UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Aufgabenblatt 7

(Zu bearbeiten bis 12.06.2020)

Vorlesung Verteilte Systeme Sommersemester 2020

Aufgabe 1: Uhrendrift

a) Beschreiben Sie kurz eine Situation, in der das Auseinanderdriften der Uhren in einem verteilten System zu Problemen führen kann.

b) Begründen Sie, warum bei der korrigierten UTC in bestimmten Abständen Schaltsekunden eingefügt werden. Informieren Sie sich dazu ggf. über Literatur oder Internet.

Aufgabe 2: Uhrensynchronisation

Wie können die Uhren in zwei Computern, die über ein Netzwerk verbunden sind, ohne Nutzung einer externen Zeitquelle (z.B. GPS) synchronisiert werden? Was läßt sich über die erreichbare Genauigkeit aussagen? Welche Faktoren schränken diese ein?

Warum können selbst mit Hilfe von GPS keine ganz exakt synchronen Uhren auf zwei entfernt stehenden Rechnern realisiert werden?

Aufgabe 3: Programmierung: Uhrensynchronisation

Im Archiv `u07Files.zip`[1] auf der Vorlesungswebseite finden Sie eine einfache Klasse zur Realisierung einer genauen Uhr (mit Zeitangabe in ns), sowie einen Server, der diese Uhrzeit per RMI zur Verfügung stellt und einen zugehörigen Client. Der Client prüft, ob seine lokale Uhr mit der des Servers gut genug synchronisiert ist, indem er drei Zeitstempel betrachtet: zur Zeit t_1 erfolgt der RMI-Aufruf, zur Zeit t_2 die Bearbeitung im Server, zur Zeit t_3 kehrt der Aufruf im Client zurück. Man würde hier die Abfolge $t_1 \leq t_2 \leq t_3$ erwarten. Da t_2 aber mit einer anderen Uhr als t_1 und t_3 gemessen wurde, sind Verletzungen dieser Erwartung bei unsynchronisierten Uhren möglich.

Realisieren Sie die Methode `syncClock()` in `ClockClient`, damit der Client seine lokale Uhr mit der des Servers synchronisiert. Da Sie die Systemuhr nicht verstellen können, müssen Sie ggf. auch Erweiterungen der Klasse `Clock` vornehmen. Testen Sie Ihr Programm auf zwei verschiedenen Rechnern (z.B. `bsclk01` und `bsclk02`).

Welchen Grund kann es geben, dass die Uhren bereits vor der Ausführung der Methode `syncClock()` hinreichend genau synchronsiert sind? Warum kann es sein, dass eine Minute nach der Uhrensynchronisation die Uhren wieder nicht genau genug übereinstimmen?

Aufgabe 4: Uhrensynchronisation

a) Ein Client versucht, eine Synchronisierung mit einem Zeit Server durchzuführen. Er hat dazu drei Anfragen an den Server ausgeführt und in der nachfolgenden Tabelle jeweils die lokale Zeit des Absendens der Anfrage, die vom Server zurückgesendete Zeit und die gemessene Round-Trip-Zeit aufgezeichnet.

[1] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/u07Files.zip`

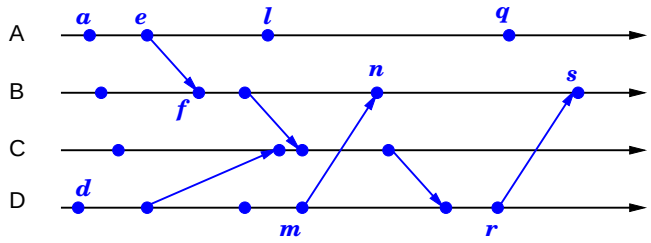
Welche dieser Zeiten sollte er verwenden, um seine Uhr zu setzen? Auf welche Zeit sollte er sie setzen? Schätzen Sie die Genauigkeit, wenn bekannt ist, dass die Zeit zwischen dem Senden einer Nachricht und dem Empfangen der Antwort in dem betreffenden System mindestens $8ms$ beträgt.

Absendezeit [hh:mm:ss]	Round-Trip-Zeit[ms]	Zeit des Servers [hh:mm:ss]
11:51:20.043	22	11:51:20.564
11:51:21.821	25	11:51:22.340
11:51:24.715	20	11:51:25.232

b) Wenn die minimale Übertragungszeit von $8ms$ gegeben sei und die Uhr eines Datei-Servers auf $\pm 1ms$ genau synchronisiert werden muss, wie hoch darf die Round-Trip-Zeit dann höchstens sein?

Aufgabe 5: Lamport- und Vektorzeit

Gegeben sei ein verteiltes Rechnersystem mit vier Rechnern sowie folgendem Ablauf von Empfangs-, Sende- und lokalen Ereignissen:



a) Geben Sie Ereignispaare an, über deren kausale Reihenfolge (im Sinne der Lamport'schen *Happened-Before-Relation*) Sie keine Aussage machen können. Können Sie eine Aussage machen, ob das Ereignis d das Ereignis n (tatsächlich) kausal beeinflusst hat?

b) Nehmen Sie nun an, dass das System Lamport-Uhren für jeden Rechner realisiert. Tragen Sie für jedes Ereignis die entsprechenden Lamport-Zeitstempel in das obige Diagramm ein.

c) Was können Sie aus den Lamport-Zeitstempeln der Ereignisse e und r schließen?

d) Tragen Sie nun für die Ereignisse a, f, l, m und s die Vektorzeiten ein.

e) Was können Sie aus den Vektor-Zeitstempeln der Ereignisse l und m bzw. aus den Vektor-Zeitstempeln der Ereignisse f und s schließen?

Aufgabe 6: *Happened-Before*-Relation, Lamport- und Vektorzeit

Drei Prozesse A, B und C erzeugen jeweils die Ereignisse $a_1, a_2, ..., b_1, b_2, ...$ und $c_1, c2, ...$

a) Nehmen Sie an, daß die Ereignisse $a_1, a_2, a_3, a_4, b_1, b_2, c_1, c_2$ aufgetreten sind und die Lamport-Zeiten $L(a_1) = 1$, $L(a_2) = 2$, $L(a_3) = 3$, $L(a_4) = 4$, $L(b_1) = 1$, $L(b_2) = 2$, $L(c_1) = 1$, $L(c_2) = 5$ haben. Welche Aussagen lassen sich über die *happened-before* Relation zwischen a_1 und b_1, c_1 und b_2, sowie a_4 und c_2 machen? Welche Aussagen können Sie jeweils über die Reihenfolge der beiden Ereignisse in der Realzeit machen?

b) Für die Ereignisse aus a) gelten nun folgende Vektorzeiten: $V(a_1) = (1, 0, 0), V(a_2) = (2, 1, 0), V(a_3) = (3, 1, 0)$, $V(a_4) = (4, 1, 0), V(b_1) = (0, 1, 0), V(b_2) = (0, 2, 1), V(c_1) = (0, 0, 1), V(c_2) = (4, 1, 2)$. Welche Aussagen lassen sich nun über die *happened-before* Relation der Ereignisse aus a) treffen? Welche Aussagen sind über die Reihenfolge in der Realzeit möglich?

c) Worin unterscheiden sich Lamport- und Vektorzeit?

Aufgabe 7: Programmierung: Vektor-Uhr

Im Archiv `u07Files.zip`[2] auf der Vorlesungswebseite finden Sie die Realisierung eines Prozeß-Systems mit mehreren Server-Prozessen. Die Server erhalten über Nachrichten Aufgaben, die sie lokal bearbeiten und an andere Server weitergeben können. Zusätzlich ist ein Client vorhanden, der Aufgaben an einen der Server sendet.

Ergänzen Sie die in der Datei `Server.java` vorgegebene Implementierung einer Vektor-Uhr und erweitern Sie den Server so, daß die vorgegebene Methode `log()` zu jedem Ereignis auch die aktuelle Vektor-Zeit mit ausgibt. Die Vektor-Zeit soll bei bei jedem Empfangsversuch (Aufruf von `receive()`) erhöht werden. Der Client soll bei der Vektor-Zeit **nicht** mit berücksichtigt werden.

Zur Vereinfachung sind in der Klasse `VectorClock` zwei Methoden `addToMessage()` und `extractFromMessage()` vorgegeben, die die Vektorzeit an einen String anfügen bzw. aus diesem extrahieren.

[2] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/u07Files.zip`