TKN Telecommunication Networks Group

Praktikum
Technische Grundlagen der Informatik IV

Aufgabenblatt Pr3

— Kausal geordnetes MQTT —

Abgabe: 3./4.6.2013

Hinweis: Auf diesem Blatt werden Eure theoretischen Kenntnisse über Vektoruhren praktisch vertieft. Bitte beachtet bei Fragen zunächst das Hinweisblatt, welches euch beim Kompilieren des Grundgerüstes hilft. Hilfreich beim Lösen dieses Aufgabenblattes sind auch Abschnitt 6.2.2 im Tanenbaum, sowie die Abschnitte 11.4 und 12.4.3 im Coulouris. Bitte beginnt rechtzeitig mit der Lösung und nutzt die Möglichkeiten zum Nachfragen und die betreuten Rechnerzeiten. Die Fragen am Ende der Teilaufgaben müssen nicht schriftlich bearbeitet werden, sondern werden im Termin mit dem Tutor diskutiert.

Aufgabe 1:

In dieser Aufgabe sollt Ihr den Client eines verteilten Message-Boards implementieren. Jeder teilnehmende Cleint-Prozess bietet dem Nutzer die Möglichkeit, Nachrichten über die Tastatur einzugeben und zu verschicken. Die Nachrichten werden an alle teilnehmenden Prozesse gesendet.

Zur Übertragung soll das MQ Telemetry Transport (MQTT) Protokoll verwendet werden, welches einen einfachen Publish/Subscribe Mechanismus implementiert und auch in der aktuellen Forschung eine Rolle spielt. Zur Vereinfachung wird euch dazu ein Grundgerüst in C vorgegeben, welches die mosquitto MQTT Library verwendet. Auch der MQTT-Broker (Server) wird euch bereitgestellt. Eine Anleitung, die die Verwendung erklärt, findet Ihr im Hinweisblatt.

a) Schaut euch das beigefügte Grundgerüst des Clients in C an. In dieser Aufgabe sollt Ihr den Client so vervollständigen, dass er Nachrichten von der Tastatur liest und an den MQTT Broker sendet. Empfangene MQTT Nachrichten von anderen Clients sollen auf der Konsole ausgegeben werden, so dass ein Chat-Programm entsteht. Detailierte Informationen zu MQTT sollten zum Lösen der Aufgabe nicht nötig sein. Eine kurze Einführung findet sich aber im Hinweisblatt.

   In großen Netzen, insbesondere im Internet, kann es passieren, dass sich Nachrichten gegenseitig überholen. Deshalb haben wir in unseren Broker ebenfalls eine Verzögerung eingebaut, welche dieses Verhalten übertrieben nachbildet.

   Wie kommt es dazu, dass sich Nachrichten überholen? Fallen euch Möglichkeiten ein, damit Nachrichten in der richtigen Reihenfolge ausgegeben werden? Welche Probleme könnten auftreten?

b) Wegen der eingebauten Verzögerung kommen Nachrichten in der Regel nicht in der richtigen Reihenfolge bei den Empfängern an. Um eine mögliche (!) Kausalität zwischen den Nachrichten sicherzustellen, habt Ihr in der Vorlesung das Konzept der Vektoruhren kennen gelernt. In diesem Aufgabenteil soll der Client aus Teil a um eine Vektoruhrimplementierung erweitert werden, so dass die Nachrichten jeweils kausal geordnet angezeigt werden.

Mit welchen Datenstrukturen ihr die Vektoruhren implementiert, ist Teil der Aufgabe und euch überlassen. Geht allerdings von festen Teilnehmergruppen aus (keine Joins/Leaves), deren Mitglieder bekannt sind (z.B. im Quelltext oder über eine einfache Textdatei konfigurierbar). Die „happened-before"-Relation soll nur durch das Versenden von Nachrichten charakterisiert werden (d.h. nur bei diesem Ereignis wird die jeweilige logische Uhr inkrementiert).

Ihr könnt euch bei der Implementierung an folgende Leitfragen halten:

(a) Welche Datenstruktur wollt Ihr für die Vektoruhren nutzen? Wie wird diese an die Nachrichten angehängt?

(b) Ihr müsst Nachrichten gegebenenfalls zurückhalten, bis fehlende vorherige Nachrichten ankommen. Welche Datenstruktur solltet ihr hier benutzen, um Nachrichten zwischenzuspeichern?

(c) Welche Bedingungen müssen erfüllt sein, damit Ihr zwischengespeicherte Nachrichten an den Benutzer ausgeben könnt, wenn die kausale Beziehung erhalten bleiben soll?

Sind eure Nachrichten nun geordnet? Wo liegen die Grenzen der Vektoruhren?