BAI5 SoSe2024

Ausarbeitung eines kausalen Multicasts

Verteilte Systeme - gelesen von
Prof. Dr. Christoph Klauck

Kristoffer Schaaf (2588265)

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik
Hochschule für angewandte Wissenschaften Hamburg

Inhaltsverzeichnis

1 Theorie

1.1 CBCast Algorithmus

In einem Netzwerk laufen verschiedene Prozesse auf verschiedenen Knoten und teilen sich keinen Speicherplatz. Die Interaktion zwischen den verschiedenen Prozessen läuft soweit ausschließlich über die Weitergabe von Nachrichten und kein Prozess kennt das Verhalten anderer Prozesse [Bab12]. Der *CBCast* (Chain-Based Broadcast) Algorithmus ist ein Algorithmus der im Bereich der verteilten Systeme zum Einsatz kommt und eine Lösung für genau diese Prozessinteraktion implementiert. Genutzt wie zum Beispiel vom ISIS Projekt [BC91] hat er sich in der Vergangenheit bereits mehrfach renommiert.

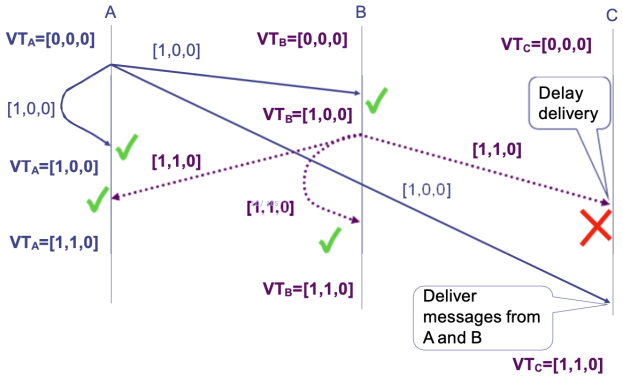

Abbildung 1: CBCAST [Kla24]

In Abb. 1 zu sehen ist ein beispielhafter Ablauf des *CBCASTs* mit drei Prozessen A, B und C. VT zeigt die Vektoruhren der jeweiligen Prozesse. Hier wird der eigene Zustand und der der anderen Prozesse individuell gespeichert. Durch diese Uhr können die Prozesse trotz fehlendem geteilten Speicherplatz erkennen, ob sie mit den anderen Prozessen synchronisiert sind.
Im ersten Schritt verschickt A eine Nachricht an alle Teilnehmer im Netzwerk. Angehängt wird die eigene Vektoruhr - nun mit $A = 1$ erhöht, da dieser Prozess die Nachricht verschickt hat. Wichtig hierbei ist, dass der sendende Prozess die Nachricht immer zusätzlich an sich selber schickt, um eine sichere Synchronisation sicherstellen zu können. In Abb. 1 empfangen Prozess A und B nun die von A gesendete Nachricht. Die Vektoruhren werden verglichen und da jeweils nur ein Zeiger um 1 erhöht wurde, nehmen die Prozesse die gesendete Nachricht an. Nachdem B die Nachricht von A empfangen hat, schickt auch dieser Prozess eine Nachricht an alle Teilnehmer. A und B können diese empfangen. Beim Vergleichen der Vektoruhr von C und der von B mitgeschickten ist aber nun eine zu große Differenz. Zwei Zeiger sind jeweils um 1 erhöht, da B bereits die Nachricht von A empfangen hat. Bei C fehlt diese noch, deshalb blockiert C. Die Nachricht von A welche daraufhin eintrifft nimmt C dann an.
Die Zeiger in den Vektoruhren sind in vielen Implementierungen Zeitstempel der zuletzt empfangenen Nachricht.

1.2 Kommunikationseinheit

1.3 Vektoruhr-ADT

1.4 Vektoruhr Zentrale/Tower

1.5 Ungeordneter Multicast

Ein Multicast verteilt Nachrichten an Teilnehmer in einem Netzwerk. Der Unterschied zum Broadcast besteht darin, dass beim Broadcast Inhalte verbreitet werden, die – mit geeigneter Empfangsausrüstung – jeder ansehen kann, wohingegen beim Multicast vorher eine Anmeldung beim Sender erforderlich ist [Wik23].
Der TowerCBC (siehe Abb. 2) übernimmt in dieser Ausarbeitung die Aufgabe des ungeordneten Multicasts. Ungeordnet ist dieser, weil die Nachrichten nicht in der Reihenfolge weitergegeben werden, in der sich die jeweiligen Teilnehmer/Prozesse beim TowerCBC registriert haben.

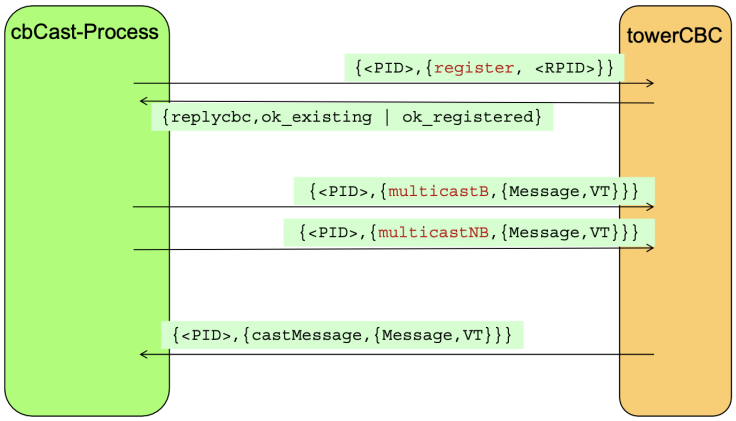

Abbildung 2: ungeordneter Multicast [Kla24]

In Abb. 2 ist eine abstrakte Kommunikation eines Prozesses mit dem Multicast dargestellt. Über *register* meldet sich der *cbCast-Prozess* beim *towerCBC* an. Dieser bestätigt die Anmeldung. Über *multicastB (blockierend)* oder *multicastNB (nicht blockierend)* kann der Prozess nun Nachrichten über den Multicast an alle Teilnehmer im Netzwerk schicken. Der Multicast wieder kann mit *castMessage* Nachrichten an die Teilnehmer schicken.

2 Entwurf

3 Realisierung

4 Analyse

4.1 Korrektheitsbeweis

4.2 Komplexitätsanalyse

5 Fazit

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich diese schriftliche Ausarbeitung meiner Hausarbeit selbstständig und ohne fremde Hilfe verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe sowie die aus fremden Quellen (dazu zählen auch Internetquellen) direkt oder indirekt übernommenen Gedanken oder Wortlaute als solche kenntlich gemacht habe. Zudem erkläre ich, dass der zugehörige Programmcode von mir selbständig implementiert wurde ohne diesen oder Teile davon von Dritten im Wortlaut oder dem Sinn nach übernommen zu haben. Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher nicht veröffentlicht.

(Ort, Datum) (Unterschrift)

Abbildungsverzeichnis

Literaturverzeichnis

[Bab12] Seyed Morteza Babamir. "Specification and verification of reliability in dispatching multicast messages". In: *The journal of supercomputing* 63.2 (2012), S. 612. URL: https://doi.org/10.1007/s11227-012-0834-2.

[BC91] Kenneth Birman und Robert Cooper. "The ISIS project: real experience with a fault tolerant programming system". In: *SIGOPS Oper. Syst. Rev.* 25.2 (Apr. 1991), S. 103–107. ISSN: 0163-5980. DOI: 10.1145/122120.122133. URL: https://doi.org/10.1145/122120.122133.

[Kla24] Christoph Klauck. *Aufgabe HA*. 2024.

[Wik23] Wikipedia. *Multicast*. [Online; accessed 16-May-2024]. 2023. URL: https://de.wikipedia.org/wiki/Multicast.

A Anhang