Ergebnisbericht Verteilte Systeme - Übung 2 - 1540051

19. Juli 2024

Einleitung

In der Übung 2 ging es um die Nutzung des Simulators sim4da-v2, der ein Verteiltes System simuliert. In diesem Fall werden mehrere Akteure generiert, die mit einer gewissen Wahrscheinlichkeit Nachrichten an eine zufällige Teilmenge der anderen Akteure sendet. Nach dem Senden geht der Akteur in einen passiven Zustand. Sobald er eine Nachricht von einem anderen Akteur erhält, geht er wieder in den aktiven Zustand über und verschickt wieder mit einer gewissen Wahrscheinlichkeit, die sich bei jedem Senden verringert, Nachrichten an eine Teilmenge der anderen Akteure.

In der dritten Aufgabe soll dann erkannt werden, dass irgendwann alle Akteure passiv sind und sich keine Nachrichten mehr auf dem Weg befinden, die einen Akteur aufwecken könnten. Dazu soll ein neuer Knoten erstellt werden, der sich „Observer“ nennt. Sobald er erkennt, dass alle passiv sind und bleiben, soll er den Simulator stoppen.

Aufgabe 2

Hierbei habe ich eine Klasse Akteur geschrieben, die von der Simulator-Klasse „Node“ erbt. In der Klasse habe ich eine private Klasse deklariert, die von „Thread“ erbt und darauf wartet, dass eine Nachricht eintrifft. Sobald das geschehen ist wird eine Methode aufgerufen, die den Akteur aufweckt und eine Nachricht an eine Teilmenge der anderen Akteure sendet.

Zudem habe ich eine zweite Klasse geschrieben, die die Akteure erstellt und den Simulator startet. Nach dem erstellen der Akteure sorgt die Klasse dafür, dass die ersten Nachrichten versendet werden. Außerdem enthält diese Klasse eine Liste mit allen Akteuren. Das ist wichtig, um die Teilmengen bestimmen zu können und wird auch beim Observer später wichtig. Zusätzlich gibt die Klasse den Akteuren ihre Nummer mit, damit die Akteure keine Nachrichten an sich selber senden.

Das schwierigste war, eine zufällige Teilmenge aus den Akteur auszuwählen. Dafür habe ich das Array mit den Akteuren genommen und in eine Liste umgewandelt, denn so konnte ich eine Funktion aufrufen, die die Liste in einer zufälligen Reihenfolge neu anordnet. Da-

nach musste ich allerdings die Liste wieder in ein Array umwandeln, damit der Akteur an die zufälligen x ersten Akteure der neuen Liste eine Nachricht senden kann.

Aufgabe 3

Zum Bearbeiten dieser Aufgabe habe ich eine Klasse „Observer“ erstellt, die ebenfalls von „Node“ erbt und zusätzlich die eingehenden und ausgehenden Nachrichten und auch die aktiven Akteure zählt. Um das zu erreichen hat diese Klasse drei statische Variablen, die von den Akteuren aktualisiert werden, sobald sie eine Nachricht senden, eine Nachricht empfangen oder ihren Zustand auf aktiv oder passiv ändern.

Auch hier gibt es eine innere Klasse, die einen Thread Counter bereitstellt. Dieser prüft kontinuierlich, ob die Aktivitäten zum Stillstand gekommen sind. Wenn das der Fall ist, wird der Simulator gestoppt.

Fazit

Die Simulation zeigt, dass der Code wie erwartet funktioniert: Die Knoten wechseln korrekt die Zustände und tauschen die Nachrichten wie geplant aus. Der Observer erkennt die Terminierung korrekt und beendet im Anschluss den Simulator.