M111-BIG DATA MANAGEMENT

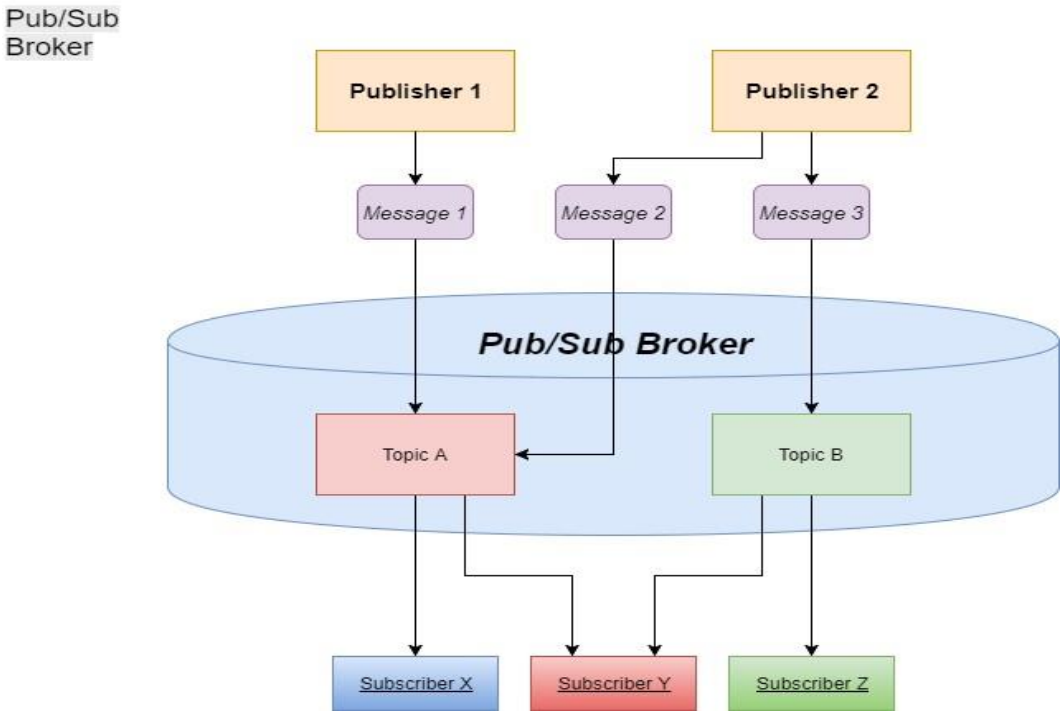
Spring 2022-Programming Assignment

First Name: Dimitris

Last Name: Orinos

7115132100006 (ic121006)

In this assignment is referred about the implementation of a simple form of a publish-subscribe system (pub-sub system) in programming language python. We implemented this system in case we only have one subscriber and one publisher connecting and communicating through a broker based on the following scheme.



If we have only the Subscriber X and the Publisher 1, each subscription has one subscriber, meaning message or subset of messages will be load-balanced across it, with each subscriber receiving a subset of the messages or a single message. Message 1 comes from Publisher 1 and is sent to Subscriber 1 via Subscription 1 into the Topic A.

Firstly about the subscriber and publisher the arguments that it is needed in order to run the sub.py, are passed through the function getopt (import getopt (python library)), the r argument that indicates the port of this specific subscriber or publisher, it is not used it because it is not needed for this approach:

```
subscriber -i ID -r sub_port -h broker_IP -p port [-f
command_file]

 publisher -i ID -r sub_port -h broker_IP -p port [-f
 command_file]
```

(All the above parameters are explained in the README)

After using the getopt function, the files (susbcriber1.txt or publisher1.txt) are opened and after some processing(opening file, splitting with split() method) each one command ,which has time for waiting after connecting with the broker (must be greater or equal to zero) the command (pub, sub, unsub) and the topic for publishing a message or subscribing, is sent to the broker into the form: `SUB_ID COMMAND TOPIC` (subscriber) or `SUB_ID COMMAND TOPIC MESSAGE` (publisher).

In the same way through the input() function the user can type these types of commands (time, topic, pub, sub etc) from the keyboard, with the difference that the time for waiting it is given through a while loop until a greater or equal to zero value is given. After that it is asked, if the user wants to continue to input “y” and the same process revised. In other case publisher or subscriber (put a letter different than y) disconnect from the broker. Also through a while loop is required the commands which are send to the broker to be only pub or sub or unsub.

The connection between publisher or subscriber is made through socket programming which consists a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server (broker) forms the listener socket while the client (publisher, subscriber) reaches out to the server. It is started by importing the socket library and making a simple socket:

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

where first parameter is AF_INET and the second one is SOCK_STREAM. AF_INET refers to the address-family ipv4. The SOCK_STREAM means connection-oriented (e.g TCP) protocol.

Also there is the SO_REUSEADDR option, which allows the broker to use the same port after an old connection was closed (normally, you would have to wait for a few minutes). About sockets also these methods are used:

- s.bind() , where with his method binds address (hostname, port number pair) to socket,
- s.listen() :this method sets up and start listener (s), When we have s.listen(10), this means that up to 10 listeners (requests) is allowed,
- s.accept():this passively accept subscriber/publisher connection, waiting until connection arrives (blocking)
- s.connect() :this method actively initiates broker connection from the side of the subscriber or the publisher.

Basically the publisher and subscriber send to the broker commands either from the file either from the keyboard through their main function (main()). Also they received relative messages from the broker in cases of publishing of a message or subscribing to a topic (through the command sock.recv(1024)) through an infinitive while loop, where in case of the subscriber the messages, that published into a topic that the subscriber has subscribed, are forwarded through the help of a list, which is created outside the two threads, in which the two connections of the broker with publisher and subscriber are kept, which is created outside the two threads. Basically through this list we can find the connection in order to forward each message with its topic in a whole message
About the broker in our approach it is the main program that makes this system reliable.

The broker uses multithreading in order to send messages to publisher and subscriber and more specifically for each subscriber or publisher a thread is created subthread or pubthread correspondingly. These threads are run in parallel. The parameters passed in the same way as in the publisher and subscriber.

About the pubthread the message `SUB_ID COMMAND TOPIC MESSAGE` is splitted through the split() method and a dictionary is created which has the form {topic: message} in order to show that the message publishing has done and every

time it is emptied in order to confirm each publishing and also the broker through this thread two messages are send the first is an OK message about publishing message to the topic and the second in order to inform about the message received for the specific topic. All these happens only if the command that is send from the publisher is pub. We assume that topic is only one word for each publisher and subscriber and also messages are of the form This is a message without commas.

About the subthread the message `SUB_ID COMMAND TOPIC` is splitted through the split() method. Also a list about keeping the topics that the subscriber has an interest, a dictionary is created which has the form {sub_id: topics} in order the broker to keep track of the topics and of the subscriber that subscribes.If the list about topics had a topic more than one times as an element, that means that there are more than one tries to subscribe on a topic and that is something we assume to be not allowed.

Also it is created another list in order to keep track which subscriber is inserting in system. This means that maybe an subscriber id is inserted more than one times, but this is not important for this approach, because each time we have only one publisher and one subscriber connected through the broker in the network. For better performance of the broker algorithm we use about the topic and message sent by the broker to subscriber an global variable named t_m, where it is given outside threads a specific value and we assume that the time waiting is 0 and the name of the publisher is the same with the name that we write in the terminal in order to start running the publisher (line with the arguments).

If the same subscriber id inserted consecutive times the communication of this subscriber with the broker is going to be continued at the same port r which is given automatically from the terminal of the broker. In the case of the insertion of different subscriber identifiers each time, then the list for the topics and the dictionary are emptied in order to have the new subscriber identifier to communicate.

The messages are not stored in this approach so every message that is send for communication from subscriber or publisher with the broker after the end of the communication is missed. For example if in our program from terminals we run from keyboard two or more consecutive times only the last message is

received from the subscriber in case of subscription to a topic, and subscriber could be lost an message that has an interest, if the consecutive messages that are published are referred to the same topic.