6 Übungsaufgabe #6: Verteilte Synchronisation

In dieser Übungsaufgabe soll die Koordinierung von Zugriffen auf eine gemeinsame Ressource in einem verteilten System näher betrachtet werden: Ziel dieser Aufgabe ist es, den Algorithmus von Lamport zu implementieren. Der Algorithmus ermöglicht Synchronisation mehrerer Teilnehmer ohne die Notwendigkeit eines zentralen Anlaufpunktes. Die Umsetzung erfolgt in Form von Protokollschichten für das bereits aus vorherigen Übungsaufgaben bekannte JGroups. Weiterhin sind die Funktionen des Algorithmus durch eine einheitliche Schnittstelle zugänglich zu machen. Diese kann von den bereitgestellten Testfällen (siehe Tafelübung) verwendet werden, um die korrekte Funktion der Implementierung zu überprüfen. Alle Klassen sind dabei im Subpackage `vsue.distlock` abzulegen.

Hinweis:

- Die Methoden einer Protokollschicht können parallel in unterschiedlichen Threads zur Ausführung kommen. Daher ist immer auf korrekte Synchronisation aller Datenstrukturen zu achten!

6.1 Logische Uhren (für alle)

Grundlage für den eigentlichen Synchronisationsalgorithmus bildet zunächst eine Protokollschicht zur Verwaltung einer logischen Uhr. Diese ermöglicht den höheren Schichten festzustellen, zu welchem logischen Zeitpunkt eine Nachricht verschickt wurde, woraus in Teilaufgabe 6.2 eine konsistente Totalordnung gebildet werden soll.
Den Kern der Protokollschicht soll ein Zähler bilden, dessen aktueller Stand an jede versendete Nachricht in Form eines zusätzlichen Headers vom Typ `VSClockHeader` angefügt wird. Der interne Zählerstand wird abhängig von den gesendeten und empfangenen Nachrichten gemäß den Vorschriften für eine Lamport-Uhr (siehe Tafelübung) angepasst. Nutzer des Protokolls sollen durch den Aufruf der statischen Methode `getMessageTime()` in der Lage sein, den logischen Zeitstempel aus der angegebenen Nachricht zu extrahieren.

Aufgabe:

→ Implementierung der logischen Uhr durch Erweiterung der vorgegebenen Klasse `VSLogicalClockProtocol`

6.2 Algorithmus von Lamport (für alle)

Auf Basis der logischen Uhren soll nun in einer höheren Protokollschicht der eigentliche Algorithmus von Lamport (siehe Tafelübung) implementiert werden. Dazu ist eine Liste mit bislang empfangenen Sperranfragen zu verwalten, die nach den logischen Zeitstempeln der Anfragen sortiert ist. Steht die eigene Anfrage am Anfang der Liste und ist kein anderer Teilnehmer mehr in der Lage eine Nachricht mit kleinerem Zeitstempel zu versenden, steht die gemeinsame Ressource solange zur Verfügung, bis diese explizit durch Versenden einer Nachricht an alle wieder freigegeben wurde.
Aufgabe der Protokollschicht ist es nun, sich um den notwendigen Nachrichtenaustausch zu kümmern und Methoden bereitzustellen, die das Anfordern und Freigeben einer Sperre ermöglichen. Weiterhin soll die Registrierung eines Objekts vom Typ `VSLamportLock` an der Protokollschicht möglich sein, welches aktiviert wird sobald die gemeinsame Ressource nach einer Sperranfrage verfügbar ist.

Aufgabe:

→ Implementierung des Algorithmus durch Erweiterung der vorgegebenen Klasse `VSLamportLockProtocol`

Hinweise:

- Bei gleichem Zeitstempel zweier Anfragen entscheidet die Rechneradresse über die Reihenfolge der Zuteilung.
- Nachrichten, welche zur Abwicklung des Lamport-Algorithmus ausgetauscht werden müssen, sollen keinen Einfluss auf höhere Protokollschichten haben.
- Mehrere unabhängige Locks müssen nicht unterstützt werden.

6.3 Schnittstelle zur Programmierung (für alle)

Um die neue Protokollschicht in Programmen einfach verwenden zu können, soll schließlich noch eine neue Klasse `VSLamportLock` implementiert werden, welche folgende Methoden anbietet:

```
public class VSLamportLock {
    public VSLamportLock(JChannel channel);
    public void lock();
    public void unlock();
}
```

Der Konstruktor registriert sich dabei an der Implementierung des Lamport-Algorithmus im Protokollstapel, so dass dieser benachrichtigt wird sobald die Sperre zugewiesen wurde. Die Methode `lock()` löst eine Sperranfrage aus und blockiert daraufhin den aktuellen Thread so lange, bis die Benachrichtigung über die Verfügbarkeit vorliegt. Die `unlock()`-Methode ermöglicht die Freigabe der gemeinsamen Ressource.

Abgabe: am 29.7. in der Rechnerübung