Protocol Buffers

Wie in der Vorlesung besprochen, automatisieren die *Protocol Buffers* [1] die Serialisierung und De-Serialisierung von Nachrichten und können so als Basis für eine nachrichten-orientierte Kommunikation dienen.
Basierend auf dem Socket–Client und dem Socket-Server der Serie 2 soll nun ein *Log-Server* entstehen, der Nachrichten in einem Log-Format, das wir mit Protocol-Buffers definieren wollen, entgegen nimmt und in einem Log-File protokolliert.

12. Laden Sie bitte Protocol-Buffers von der Protocol-Buffers-Homepage [1] (Datei `protobuf-2.6.1.tar.gz`, auch auf dem Handout-Server).

13. Wir müssen Protocol-Buffers selbst übersetzen:

    ```
    % mkdir $HOME/protobuf
    % ./configure --prefix=$HOME/protobuf
    % make; make install
    ```

14. Definieren Sie bitte das Nachrichten-Format für den Log-Server in Form eines `.proto`-Files, beispielsweise mit dem Namen `LogMessage.proto`. Es solle diese Einträge haben:

    - einen Zeitstempel, der angibt, wann der Log-Eintrag erzeugt wurde,
    - optional einen Identifikator, der angibt, wer diesen Log-Eintrag erzeugt hat und wo das war,
    - eine Dringlichkeitsstufe (`DEBUG`, `INFO`, `WARN`, `ERROR`, `FATAL`), die angibt, wie die Nachricht einzustufen ist und
    - einen einzeiligen Text, den Inhalt des Log-Eintrags

15. Generieren Sie bitte aus Ihrer Nachrichtendefinition Java-Code (`protoc --java_out=. LogMessage.proto`)

16. Ändern Sie bitte Ihren Socket-Client so ab, dass er entsprechende Log-Nachrichten versendet.

17. Ändern Sie bitte Ihren Socket-Server so ab, dass er Log-Nachrichten empfängt und sie in geeigneter Weise in eine Protokoll-File schreibt (je Log-Eintrag eine Zeile).

RPC mit Protocol Buffers

Wenn wir Remote-Procedure-Calls[1] mit Protocol-Buffers realisieren wollen, so müssen wir in der Anfrage mitteilen, welche Operation wir aufrufen wollen. Dies führt zu Nachrichten-Definitionen der Art [2]:

```
message RPC_Request {
  enum Operation { OP1=1; OP2=2; ... }
  required Operation operation = 1;
  optional Op1_request op1_arg = 2;
  optional Op2_request op2_arg = 3;
  optional Op3_request op3_arg = 4;
  ...
}
```

Das Feld `operation` legt fest, welche Operation aufgerufen werden soll. Beim konkreten Aufruf wird in in der RPC-Anfrage-Nachricht in Abhängigkeit des Wertes von `operation` *einer* der optionalen Parameter `op1_arg`, `op2_arg`, ... angegeben, der die Anfrage-Parameter für die gewählte Operation enthält. Die Typen `Op1_request`, `Op2_request`, ... von `op1_arg`, `op2_arg`, ... können zusammengesetzte Nachrichten sein, so dass effektiv auch mehr als ein Parameter übergeben werden kann.
RPC-Antworten sind ganz normale Protocol-Buffer-Nachrichten, die vom Server nach Ausführung der Operation an den Client verschickt werden.
Wir wollen den einfachen Datenbank-Server der Serie 3 nun mit Protocol-Buffers realisieren.
Der Protocol–Buffer–Datenbank–Server soll wiederum drei Operationen unterstützen:

`getRecord` liest anhand eines Indexes (vom Typ `int32`) einen Datensatz (vom Typ `string`) aus der Datenbank aus,

`addRecord` schreibt einen Datensatz unter einem Index in die Datenbank und

`getSize` ermittelt, wieviele Datensätze gespeichert sind.

18. Definieren Sie bitte das Nachrichten-Format für den Datenbank-Server mit Protocol-Buffers so, dass er `getRecord`, `addRecord` und `getSize` unterstützt.

19. Modifizieren Sie bitte Ihren Socket–Server aus Aufgabe 6 bzw. 8 so, dass er so definierte Protocol-Buffers-Anfragen akzeptiert und passend beantwortet. Auch mit Protocol-Buffers haben wir die Wahl, unmittelbar Protocol-Buffers-Nachrichten über den Socket zu verschicken, oder aber sie wiederum über HTTP zu übertragen. Bitte diskutieren Sie erneut die Unterschiede beider Vorgehen.

20. Modifizieren Sie bitte Ihren Socket–Client aus Aufgabe 5 bzw. 9 so, dass er passende Protocol-Buffer–Anfragen stellen und die Antworten passen verarbeiten kann.

21. Der Socket–Client soll die Daten der folgenden Tabelle in die Datenbank schreiben:

4101	Appen
4102	Ahrensburg
4103	Wedel
4104	Aumühle
4105	Seevetal
4106	Quickborn

    und Datensätze für die Indizes 4103 und 4107 lesen.

22. Rufen Sie bitte auch die Operation `getSize()` auf, um festzustellen, ob die richtige Anzahl von Datensätzen vorhanden ist.

Links

[1] `https://developers.google.com/protocol-buffers/`
[2] `https://developers.google.com/protocol-buffers/docs/techniques#union`
[3] `https://developers.google.com/protocol-buffers/docs/proto#services`

[1] Jüngere Versionen von Protocol-Buffers unterstützen die Definition von Request/Response-Paaren auch direkt über das `rpc`–Schlüsselwort, vgl. [3]. Wir verwalten RPC-Nachrichten hier *per Hand*.