

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



BÁO CÁO BÀI TẬP LỚN MÔN LẬP TRÌNH MẠNG

ĐỀ BÀI: Xây dựng chương trình kiểu
publish/subscribe như MQTT cho các thiết bị
cảm biến

Sinh viên:

- Hoàng Minh Đức Anh - 18020003
- Nguyễn Chí Thành - 18020053
- Nguyễn Hải Long - 18020037

Hà Nội, tháng 12 năm 2021

MỤC LỤC

- 1. Yêu cầu bài toán**
- 2. Quá trình làm việc**
- 3. Chi tiết về chương trình**
- 4. Protocol subscriber**

1. Yêu cầu bài toán

- I. Chương_trình_hiển_thị_thông_tin_cảm_biến_A kết nối với Broker, đăng ký nhận thông tin tại topic /locationA/sensorA (locationA, sensorA: tham số có thể tùy biến)
- II. Chương_trình_sinh_dữ_liệu_cảm_biến_A kết nối với Broker, tự động sinh dữ liệu cảm biến và publish dữ liệu lên Broker vào topic /locationA/sensorA
- III. Chương_trình_hiển_thị_thông_tin_cảm_biến_A nhận dữ liệu được publish vào topic /locationA/sensorA tức dữ liệu của cảm biến A và hiển thị thông tin cho người dùng
- IV. Nhiều chương trình sinh dữ liệu cảm biến cùng kết nối và publish dữ liệu lên Broker với các topic khác nhau
- V. Nhiều chương trình sinh dữ liệu cảm biến cùng kết nối vào Broker và mỗi chương trình nhận và hiển thị các thông tin từ một topic khác nhau theo thời gian thực
- VI. (thêm) Thi công vẽ UI để nhìn chương trình dễ dàng và đẹp hơn.

2. Quá trình làm việc

- Hợp team, đọc kỹ đề bài, phân tích đặc tả yêu cầu, từ đó chia việc cho từng thành viên.
- Phân chia công việc:
 - + Thành: Đảm nhiệm thiết kế Protocol và Broker
 - + Long: Đảm nhiệm Publisher
 - + Đức Anh: Đảm nhiệm Subscriber
- Timeline:
 - + 27/11: Hợp team, phân chia công việc. Chốt Protocol. Bắt đầu làm Broker.
 - + 01/12: Hoàn thành phần lõi của Broker.
 - + 02/12: Bắt đầu làm Publisher và Subscriber.
 - + 10/12: Hoàn thành phần lõi của Publisher (gửi, nhận, sinh thông báo) và Subscriber (Sử dụng giao diện dòng lệnh).
 - + 11/12: Sửa một vài lỗi nhỏ của Publisher và Broker. Bắt đầu làm GUI cho Subscriber.
 - + 12/12: Hoàn thành cửa sổ hiển thị Output từ Server của Subscriber.
 - + 13/12: Hoàn thành cửa sổ nhập Command, cửa sổ nhập IP + Port để kết nối cho Subscriber
 - + 14/12: Publisher có thể sinh ngẫu nhiên nhiều topic hơn. Bắt đầu viết báo cáo
 - + 16/12: Hoàn thiện báo cáo.

3. Mô tả giao thức

Quá trình gửi nhận tin nhắn sẽ gồm 3 thành phần tham gia:

- Subscriber: Đóng vai trò là Client đăng ký nhận tin nhắn từ 1 hoặc nhiều topic.
- Publisher: Đóng vai trò là Client tạo ra các tin nhắn và gửi tới Broker
- Broker: Đóng vai trò là Server tiếp nhận tin nhắn từ Publisher, sau đó gửi tới Subscriber dựa theo topic của bản tin.

Các câu lệnh trong giao thức:

- CONNECT: Client cần gửi lệnh này sau khi tạo kết nối tới Server. Nếu kết nối được chấp nhận, Server trả về CONNACK [ClientId].
- SUBSCRIBE [Topic]: Client đăng ký nhận thông báo từ Topic. Nếu đăng ký được chấp nhận, Server trả về SUBACK [Topic].
- UNSUBSCRIBE [Topic]: Client huỷ đăng ký nhận tin từ Topic. Nếu huỷ đăng ký thành công, Server trả về UNSUBACK [Topic].
- PUBLISH [Topic] [Message]: Client gửi Message tới Topic. Nếu gửi thành công, Server sẽ trả về PUBACK. Sau khi nhận được lệnh PUBLISH, Server sẽ gửi Message tới các Client đã đăng ký Topic theo dạng: PUBLISH [Topic] [Message].

4. Chi tiết về chương trình

Ngôn ngữ chính : Java. Thư viện hỗ trợ thêm: Javafx cho phần GUI của Subscribe.

4.1. Mô tả hoạt động của Server:

- Khi khởi động sẽ sinh ra 2 thread chính
 - + 1 thread để lắng nghe kết nối của Client
 - + 1 thread để gửi thông báo PUBLISH tới Client
- Ở luồng thứ nhất, khi có kết nối tới từ Client, Server sẽ sinh ra 1 thread mới để tiếp nhận và xử lý các tin nhắn tới:
 - + Khi nhận được CONNECT, Server sinh ra 1 UUID và dùng nó để định danh client
 - + Khi nhận được SUBSCRIBE, Server thêm ClientId vào danh sách đăng ký topic đó. Sử dụng ConcurrentHashMap có Key là tên topic và Value là 1 Set các ClientId đăng ký topic đó.
 - + Khi nhận được UNSUBSCRIBE, Server sẽ xoá Client đó ra khỏi danh sách đăng ký.
 - + Khi nhận được PUBLISH, Server sẽ thêm message vào message queue để đợi được gửi tới các Client đã đăng ký. Message queue sử dụng kiểu dữ liệu là BlockingQueue.

- + Khi Client ngắt kết nối, Server sẽ xóa Client khỏi danh sách đang hoạt động và danh sách đăng ký của các topic đã đăng ký, sau đó dừng thread.
- Ở luồng thứ hai, Server sẽ chạy một vòng lặp liên tục kiểm tra message queue xem có tin nhắn cần gửi không, nếu có thì lấy danh sách các Client đã đăng ký topic đó và gửi tin nhắn tới Client.

4.2. Mô tả hoạt động của Publisher:

Nhiệm vụ của Publisher ở trong bài tập lớn lần này là tự sinh ra các câu thông báo và gửi nó đến Server. 2 công việc trên sẽ được tách riêng ra 2 thread để không ảnh hưởng đến nhau: MessageGenerator và ServerInteractHandler.

Hai luồng sẽ thực hiện thao tác chung trên một queue, nên để tránh gây tắc nghẽn, chương trình đã sử dụng BlockingQueue vì nó có thể triển khai dễ dàng và an toàn chia sẻ giữa các thread. BlockingQueue này sẽ được khởi tạo ở Class Publisher và truyền cho 2 thread. Nếu đúng logic thì ở Publisher sẽ tạo 2 thread riêng biệt, nhưng trong bài này nhóm em đã thảo luận để thống nhất chung là chỉ khi kết nối được server thì mới bắt đầu tạo thông báo, nên luồng MessageGenerator đã được chuyển vào khởi chạy trong thread ServerInteractHandler. Chi tiết về 2 luồng như sau:

- Luồng tạo thông báo: Class tạo thông báo chấp nhận 2 tham số khởi tạo là queue chung và định nghĩa của tín hiệu hết thông báo. sinh ra dữ liệu với hàm generateMessage(), đẩy vào queue. Hàm này sẽ sinh ra thông báo random topic và random câu thông báo. Trong chương trình thì có thực hiện delay để tránh sinh quá nhiều dẫn đến gửi quá nhiều request đến server.
- Luồng tương tác với Server: Class tiếp nhận các tham số khởi tạo bao gồm địa chỉ server, cổng server, queue và định nghĩa của tín hiệu hết thông báo. Sau đó từ các tham số này class này sẽ kết nối với Server. Khi kết nối thành công theo protocol thì sẽ bắt đầu tạo thread sinh tin nhắn. Bằng việc sử dụng hàm take() thì thread này sẽ chờ cho đến khi queue có các thông báo. Ở đây có sử dụng một trường đánh dấu là lastSentSuccess. Nếu gửi dữ liệu thành công cho server sẽ nhận được “PUBACK” thì lastSentSuccess = true, lúc này ta tiến hành lấy thông báo tiếp theo. Còn trong trường hợp không nhận được “PUBACK” hoặc quá thời gian mà Server không giao tiếp gì thì tiến hành gửi lại thông báo cũ.

4.3. Mô tả hoạt động của Subscriber:

Subscriber có các chức năng đăng ký nhận thông tin từ Publisher thông qua Broker. Xây dựng Subscriber khá đơn giản với tính năng Gửi và Nhận thông báo thông qua cửa sổ giao diện người dùng. Sau khi kết nối tới Broker thì sinh ra thêm 1 thread có nhiệm vụ liên tục đọc thông tin từ Server gửi về để cập nhật vào cửa sổ Output. GUI của Subscriber được xây dựng bằng thư viện JavaFx.

5. Tổng kết

Source code: <https://github.com/eruchii/NetworkProgramming-MQTT>

Thiết lập server ở cổng 9000.

Ở phía Publisher : Nhập IP và Port. Ta sẽ thấy thông báo Done Connect To Server - thông báo connect thành công với server

```
PS D:\Dev\Java\NetworkProgramming-MQTT> & "C:\Users\1\AppData\Local\Programs\Eclipse Foundation\jdk-11.0.12-hotspot\bin\java.exe" "-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=321" "@C:\Users\1\AppData\Local\Temp\cp_81ul7yqixy9h3lwjfuozn43m_argfile" "Publisher.Publisher"
Nhập IP: 192.168.1.73
Nhập Port: 9000
Done connect to server
```

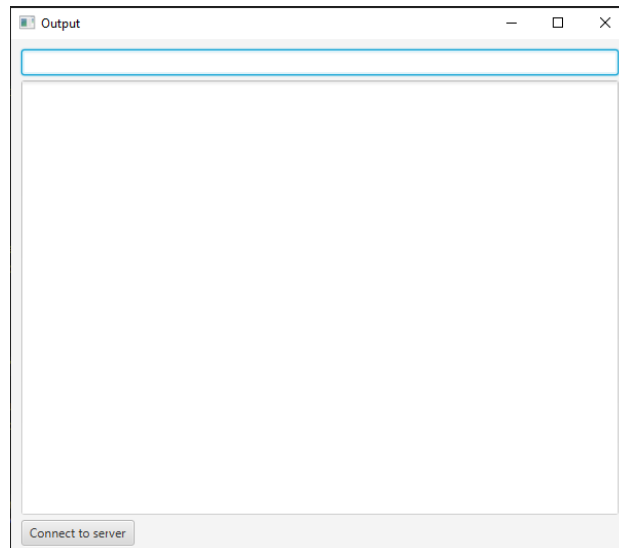
Sau khi connect thành công thì Publisher bắt đầu gửi thông báo:

```
Sending publish: PUBLISH AirQuality The AirQuality is 28
2
Sending publish: PUBLISH Humidity The Humidity is 85
2
Sending publish: PUBLISH Humidity The Humidity is 53
1
0
Sending publish: PUBLISH Temperature The Temperature is 64
Sending publish: PUBLISH Pressure The Pressure is 98
```

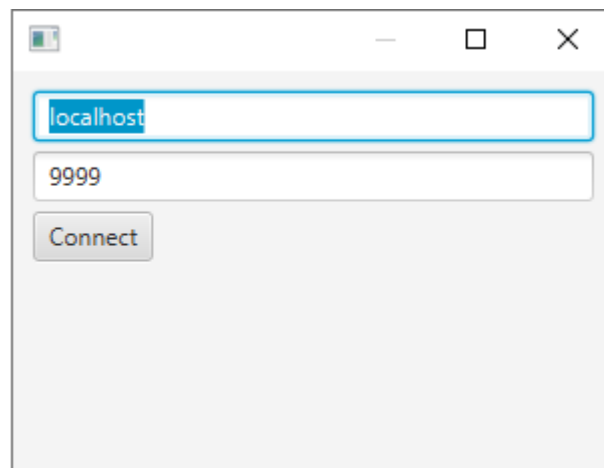
Phía Server cũng nhận được các thông báo publish:

```
PUBLISH AirQuality The AirQuality is 90
PUBLISH Pressure The Pressure is 99
PUBLISH Temperature The Temperature is 3
PUBLISH AirQuality The AirQuality is 9
PUBLISH Humidity The Humidity is 19
PUBLISH AirQuality The AirQuality is 53
PUBLISH Temperature The Temperature is 44
PUBLISH Humidity The Humidity is 31
PUBLISH Humidity The Humidity is 98
```

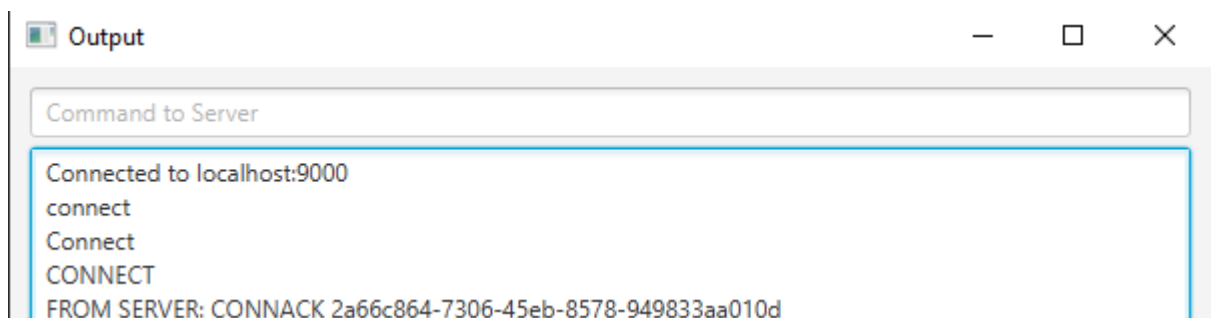
Giao diện Subscriber:



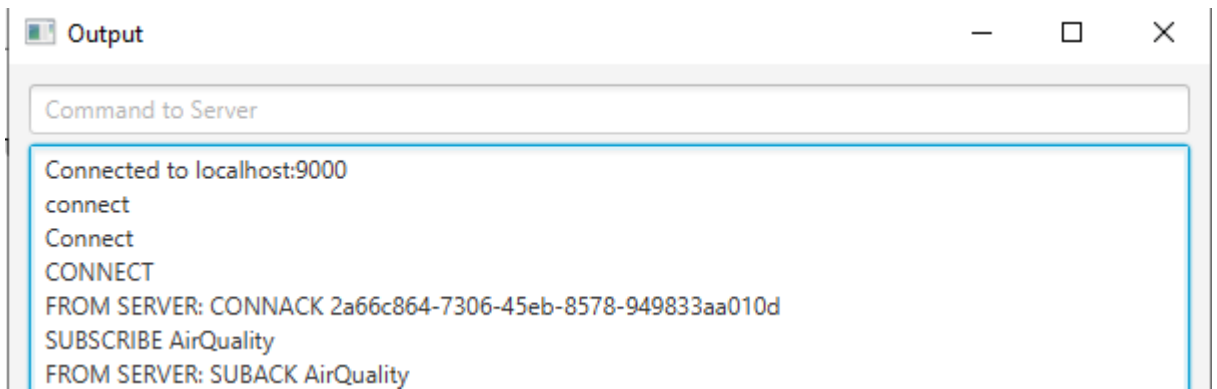
Nhấn nút Connect To server. Cửa sổ này hiện ra :



Ta nhập IP và Port. Sau đó nhấn Connect. Khi đó ta nhập vào lệnh CONNECT để kết nối với server. Nếu thành công server sẽ trả lại UUID của client:



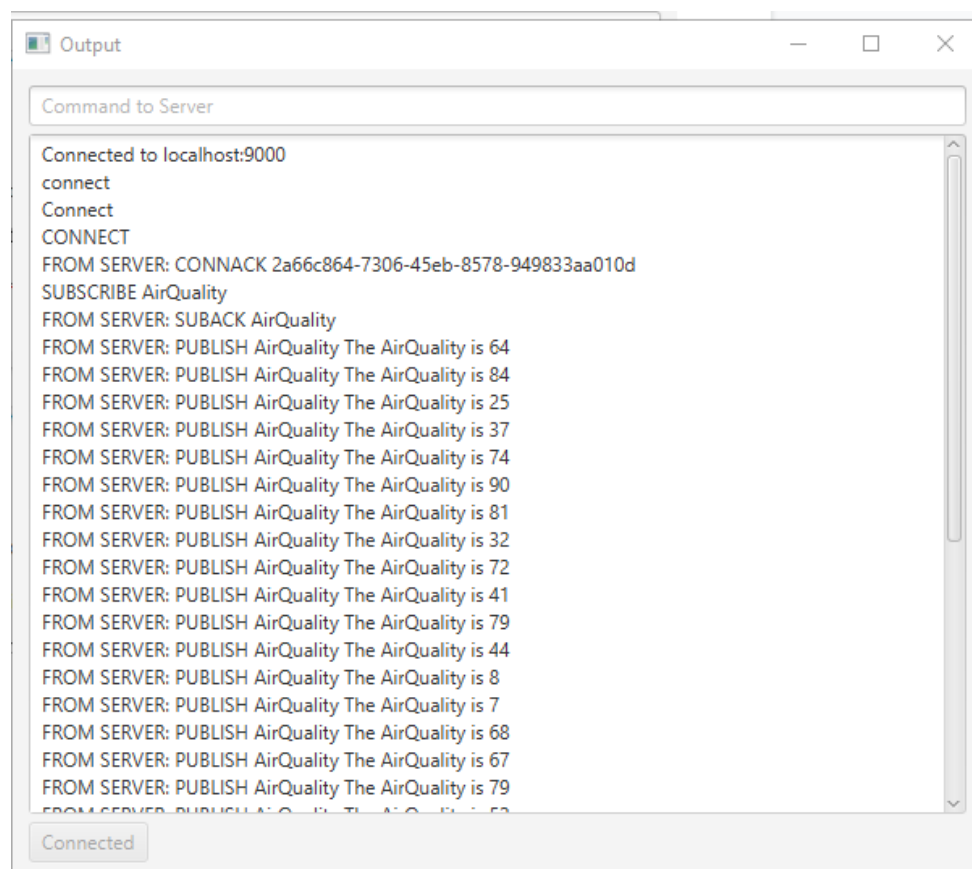
Tiếp theo ta nhập SUBSCRIBE <topic> để tiến hành Subscribe với topic. Nếu đúng cú pháp thì server sẽ trả lại SUBACK cùng tên topic.



```
Output
Command to Server

Connected to localhost:9000
connect
Connect
CONNECT
FROM SERVER: CONNACK 2a66c864-7306-45eb-8578-949833aa010d
SUBSCRIBE AirQuality
FROM SERVER: SUBACK AirQuality
```

Sau đó server sẽ bắt đầu nhận thông báo từ Publisher và gửi đến cho Subscriber đúng với topic Subscriber đăng ký:



```
Output
Command to Server

Connected to localhost:9000
connect
Connect
CONNECT
FROM SERVER: CONNACK 2a66c864-7306-45eb-8578-949833aa010d
SUBSCRIBE AirQuality
FROM SERVER: SUBACK AirQuality
FROM SERVER: PUBLISH AirQuality The AirQuality is 64
FROM SERVER: PUBLISH AirQuality The AirQuality is 84
FROM SERVER: PUBLISH AirQuality The AirQuality is 25
FROM SERVER: PUBLISH AirQuality The AirQuality is 37
FROM SERVER: PUBLISH AirQuality The AirQuality is 74
FROM SERVER: PUBLISH AirQuality The AirQuality is 90
FROM SERVER: PUBLISH AirQuality The AirQuality is 81
FROM SERVER: PUBLISH AirQuality The AirQuality is 32
FROM SERVER: PUBLISH AirQuality The AirQuality is 72
FROM SERVER: PUBLISH AirQuality The AirQuality is 41
FROM SERVER: PUBLISH AirQuality The AirQuality is 79
FROM SERVER: PUBLISH AirQuality The AirQuality is 44
FROM SERVER: PUBLISH AirQuality The AirQuality is 8
FROM SERVER: PUBLISH AirQuality The AirQuality is 7
FROM SERVER: PUBLISH AirQuality The AirQuality is 68
FROM SERVER: PUBLISH AirQuality The AirQuality is 67
FROM SERVER: PUBLISH AirQuality The AirQuality is 79
FROM SERVER: PUBLISH AirQuality The AirQuality is 62
```

Connected