Verteilte Systeme, Übungsblatt 2, Sommer 2024

Abgabe: Benedikt Maria Beckermann, Mtr.Nr. 1599203, 06.07.2024

- Link zum Repository: https://github.com/greenBene/24S-DS-A02/

Aufgabe 2

In Aufgabe 2 sollen wir basierend auf dem sim4da Framework eine verteilte Anwendung schreiben, die aus Actor Nodes besteht, die sich gegenseitig "Firework" Nachrichten schicken. Wenn eine "Firework" Nachricht ankommt, soll der Actor Node, der die Nachricht bekommt, aktiviert werden und mit einer bestimmten Wahrscheinlichkeit an ein zufälliges Subset von anderen Actor Nodes weitere "Firework" Nachrichten schicken. Diese Wahrscheinlichkeit soll nach jeder ankommenden "Firework" Nachricht sinken.

Meine Implementierung hierzu ist im Repository unter "./src/click/greenbene/uni/ds/a02/Fireworks.java" zu finden. Ich habe die Klasse ActorNode erstellt, die die Klasse Node des sim4da Framework erweitert. Wenn eine Instanz des Node aktiviert wird (Simulator.simulate()), wartet sie bis zu 5 Sekunden, bevor sie zu einem zufälligen Subset von zwischen 0 und 5 anderen ActorNodes Firework Nachrichten sendet und wartet anschließend auf ankommende Nachrichten.
Wenn die Instanz einen "Firework" Nachricht bekommt, werden mit einer Wahrscheinlichkeit von anfangs 50% wie zu Beginn weiter "Firework" Nachrichten an ein zufälliges Subset von bis zu 5 anderen Actor Nachrichten geschickt. Unabhängig davon, ob weiter "Firework" Nachrichten gesendet wurden, sinkt die Wahrscheinlichkeit, dass zukünftige ankommende "Firework" Nachrichten bei der Instanz zum Versenden von "Firework" Nachrichten führen um 50%.

Aufgabe 3

Für Aufgabe 3 sollen wir die Lösung von Aufgabe 2 um eine Nachrichtenbasierte Terminierungserkennung erweitern.

Hierzu habe ich das Doppelzählverfahren gewählt (siehe im Repository die Datei "src/click/greenbene/uni/ds/a02/FireworksNoMore.java"). Ein neuer Observer Node sendet an alle Actor Node eine "GETSTATE" Message und wartet darauf, dass alle Actor Nodes ihren State via Nachricht zurück melden. Diese "SENDSTATE" Nachrichten beinhalten den Namen des Actor Nodes, die Anzahl der gesendeten und verschickten "Firework" Nachrichten, und seinen Status (Active, Passiv). Dies wird wiederholt, bis in zwei aufeinanderfolgenden Runden alle Actor Nodes passiv sind, die gleiche Anzahl an versendeten und verschickten Nachrichten haben, und mindestens eine Nachricht verschickt wurde. Wenn dies erkannt wird, erkennt der Observer Node, dass das System terminiert ist

und verschickt entsprechende Nachrichten an alle Actor Nodes, damit diese aufhören, auf ankommende Nachrichten zu warten.
Das Versenden der Terminierungsnachrichten ist nicht relevant für den Terminierungsalgorithmus an sich, allerdings ist mir während der Entwicklung der Lösung aufgefallen, dass die Simulation trotz des Parameters "duration_in_seconds" in dem Methodenaufruf "Simulator.simulate(duration_in_seconds)" nicht terminiert, da die Nodes noch in Endlosschleifen sind. Ich bin mir unsicher, ob dies das gewünschte Verhalten der Simulation ist. Um zu vermeiden, dass ich jede Ausführung des Programms manuell beenden muss, habe ich deshalb die Terminierungsnachrichten eingebaut. Ich würde vorschlagen sim4da so zu erweitern, dass nach dem Ablauf der "duration_in_seconds" alle Node Threads automatisch gestoppt werden und anschließend auszugeben, wie viele der Node Threads noch am Laufen waren. Leider komme ich wegen anderer Abgaben nicht dazu, diesen Vorschlag zu implementieren.