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 zumindest in Teilen selbst implementiert werden.

5.1 Änderung der Gruppenzusammensetzung (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)`. Erstere wird von Schichten, die im Protokoll-Stack weiter unten liegen, aufgerufen, beispielsweise wenn Nachrichten über die Gruppenkommunikation empfangen werden. Hingegen wird letztere 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 gespeichert werden. Zum anderen ist beim Auftreten des Ereignisses `SET_LOCAL_ADDRESS` die eigene von JGroups intern verwendete Adresse festzuhalten. Als Grundlage soll dazu die im Pub-Verzeichnis 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 Ereignisses `VIEW_CHANGE`: 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 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 stets über den Leader geleitet werden

→ Anpassen der Datei my_hosts, um Testrechner auszuwählen (mind. 5 Hostnamen faui*; eine Zeile je Host)

→ Testen der Implementierung mit Hilfe der bereitgestellten Skripte (`distribute.sh`, um Test zu starten; `checklogs.sh`, um Ausgabe zu überprüfen)

Hinweise:

- Protokollschichten können Nachrichten für interne Zwecke *Header* hinzufügen, die mit der Nachricht versendet werden und so benötigte Informationen transportieren können.
- Folgende Hilfsklassen werden vorgegeben: `VSMsgID`, als ID für jede gesendete Nachricht; `VSTotalOrderMsgType`, um Protokoll-interne Nachrichten zu unterscheiden; `VSTotalOrderHeader`, als Header für die Protokollschicht. Sofern nötig können diese Klassen auch für die eigene Implementierung angepasst werden.
- Änderungen der aktuellen Sicht der Gruppe, die nach der Initialisierungsphase auftreten, werden aus Gründen der Vereinfachung nicht betrachtet.

5.3 Multicast mit Bestätigung (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, kann nicht mehr nachvollzogen werden, welche Nachrichten er kurz vor dem Ausfall bereits nach oben weitergereicht hat. Um zu gewährleisten, dass auch fehlerhafte Teilnehmer stets die Reihenfolge der Nachrichten einhalten, muss die 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 erst zwischengespeichert werden. Der Empfang der Nachricht wird bestätigt, indem eine `ACK`-Nachricht an alle Gruppenteilnehmer versendet wird. Erst wenn ein `ACK` von allen Teilnehmern empfangen wurde, darf eine Übergabe der Nachricht an höhere Schichten stattfinden.

Aufgaben:

→ Die bestehende Implementierung ist durch Empfangsbestätigungen zu erweitern.
→ Die korrekte Funktionsweise ist mittels der oben genannten Testanwendung und geeigneten Bildschirmausgaben zu überprüfen.

Hinweise:

- Die vom Leader vorgegebene Ordnung darf durch das Zwischenspeichern nicht verletzt werden.
- `ACK`s können den entsprechenden Nachrichten über deren IDs zugeordnet werden.
- Der Einfachheit halber soll auf ein `ACK` von jedem Teilnehmer gewartet werden und nicht lediglich auf die einer Mehrheit.
- `ACK`s können auch schon vor der eigentlichen Nachricht eintreffen.

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. Dies erzeugt nicht nur zusätzliche Last im Netzwerk, sondern ebenso beim Leader. Eine andere Möglichkeit besteht darin, die Nachrichten direkt von den Teilnehmern verschicken zu lassen und lediglich die Ordnung über den Leader zu erstellen. Hierbei versendet der Leader die Reihenfolge festlegende Ordnungsnachrichten, wenn er die ursprüngliche Nachricht empfängt. Somit müssen die Teilnehmer nicht nur auf die eigentlichen Nachrichten und die `ACK`s warten, ehe sie empfangene Nachrichten an höhere Schichten weiterreichen, sondern ebenso auf die Nachrichten mit der Ordnung.

Aufgaben:

→ Umsetzung der alternativen Herangehensweise
→ Testen der eigenen Implementierung

Hinweise:

- Die in den anderen Teilaufgaben erstellte Implementierung soll so weit wie möglich wiederverwendet werden und selbst weiterhin lauffähig bleiben.
- Eigentliche Nachrichten, `ACK`s und Ordnungsnachrichten können in beliebiger Reihenfolge bei den Teilnehmern ankommen.

Abgabe: am 22.7.2011 in der Rechnerübung

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