

## Übungsblatt 5

**Ausgabe:** 22.11.2011

**Abgabe:** 02.12.2011

---

**Aufgabe 1** (RPC-Framework mit UDP und Reflections): (20 Punkte)

Ziel dieser Aufgabe ist es, ein RPC-Framework mit UDP zu programmieren, welches Fernaufrufe von statischen Methoden ermöglicht (call-by-value).

- a) (7 Punkte) Laden Sie sich das Framework für die Aufgabe von der Veranstaltungsseite herunter. Benutzen Sie zum Lösen der Aufgaben die vorgegebenen Interfaces und Klassen. Das Manipulieren bestehender Interfaces/Klassen ist nicht erlaubt, lediglich das Hinzufügen von Methoden-Rümpfen, Methoden oder neuen Klassen. Machen Sie sich mit dem Reflection Tutorial vertraut. Informieren Sie sich im Besonderen über die Methoden *Class.forName*, *Class.getMethod* und *Method.invoke*. Implementieren Sie die *RPCServiceProvider*-Schnittstelle und testen Sie lokal mit Aufrufen, wie z.B. *remote.call("testpackage.Testclass", "myMethod")*.

**Hinweise:**

- Damit Sie die gewünschte Methode finden, müssen Sie die Klassen der Aufruf-Parameter herausfinden. Dies ist mit *Object.getClass* möglich.
  - Denken Sie daran, dass der Anwender des Framework auch über Versagen informiert werden möchte. Also sollten Sie eine *RPCException* werfen, wenn ein Versagen aufgedeckt wird. Dabei darf natürlich die eigentliche Exception (davon kann es sehr viele verschiedene geben) nicht verloren gehen.
  - Wie Sie beim Testen feststellen sollten, kann man in Java (ohne Tricks) keine primitiven Datentypen übertragen. Weil statische Methoden in der Java-Library aber meistens auf primitiven Datentypen operieren und es bei Benutzung dieses Frameworks sehr umständlich wäre, für jede dieser Methoden einen Wrapper zu schreiben, gibt es einen Workaround.  
Mit *RPCSecrets.warpToPrimitiveClass* können Sie aus einer Objekt-Klasse (falls möglich) eine Primitiv-Klasse erstellen und damit dann auch Methoden mit primitiven Parametern aufrufen.
- b) (7 Punkte) Jetzt soll das Aufrufen von Methoden (unter Benutzung des *RPCLocalServiceProvider* aus Aufgabenteil a) über das Netzwerk funktionieren.

Die Kommunikation zwischen dem *RPCServerServiceProvider* und dem Client soll auf der Übertragung von UDP-Paketen basieren, welche Protobuf-Nachrichten beinhalten. Schauen Sie sich also die Definitionen in **rpc.proto** an.

Implementieren Sie nun den **Server**-Teil.

**Hinweise:**

- Dieser sollte in einem eigenen Thread laufen.
- Benutzen Sie zum (De-)Serialisieren der Parameter die Methoden der Klasse *RPCSecrets*.
- Achten Sie auf geeignetes Exception-Handling. Exceptions des *RPCLocalServiceProvider* sollten zurück an den Aufrufer übertragen werden, IOExceptions von dem DatagramSocket nicht.

- c) (6 Punkte) Implementieren Sie die *RPCServiceProvider*-Schnittstelle erneut (der **Client** des Frameworks). Diesmal soll jede Anfrage in ein RPC umgewandelt werden und an den *RPCServerServiceProvider* verschickt werden. Serialisieren Sie dazu zuerst alle Parameter, bauen sie eine *RPCCall*-Nachricht und schicken Sie diese an eine im Konstruktor (von außen) gesetzte Adresse. Empfangen Sie nun ein UDP-Paket (wir ignorieren hier den Fakt, dass jemand Fremdes Pakete an uns schicken könnte) und verarbeiten Sie das Ergebnis.

**Hinweise:**

- Achten Sie auch hier wieder auf sinnvolles Exception-Handling.
- Benutzen Sie *Builder.mergeFrom(byte[] data, int off, int len)*, wenn Ihnen das Parsen von Protobuf-Nachrichten ein Problem bereitet.

- optional: d) (2 Punkte) Bei dieser Implementierung kann nur eine Anfrage gleichzeitig verarbeitet werden. Wie kann man das Problem lösen? Lösen Sie es! Achten Sie auf geeignete Sperr-Synchronisierung.

- optional: e) (4 Punkte) Bei dieser Implementierung kann es dazu kommen, dass ein UDP-Paket verloren geht und der Client dann "feststeckt". Wie kann man das Problem lösen? Lösen Sie es!

Timeouts und Retransmission oder TCP wäre ein guter Ansatz. Allerdings muss man sich dann die Frage stellen, ob man die ganze Zeit eine Verbindung aufrecht erhält, oder ob für jede Anfrage die Verbindung wiederherstellt.

**Achtung!** Sollten Sie TCP benutzen wollen, müssen Sie neue Klassen schreiben. Wenn Ihre Abgabe nur eine TCP-Lösung enthält, erhalten Sie weniger Punkte in den anderen Aufgaben-Teilen.