Architektur verteilter Anwendungen

Übung 1

Christoph Geidt

3584097

Idee

Die Implementierung der Knoten wurde in Java vorgenommen. Grund für die Wahl von Java war die starke Typisierung der Programmiersprache, was zu einer guten Wartbarkeit und Verständlichkeit des Codes beiträgt. Dies war für mich von hoher Relevanz, um gegebenen Falls den geschriebenen Code in den weiteren Übungen verwenden zu können.

Ein weiteres Argument für die Verwendung von Java war die Plattformunabhängigkeit.

Nachrichtenformat

Da in den Übungsaufgaben nur meine Knoten mit der Java-Implementierung untereinander kommunizieren, wurde zur Nachrichtenübermittlung die von Java mitgebrachte Serialisierung verwendet.

Dies wurde umgesetzt, indem alle Nachrichten das Interface „Serializable“ implementieren.

Wäre es erforderlich, dass die Knoten mit in anderen Programmiersprachen geschriebenen Knoten kommunizieren, müsste ein anderes Nachrichtenformat wie JSON, XML oder YAML verwendet werden.

Softwarestruktur

GraphManager

Der GraphManager ist eine Hilfsklasse für die Knoten die Informationen auf Basis der Graph- und der Konfigurationsdatei liefert.

Das ist einmal eine Methode die für eine übergebene ID alle Nachbarn im Graphen sowie eine Methode die für eine ID alle Hostinformationen zurückgibt.

NodeManager

Der NodeManager startet den Knoten und steuert ihn. Er liest Eingaben ein und führt die entsprechende Funktionalität auf dem Knoten aus.

Als Eingabeparameter bzw. Optionen nimmt der NodeManager folgendes entgegen:

- **id**
  Die ID des eigenen Knoten. Der Knoten, der gestartet wird.
- **hosts**
  Konfigurationsdatei mit den Knoteninformationen
- **graph**
  Datei mit dem Graph
- **rumor**
  Anzahl nach wie häufigen Hörens eines Gerüchtes das Gerücht geglaubt werden soll.

Logger

Der Logger gibt einen übergebenen Text mit einem aktuellen Zeitstempel aus. Hierbei wird unterscheiden zwischen „log“, „err“ und „debug“.

- **log**
  Wird immer ausgegeben. Hier wird es zur Ausgabe verwendet ob ein Knoten das Gerücht glaubt.
- **err**
  Wird verwenden wenn es zu einem Fehler kommt. Zum Beispiel in einem catch-Block.
- **debug**
  Wird nur ausgegeben wenn beim Initialisieren des Loggers der Modus auf „Debug“ gesetzt wird. Dies kann als Parameter beim Starten des Knoten übergeben werden.

Kommunikation

Unter die Kommunikation fallen alle Komponenten die etwas mit der Kommunikation der Knoten untereinander zu tun haben.

Zur Kommunikation wurden Sockets(TCP) verwenden.

Server

Im Serverteil wird ein Socket in einem neuen Thread geöffnet, der auf Nachrichten von anderen Knoten wartet.

Beim Empfang einer Nachricht werden Änderungen mittels Referenz auf den eigenen Koten aus dem Nebenläufigen Thread vorgenommen.

Client

Im Clientteil wird in einem neuen Thread eine Socketverbindung aufgebaut und die Nachricht an einen anderen Knoten gesendet.

Nachrichten

Für die Kommunikation wird eine Nachricht verwenden, die zur Übertragung mittels des in Java vorhandenen Serialisierers serialisiert wird.

Es wurde ein Container „Message“ programmiert, der Seder-ID, den Typ der Nachricht und die eigentliche Nachricht an sich enthält. Die Nachricht an sich muss das Interface „Serializable“ implementieren, sodass sichergestellt ist, dass die Nachricht auch noch serialisiert werden kann.

Konfigurationsdatei

Zum Einlesen der Hostinformationen der anderen Knoten wird eine Konfigurationsdatei im JSON-Format verwendet. Sie besteht aus einem Array welches Objekte mit den Attributen „id“, „hostname“ und „port“ enthält. Jedes Objekt entspricht einem Knoten.

Zum Parsen der JSON-Objekte wird die von gson-Bibliothek von google verwendet. Bei Fehlern in der Konfigurationsdatei wird eine entsprechende Fehlermeldung ausgegeben.

Im Folgenden ist ein Beispiel einer Konfigurationsdatei zu sehen:

```
[
    {
        "id":"1",
        "hostname":"127.0.0.1",
        "port":55551
    },
    {
        "id":"2",
        "hostname":"127.0.0.1",
        "port":55552
    },
    {
        "id":"3",
        "hostname":"127.0.0.1",
        "port":55553
    },
    {
        "id":"4",
        "hostname":"127.0.0.1",
        "port":55554
    },
    {
        "id":"5",
        "hostname":"127.0.0.1",
        "port":55555
    }
]
```

Besonderheiten der Implementierung
Aufgrund des Containers zur Nachrichtenübertragung kann ganz einfach auch eine andere Nachricht übertragen werden. Es sind keine Änderungen an der Struktur erforderlich um das Programm zur Simulation mit anderen Nachrichten außer zu verwenden.

Zum Starten mehrerer Knoten und zum Durchführen der Tests wurde ein Batch-Skript geschrieben:

```
:: Variablen aus Aufgabenstellung
set n=5
set m=7
set c=3

:: Erzuegen eines Graphen mit Hilfe mit dem GraphGen Programm
java -jar C:\Users\cgeidt\AVA\GraphGen\dist\GraphGen.jar -nodes %n% -edges %m% -output "C:\Users\cgeidt\Desktop\AVA\nodegraph"

:: Erstellen einer Bilddatei des erzuegten Graphen
dot -Tpng nodegraph -o nodegraph.png

:: Öffnen der erzeugten Bilddatei
nodegraph.png

:: Starten der einzelnen Nodeinstanzen
for /l %%x in (1, 1, %n%) do (
   echo %%x
   start "Node %%x" java -jar C:\Users\cgeidt\AVA\ava_ueb01\out\artifacts\ava_ueb01_jar\ava_ueb01.jar -id %%x -hosts hostlist.json -graph nodegraph -rumor %c%
)
```

Beispielabläufe

Die Beispielabläufe aus Übung 1 Aufgabe 4 ergaben folgende Ergebnisse:

n	m	c	Anzahl der glaubenden Knoten(Initiator glaubt immer)
3	3	2	3
4	4	2	3
4	5	2	4
4	6	2	4
5	4	2	1
5	5	2	3
5	6	2	5
6	6	2	3
6	7	2	4
6	8	2	5
6	9	2	5
6	10	2	6
4	5	3	2
4	6	3	4
5	4	3	1
5	5	3	1
5	6	3	2
6	6	3	1
6	7	3	2
6	8	3	3
6	9	3	4
6	10	3	5

Fazit

Es ist während der Implementierung zu keinen größeren Problemen gekommen. Die Sockets in Java haben sich gut benutzen lassen und vorhandene Serialisierung von Objekten mittels der in Java vorhandenen Mittel hat ebenfalls gut funktioniert.

Aufgefallen ist, dass das Debuggen durch das Verwenden mehrere Threads sich als schwieriger gestaltet.

Beim Betrachten der Ergebnisse der Beispielabläufe ist zu sehen, dass das Verhältnis zwischen Kanten(m) und Knoten(n) sowie der Konstanten(c) eine Rolle spielt wie viele Knoten das Gerücht glauben. Ist die Anzahl der Kanten im Verhältnis zu den Knoten groß glauben mehr Knoten das Gerücht. Bei steigendem c wird die Anzahl der Knoten die das Gerücht glauben geringer.