5 Übungsaufgabe #5: Zuverlässige Gruppenkommunikation

In Übungsaufgabe 4 wurde ein Dienst unter Verwendung von *JGroups*, einer Bibliothek für zuverlässige Gruppenkommunikation, aktiv repliziert. JGroups ist modular aufgebaut und kann durch das Hinzufügen und Entfernen von *Protokollschichten* im *Protokoll-Stack* an die jeweiligen Erfordernisse angepasst werden. So wurde für die aktive Replikation ein *Sequencer* eingesetzt, der eine totale Ordnung auf Anfragen von verschiedenen Clients sicherstellte. Um die Interna einer Gruppenkommunikation näher kennenzulernen, soll in dieser Übungsaufgabe ein solcher Sequencer in Teilen selbst implementiert werden.

5.1 Vorbereitung: Verwaltung von Teilnehmeradressen (für alle)

Protokollschichten werden in JGroups in Klassen realisiert, die sich von `org.jgroups.stack.Protocol` ableiten. Die Kommunikation zwischen den Schichten erfolgt dabei über verschiedene *Ereignisse*. Die wichtigsten Methoden sind in diesem Zusammenhang `Object up(Event evt)` und `Object down(Event evt)`: Die `up`-Methode wird von Schichten, die im Protokoll-Stack weiter unten liegen, aufgerufen, beispielsweise wenn Nachrichten über die Gruppenkommunikation eintreffen. Die `down`-Methode wird dagegen von höher liegenden Schichten unter anderem zum Versand von Nachrichten verwendet.

In der ersten Teilaufgabe sollen Nachrichten abgefangen und behandelt werden, die die aktuelle Sicht eines Teilnehmers auf die Gruppe betreffen. Zum einen sollen bei einer Änderung der Gruppenzusammensetzung bzw. beim dadurch ausgelösten `VIEW_CHANGE`-Ereignis die wichtigsten Informationen (aktuelle View, Leader-Adresse) gespeichert werden. Zum anderen ist beim Auftreten des Ereignisses `SET_LOCAL_ADDRESS` die eigene von JGroups intern verwendete Adresse des Teilnehmers festzuhalten. Als Grundlage der Implementierung soll dazu die im Pub-Verzeichnis unter `/proj/i4vs/pub/aufgabe5` bereitgestellte Klasse `VSTotalOrder` dienen.

Aufgaben:

→ Erweiterung der Klasse `VSTotalOrder` um eine Methode, die beim Auftreten eines `SET_LOCAL_ADDRESS`-Ereignisses die lokale Adresse des Teilnehmers abspeichert

→ Behandlung des `VIEW_CHANGE`-Ereignisses: Speichern der aktuellen Sicht und der Adresse des Leader der Gruppe, sowie bestimmen, ob der jeweilige Teilnehmer selbst der Leader ist

Hinweise:

- Jedem `JChannel`-Objekt ist eine eigene Instanz des Protokoll-Stack zugeordnet, wobei sich diese wiederum aus Instanzen derjenigen Klassen zusammensetzt, die die Protokollschichten implementieren. Das heißt: Jedes `JChannel`-Objekt verwendet intern genau eine `VSTotalOrder`-Instanz. Es ist davon auszugehen, dass diese Instanz von mehreren Threads gleichzeitig verwendet werden kann. Entsprechend muss der Zugriff auf interne Zustände stets synchronisiert werden!
- Ereignisse von JGroups haben in den meisten Fällen ein Argument, das die zugehörigen Nutzdaten enthält; beispielsweise kann bei einem `VIEW_CHANGE`-Ereignis mittels `(View) evt.getArg()` auf die aktuelle Sicht zugegriffen werden.
- Als Leader soll das erste Gruppenmitglied einer View dienen (`view.getMembers().firstElement()`).

5.2 Multicast mit totaler Ordnung (für alle)

Im nächsten Schritt ist die Klasse `VSTotalOrder` derart anzupassen, dass von ihr ein Sequencer realisiert wird. Dabei kann auf die Funktionalität von im Protokoll-Stack weiter unten liegenden Schichten aufgebaut werden: Das JGroups-Protokoll `NAKACK` setzt bereits eine FIFO-Ordnung um. Demnach ist sichergestellt, dass bei sämtlichen Teilnehmern einer Gruppe die Nachrichten eines jeden Teilnehmers in der Reihenfolge ankommen, in der sie von diesem versendet wurden. Damit wird jedoch nichts darüber ausgesagt, in welcher Reihenfolge Nachrichten von verschiedenen Teilnehmern weitergereicht werden. Wenn indes alle Nachrichten erst an den Leader der Gruppe gesendet und von diesem anschließend verteilt werden, verhält sich die FIFO- wie eine totale Ordnung.

Im Einzelnen sind folgende Schritte durchzuführen: Soll eine Multicast-Nachricht versendet werden, ist diese abzufangen und an den Leader zu senden. Dazu ist sie serialisiert an eine neu zu erstellende Nachricht anzuhängen. Der Leader leitet die Nachricht anschließend an alle Teilnehmer in der Gruppe weiter. Schließlich müssen die Teilnehmer die Originalnachricht wiederherstellen und an die höheren Protokollschichten weiterreichen.

Aufgaben:

→ Erweiterung der Klasse `VSTotalOrder` derart, dass Nachrichten nicht direkt versendet, sondern intern stets über den Leader geleitet werden

→ Testen der Implementierung mit Hilfe der im Pub-Verzeichnis bereitgestellten Skripte (`distribute.sh`, um einen Test zu starten; `checklogs.sh`, um die Ausgabe zu überprüfen); Anpassen der Datei `my_hosts`, um Testrechner auszuwählen (mindestens fünf Hostnamen `faui<`*`id`*`>`; eine Zeile je Host)

Hinweise:

- Zur Weitergabe von protokollspezifischen Informationen erweitern Protokollschichten in JGroups Nachrichten um eigene *Header*. Im Pub-Verzeichnis steht mit `VSTotalOrderHeader` eine Beispielklasse für einen solchen Header zur Verfügung, die für die eigene Implementierung verwendet und bei Bedarf beliebig erweitert werden kann. Darüber hinaus sind die Hilfsklassen `VSMsgID` (zur Identifizierung von Nachrichten) und `VSTotalOrderMsgType` (zur Unterscheidung verschiedener Nachrichtentypen) vorgegeben.
- Vereinfachend darf angenommen werden, dass sich die Gruppenzusammensetzung nach der Initialisierungsphase nicht mehr ändert. Insbesondere muss auch der Fall, dass die Leader-Rolle während des Multicasts einer Nachricht auf einen anderen Teilnehmer übergeht, nicht betrachtet werden.

5.3 Uniformer Multicast (für alle)

Die bisherige Implementierung stellt sicher, dass alle korrekten Teilnehmer Nachrichten in der gleichen Reihenfolge an höhere Schichten übergeben. Sollte jedoch beispielsweise der Leader während der Weiterleitung einer Nachricht ausfallen, lässt sich für die anderen Teilnehmer nicht mehr nachvollziehen, welche Nachrichten er vor seinem Ausfall noch nach oben weitergereicht hat, und welche nicht mehr. Um zu gewährleisten, dass auch fehlerhafte Teilnehmer stets die Reihenfolge der Nachrichten einhalten, muss eine Mehrheit aller Teilnehmer den Empfang einer Nachricht bestätigt haben, ehe sie weiter verarbeitet werden darf.

Das bedeutet: Wird eine weitergeleitete Nachricht vom Leader empfangen, muss diese zunächst zwischengespeichert werden. Der Empfang der Nachricht wird bestätigt, indem eine `ACK`-Nachricht an alle Gruppenteilnehmer versendet wird. Erst wenn ein `ACK` von der Mehrheit aller Teilnehmer empfangen wurde, darf eine Übergabe der Nachricht an höhere Schichten stattfinden.

Aufgaben:

→ Erweiterung der bestehenden Implementierung durch Empfangsbestätigungen
→ Überprüfung der korrekten Funktionsweise mittels `distribute.sh` und geeigneten Bildschirmausgaben

Hinweise:

- Die vom Leader vorgegebene Ordnung darf durch das Zwischenspeichern nicht verletzt werden.
- `ACK`s lassen sich den entsprechenden Nachrichten über deren IDs zuordnen.
- Es ist zu beachten, dass bei einem Teilnehmer die Empfangsbestätigungen von anderen Teilnehmern für eine Nachricht bereits vor der eigentlichen Nachricht eintreffen können.
- Unvollständige Nachrichten gilt es vor dem Weiterreichen in einer geeigneten Datenstruktur zu verwalten.

5.4 Optimierung für größere Nachrichten (optional für 5,0 ECTS)

Der bisherige Ansatz sämtliche Nachrichten vom Leader verteilen zu lassen hat gerade bei größeren Nachrichten einige Defizite. So werden die Originalnachrichten erst vom eigentlichen Sender zum Leader übertragen und anschließend von diesem an alle anderen Teilnehmer. Dies erzeugt nicht nur zusätzliche Last im Netzwerk, sondern ebenso beim Leader. Eine alternative Implementierung besteht darin, die Nachrichten direkt von den Teilnehmern verschicken zu lassen und lediglich die Ordnung über den Leader zu erstellen. Hierbei versendet der Leader eine die Reihenfolge festlegende Ordnungsnachricht, sobald er die ursprüngliche Nachricht empfangen hat. In der Konsequenz heißt das, dass ein Teilnehmer bei diesem Ansatz also nicht nur auf die eigentliche Nachricht und die dazugehörigen Empfangsbestätigungen, sondern auch auf die korrespondierende Ordnungsnachricht warten muss, bevor er die Nachricht an höhere Schichten weiterreichen darf.

Aufgaben:

→ Umsetzung der alternativen Herangehensweise
→ Testen der eigenen Implementierung

Hinweise:

- Die in Teilaufgabe 5.3 erstellte Implementierung soll so weit wie möglich wiederverwendet werden.
- Die hier implementierte Optimierung für große Nachrichten soll sich (z. B. mittels einer klassenspezifischen Boolean-Konstanten) statisch an- bzw. ausschalten lassen können.
- In der optimierten Variante darf ein Teilnehmer erst dann ein `ACK` für eine Nachricht senden, wenn ihm sowohl die eigentliche Nachricht als auch deren Ordnungsinformation vorliegt.
- Es ist zu beachten, dass die Nachricht mit den Nutzdaten sowie die dazugehörigen Empfangsbestätigungen und Ordnungsnachrichten in beliebiger Reihenfolge bei einem Teilnehmer ankommen können.

Abgabe: am Mi. 2.7.2014 in der Rechnerübung

Die für diese Übungsaufgabe erstellten Klassen sind in einem Subpackage `vsue.totalorder` zusammenzufassen.