Message Broker

Reto Hablützel
Zürcher Hochschule für Angewandte Wissenschaften, Studiengang Informatik (29. Juni 2014)

Ein Message Broker ist eine zentrale Kommunikationsschnittstelle für mehrere Programme, die in verschiedenen Programmiersprachen implementiert sein können. Nebst der Möglichkeit heterogene Systeme zu verbinden, kann jedoch durch einen Message Broker auch eine verteilte Architektur realisiert werden, indem jeder Client einen Teil der Funktionalität übernimmt. Damit verschiedenste Programme miteinander kommunizieren können, bieten Message Broker typischerweise Schnittstellen mit verschiedenen Protokollen. Eines davon ist STOMP, welches sehr einfach gehalten und spezifisch für dynamische Programmiersprachen entwickelt wurde. Im Rahmen dieses Seminars habe ich mich deshalb für die Implementation von einem Message Broker entschieden. Da das Kernthema Concurrency ist und ich nicht zu viel Zeit mit dem Protokoll verschwenden wollte, habe ich eine abgespeckte Version von STOMP als Protokoll implementiert.

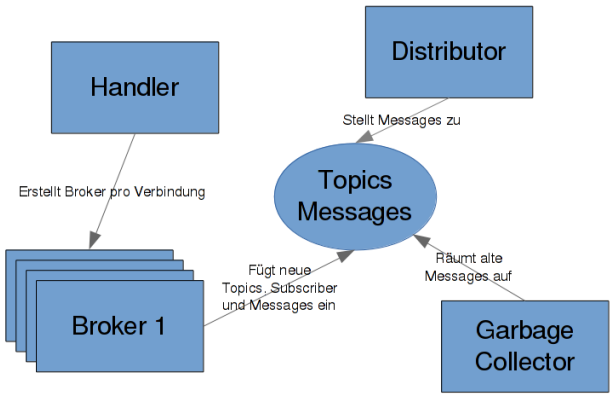

Abbildung 1: Komponentenübersicht

Abbildung 1 zeigt eine Übersicht der Komponenten, die Teil vom Message Broker sind. Als zentrale Speicherstruktur wurde eine Liste von Topics sowie eine Liste von Messages verwendet. Diese sind jeweils als verkettete Liste implementiert. Zu jedem Topic ist eine Liste von sogenannten Subscribern assoziiert, welche die Nachrichten erhalten. Jede Nachricht hält sich eine Liste von Statistiken zu jedem Zustellungsversuch, der an einen Subscriber unternommen wurde. Aufgrund dessen kann der Garbage Collector entscheiden, ob eine Nachricht abgeräumt werden kann und der Distributor, welcher die Nachrichten zustellt, ob eine Zustellung erneut versucht werden soll. Für die Locks auf den Listen wurde ein Read/Write Lock verwendet. Als Beispiel wird folgend der Ablauf vom Garbage Collector aufgezeigt. Die restlichen Komponenten interagieren mit den Listen nach dem gleichen Konzept.

Der Garbage Collector holt sich einen Read Lock auf die Liste der Zustellungsstatistiken. Wenn er sieht, dass für eine Nachricht an einen Subscriber kein erneuter Zustellungsversuch unternommen werden wird (z.B. weil die Nachricht bereits erfolgreich zugestellt wurde), merkt er sich dieses Element in einer Liste. Nachdem er die ganze Liste von Elementen gesammelt hat, werden diese in einem zweiten Schritt von der Liste entfernt. Für das Entfernen ist ein Write Lock notwendig. Als nächstes wird nach dem gleichen Verfahren die Liste der Messages durchsucht und dann aufgeräumt. Diese Aktionen werden jeweils in eine Sammel- und Aufräumphase unterteilt, weil so die Liste in einem ersten Schritt nur mit einem Read Lock geschützt werden muss, wodurch die anderen Komponenten nicht vom Zugriff abgehalten werden. Falls im ersten Schritt nichts zum Aufräumen gefunden wird, muss die Liste nie mit einem Write Lock geschützt werden und bleibt somit für die Anderen zugreifbar.