQ

Ping-Pong :

Our algorithm work with channels not cleaned, but you must run 2 instances to clean all the old messages.

```
rabbitmqctl purgue_queue A
rabbitmqctl purgue_queue B
```

We've tested locally on our computers and on the rabbitmq as a service, to test if we can communicate far away.
The code works the same locally or online. We just need to change the setHost into setURI.

We have 2 queues: A and B. The queue A is the default one, both of the clients connect here to consume the remaining messages, and they go to channel B to not consume their own message.

Algorithm view in the course:

1st rule : req? <START> && st=IDLE ⟶ st=WAITING ; out!<init(ID())>

2nd rule : in? <init(id)> && id>ID() ⟶ st=START ; out!<OK_INIT>

3rd rule : in? <init(id)> && id<ID() ⟶ st=WAIT ; out!<init(ID())>

4th rule : in? <OK_INIT> ⟶ st=START ; out!<PING>

5th rule : in? <PING> ⟶ out!<PONG>

6th rule : in? <PONG> ⟶ out!<PING>

Our Algorithm:

1st rule : req? <START> && st=IDLE ⟶ st=WAITING ; out!<"START" +ID> ;out!<"START" +ID>

2nd rule : in? <"START" + sID> && sID > ID() ⟶ st=WAITING ; out!<"SECOND" + ID>

3rd rule : in? <"START" + sID> && sID < ID() ⟶ st=WAITING ; out!<"FIRST" + ID>

4th rule : in? <"SECOND" + sID> ⟶ st=START ;

5th rule : in? <"FIRST" + sID> ⟶ st=START ; out!<"PING" + ID>

6th rule : in? <"PING" + sID> ⟶ out!<"PONG" + ID>

7th rule : in? <"PONG" + sID> ⟶ out!<"PING" + ID>

We can see differences in the 2 algorithms due to the implementation in rabbitmq requiring management of the different queues.
The messages SECOND and FIRST are used to decide if a client consume on A and publish on B or consume on B and publish on A
This intermediate management required the addition of another rule to put the channel in the right state when the ping-pong begin.