

Aufgabenblatt 7

Musterlösung

Vorlesung Verteilte Systeme

Wintersemester 2024/25

Aufgabe 1: Uhrendrift

- Ein einfaches Beispiel ist das verteilte Kompilieren eines Programms mittels `make`. `Make` wird über ein sogenanntes Makefile gesteuert, in dem Abhängigkeiten z.B. zwischen Quell- und Objektdateien sowie Regeln, die angeben, wie Zielformate aus Quelldateien erzeugt werden können, spezifiziert sind. Aus Optimierungsgründen wird eine solche Regel nur angewendet, wenn sich seit der letzten Anwendung die Quelldatei verändert hat, damit eben nur wirklich notwendige Kompilierungsvorgänge stattfinden. Dabei orientiert sich `Make` an den im Dateisystem abgelegten Zeitstempeln der Dateien, die die letzte Änderung angeben. Weichen nun die Uhren auf den einzelnen beteiligten Rechnern voneinander ab, so kann es zu inkonsistenten Zuständen kommen, z.B. dazu, dass eine Quelldatei ein älteres Datum als die zugehörige Objektdatei aufweist, obwohl der Quellcode nach dem letzten Übersetzungsvorgang verändert wurde. Fehlerhafterweise würde dann die erneute Kompilierung dieser Datei ausbleiben.
- Mit den zusätzlich eingefügten Schaltsekunden wird erreicht, dass manche Tage statt 86400 genau 86401 Sekunden dauern. Notwendig ist dies, da eine Erddrehung eben nicht exakt $24 * 60 * 60$ Sekunden dauert, sondern einen kleinen Sekundenbruchteil länger. Schaltsekunden haben also nichts mit einer evtl. Ungenauigkeit der Atomuhr zu tun, sondern dienen lediglich einer Anpassung an den Kalender.

Aufgabe 2: Uhrensynchronisation

Es gibt mehrere Möglichkeiten zur Synchronisierung verteilter Uhren. Eine von diesen ist Cristians Protokoll. Ein Client-Prozess P fordert die Zeit in einer Nachricht m_r von einem Zeit-Server S an, und empfängt den Zeitwert t in einer Nachricht m_t von S . Das Problem ist, daß wegen der Signallaufzeit von m_t die Zeit t bis zur Ankunft bei P schon wieder veraltet ist. P mißt daher die Round-Trip-Zeit T_{round} vom Versenden von m_r bis zum Empfang von m_t und setzt seine Uhr auf $t + T_{round}/2$ ($T_{round}/2$ ist die geschätzte Laufzeit der Nachricht m_t).

Die Genauigkeit des Verfahrens wird eingeschränkt durch:

- die Annahme, daß die Laufzeiten von m_r und m_t gleich sind (systematischer Fehler).
- die Streuungen in den Nachrichtenlaufzeiten (statistischer Fehler).

In einem LAN entstehen Streuungen durch den Wettbewerb um das Medium (Kollisionen, Pufferzeiten in Switches) und durch die Verarbeitung der Nachrichten im Betriebssystem von P und S . Für ein LAN ist die Genauigkeit vermutlich innerhalb $1ms$. Im Internet entstehen zusätzliche Ungenauigkeiten, weil die Nachrichten in den Routern verzögert werden. Für WANs ist die Genauigkeit vermutlich innerhalb $5 - 10ms$.

Auch mit Hilfe von GPS ist eine exakte Synchronisierung von Uhren nicht möglich. Wegen der bekannten Position von Satellit und GPS-Empfänger kann die Nachrichtenlaufzeit zwar sehr genau berechnet werden, schwankt aber z.B. aufgrund von Wetterbedingungen. Zudem sind auch die Positionen mit Fehlern behaftet.

Aufgabe 3: Programmierung: Uhrensynchronisation

Die Programmierlösung dieser Aufgabe finden Sie im Archiv [107Files.zip](#)¹ auf der Vorlesungswebseite.

¹<http://www.bs.informatik.uni-siegen.de/web/wismueller/v1/vs/107Files.zip>

Durch den unterschiedlich schnellen Gang der lokalen Uhren (Drift) muss die Synchronisation regelmäßig wiederholt werden.

Aufgabe 4: Uhrensynchronisation

- a) Der Client sollte die Anfrage mit der minimalen Round-Trip-Zeit von $20ms$ wählen $= 0.02s$. Falls er seine Uhr unmittelbar nach Ankunft der Antwort setzen kann, würde er seine Uhr auf $t_{Server} + T_{round}/2$, also auf $11:51:25.232 + 0.02/2 = 11:51:25.242$ Uhr setzen. Da der Client in der Regel aber erst später (nach einigen Anfragen) diejenige mit der minimalen Round-Trip-Zeit bestimmen kann, muß er wie folgt vorgehen: Angenommen, die lokale Zeit des Clients ist t . Er setzt dann die Uhr auf $t + (t_{Server} + T_{round}/2 - t_{Ankunft}) = t + (t_{Server} + T_{round}/2 - (t_{Absenden} + T_{round})) = t + 25.242 - 24.734 = t + 0.508$, d.h., die Uhr wird um $508ms$ vorgestellt.

Die Genauigkeit ist $\pm 10ms$.

Wenn die minimale Übertragungszeit $8ms$ beträgt, dann bleibt die Einstellung die selbe, aber die Genauigkeit verbessert sich auf $\pm 2ms$.

- b) Um die Uhr innerhalb von $1ms$ zu synchronisieren, wenn eine Übertragungszeit von mindestens $8ms$ gegeben sei, ist die maximale Round-Trip-Zeit $T_{round} = 2 \cdot 1ms + 2 \cdot 8ms = 18ms$.

Aufgabe 5: Lamport- und Vektorzeit (Pflichtaufgabe für 6 LP, Abgabe über moodle!)

Aufgabe 6: Happened-Before-Relation, Lamport- und Vektorzeit (Pflichtaufgabe für 6 LP, Abgabe über moodle!)

Aufgabe 7: Programmierung: Vektor-Uhr

Die Lösung dieser Aufgabe finden Sie im Archiv [107Files.zip](http://www.bs.informatik.uni-siegen.de/web/wismueller/v1/vs/107Files.zip)² auf der Vorlesungswebseite.

²<http://www.bs.informatik.uni-siegen.de/web/wismueller/v1/vs/107Files.zip>