

Ergebnisbericht zur Übung 2 Verteilte Systeme

Aufgabe 2: Fireworks

Ziel der Aufgabe

In dieser Aufgabe sollte eine einfache verteilte Anwendung basierend auf dem Simulator aus Aufgabe 1 entworfen werden. Diese Anwendung besteht aus einer konfigurierbaren Anzahl von Knoten, die als Aktoren modelliert werden. Die Knoten sollten nur im aktiven Zustand Nachrichten versenden und im passiven Zustand durch eine eintreffende Nachricht wieder aktiviert werden können.

Implementierung

FireworkNode

Die Klasse `FireworkNode` **erweitert die Basisfunktionalität der `Node.java`** und modelliert das Verhalten eines Knotens in der verteilten Anwendung. Jeder Knoten wechselt zwischen aktivem und passivem Zustand und sendet bzw. empfängt `Firework`-Nachrichten.

Wichtige Methoden und ihre Funktionen:

engage:

- Diese Methode steuert das Verhalten des Knotens. Zu Beginn wartet der Knoten eine zufällige Zeitspanne, bevor er seine erste Nachricht sendet. Anschließend wechselt der Knoten in den passiven Zustand und wartet auf eingehende Nachrichten. Wenn der Knoten eine `Firework`-Nachricht empfängt, wird er aktiviert und entscheidet basierend auf einer Wahrscheinlichkeit `p`, ob er eine weitere Nachricht sendet. Nach jeder Entscheidung wird `p` verkleinert. Diese Schleife wiederholt sich, bis der Knoten eine vordefinierte Anzahl von Iterationen (`Max_Iterations`) erreicht hat, nach der er seine Operationen beendet.

sendFireworkMessage:

- Sendet eine `Firework`-Nachricht an eine zufällige Teilmenge der anderen Knoten im Netzwerk. Diese Methode stellt sicher, dass Nachrichten nur an andere Knoten gesendet werden und nicht an sich selbst.

handleFireworkMessage:

- Behandelt den Empfang von `Firework`-Nachrichten. Wenn der Knoten eine solche Nachricht empfängt, wird er aktiv und es wird geprüft, ob er basierend auf der aktuellen Wahrscheinlichkeit `p` eine weitere Nachricht sendet. Nach jeder Entscheidung wird `p` reduziert, um die Wahrscheinlichkeit zu verringern, dass der Knoten erneut eine Nachricht sendet.

FireworkSimulation

Die Klasse `FireworkSimulation` **erweitert die Basisfunktionalität der Simulator**, initialisiert die Knoten und startet die Simulation.

Wichtige Funktionen:

Initialisierung:

- **Die Anzahl der Knoten wird vom Benutzer eingegeben.** Anschließend werden die Knoten mit ihren Namen und der initialen Wahrscheinlichkeit für das Senden weiterer Nachrichten erstellt und initialisiert.

Simulation starten:

- Die Simulation wird für eine **definierte Dauer (Max_Iterations)** gestartet. Während dieser Zeit führen die Knoten ihre definierten Operationen aus, senden und empfangen Nachrichten und wechseln zwischen aktivem und passivem Zustand.

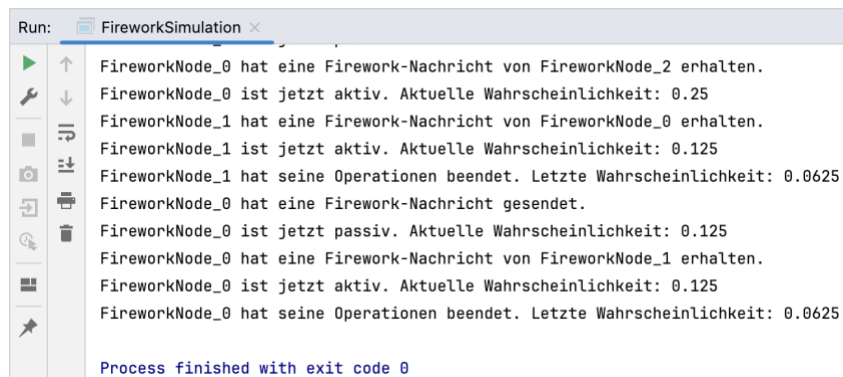
Beenden der Simulation:

- Nach Ablauf der Simulationsdauer wird der Simulator heruntergefahren und alle Ressourcen werden freigegeben. Also nach eine bestimmte Zeit X gibt es keine aktiven Knoten mehr und werden keine Operationen mehr durchgeführt. *Allerdings manchmal terminiert das Programm selbst, und manchmal muss man ihn selber terminieren. Da Terminierung im Aufgabe 3 gemacht werden musste, wurde hier auf weitere Implementierung verzichtet.*

Ergebnisse der Simulation

Die Konsolenausgaben zeigten den Ablauf der Simulation und den Zustand der Knoten:

- Knoten wechselten korrekt zwischen aktivem und passivem Zustand.
- Nachrichten wurden gesendet und empfangen, um Knoten zu aktivieren.
- Die Wahrscheinlichkeit für das erneute Senden von Nachrichten wurde korrekt reduziert.
- Knoten beendeten (aber nicht immer) ihre Operationen nach der festgelegten Anzahl von Iterationen.



```
Run: FireworkSimulation x
FireworkNode_0 hat eine Firework-Nachricht von FireworkNode_2 erhalten.
FireworkNode_0 ist jetzt aktiv. Aktuelle Wahrscheinlichkeit: 0.25
FireworkNode_1 hat eine Firework-Nachricht von FireworkNode_0 erhalten.
FireworkNode_1 ist jetzt aktiv. Aktuelle Wahrscheinlichkeit: 0.125
FireworkNode_1 hat seine Operationen beendet. Letzte Wahrscheinlichkeit: 0.0625
FireworkNode_0 hat eine Firework-Nachricht gesendet.
FireworkNode_0 ist jetzt passiv. Aktuelle Wahrscheinlichkeit: 0.125
FireworkNode_0 hat eine Firework-Nachricht von FireworkNode_1 erhalten.
FireworkNode_0 ist jetzt aktiv. Aktuelle Wahrscheinlichkeit: 0.125
FireworkNode_0 hat seine Operationen beendet. Letzte Wahrscheinlichkeit: 0.0625

Process finished with exit code 0
```

Aufgabe 3: No more Fireworks?

Ziel der Aufgabe

Das Ziel der Aufgabe war die Erweiterung der Lösung aus Aufgabe 2 um eine nachrichtenbasierte Terminierungserkennung. Ein zusätzlicher Knoten, der Observer, sollte feststellen, dass alle Aktoren passiv sind und keine weiteren Firework-Nachrichten unterwegs sind.

Implementierung

Disclaimer: Da es mir jetzt nicht klar war, ob die Wahrscheinlichkeiten individuell für jeden Knoten separat sein müssen oder alle Knoten sich eine gemeinsame synchronisierte Wahrscheinlichkeit teilen müssen, wurden in diesem Aufgabenteil zwei Ansätze implementiert, um die Wahrscheinlichkeit zu handhaben, dass ein Knoten eine Firework-Nachricht sendet. **Der erste Ansatz** verwendet eine globale Wahrscheinlichkeit, die für alle Knoten gleich ist, da sie sich diese sozusagen teilen und bei jeder Entscheidung reduziert wird. Dies wurde in **ObserverFireworkNode.java** implementiert. **Der zweite Ansatz** verwendet eine individuelle Wahrscheinlichkeit für jeden Knoten, was eine realistischere Simulation ermöglicht, da Knoten in einem verteilten System oft unterschiedlich agieren und unterschiedliche Wahrscheinlichkeiten für bestimmte Aktionen haben. Dies wurde in **IndividualObserverFireworkNode.java** implementiert. **Die sim4da-Log-Dateien wurden ebenfalls geprüft, um sicherzustellen, dass alles korrekt funktioniert.**

Erweiterungen:

engage: Steuert das Verhalten des Knotens. Der Knoten wechselt zwischen aktivem und passivem Zustand, sendet Nachrichten und empfängt sie. **Die Methode aktualisiert den Zustand des Knotens beim Observer.**

-

Globale Wahrscheinlichkeit (ObserverFireworkNode.java)

Dieser Ansatz verwendet eine **globale synchronisierte Wahrscheinlichkeit, die für alle Knoten gleich ist**. Die Wahrscheinlichkeit wird bei jeder Entscheidung, ob eine Firework-Nachricht gesendet wird, reduziert.

ObserverFireworkNode: Diese Klasse erweitert die Node-Klasse und implementiert das Verhalten der Knoten mit **einer gemeinsam teilbaren globalen Wahrscheinlichkeit**.

-

Individuelle Wahrscheinlichkeit (IndividualObserverFireworkNode.java)

Dieser Ansatz verwendet eine individuelle Wahrscheinlichkeit für jeden Knoten. Dies **ermöglicht eine realistischere Simulation**, da Knoten in einem verteilten System oft unterschiedlich agieren und unterschiedliche Wahrscheinlichkeiten für bestimmte Aktionen haben.

IndividualObserverFireworkNode: Diese Klasse erweitert die Node-Klasse und implementiert das Verhalten **eines Knotens mit einer individuellen Wahrscheinlichkeit**.

ObserverNode.java

Der ObserverNode überwacht die Zustände der anderen Knoten und stellt fest, ob alle Knoten passiv sind und keine Nachrichten mehr unterwegs sind.

- ObserverNode: Diese Klasse implementiert den Observer-Knoten.
- updateState: Aktualisiert den Zustand eines Knotens und überprüft, ob alle Knoten passiv sind.
- checkTermination: Überprüft, ob alle Knoten passiv sind und keine Nachrichten mehr gesendet werden. Wenn alle Knoten passiv sind, wird die Simulation beendet.

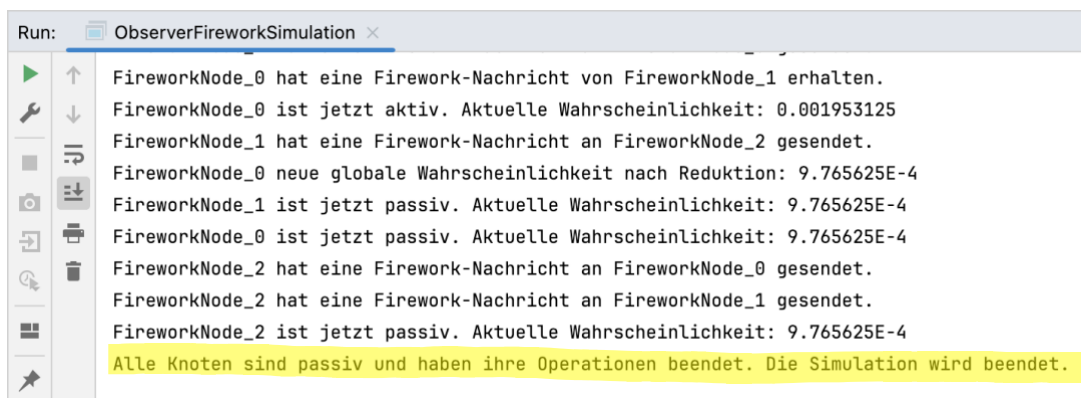
ObserverFireworkSimulation.java

Diese Klasse ermöglicht die Simulation der Knoten in einem verteilten System. **Benutzer können wählen, ob sie Knoten mit einer globalen oder individuellen Wahrscheinlichkeit verwenden möchten.**

Ergebnisse der Simulation

Die Konsolenausgaben zeigten den Ablauf der Simulation und den Zustand der Knoten:

- Knoten wechselten korrekt zwischen aktivem und passivem Zustand.
- Nachrichten wurden gesendet und empfangen, um Knoten zu aktivieren.
- Der Observer-Knoten stellte erfolgreich fest, wenn alle Knoten passiv waren und keine Nachrichten mehr gesendet wurden, und beendete die Simulation.



```
Run: ObserverFireworkSimulation x
FireworkNode_0 hat eine Firework-Nachricht von FireworkNode_1 erhalten.
FireworkNode_0 ist jetzt aktiv. Aktuelle Wahrscheinlichkeit: 0.001953125
FireworkNode_1 hat eine Firework-Nachricht an FireworkNode_2 gesendet.
FireworkNode_0 neue globale Wahrscheinlichkeit nach Reduktion: 9.765625E-4
FireworkNode_1 ist jetzt passiv. Aktuelle Wahrscheinlichkeit: 9.765625E-4
FireworkNode_0 ist jetzt passiv. Aktuelle Wahrscheinlichkeit: 9.765625E-4
FireworkNode_2 hat eine Firework-Nachricht an FireworkNode_0 gesendet.
FireworkNode_2 hat eine Firework-Nachricht an FireworkNode_1 gesendet.
FireworkNode_2 ist jetzt passiv. Aktuelle Wahrscheinlichkeit: 9.765625E-4
Alle Knoten sind passiv und haben ihre Operationen beendet. Die Simulation wird beendet.
```