Dang Quang Tran 1113204
Manh Tan Doan 1113588

MESSPROTOKOLL PRAKTIKUM 04

I. Einführung

Das vorliegende Protokoll beschreibt die Implementierung einer Bankenrettungssimulation unter Verwendung von Message-Oriented-Middleware (MOM) wie MQTT. Das Ziel ist es, gefährdete Banken mithilfe einer Gruppe von Banken zu retten. Dabei werden sowohl funktionale Anforderungen (Features) als auch nichtfunktionale Anforderungen (Performance) berücksichtigt.

II. Neue Features

1. Broker

Der Broker ist ein MQTT-Service, der auf Mosquitto basiert und als Docker-Container ausgeführt wird. Der Broker fungiert als zentraler Kommunikationskanal für die Banken. Er ermöglicht den Austausch von Nachrichten zwischen den Banken durch das Publish/Subscribe-Muster.

```
persistence false
listener 1883
log_type all
log_type debug
allow_anonymous true
connection_messages true
```

Mosquitto-config file

```
FROM eclipse-mosquitto
COPY src/main/java/MOM/mosquitto.conf /mosquitto/config/mosquitto.conf
```

Mosquitto Dockerfile

2.Publishing/Subscribing

Jede Bank nimmt sowohl die Rolle eines Publishers als auch eines Subscribers ein. Dies bedeutet, dass jede Bank in der Lage ist, Informationen an den Broker zu veröffentlichen und Nachrichten von anderen Banken zu abonnieren. Die Banken veröffentlichen regelmäßig ihre Gesamtwerte als Nachrichten auf bestimmten Topics, während sie gleichzeitig Nachrichten von anderen Banken abonnieren, um Informationen zu erhalten, die für ihre Entscheidungsfindung relevant sind.

Dang Quang Tran 1113204
Manh Tan Doan 1113588

```java
import org.eclipse.paho.client.mqttv3.MqttException;

2 usages  1 implementation  stdatrann
public interface Publisher {
    1 implementation  stdatrann
    void publish(String topic, String message) throws MqttException;
}
```

Publisher Interface

```java
import org.eclipse.paho.client.mqttv3.MqttException;

2 usages  1 implementation  stdatrann
public interface Subscriber {
    1 implementation  stdatrann
    void subscribe(String topic) throws MqttException;
    1 usage  1 implementation  stdatrann
    void unsubscribe(String topic) throws MqttException;
    1 implementation  stdatrann
    void disconnect() throws MqttException;
}
```

Subcriber Interface

Der Prozess sieht wie folgt aus:

- Jede Bank veröffentlicht periodisch ihre Gesamtwerte auf einem spezifischen Topic.
- Der Broker empfängt die veröffentlichten Nachrichten und speichert sie ab.
- Die Banken, die das entsprechende Topic abonniert haben, erhalten automatisch die veröffentlichten Nachrichten vom Broker.
- Die Banken können die empfangenen Informationen nutzen, um Entscheidungen im Zusammenhang mit der Rettung gefährdeter Banken zu treffen.

Dang Quang Tran 1113204
Manh Tan Doan 1113588

```
2 usages  stdatrann
public void publishBankValue(double value) throws MqttException, UnknownHostException {
    String topic = "bank-values/";
    publish(topic,  message: bank.getThisBankIP() + ":" + bank.getThisBankPortThrift() + "/" + value);
}

1 usage  stdatrann
public void subscribeToBankValues() throws MqttException, UnknownHostException {
    String topic = "bank-values/";
    subscribe(topic);
}
```

Durch die Verwendung des Publish/Subscribe-Musters und des zentralen Brokers wird eine effiziente und zuverlässige Kommunikation zwischen den Banken ermöglicht.

III. Funktionsanforderungen

1. Regelmäßige Veröffentlichung der Gesamtwerte der Banken:
Jede Bank agiert als Publisher und Subscriber im MQTT-Netzwerk und veröffentlicht regelmäßig ihre aktuellen Gesamtwerte. Diese Veröffentlichungen dienen dazu, den anderen Banken aktuelle Informationen über den Zustand der Banken zur Verfügung zu stellen.

```
Bank_02          | Received subscribed message: bank-values/ - 172.20.1.1:7000/2100714.0
Bank_02          | Received subscribed bank value: 172.20.1.1:7000 - 2100714.0
Bank_01          | Received subscribed message: bank-values/ - 172.20.1.1:7000/2100714.0
Bank_01          | Received subscribed bank value: 172.20.1.1:7000 - 2100714.0
Bank_03          | Received subscribed message: bank-values/ - 172.20.1.1:7000/2100714.0
Bank_03          | Received subscribed bank value: 172.20.1.1:7000 - 2100714.0
```

2. Bankenrettung durch andere Banken:
Eine gefährdete Bank kann nur durch die Unterstützung mehrerer anderer Banken gerettet werden. Die Banken müssen individuell entscheiden, ob sie bereit sind, Geld an die gefährdete Bank zu überweisen. Wenn eine Bank die Rettung ablehnt, wird der Rettungsprozess für die gefährdete Bank abgelehnt.

Dang Quang Tran 1113204
Manh Tan Doan 1113588

```
Bank_01             | Update Total Value: -9.9979688219186O6E9
Bank_01             | SENDING HELP REQUEST
```

```
Bank_01             | Response: APPROVED came in 45ms
Bank_01             | 172.20.1.3 success to rescue
```

```
Bank_01             | Response: APPROVED came in 72ms
Bank_01             | 172.20.1.2 success to rescue
```

3. Two Phase Commit (2PC) für die Rettung:

Um sicherzustellen, dass die Überweisungen zur Rettung koordiniert und abgestimmt werden, wird der Two Phase Commit (2PC) Algorithmus eingesetzt. Hierbei werden mehrere Banken gefragt, ob sie bereit sind, Geld an die gefährdete Bank zu überweisen. Erst wenn ausreichend viele Banken zustimmen, werden die Überweisungen bestätigt und ausgeführt. Für die Überweisungen wird das Thrift-Framework verwendet, wie im vorherigen Praktikum (Praktikum 3) beschrieben. Die Höhe der Überweisung wird durch den Leader, also die Bank, die die Rettung koordiniert, festgelegt.

```java
case "bail-out/": {
    if(isLeader){
        int totalAmountOfBanks = bank.getAssociatedBanks().size();
        String[] bankInfo = message.toString().split( regex: "/");
        String bankAddress = bankInfo[0];
        String bailAmount = bankInfo[1];

        if(bailAmount.equals("ACCEPTED")){
            acceptMessagesReceived++;
        }
        if(bailAmount.equals("DENIED")){
            publish( topic: "bail-out-denied/",  message: "DENIED");
            break;
        }

        if(acceptMessagesReceived == totalAmountOfBanks){
            publish( topic: "bail-out-granted/", bailAmount);
        }
```

Dang Quang Tran 1113204
Manh Tan Doan 1113588

```
            if(bank.getTotalValue() ≤ Double.parseDouble(bailAmount)){
                publish( topic: "bail-out/",  message: "DENIED");
            }
            else{
                choseToBail = true;
                publish( topic: "bail-out/",  message: "ACCEPTED");
            }
        }
        break;
    }
    case "bail-out-granted":
        if(choseToBail) {
            String bailAmount = message.toString();
            System.out.println("Received bail-out-granted message: " + bailAmount);

            setBailAmount(Double.parseDouble(bailAmount) / (bank.getAssociatedBanks().size()));
            choseToBail = false;
        }
```

4. Bank-Leader Auswahl:

Zur Koordination der Rettung wird der Raft-Algorithmus eingesetzt, um einen Bank-Leader zu bestimmen. Der Bank-Leader ist für die Koordination des Rettungsprozesses zuständig. Wenn eine Rettung ausgelöst wird, sendet die betroffene Bank eine entsprechende Nachricht an alle Banken, die das entsprechende Thema abonniert haben. Anschließend wird die Bank mit dem höchsten Gesamtwert als Bank-Leader ausgewählt, der die Rettung koordiniert.

```
if (hasThisBankTheHighestValue()){
    requestVote();
    Thread.sleep( l: 2000);
    checkVotes();
}
```

Dang Quang Tran 1113204
Manh Tan Doan 1113588

```java
private void checkVotes() throws UnknownHostException {
    //get bank address with the highest vote in votes
    String bankAddressWithHighestVote = "";
    int highestVote = 0;
    for(String bankAddress : votes.keySet()){
        if(votes.get(bankAddress) > highestVote){
            highestVote = votes.get(bankAddress);
            bankAddressWithHighestVote = bankAddress;
        }
    }

    //check if this bank is the bank with the highest vote
    if(bankAddressWithHighestVote.equals(bank.getThisBankIP() + ":" + bank.getThisBankPortThrift())){
        isLeader = true;
        System.out.println("This bank is the leader");
    }
    else{
        isLeader = false;
    }
}
```

Alternativ kann auch die Bank, die gerettet werden muss, automatisch zum Leader der Rettung ernannt werden.

III. Nichtfunktionale Anforderungen

1. Performanzmessung:

Um die Performance des Rettungsprozesses zu bewerten, wird eine Zeitmessung implementiert. Die Zeit zwischen der Rettungsanfrage einer betroffenen Bank und der Entscheidung des Leaders wird erfasst, um festzustellen, wie lange es dauert, den gesamten Prozess abzuschließen. Dies ermöglicht die Evaluierung der Effizienz des Rettungsmechanismus und gibt Aufschluss über die Leistungsfähigkeit des Systems.

Dang Quang Tran 1113204
Manh Tan Doan 1113588

```
Bank_01              | Update Total Value: -9.9979688219186O6E9
Bank_01              | SENDING HELP REQUEST
```

```
Bank_01              | Response: APPROVED came in 45ms
Bank_01              | 172.20.1.3 success to rescue
```

```
Bank_01                | Response: APPROVED came in 72ms
Bank_01                | 172.20.1.2 success to rescue
```