3 Übungsaufgabe #3: Fernaufrufsemantiken

In den bisherigen Aufgaben wurde davon ausgegangen, dass die für einen Fernaufruf notwendige Kommunikation zwischen zwei Rechnern stets fehlerfrei abläuft. Diese Annahme lässt sich in realen Netzwerken jedoch üblicherweise nicht aufrechterhalten, weshalb Fernaufrufsysteme so ausgelegt werden sollten, dass sie auch beim Auftreten von Netzwerkfehlern korrekt funktionieren. Wie in der Vorlesung erläutert, lässt sich dies durch die Implementierung geeigneter Fernaufrufsemantiken erreichen.

Ziel dieser Übungsaufgabe ist es, das bestehende Fernaufrufsystem so zu erweitern, dass temporäre Kommunikationsfehler toleriert werden. Hierzu sollen die beiden Semantiken *Last-of-Many* und *At-Most-Once* unterstützt werden. Die Tolerierung von Rechnerausfällen wird im Rahmen dieser Übungsaufgabe nicht betrachtet, da sie zusätzliche Mechanismen, zum Beispiel zur persistenten Sicherung des Anwendungszustands, erforderlich macht.

3.1 Methodenspezifische Auswahl der Fernaufrufsemantik (optional für 5,0 ECTS)

Die unterschiedlichen Charakteristika verschiedener Fernaufrufsemantiken (z. B. die Anzahl möglicher Ausführungen) führen dazu, dass im Allgemeinen nicht dieselbe Semantik für alle Operationen eines Diensts anwendbar ist. Das eigene Fernaufrufsystem soll daher die Möglichkeit bieten, mittels einer Annotation `VSRPCSemantic` für jede Methode einer Remote-Schnittstelle einzeln festzulegen, welche Semantik jeweils zum Einsatz kommen soll:

```
public @interface VSRPCSemantic {
    VSRPCSemanticType value();
}
```

`VSRPCSemanticType` ist dabei eine Enum, die zur Unterscheidung der beiden im Kontext dieser Übungsaufgabe relevanten Fernaufrufsemantiken zwei Werte umfassen soll: `LAST_OF_MANY` und `AT_MOST_ONCE`.

Aufgaben:

→ Bereitstellung der Annotation `VSRPCSemantic`
→ Annotierung der Methoden der Schnittstelle `VSAuctionService` des Auktionsdiensts

Hinweis:

- Bei der Umsetzung von Last-of-Many und At-Most-Once in den folgenden Teilaufgaben 3.2 und 3.3 ist auf Modularität zu achten. Die Implementierung sollte so gestaltet sein, dass theoretisch weitere Semantiken integriert werden können, ohne die dann bestehenden Klassen signifikant modifizieren zu müssen.

3.2 Last-of-Many (für alle)

In dieser Teilaufgabe sind die Fernaufruf-Stubs und -Skeletons so zu erweitern, dass sie die Semantik *Last-of-Many* unterstützen. Diese Semantik ist im Wesentlichen durch die beiden folgenden Merkmale charakterisiert:

- Die von einem Client per Fernaufruf angestoßene Operation wird auf Server-Seite nicht nur einmal, sondern unter Umständen (d. h. im Fehlerfall) mehrmals ausgeführt.
- Dem Client werden nach jeder Ausführung stets die neuesten Ergebnisse zurückgeliefert.

Diese Eigenschaften der Last-of-Many-Semantik werden dabei durch folgende Vorgehensweise erreicht:

- Erhält ein Client nach dem Absenden einer Anfrage vom Server innerhalb eines vordefinierten Zeitintervalls keine Antwort, sendet er die Anfrage erneut.
- Empfängt der Server eine bereits bearbeitete Anfrage, so führt er die betreffende Operation ein weiteres Mal lokal aus. Das auf diese Weise erhaltene Ergebnis sendet er anschließend an den Client zurück.
- Ein Client wartet so lange, bis eine Antwort auf die letzte von ihm geschickte Anfrage zurück kommt. Diese reicht er an den Aufrufer weiter. Alle anderen Antwortnachrichten werden verworfen.

Die Umsetzung von Last-of-Many macht es erforderlich, den Server in die Lage zu versetzen, einzelne Fernaufrufe eines Clients eindeutig zu unterscheiden (→ Fernaufruf-ID). Darüber hinaus muss es dem Server möglich sein, die zum selben Fernaufruf gehörigen Anfragenachrichten auseinander zu halten (→ Einsatz von Sequenznummern).

Aufgabe:

→ Implementierung der Last-of-Many-Semantik

Hinweis:

- Auf Client-Seite soll eine maximale Anzahl von gesendeten Anfragen definiert werden können, nach der ein Fernaufruf als erfolglos abgebrochen wird, falls bis dahin kein valides Ergebnis vorliegt. Das Scheitern eines Fernaufrufs ist dem Aufrufer durch eine geeignete `RemoteException` zu signalisieren.

3.3 At-Most-Once (optional für 5,0 ECTS)

In dieser Teilaufgabe soll als weitere Semantik *At-Most-Once* zur Verfügung gestellt werden. Diese Semantik stellt sicher, dass der Server jede zu einer einzelnen Anfrage gehörende Operation nur höchstens einmal ausführt. Hierzu muss der Server in der Lage sein, anhand von Fernaufruf-IDs folgende Situationen zu unterscheiden:

- *Neue Anfrage*: Der Server erhält eine Anfrage mit einer ihm noch unbekannten Fernaufruf-ID. → Nach Ausführung der entsprechenden Operation sendet er das Ergebnis an den Client; zusätzlich speichert er das Ergebnis bei sich lokal ab.
- *Alte Anfrage*: Der Server erhält eine Anfrage mit einer ihm bereits bekannten Fernaufruf-ID. → Anstatt die Anfrage erneut zu bearbeiten und die Operation ein weiteres Mal auszuführen, sendet der Server das bereits vorliegende, lokal gespeicherte Ergebnis an den Client.

Da der Server über begrenzten Speicherplatz verfügt, ist ein Garbage-Collection-Mechanismus für nicht mehr benötigte Ergebnisse zu realisieren, der den Speicher auf Server-Seite zuverlässig aufräumt. Die gewählte Strategie soll dabei nur Ergebnisse verwerfen, deren zugehörige Fernaufrufe bereits garantiert beendet sind.

Aufgabe:

→ Implementierung der *At-Most-Once*-Semantik

Hinweis:

- Es darf angenommen werden, dass ein Client keinen Bedarf mehr für das Ergebnis eines Fernaufrufs hat, sobald er einen weiteren Fernaufruf durchführt.
- Im Fall eines Verbindungsabbruchs soll der Client die noch verbleibenden Anfragewiederholungen über eine neue Verbindung senden.

3.4 Testen der Implementierung (für alle)

Die Integration der in den vorherigen Teilaufgaben realisierten Semantiken in das bestehende Fernaufrufsystem soll abschließend unter Verwendung des Auktionsdiensts aus den vorherigen Übungsaufgaben getestet werden. Dabei sind folgende Netzwerkfehler zu berücksichtigen:

- Verlust von Nachrichten
- Verzögerung einzelner Nachrichten

Da die aktuelle Implementierung des Kommunikationssystems aufgrund des verwendeten TCP-Protokolls diese Fehler bisher toleriert, sollen bestimmte Fehlersituationen bewusst herbeigeführt werden. Zu diesem Zweck ist das Kommunikationssystem an geeigneten Stellen so zu sabotieren, dass es Nachrichten verwirft oder verzögert. Hierzu eignet sich insbesondere die Klasse `VSObjectConnection`, da in `{send,receive}Object()` eine direkte Zugriffsmöglichkeit auf einzelne Nachrichtenobjekte besteht. Es ist daher zum Beispiel sinnvoll, die Fehlererzeugung in einer Unterklasse `VSBuggyObjectConnection` zu realisieren.

Aufgaben:

→ Sabotage des Kommunikationssystems
→ Implementierung geeigneter Testfälle

Hinweise:

- Mit den Testfällen sind nicht nur die Fehlerarten bei der Nachrichtenübermittlung einzeln, sondern auch Kombinationen aus diesen Fehlern abzudecken.
- Auf welche Weise die geforderten Kommunikationsfehler tatsächlich simuliert werden, ist freigestellt. Die Verwendung einer `VSBuggyObjectConnection` dient in diesem Zusammenhang nur als Vorschlag.
- Es darf angenommen werden, dass gesendete Anfrage- und Antwortnachrichten entweder vollständig oder überhaupt nicht auf der Gegenseite ankommen. Der teilweise Verlust von Nachrichten bzw. von einzelnen Objekten einer Nachricht muss nicht betrachtet werden.

Abgabe: am Mi., 21.6.2017 in der Rechnerübung

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