Assignment in Big Data Management

Name: Sofia Kalogiannidi
AM: 7115132100002

In this report, I will briefly describe the details of my implementation

Data structures used

I used 3 data structures:

- **subscribers_dict:** It is a dictionary of lists. The keys are the topics and the values for every topic is a list with the ids of all the subscribers to this topic. For example, if s1 and s2 are subscribed to topic #hello and s3 is subscribed to topic #magic, then the subscribers_dict will look like this:

  {#hello:[s1,s2],#magic:[s3]}
- **publishers_dict:** I know that this was not wanted in the exercise, but I could not think of another idea as I implemented everything from scratch. The broker also saves the messages obtained from publishers into a dictionary of lists .The keys are the topics again and the values for every topic is a list with the messages published from all publishers for this topic.
  For example, if p1 published ‘This is the first message’ for topic #hello ,p2 published ‘This is the second message’ for topic #hello and p3 published ‘I am Uri Geller’ for topic #magic then the publishers_dict will look like this :

  {#hello:[‘This is the first message’,’This is the second message’],#magic:[‘I am Uri Geller’]}
- **already_sent:** Again this is a dictionary of lists. In this case, the keys are the ids of the subscribers and the value for every subscriber is a list of all the messages that the broker has already sent , in order not to send them again.
  For example if s1 has already received for topic #hello the message ‘This is the first message’ and for topic #magic ‘I am Uri Geller’, while s3 has received ‘I am Uri Geller’,then the already_sent will look like this :
  {s1:[‘This is the first message’,’I am Uri Geller’],s3:[‘I am Uri Geller’]}

BROKER

It is important to describe the performance of the broker because publisher and subscriber perform standard steps as described in the exercise.

1. General steps

- Broker can accept up to five connections with subscribers and up to five connections with publishers.In order to achieve that I used two variables called subscribersCount and publishersCount respectively.These variables are used as counters.They are initialized with 0 and are increased by 1 every time a new subscriber or publisher arrives. Instead of a while True loop , I used the condition while subscribersCount<=5 and publishersCount<=5.

- Also, subscribers_dict and publisher_dict are initialized.

2. Dealing with publisher.

- Every time a new publisher arrives, a new thread starts and the function on_new_publisher is being called. This function accepts as argument the connection details in order to be able to send and receive message to this specific publisher.
- When publisher sends a request,e.g p1 pub #hello This the first message, the broker prints the message:

```
Message received from publisher: P1 pub #hello This is the first message
```

- Then, function publisher_request is being called.This function takes as argument the message received from publisher. It splits the message into words in order to obtain the topic and the actual message published.Then, it checks if publishers_dict.get(topic)==None, which means that there is not a previous publisher to publish something on this topic.If that is the case, then it creates a list containing the message and adds a new key-value pair in publishers_dict with the format topic:[message].If there is already a message with that topic, it gets the list with the messages and just appends the new message.
- After, that broker prints

```
Sending aknowledgement to the publisher
```

and sends OK to publisher

On the other hand, publisher receives the message OK and also prints:

```
Acknowledgement received from
server
OK
```

3. Dealing with subscriber.

- When a connection with a new subscriber is being established,a new thread starts and the function on_new_subscriber is being called with the connection details.
- When broker receives a message of the format s1 sub/unsub #hello , the function subscriber_request is being called. This function splits the string in order to take subscriber id ,sub/unsub and the topic. Then if the second word was sub, it investigates if subscribers_dict.get(topic)==None.If that is the case,there is no other subscriber, subscribed to this topic ,so it creates key-value pair of the format topic:[id] and inserts it into subscribers_dict dictionary. If there is already a subscriber for this topic ,it just takes the list of the key topic and appends the subscriber in the list. In case that the second word is unsub, it first checks that the subscriber with that id is indeed subscribed in that topic and then it removes the id from the list of that topic. Finally, it returns the subscriber's id.
- Then, broker gets the subscriber's id returned and looks in already_sent dictionary. If already_sent.get(id)==None, it means that this subscriber is new and has just arrived,so the broker adds a new entry to already_sent dictionary of the format id:[] as no message has been sent to this subscriber.
- Broker prints the message

```
Sending aknowledgement to the
subscriber
```

and subscriber prints

```
Acknowledgement received from
server
OK
```

- Afterwards, broker calls the function send_topics with arguments the id of the subscriber and already_sent[id].This function, takes every key of publishers_dict aka every topic with published message and checks at first if there is at least one user subscribed to this topic and then if the subscriber's id is in subscribers_dict[topic].In that case it adds to a new dictionary called user_dict the key-pair value topic:[message1,message2] as they are in publishers_dict.In other words,it actually copies to a new dictionary all the messages published (publisher_dict) in topics that the subscriber has been subscribed (subscriber_dict). As a last step it filters out all the messages that have already been sent to that subscriber by checking already_sent dict.The final dictionary called user_dictfinal is returned.
- Broker takes that dictionary with all the messages that must be sent. Then, it calls the function count_messages with that dictionary as parameter. This function counts and returns the number of messages that the subscriber must receive. Broker prints:

```
I need to send 3 messages
```

- As a next step, broker sends to the subscriber through the socket the number of messages that will be sent. This happens so that the subscriber opens a for loop (with this specific number of repetitions) and receives specific messages because the number of messages sent is varied every time.
- Subscriber prints

```
Messages that I haven't seen :
3
```

- Broker sends in a for loop of certain repetitions all the messages and subscriber receives the messages in a for loop with the same number of repetitions.Then subscriber prints:

```
Received message for topic #hello:This is the first message

Received message for topic #hello : This is the second message

Received message for topic #magic : I am Uri Geller
```

Drawbacks of my implementation:

1. Broker saves the messages from publishers
2. subscribers receive all the messages for the topics they have subscribed not instantly every time a message is published, but every time they make a new sub/unsub request.

Good points of my implementation:

1. There can be multiple subscribers and publishers
2. Subscribers receive every time only the messages that they have not seen before.
3. All the messages required are being printed
4. They can also be called with files as asked.

Screenshots from execution:

Before, finishing let's see a trial execution with the appropriate screenshots provided.
At first, we start our broker.

```
sofia@sofia-B5400:~/Desktop/ntoulas$ python3 broker.py -s 9000 -p 8000
subsciber port is: 9000
publisher port is: 8000
Socket created.
Waiting for Publishers/Subscribers.
```

At first a new subscriber s5 connects with broker and wants to subscribe to topic #hello.
SUBSCRIBER

```
id is: s5
port is: 8000
Ip pf Broker is: 127.0.0.1
Port of Broker is: 9000
filename is: blaaa
Socket created.
Connection established with the server.
This file does not exist
Give a subscribe/unsubscribe command
3 sub #hello
Sleeping for 3
Sending msg to the server: s5 sub #hello
Acknowledgment received from the server:
 OK
Messages that I haven't seen:
0
Give a subscribe/unsubscribe command
```

As no messages have been published for topic #hello ,it receives 0 messages.

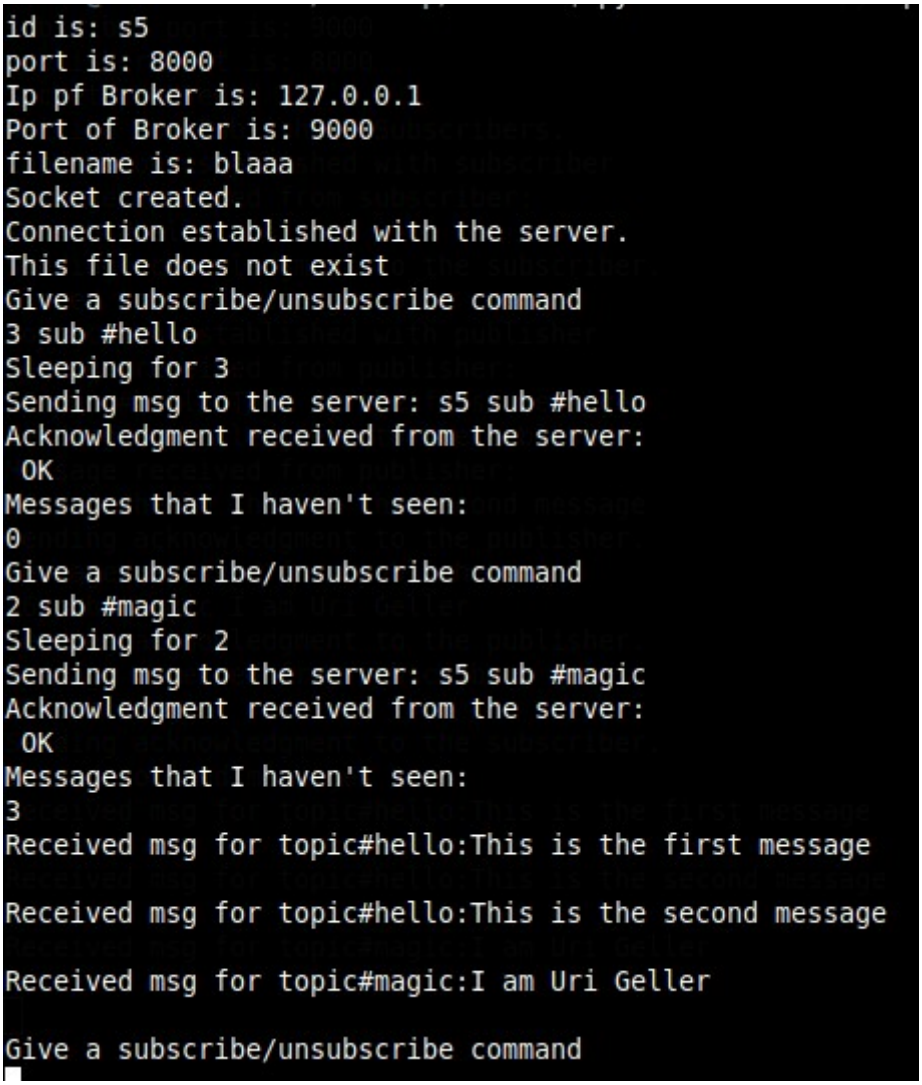
Then a publisher p6 publishes 3 messages
1 for topic #hello : This is the first message
1 for topic #hello : This is the second message
1 for topic #magic : I am Uri Geller
PUBLISHER

```
id is: p6
port is: 8000
Ip pf Broker is: 127.0.0.1
Port of Broker is: 8000
filename is: blaaa
Socket created.
Connection established with the server.
This file does not exist
Give a publish command
3 pub #hello This is the first message
Sleeping for 3
Sending msg to the server: p6 pub #hello This is the first message
Acknowledgment received from the server:
 OK
Published message for topic #hello : This is the first message
Give a publish command
2 pub #hello This is the second message
Sleeping for 2
Sending msg to the server: p6 pub #hello This is the second message
Acknowledgment received from the server:
 OK
Published message for topic #hello : This is the second message
Give a publish command
3 pub #magic I am Uri Geller
Sleeping for 3
Sending msg to the server: p6 pub #magic I am Uri Geller
Acknowledgment received from the server:
 OK
Published message for topic #magic : I am Uri Geller
Give a publish command
```

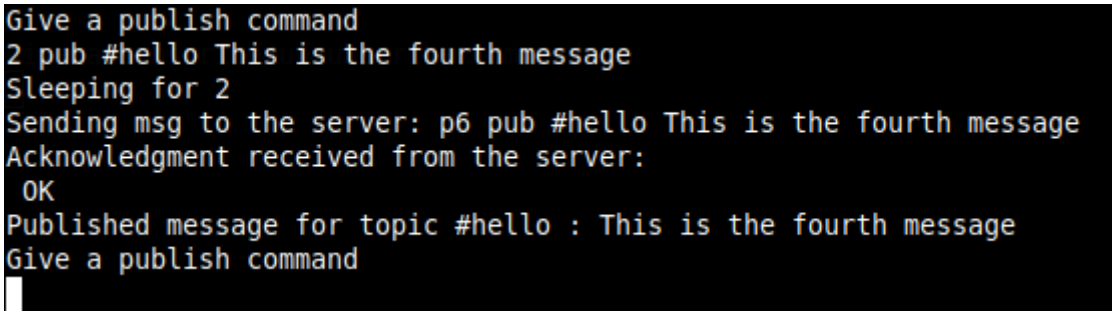
BROKER

```
subsciber port is: 9000
publisher port is: 8000
Socket created.
Waiting for Publishers/Subscribers.
Connection established with subscriber
Message received from subscriber:
s5 sub #hello
Sending acknowledgment to the subscriber.
I need to send messages 0
Connection established with publisher
Message received from publisher:
p6 pub #hello This is the first message
Sending acknowledgment to the publisher.
Message received from publisher:
p6 pub #hello This is the second message
Sending acknowledgment to the publisher.
Message received from publisher:
p6 pub #magic I am Uri Geller
Sending acknowledgment to the publisher.
```

Then s5 subscribes to #magic.
SUBSCRIBER



As we can see it now receives the 3 messages
Publisher now publishes 1 message for topic #hello : This is the fourth message
PUBLISHER



If s5 is unsubscribed from #hello, he will not see that message

SUBSCRIBER

```
Give a subscribe/unsubscribe command
3 unsub #hello
Sleeping for 3
Sending msg to the server: s5 unsub #hello
Acknowledgment received from the server:
 OK
Messages that I haven't seen:
0
Give a subscribe/unsubscribe command
```

Finally,if another subscriber s6 connects and subs to #hello and to #magic he will receive all the messages

```
id is: s6
port is: 8000
Ip pf Broker is: 127.0.0.1
Port of Broker is: 9000
filename is: blaaa
Socket created.
Connection established with the server.
This file does not exist
Give a subscribe/unsubscribe command
3 sub #hello
Sleeping for 3
Sending msg to the server: s6 sub #hello
Acknowledgment received from the server:
 OK
Messages that I haven't seen:
3
Received msg for topic#hello:This is the first message

Received msg for topic#hello:This is the second message

Received msg for topic#hello:This is the fourth message

Give a subscribe/unsubscribe command
2 sub #magic
Sleeping for 2
Sending msg to the server: s6 sub #magic
Acknowledgment received from the server:
 OK
Messages that I haven't seen:
1
Received msg for topic#magic:I am Uri Geller

Give a subscribe/unsubscribe command
```