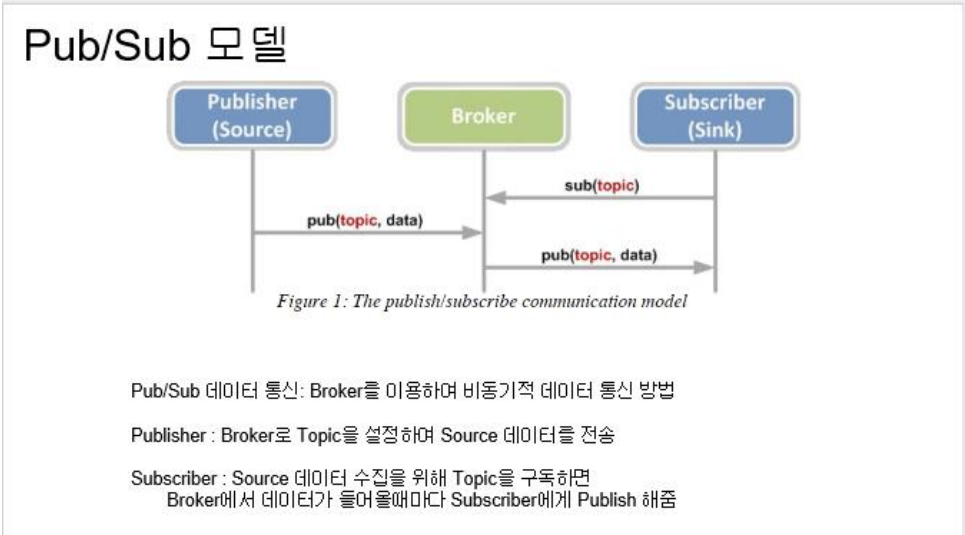
2015104220 컴퓨터공학과 정종윤

네트워크 통신 대전이 가능한 포켓몬 배틀 프로그램

제안서에서는 어떤 프로그램을 만들 것인지에 대한 기본 규칙과 네트워크 구조, 시뮬레이션, 개발 도구 등에 대해 서술하였다. 이번 중간 보고서에서는 지금까지 제안서에 기술한 내용을 구현하기 위해 구체적으로 어떤 공부를 했는지, 그리고 앞으로의 계획에 대해 작성하려 한다.

지금까지 한 것

- 브로커를 이용한 비동기적 데이터 통신 방법인 Pub-Sub 모델을 공부했다.



- 오픈 소스 MQTT Broker 인 Mosquitto 에 대해 공부했다. 이를 구현하기 위해, 오픈 소스 클라이언트 도구인 Paho 라는 Library 를 이용했다.

- Java 로 Server 와 Client 간에 메시지를 주고 받을 수 있는 프로그램을 구현해보았다. Publish 된 메시지가 Client 에게 도착함을 알 수 있다.

```java
package project;

import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

public class app {
	public static void main(String[] args) throws Exception {
		// Init
		// Get Address
		String userName = "User";
		String Pub_Topic = "test/hobby";
		String Sub_Topic = "test/hobby";
		String broker = "tcp://127.0.0.1:1883";

		MemoryPersistence persistence = new MemoryPersistence();
		// Get Address
		MqttConnectOptions connOpts = new MqttConnectOptions();
		connOpts.setCleanSession(true);
		connOpts.setUserName("hubiss");
		connOpts.setPassword("abcdefg".toCharArray());

		MqttClient sampleClient = new MqttClient(broker, userName, persistence);

		if (!sampleClient.isConnected()) {
			System.out.println("Connecting...");
			sampleClient.connect(connOpts);
		}

		// Sub
		System.out.println("Connected");
		Subscribe(sampleClient, Sub_Topic, 1);
		System.out.println("Subscribe");

		// Pub
		int pub_qos = 1;
		Publish(sampleClient, Pub_Topic, " Messege No.1 ", pub_qos, false);
		Publish(sampleClient, Pub_Topic, " Messege No.2 ", pub_qos, false);
		Publish(sampleClient, Pub_Topic, " Messege No.3 ", pub_qos, false);
		Publish(sampleClient, Pub_Topic, " Messege No.4 ", pub_qos, false);
		Publish(sampleClient, Pub_Topic, " Messege No.5 ", pub_qos, false);
		Publish(sampleClient, Pub_Topic, " Messege No.6 ", pub_qos, false);
		sampleClient.disconnect();
		System.out.println("disconnect");
		sampleClient.close();
		System.out.println("close");
	}

	public static int Publish(MqttClient sampleClient, String Topic, String
Message, int qos, boolean retain) {
		try {
			System.out.println("Publishing message: " + Message);
			MqttMessage message = new MqttMessage(Message.getBytes());
			message.setQos(qos);
			sampleClient.publish(Topic, Message.getBytes(), qos, retain);

			// sampleClient.publish(Topic, message);
			System.out.println("Message published");
		} catch (MqttException e) {
			// TODO Auto-generated catch block
			System.out.println(" Dead");
			e.printStackTrace();
		}
		return 0;
	}

	public static int Subscribe(MqttClient sampleClient, String Topic, int qos) {
		try {
			sampleClient.setCallback(new MqttCallback() {
				@Override
				public void messageArrived(String arg0, MqttMessage message)
throws Exception {
					System.out.println("New Message is Arrived : " +
message.toString());
				}

				@Override
				public void deliveryComplete(IMqttDeliveryToken arg0) {
					try {
						arg0.waitForCompletion();
					} catch (MqttException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					}
					System.out.println("deliveryComplete - " +
arg0.isComplete());
				}

				@Override
				public void connectionLost(Throwable arg0) {
				}
			});

			System.out.println("Message Subscribed");
			sampleClient.subscribe(Topic, qos);
			System.out.println("Message Subscribed done");

		} catch (MqttException e) {
			// TODO Auto-generated catch block
			System.out.println(" Dead");
			e.printStackTrace();
		}
		return 0;
	}
}
```

```
Connecting...
Connected
Message Subscribed
Message Subscribed done
Subscribe
Publishing message:  Messege No.1
Message published
Publishing message:  Messege No.2
deliveryComplete - true
New Message is Arrived :  Messege No.1
Message published
Publishing message:  Messege No.3
deliveryComplete - true
New Message is Arrived :  Messege No.2
Message published
deliveryComplete - true
Publishing message:  Messege No.4
New Message is Arrived :  Messege No.3
Message published
Publishing message:  Messege No.5
deliveryComplete - true
New Message is Arrived :  Messege No.4
Message published
Publishing message:  Messege No.6
deliveryComplete - true
New Message is Arrived :  Messege No.5
Message published
deliveryComplete - true
New Message is Arrived :  Messege No.6
disconnect
close
```

- 이러한 프로그램을 Docker 를 이용해 Linux 환경에서 구현된 컨테이너를 만들어 보았다.

앞으로 할 것

- 주어진 상황 내에서만 메시지 입력과 전송이 이루어지도록 제어하는 기능 구현
- 사용자의 입력을 받아 연산을 처리하는 함수 구현
- 인터페이스 개선 및 구체적인 프로그램 구조 설계