Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Modellierung und Verifikation des Token Ring Algorithmus

Birger Kamp und Maria Lüdemann
Formale Simulation und Verifikation verteilter Algorithmen
Sommersemester 2016

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Birger Kamp und Maria Lüdemann

Formale Simulation und Verifikation verteilter Algorithmen
Sommersemester 2016

Modellierung und Verifikation des Token Ring Algorithmus eingereicht
im Rahmen des Projekts New Storytelling
im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer: Sascha Kluth

Abgegeben am 05. Februar 2015

Inhaltsverzeichnis

1. Einleitung

Dieses Dokument beschreibt die Ergebnisse des SVA Praktikums. Dabei sollte ein verteilter Algorithmus gewählt werden um ihn dann in mehreren Schritten in einem Petri Netz zu modellieren, zu spezifizieren und seine Korrektheit zu zeigen.

1.1. Aufteilung

Die Arbeit an diesem Dokument teilt sich wie folgt auf:

Teil 1	Name
Teil 2	Name

Tabelle 1: Arbeitsteilung

2. Hauptteil

2.1. Der Token Ring Algorithmus

Der Token Ring Algorithmus ist ein Wahlalgorithmus der von Chang und Roberts 1979 entworfen wurde. Er kann verteilt auf mehreren Clienten verwendet werden die in einer Ring-Topologie miteinander verbunden sind. Das Ziel des Algorithmus ist, bei Ausfall des Master-Clienten im Netz einen neuen zu wählen.

Voraussetzungen

Damit der Algorithmus auf eine Ring-Topologie angewandt werden kann, müssen folgende Voraussetzungen im Netz gegeben sein:

- Jeder Client kennt seinen Nachfolger
- Jeder Client ist mit seinem Nachfolger verbunden, sodass er mit ihm kommunizieren kann
- Jeder Client hat eine eindeutige ID
- Jeder Client kennt die gesamte Ring-Topologie

Ablauf

Der Algorithmus startet wenn der Master-Client ausfällt. Der Vorgänger des ausgefallenen Master-Clients baut eine Verbindung zum Nachfolger des ausgefallenen Master-Clients auf, sodass die Ring-Topologie wieder vollständig ist.

Der Client, der den Ausfall bemerkt, startet die Wahl in dem er seinem Nachfolger eine Nachricht mit seiner ID und der Info dass es sich um eine Wahl handelt schickt. Dieser nimmt die Nachricht und überprüft ob seine eigene ID darin vor kommt. Falls nicht, hängt er seine eigene ID hinten an und schickt die vervollständigte Nachricht an seinen Nachfolger.

Wenn ein Client feststellt, dass seine eigene ID bereits in der Nachricht vorhanden ist, nimmt er die höchste ID aus der Liste der gesammelten IDs in der Nachricht. Anschließend sendet er eine "GewähltNachricht mit der höchsten ID an seinen Nachfolger. Der Empfänger der "GewähltNachricht merkt sich, dass der gewählte Client nun der neue Master ist und sendet seinem Nachfolger die gleiche "GewähltNachricht. Jeder wird somit benachrichtigt was die höchste ID ist. Kommt die "Gewählt"Nachricht wieder am Initiator der "GewähltNachricht an, wird die Wahl erfolgreich beendet und der Algorithmus ist terminiert.

Eigenschaften

Laufzeit Welche Grundlegenden Eigenschaften hat der Algorithmus was tut er und warum, wofür?

2.2. Spezifikation

Damit ein Modell erstellt werden kann, das den Algorithmus abbildet, müssen zunächst die charakteristischen Eigenschaften des Algorithmus bestimmt werden.

Der Algorithmus (s. Abschnitt 2.1) lässt sich in folgende drei Phasen einteilen:

Phase 1: Ein Client bemerkt den Ausfall des bisherigen Masters

Phase 2: Sammeln aller beteiligten Client-IDs

Phase 3: Bekanntgeben des neuen Masters

Der jeweilige Ablauf der Phasen lässt sich mit folgenden Punkten spezifieren:

Phase	Eigenschaft	Nr.
Phase 1: Master-Ausfall bemerkt	Sendet Nachricht zum Wählen und hängt seine eigene ID daran	1
Phase 2: Wahl	Client, der Wahl-Nachricht erhält, hängt seine eigene ID an die Nachricht	2
	Client sendet die erweiterte Nachricht an seinen Nachfolger	3
	Sobald ein Client eine Wahl-Nachricht erhält, in der seine eigene ID bereits enthalten ist, geht der Algorithmus in Phase 3 über	4
Phase 3: Neuen Master mitteilen	Der Client, der feststellt, dass Phase 2 vorbei ist, sendet eine Nachricht mit dem neuen Master an seinen Nachfolger	5
	Ein Client, der die Nachricht über einen Master erhält, merkt sich den neuen Master	6
	Ein Client, der die Nachricht über einen Master erhält, teilt seinem Nachfolger diese Nachricht mit	7
	Sobald der Client, der Phase 3 eingeleitet hat, die Nachricht über den neuen Master erhalten hat, terminiert der Algorithmus	8

Tabelle 2: Spezifikation der Phasen des Algorithmus

2.3. Modellierung

Wie haben wir ihn modelliert -Gefärbtes netz - Ids - Guards Erklären an wo die spezifizierten Punkte im Netz zu finden sind.

2.3.1. Das Netz

Hier Netzbild einbinden

2.4. Korrektheit

Um zu zeigen, dass das erstellte Modell dem Algorithmus entspricht, wird im Folgenden die Korrektheit bewiesen. Dazu wird die Existenz der spezifizierten Eigenschaften aus Tabelle 2 im erstellten Petri-Netz durch CTL-Ausdrücke geprüft.

2.4.1. Umwandlung CPN zu PN

Das in Abschnitt 2.3 erstellte Modell ist ein farbiges Petri-Netz (engl: Colored Petri Net, kurz: CPN), daher lässt sich darauf nicht die CTL anwenden. Um die CTL verwenden zu können, muss vorher das erstellte Netz in ein einfaches Petri-Netz umgewandelt werden.

Das Modell-CPN verwendet einige Transition-Guards und einen Datentyp mit einem begrenzten Wertraum. Diese Logik ist nicht in einfachen Petri-Netzen erlaubt, daher muss sie umgewandelt werden.

Wenn im CPN auf der Stelle *P1* ein Token mit dem Wert *3* liegt, wird dies im PN dargestellt indem auf der Stelle *P1_3* ein Token liegt. Es gibt daher für *P1* für jeden Wert des Datentyps eine Stelle. Die Transition-Guards des CPN werden im PN durch komplexe Stellen-Transitions-Schaltungen abgebildet.

Dadurch wird das PN im Vergleich zum CPN sehr groß und unübersichtlich.

2.4.2. Erreichbarkeitsgraph

Der Erreichbarkeitsgraph wird auf dem einfachen Petri-Netz gebildet, da die CTL-Ausdrücke auch auf dem einfachen Petri-Netz geprüft werden.

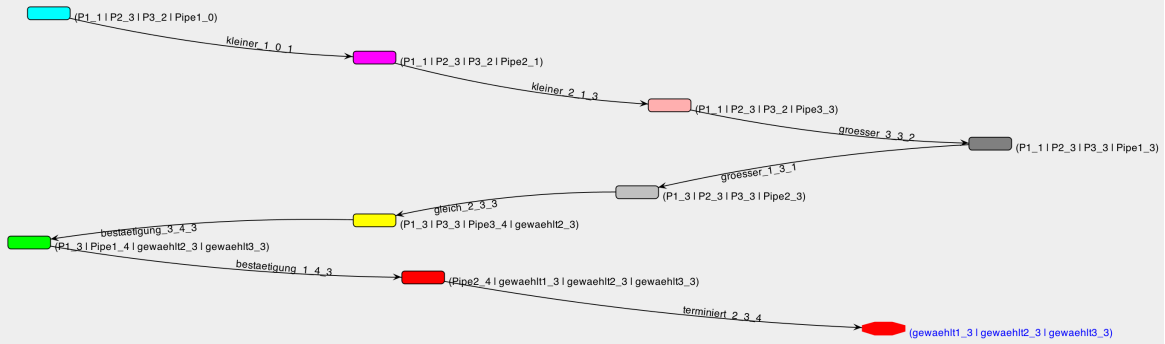

Abbildung 1: Erreichbarkeitsgraph des PN

In Abbildung 1 ist der generierte Erreichbarkeitsgraph zu sehen. Es fällt auf, dass der Graph keine Abzweigungen hat, sondern nur einen Pfad enthält. Das bedeutet, dass der Algorithmus bei gleichen Bedingungen sich deterministisch verhält.

Würde man die Bedingungen verändern, sodass bspw. ein anderer Client den Master-Ausfall bemerkt oder ein anderer Client im Netzwerk hat die höchste ID, dann würde sich die Reihenfolge der Kanten im Erreichbarkeitsgraph verändern und die Knoten würden andere Stellen beinhalten. Der resultierende Erreichbarkeitsgraph wäre allerdings nach wie vor geradlinig.

2.4.3. Prüfung durch CTL

Auf Grundlage des einfachen PN kann nun die Korrektheit in Bezug auf die Modell-Eigenschaften aus Tabelle X festgestellt werden. Die Übersicht über alle verwendeten Ausdrücke ist in Tabelle A im Anhang zu finden.

Die CTL-Ausdrücke lassen sich fehlerfrei auf das einfache Petri-Netz anwenden. Daher entspricht das erstellt CPN dem Token-Ring-Algorithmus von Chang&Pang.

3. Zusammenfassung und Ausblick

Netz flachklopfen um über IDs ein beliebig großes Netz generieren zu können um zu zeigen, dass auch bei steigender Clienten Zahl der Erreichbarkeitsbaum gradlinig deterministisch bleibt Probleme mit Snoopy bezgl. CTL und CPN erklären

A. Korrektheitsbeweis