universität ulm

Verteilte Systeme

Institut für Verteilte Systeme | Wintersemester 2025/2026
Alexander Heß, Prof. Dr.-Ing. Franz J. Hauck

Übungsblatt 2: ZeroMQ

Abgabetermin: 06.11.2025, 16:00Uhr

Aufgabe 1: Number Guessing Game (4)

In dieser Aufgabe sollen Sie sich zunächst mit der JeroMQ Bibliothek[1] vertraut machen, welche eine Java Implementierung von ZeroMQ bereitstellt.

Dazu sollen Sie einen Client für ein kompetitives Zahlrenratespiel entwickeln. Hierfür steht ein weiterer Server bereit, der JMQ Requests erwartet und unter der Adresse **vs.lxd-vs.uni-ulm.de:27401** erreichbar ist. Der Server generiert immer wieder neue zufällige positive ganze Zahlen, sowie dazugehörige `ID`s die das aktuelle Spiel identifizieren, jeweils aus dem Intervall $[1, 2^{63} - 1]$. Der Server erwartet Anfragen als Strings mit folgendem Format "`<gameID>:<guess>`", terminiert mit einem Zeilenumbruch. Bei der ersten Anfrage sollten Sie eine zufällige `gameID` wählen um die tatsächliche aktuelle `gameID` vom Server zu erhalten.

Ziel des Spiels ist es die aktuelle Zufallszahl zuerst zu erraten. Eine kleine zusätzliche Schwierigkeit ist hierbei, dass die aktuelle Zahl bei jedem Fehlversuch eines Spielers um Eins erhöht wird. Der Server unterscheidet folgende Fälle und gibt entsprechend die folgenden Antworten zurück:

- **Die Zahl wurde bereits von einem Mitspieler erraten:**
  "This number has been guessed already. The current gameID is <gameID>"
- **Die übergebene `gameID` ist unbekannt:**
  "GameID unknown! The current gameID is <gameID>"
- **Die übergebene Zahl ist zu klein:**
  "Attempt: <numAttempts> - too small"
- **Die übergebene Zahl ist zu groß:**
  "Attempt: <numAttempts> - too large"
- **Die übergebene Zahl ist korrekt:**
  "Correct guess after <numAttempts> attempts! The new gameID is <gameID>"
- **Die Syntax der Anfrage ist nicht korrekt:**
  "Invalid request format! Expected request syntax - <gameID>:<guess>"

[1] https://mvnrepository.com/artifact/org.zeromq/jeromq/0.6.0

Aufgabe 2: Factorization Challenge (6)

In dieser Aufgabe soll es ebenfalls um ein kompetitives Spiel gehen, dieses Mal dreht es sich jedoch um die Primfaktorzerlegung. Hier sollen Sie nun einen Client implementieren, welcher zusammengesetzte Zahlen der Form $n = p \cdot q$ von einem Publisher-Dienst entgegen nimmt und deren Faktoren p und q berechnet. Bei den zwei Faktoren p und q handelt es sich um zufällig generierte, verschiedene Primzahlen mit einer Länge von 32 Bit.

Für diese Aufgabe steht unter **vs.lxd-vs.uni-ulm.de:27378** ein ZeroMQ Publisher Dienst bereit, welcher nach einer Subscription zusammengesetzte Zahlen $n = p \cdot q$ zur Basis 10 als Strings versendet. Außerdem ist unter **vs.lxd-vs.uni-ulm.de:27379** ein ZeroMQ Reply Socket gebunden, welcher die Faktoren im Format "n:p:q" und ebenfalls zur Basis 10 entgegen nimmt.

Für diese Aufgabe finden Sie im Moodle die Klasse `Fermat.java`, welche eine Methode für die Faktorisierung basierend auf Fermat's Algorithmus bereitstellt.

1. Die Klasse `Controller` soll sich mit Hilfe eines ZeroMQ Subscribe Sockets mit dem bereitgestellen Dienst verbinden, immer wieder neue Challenges entgegen nehmen, und diese mittels ZeroMQ Push Sockets an eine `Worker` Instanz zur Faktorisierung weiterleiten. Über einen ZeroMQ Pull Socket sollen die entsprechenden Faktoren dann wieder entgegen genommen und über einen ZeroMQ Request Socket an den Challenge Publisher als String mit dem Format "n:p:q" zurückgegeben werden.
2. In der Klasse `Worker` sollen die Challenges über ZeroMQ Pull Sockets entgegen genommen und faktorisiert werden. Hierzu können Sie die Funktionalität der bereitgestellten Klasse `Fermat` nutzen, oder selbst eine effizientere Faktorisierungsstrategie implementieren. Nach der Faktorisierung sollen die Faktoren dann mittels ZeroMQ Push Sockets an den `Controller` zurückgegeben werden.
3. In der `Test` Klasse sollen Sie nun die Funktionalität ihrer Implementierung demonstrieren, indem Sie eine `Controller` und eine `Worker` Instanz starten.

Die Architektur ihrer Implementierung sollte folgendermaßen aussehen: