Time synchronisation with NTP

Matthias Kastner
Betreuer: Dirk Haage
Seminar Innovative Internet Technologien und Mobilkommunikation WS09/10
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: kastner@in.tum.de

ABSTRACT
Heutzutage ist beinahe jeder Computer vernetzt. In diesen, sogenannten verteilten Systemen spielt eine sehr genaue Zeitsynchronisation eine entscheidende Rolle. Aus diesem Grund wird dem Bereitstellen von exakten Uhren eine zunehmend große Bedeutung geschenkt. Seit langer Zeit wird das Protokoll NTP entwickelt, welches durch seine Flexibilität überzeugt. Doch auch dieser Ansatz hat seine Grenzen, welche in dieser Arbeit aufgezeigt und Alternativen gefunden werden sollen.

Keywords
Anwendungsgebiete NTP, Marzullo's Algorithmus, Berkeley Algorithmus, Cristian's Algorithmus, NTPD, Sicherheit von NTP, NTP-Header

1. EINLEITUNG
Eine genaue Zeitsynchronisierung benötigt man immer dann, wenn zwei Ereignisse/Prozesse in einem verteilten System in einer bestimmten Reihenfolge abgearbeitet werden müssen. In verteilten Systemen ist es nicht möglich, sämtliche Informationen (wie auch die Systemzeiten) zentral zu speichern. Schon kleinste Ungenauigkeiten können die Daten komplett unbrauchbar machen.[9]
Als Anwendungsgebiet kann man sich beispielsweise eine verteilte Datenbankanwendung vorstellen, wo ein Rechner Tabellen anlegt und von einem zweiten diese mit Daten gefüllt werden. Nun gibt es in diesem Szenario nur die lokalen Uhren der Rechner. Ist es nun auf der Uhr von Rechner 2 später als auf der von Rechner 1, würde das System versuchen, die Daten zu füllen, obwohl noch keine Tabellen erstellt wurden. Dadurch gingen die Daten verloren und eine leere Tabelle wäre die Folge.
Als weiteres, alltägliches Beispiel kann man sich die Emission von Aktien vorstellen, welche über das Internet verkauft werden. Wenn die Uhren in einem solchen Beispiel nicht korrekt arbeiten, hätte das eine sehr starke Wettbewerbsverzerrung zur Folge, da einige Personen die Aktien früher kaufen könnten als andere.
Im ersten Kapitel werde ich näher auf die Funktionsweise von NTP und dessen Schwächen eingehen. Anschließend werde ich alternative Algorithmen, wie zum Beispiel den Berkeley Algorithmus und weitere Verfahren vorstellen. Abschließen möchte ich diese Arbeit mit einer persönlichen Stellungnahme und einem Ausblick.

2. NTP
Die Grundidee von NTP (Network Time Protocol), welches 1985 entwickelt wurde, ist, dass ein Server die Uhrzeit zentral für alle Clients sehr genau zur Verfügung stellt. Dabei wurde ein besonderes Augenmerk auf die unterschiedlichen Paketlaufzeiten der beteiligten Clients gelegt. NTP hat sich als Standard für Zeitsynchronisation (auch und gerade über das Internet) etabliert. Bei NTP ist der Zeitserver passiv, was bedeutet, dass die Clients den Server regelmäßig abfragen und der Server lediglich auf die Anfragen antwortet. Inzwischen schätzt die NIST (National Institute of Standards and Technology), dass es im Internet zwischen 10 und 20 Millionen NTP-Server gibt. Nahezu jedes moderne Betriebssystem stellt einen NTP-Client zur Verfügung mithilfe dessen die Zeit von einem NTP-Server abgefragt werden kann. Erwähnenswert ist, dass NTP das am längsten laufende Internetprotokoll ist. [4]

2.1 Funktionsweise
NTP verwendet das verbindungslose Protokoll UDP und arbeitet somit auf der Transportschicht (Schicht 4 im OSI-Schichtenmodell). Der Standardport 123 ist dabei für NTP reserviert. Wie die folgende Abbildung zeigt, ist die Netzwerkhierarchie von NTP in mehrere Schichten unterteilt. Schicht 0 besteht aus den Uhren (im Normalfall Atom- oder Funkuhren). Schicht 1 besteht aus den Zeitservern, die eine direkte Verbindung zu den Uhren haben. Schicht 2 und 3 sind ebenfalls Serverschichten, die die Daten weitergeben. Es kann bei der aktuellen Version NTPv4 bis zu 16 Schichten (werden auch Stratum genannt) geben. Schicht 4 sind in unserem Beispiel dann die Endanwender - also Clients. Dabei ist zu beachten, dass die Uhrzeit nach unten hin immer ungenauer wird. Aus diesem Grund wird geplant, bei NTPv5 nur mehr 8 Schichten zu erlauben. Grundlage für NTP ist die Koordinierte Weltzeit (UTC), da diese äußerst konstant ist. Es werden bei UTC keinerlei Informationen über die Zeitzone oder Sommer/Winterzeit weitergegeben.
Format für die Mitteleuropäische Zeitzone:
20090912T222050+0100 - 12.09.2009 23:20:50
Die +0100 stehen für eine Stunde Abweichung von der GMT

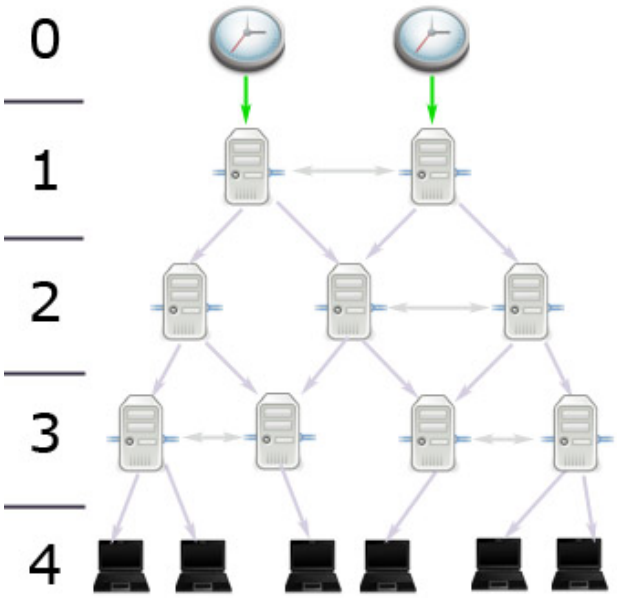

Figure 1: Schichten von NTP

(Greenwich Mean Time).
NTP arbeitet mit sogenannten Zeitstempeln, welche 64 Bits lang sind. Die 64 Bits sind folgendermaßen aufgeteilt: Die ersten 32 Bits geben die Anzahl der Sekunden seit dem 01.01.1900 00:00:00 zurück, die anderen 32 Bits den Sekundenbruchteil. Hiermit lässt sich (zumindest theoretisch) eine Genauigkeit von 233 Pikosekunden (2^{-32} Sekunden) erreichen. Mit 32 Bits kann man in etwa 136 Jahre codieren, was bedeutet, dass im Februar 2036 der Timestamp zum ersten mal überläuft (also wieder mit 0 beginnt). Allerdings wird dies zu keinen nennenswerten Konflikten führen. [6]
Ein Problem von Zeitsynchronisierung über Netzwerke liegt darin, dass durch die Nachrichtenverzögerung die Zeit am Client veraltet ist. Somit wurde nach einer möglichst genauen Schätzung gesucht, welche folgendermaßen umgesetzt wurde: Man nimmt an, dass man von A nach B in etwa in

Figure 2: Zeitverzögerung Diagramm

der gleichen Zeit kommt als von B nach A, was bedeutet, dass gilt: $T_2 - T_1 \approx T_4 - T_3$ Mittels folgender Formel kann A die Abweichnung (θ) relativ zu B berechnen.
$\theta = T_3 + \frac{(T_2-T_1)+(T_4-T_3)}{2} - T_4 = \frac{(T_2-T_1)+(T_3-T_4)}{2}$
θ muss selbstverständlich $\geqslant 0$ sein, da die Zeit nur vorwärts laufen darf. Man kann dies insofern erreichen, indem man eine gewisse Anzahl x (zB 100) an Interrupts pro Sekunde erzeugt. So müsste jeder Interrupt 10 ms hinzufügen. Muss man nun die Uhr langsamer laufen lassen, fügt man zB lediglich 8 ms hinzu, wenn sie schneller laufen soll zB 11 ms bis man die korrekte Uhrzeit erreicht hat.
Dieser Ansatz wurde 1989 von Cristian vorgeschlagen und wurde nach ihm benannt (Cristian's Algorithmus). [9]
Mit NTPv4 lässt sich weltweit eine Genauigkeit von ca. 10 ms erreichen.

2.1.1 Design von NTP

Eine große Herausforderung für die Entwicklung von NTP-Servern ist, dass diese eine sehr genaue und vor allem stabile Zeit, auch in unstabilen Netzwerken, liefern müssen. Beispielsweise kann es zu Verbindungsabbrüchen und Unerreichbarkeiten von Servern kommen. Aus diesem Grund wird auf eine große Redundanz von Servern Wert gelegt. Darüber hinaus wird die Zeit von einem großen Spektrum an Architekturen und Betriebssystemen abgefragt: Von Supercomputern mit sehr schnellen und direkten Internetanschlüssen über Heimcomputer bis hin zu Eingebetteten Systemen (Embedded Systems) über unstabile UMTS-Verbindungen. Dabei müssen für sämtliche Plattformen die selben Bedingungen gelten und zudem muss die Software äußerst einfach zu installieren sein. Weiters will man eine möglichst hohe Resistenz gegenüber Fehlern in der Implementation und Angriffen erreichen. Die Implementierung von NTP bietet wie nahezu alle modernen Protokolle die Möglichkeit, sämtliche Probleme zu loggen wodurch der Administrator einen Überblick über potentielle Fehlerquellen bekommt. [7]

2.1.2 Header von NTP

Wie sämtliche IP-Pakete hat auch NTP einen Header. In der folgenden Tabelle möchte ich ein paar der insgesamt 16 Felder nach [6] kurz beschreiben.

Name	Beschreibung
leap	Hier werden die Schaltsekunden „verwaltet". Entweder, dass es keine Warnung gibt, oder aber das die letzte Minute des Tages 59 oder 61 Sekunden lang dauert.
mode	In welchem Modus der Rechner ist - sowohl Server als auch Client werden hierbei unterschieden und zusätzlich ob es sich um eine aktive oder passive Verbindung handelt.
stratum	Im Feld stratum wird festgelegt, in welchem Stratum der NTP-Hierarchie sich der Rechner befindet. Also beispielsweise 1 (Atomuhren) oder 4 (in unserem Beispiel Clients)
rootdelay	Die Ausbreitungsverzögerung (RTT) zum NTP-Server
reftime	Zeitangabe wann die lokale Uhr das letzte Mal synchronisiert wurde

2.2 Ablauf von NTP

Wie der Zeitalgorithmus von NTP abläuft, werde ich in diesem Kapitel nach [1] beschreiben. Hier soll als erstes ein informeller Überblick geschaffen werden und anschließend werde ich einen Teil vom Intersection-Algorithmus (Marzullo's Algorithmus) im folgenden Unterkapitel erläuteren.

1. Es wird überprüft, ob die Paketdaten gültig sind
2. Filtern der Pakete um keinen unnützen Overhead bearbeiten zu müssen

3. Intersection Algorithmus
4. Entfernen der ungenauesten Uhren bis maximal zehn zur Auswahl stehen
5. Auswahl der „besten“ Quelluhren (mittels Clustering-Algorithmus)
6. Kombinieren der Uhren - Korrigieren der Fehler mittels Abschätzungen auf den „schlechten“ Uhren

2.2.1 *Marzullo's Algorithmus*

NTP verwendet eine Verfeinerung des Marzullo-Algorithmus (entwickelt von Keith Marzullo in dessen Dissertation), welchen ich in diesem Kapitel vorstellen möchte. Der Algorithmus versucht aus den vorhanden Zeitquellen das genaueste Zeitintervall (mittels der meisten Übereinstimmungen) zu bekommen. Man nimmt an, dass die verschiedenen Zeitquellen eine Zeit mit Vertrauensintervallen(+/- einer bestimmten Zeit) bieten. Man muss diese Intervalle selbstverständlich möglichst gut abschätzen. Einerseits sollten sie nicht zu großsein (Ungenauigkeit), andererseits dürfen sie aber auch nicht zu klein sein (es wird keine Übereinstimmung gefunden). Wir nehmen nun als Beispiel folgende 3 Intervalle (diese werden also von drei verschiedenen Quellen zur Verfügung gestellt):
$7 \pm 2, 9 \pm 1, 8 \pm 1$ Nun wird nach dem Intervall mit der größ-

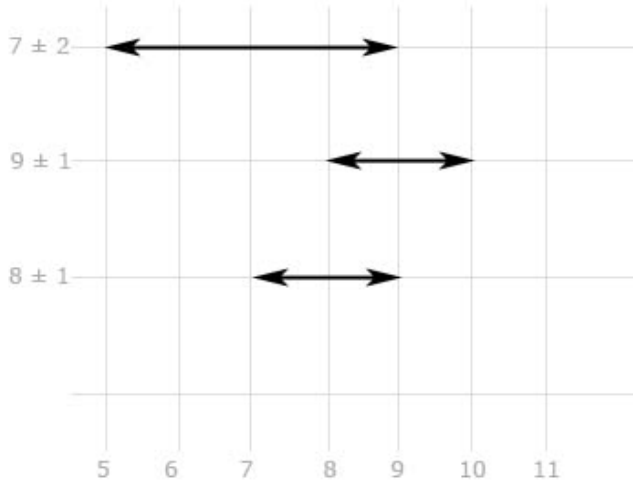

Figure 3: Marzullo Intervalle

ten Übereinstimmung gesucht.
Wie man auf der Abbildung erkennen kann, ist dies das Intervall von 8 bis 9, da alle 3 Intervalle eingeschlossen sind. Es ist keineswegs eine Voraussetzung, dass sich alle Intervalle in der Lösungsmenge befinden. In einem solchen Fall wird das Intervall mit den meisten Übereinstimmungen gewählt. Der Algorithmus hat eine lineare Laufzeit $O(m)$ was den Speicherplatz betrifft und eine zeitliche Laufzeit von $O(m * log(m))$. Deshalb gilt er als sehr effizient. [2]
An dieser Stelle möchte ich den Ablauf des Algorithmus nach

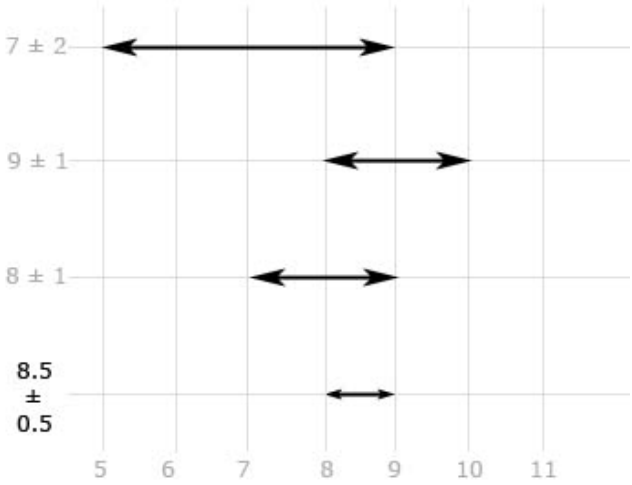

Figure 4: Lösung Marzullo

[2] beschreiben:

Es gibt folgende Variablen:

Name	Beschreibung
best	größte Überlappung der Intervalle
cnt	aktuelle Anzahl der überlappenden Intervalle
beststart	Beginn des aktuell besten Intervalls
bestend	Ende des aktuell besten Intervalls
i	index
ls	Liste der Tupeln

1. Bilden der Tupel <offset, type> - für jede Quelle gibt es einen Intervall [c-r, c+r], wobei für solche Intervalle zwei Tupel erzeugt werden. Einmal für das c-r und einmal für das c+r. Dabei kennzeichnet <c-r, -1> ein beginnendes und <c+r,+1> ein endendes Intervall.
2. Sortieren der Liste nach Offsets
3. [Initialisierung] best = 0, cnt=0
4. [Schleife] man geht in aufsteigender Reihenfolge durch die Liste der Tupel
5. [aktuelle Nummer des überlappenden Intervalls] cnt = cnt - type[i]
6. if cnt>best then
   best=cnt beststart=offset[i] bestend=offset[i+1]
7. Kommentar: Das nächste Tuple muss entweder das Ende eines Intervalls sein <c+r, +1>, wobei dieses Intervall dann das Beste ist oder aber der Beginn eines anderen Intervalls <c-r, -1>, welches im nächsten Schritt mit dem besten Intervall überschrieben wird.

8. [Schleifenende] Rückgabe [beststart, bestend] als optimales Intervall

Der Intersection-Algorithmus (die Abänderung vom Marzullo-Algorithmus) stellt zudem noch sicher, dass auf jeden Fall der Mittelpunkt der gewählten Abschätzungen im Intervall enthalten ist, was beim Marzullo-Algorithmus nicht sicher gestellt wird. Somit bekommt man womöglich ein größeres Intervall als Rückgabe.

2.3 Sicherheit von NTP

Bemerkenswerterweise kam es in den 25 Jahren, in denen NTP eingesetzt wird zu kaum nennenswerten Sicherheitsproblemen und kein einziges Problem konnte bisher auf die Spezifikation von NTP zurückgeführt werden.
NTP bietet für den User sowohl die Möglichkeit, sich auf dem Server unverschlüsselt als auch mittels symmetrischer und asymmetrischer Verschlüsselungsverfahren zu authentifizieren.
Die Zeit selbst wird ohne jegliche Verschlüsselung übertragen und steht jedem zur Verfügung. [5]

2.4 Schwächen von NTP

NTP liefert über das Internet eine Zeitgenauigkeit von 10 ms. Für den „normalen“ Heimanwender mag dies kaum eine Rolle spielen. Doch für viele professionelle Anwendungen (stellt man sich zum Beispiel eine verteilte Kernspaltung vor) kann diese Genauigkeit nicht ausreichend sein. Das Hauptproblem ist, dass Netzwerke in der Praxis weder beständig noch gleichmäßig arbeiten. Zwar wird dies berücksichtigt, jedoch können diese Ungenauigkeiten nie exakt bestimmt werden. Daraus resultiert auch die Schwachstelle von NTP, dass in den unteren Ebenen die Zeit immer stärker von der korrekten realen Uhrzeit abweicht.

2.5 NTP-Versionsübersicht

In diesem Unterkapitel möchte ich die Entwicklung von NTP vorstellen. Wie bereits erwähnt, dauert diese bemerkenswerterweise seit 1985 an und wurde maßgeblich von David L. Mills beeinflusst. Der aktuelle Internetstandard ist noch immer ntpv3 (aus dem Jahre 1992), während bereits seit 2006 eine Implementierung von ntpv4 existiert. Zur Zeit wird an ntpv5 gearbeitet.

2.5.1 Version 3

Version 3 hat weder das Protokol noch die Implementierung von Version 2 (aus dem Jahre 1989) in wesentlichen Punkten geändert. Die Motivation war viel mehr die Verfeinerung der Analyse und der Implementierungsmodelle für neue Applikationen in deutlich schnelleren Netzen. Beeindruckenderweise beachtete man zu dieser Zeit (1992) bereits Gigabitnetzwerke. Dabei wurde ein besonderes Augenmerk auf die Genauigkeit und Stabilität der Highspeed-Netzwerke gelegt. Zudem wurden die Lokaluhren-Algorithmen verbessert um den Netzwerkoverhead zu minimieren.[3]

2.5.2 Version 4

Folgende Änderungen gegenüber der Version 3 wurden in ntpv4 durchgeführt:
Erstmals wurde auch IPv6 beachet indem man einen alternativen Protokollheader vorstellte. Jedoch existiert der alte Header ebenfalls noch um eine Abwärtskompatibilität zu erreichen. Darüber hinaus wurde der Header erweitert, sodass eine Public Key-Authentifizierung möglich wurde. Die Zeitauflösung ist nun genauer als eine Nanosekunde und die Frequenzauflösung genauer als eine Nanosekunde pro Sekunde.
Weiters wurde ein „Clock discipline algorithm“ eingeführt, welcher Hardwarefrequenzungenauigkeiten bemisst.
Die Server aus dem ersten Stratum haben nun eine Genauigkeit von unter 100 Mikrosekunden. Die Server aus dem zweiten Stratum bieten immerhin noch eine Genauigkeit besser als eine Millisekunde.[6]

2.5.3 Version 5

Es gibt kaum Informationen zur Version 5 von NTP. Aus diesem Grund liegt die Vermutung nahe, dass es noch einige Jahre dauern wird bis eine Implementierung von Version 5 freigegeben wird. Bisher steht lediglich fest, dass die Genauigkeit weiterhin erhöht werden soll und dass Ungenauigkeiten von modernen Netzwerken besser beachtet werden sollen. Jedoch gibt es keine konkreten Ansätze dazu (bzw. wurde noch nichts veröffentlicht).

3. GENAUERE ZEITEN MITTELS NTP

Wie bereits erwähnt, spielt die Genauigkeit die entscheidende Rolle bei NTP.
In diesem Unterkapitel werde ich verschiedene Ansätze aufzeigen, wie man dieses Ziel erreicht. Man sollte je nach Anwendungsgebiet unterschiedliche Lösungsmöglichkeiten in Betracht ziehen. Für manche Bereiche ist es vor allem von Bedeutung, die reale Uhrzeit möglichst genau mitgeteilt zu bekommen, für andere jedoch lediglich die relative Zeit im LAN.

3.1 NTP-Server im LAN

NTP nimmt explizit immer an, dass bei der RTT (Round Trip Time) der Hinweg die gleiche Zeit wie der Rückweg benötigt. Da die Routen in modernen Netzen immer komplexer werden, ist dies jedoch keineswegs der Fall. So eine Annahme darf nur im LAN getroffen werden. Aus diesem Grund kann man seine Netzwerktopologie beispielsweise so gestalten, dass ein Rechner im LAN die NTP-Serverfunktion übernimmt und allen Rechnern die Zeit mitteilt, anstatt dass jeder Rechner beispielsweise auf einen NTP-Server im Internet zugreift. In Unixderivaten wird dieser Ansatz mittels eines Dämons (ntpd) gelöst, der sowohl die Zeit von einem Server (beispielsweise aus dem Internet) abfrägt, als auch selbst als NTP-Server bereitsteht. Dieser Dämon speichert zusätzlich auch das Wegdriften der lokalen Uhr von der Referenzzeit des Servers ab. Diese Berechnung wird bei einem erneuten Start des Dämons verwendet, wodurch eine deutlich schnellere Synchronisation möglich ist.
Ein Problem hierbei ist, dass, sollte man die reale Zeit benötigen und der NTP-Server eine sehr ungünstige (unregelmäßige) Route ins Internet hat, die Zeit auf sämtlichen Rechnern dann falsch ist. Dieser Ansatz ist deshalb vor allem dann empfehlenswert, wenn ein Rechner eine sehr direkte Verbindung zu NTP-Servern hat, während das restliche LAN sehr komplex aufgebaut ist und einige Geräte einen äußerst unbeständigen Weg ins Internet haben (zum Beispiel WLAN).

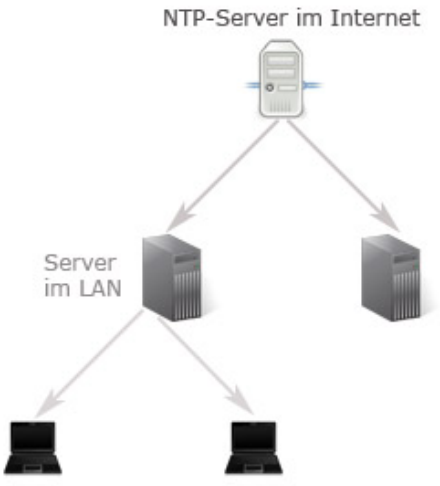

Figure 5: Server im LAN stellt Uhrzeit für alle Clients bereit

3.1.1 NTPD

Um sich die Funktionsweise einer solchen Umsetzung besser vorstellen zu können, stelle ich hier mögliche Konfigurationsdateien vom Server und Client nach [8]:

```
### Server:/etc/ntp.conf - 192.168.1.1
# Abweichungen
driftfile /var/lib/ntp/ntp.drift
# NTP-Server
server ptbtime1.ptb.de
server ptbtime2.ptb.de
# Zugriff durch NTP-Server gestatten
restrict ptbtime1.ptb.de
restrict ptbtime2.ptb.de
# Zugriff vom localhost gestatten (ntpq -p)
restrict 127.0.0.1
# Zugriff aus dem internen Netz gestatten
restrict 192.168.1.0 mask 255.255.255.0
# allen anderen Rechnern Zugriff verwehren
restrict default notrust nomodify nopeer
```

Der Server mit der IP 192.168.1.1 greift auf die beiden NTP-Server im Internet - ptbtime1.ptb.de und ptbtime2.ptb.de zu, um von dort die Zeitanfragen über den Dämon zu tätigen. Er erlaubt nur den Servern und dem lokalen Netzwerk einen Zugriff.

```
### Client:/etc/ntp.conf -
### sämtliche Clients (192.168.1.xxx)
# Abweichungen
driftfile /var/lib/ntp/ntp.drift
# NTP-Server im LAN (siehe oben)
server 192.168.1.1
# Zugriff durch NTP-Server gestatten
restrict 192.168.1.1
# Zugriff vom localhost gestatten (ntpq -p)
restrict 127.0.0.1
# allen anderen Rechnern Zugriff verwehren
restrict default notrust nomodify nopeer
```

Die Clients hingegen erlauben nur dem NTP-Server im LAN einen Zugriff - weder den restlichen Clients noch Servern aus dem Internet. Somit ist sichergestellt, dass es nur einen Server (im LAN) gibt, von dem alle Clients die Uhrzeit beziehen.
Das jeweils in der ersten Zeile angegebene Driftfile speichert das oben erwähnte Wegdriften der lokalen Uhr.

3.2 NTP-Server synchronisieren

Eine andere Möglichkeit ist, dass sämtliche Rechner im LAN auf einen NTP-Server aus einem niedrigeren Stratum zugreifen und dass anschließend die Zeit der Rechner innerhalb des LANs synchronisiert wird.
Eine Möglichkeit hierzu ist der Berkeley Algorithmus, den ich im folgenden Unterkapitel vorstellen möchte. Optimalerweise greifen sämtliche Clients auf den gleichen NTP-Server zu. Sollten alle Rechner einen sehr ähnlichen Weg in das Internet haben (zum Beispiel LAN-Verkabelung) empfiehlt sich dieser Ansatz.

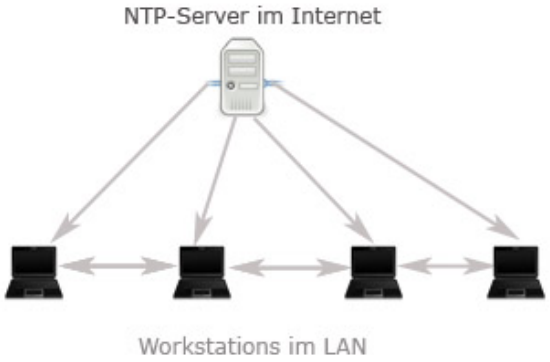

Figure 6: Alle Clients nehmen die Uhrzeit von einem NTP Server und synchronisieren sich anschließend

3.2.1 Berkeley Algorithmus

Der Berkeley Algorithmus verfolgt einen gegensätzlichen Ansatz zu NTP - anstatt einer passiven Verbindung, wählt er eine aktive. Dabei läuft ständig im Hintergrund ein Dämon, welcher die Clients stets nach ihrer Uhrzeit befragt und mittels der Antworten eine Durchschnittszeit berechnet. Dabei teilt er den schnelleren Uhren mit, sich zu verlangsamen und den langsameren teilt er die aktuelle Zeit mit, die sowohl die Clients als auch der Server anschließend übernehmen. In Abbildung 7 wird diese Funktionsweise dargestellt. [9]

Der Berkeley Algorithmus findet vor allem dann eine Verwendung, wenn nicht die genaue Uhrzeit (im Sinne der realen Uhrzeit) wichtig ist, sondern, dass alle Uhren die selbe Zeit haben. Er ist auch lediglich im LAN (Local Area Network) von Bedeutung, da selbst hier, je nach Begebenheiten, nur eine Genauigkeit von 10 ms erreicht werden kann.

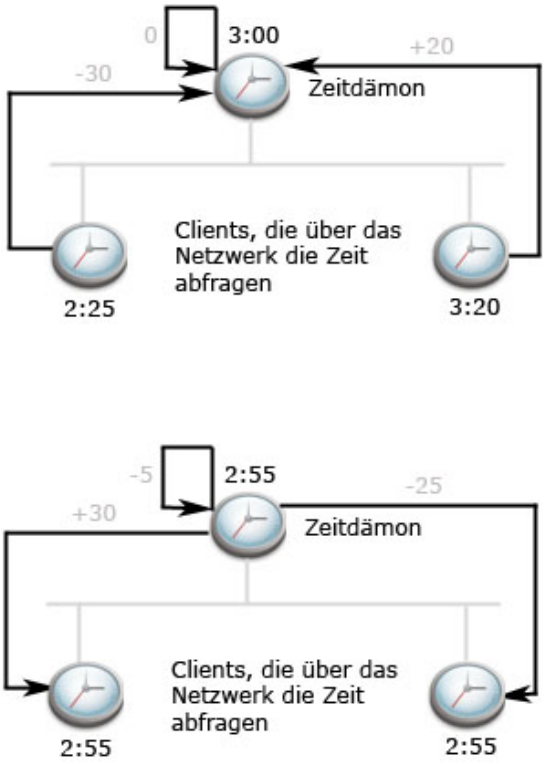

Figure 7: Berkeley Algorithmus

Auch der Berkeley Algorithmus ist wie NTP ein zentraler Algorithmus mit den damit verbundenen Nachteilen. Man könnte einen solchen Ansatz auch dezentral realisieren.

3.3 Kombination der beiden Ansätze

Selbstverständlich kann man die beiden obigen Ansätze auch kombinieren. Dies macht in der Praxis vor allem dann Sinn wenn im LAN sowohl mehrere Server, die ihre Zeit von eventuell verschiedenen NTP-Servern im Internet abfragen, als auch viele Clients sind. So lassen sich die Server synchronisieren und die zur Verfügung gestellte Zeit kann anschließend von den Clients abgefragt werden.

3.4 Weitere Kriterien für eine genaue Zeit

Neben einer möglichst direkten und stabilen Verbindung ins Internet, sollte man weiterhin darauf achten, dass man die Zeit von sehr zuverlässigen NTP-Servern abfrägt. Dabei spielt eine entscheidende Rolle, dass man mehrere Server auswählt um bei Unerreichbarkeit eines Servers immer noch eine brauchbare Zeit zu bekommen. Gewisse Begebenheiten (wie eben zum Beispiel die Stabilität des Netzwerkes außerhalb des LANs) kann man leider nicht unmittelbar beeinflussen.

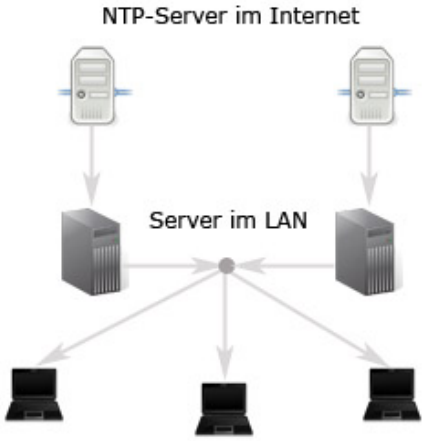

Figure 8: Kombinieren der Ansätze

4. FAZIT

Man sollte sich bewusst sein, dass eine genaue Zeit eine entscheidende Rolle in einem System spielt. Lange Zeit wurde diesem Problem eine zu geringe Aufmerksamkeit geschenkt. 1988 wurden zufällig 5722 Server und Gateways aus dem Internet überprüft, wovon 1158 ihre Zeit Clients zur Verfügung stellten. 60 Prozent der Anfragen lieferten Fehler mit mehr als einer Minute zurück, 10 Prozent mehr als 13 Minuten und ein paar sogar Fehler über 2 Jahre. [6] Glücklicherweise ist das Bewusstsein über eine genaue Zeitsynchronisation bei den Administratoren gestiegen.
Ein Hauptproblem von NTP sehe ich darin, dass es 1985 entwickelt wurde - zu einer Zeit, als das Internet noch sehr einfach aufgebaut war. Die Technik ist aber enorm schnell vorangeschritten und viele Annahmen (wie die erwähnte RTT) gelten in modernen Netzen keineswegs mehr. Gerade diesem Problem muss in den folgenden Versionen von NTP große Aufmerksamkeit geschenkt werden. Zum derzeitigen Stand der Technik denke ich, dass für eine genaue Synchronisierung, die vorgestellten alternativen Ansätze eine recht zufriedenstellende Lösung, zumindest für eine relative Uhrzeit im LAN, bieten.
Es gibt einige komplett andere Ansätze, wie zum Beispiel die Zeit über GPS zu beziehen, was theoretisch eine sehr genaue Zeit liefert. Jedoch scheitert dieser Ansatz in der Praxis, da in Häusern kein GPS-Empfang möglich ist.
Eine absolut genaue Zeit mit nicht mehr messbaren Fehlern, wird aufgrund der Komplexität der Netzwerke meiner Meinung nach ohnehin nicht so schnell möglich sein.

5. REFERENCES

[1] G. B. David Deeths. Using NTP to Control and Synchronize System Clocks - Part III: NTP Monitoring and Troubleshooting. http://www.sun.com/blueprints/0901/NTPpt3.pdf, September 2001.
[2] A. K. et al. Marzullo's algorithm. http://en.wikipedia.org/w/index.php?title=Marzullo's_algorithm&oldid=206416743, April 2006.
[3] D. L. Mills. RFC 1305 - Network Time Protocol

(Version 3) Specification, Implementation and Analysis. http://tools.ietf.org/html/rfc1305.

[4] D. L. Mills. Network Time Protocol (NTP) General Overview. http://www.eecis.udel.edu/~mills/database/brief/overview/overview.pdf, August 2004.

[5] D. L. Mills. NTP Security Model. http://www.eecis.udel.edu/~mills/database/brief/autokey/autokey.pdf, August 2004.

[6] D. L. Mills. Network Time Protocol Version 4 Reference and Implementation Guide. http://www.eecis.udel.edu/~mills/database/reports/ntp4/ntp4.pdf, Juni 2006.

[7] D. L. Mills. NTP Architecture, Protocol and Algorithms. http://www.eecis.udel.edu/~mills/database/brief/arch/arch.pdf, Juli 2007.

[8] M. Rasp. Zeitsynchronisation mit NTP (Client/Server). http://www.linux-fuer-alle.de/doc_show.php?docid=7&catid=15, September 2007.

[9] A. S. Tanenbaum. Verteilte Systeme: Prinzipien und Paradigmen. Pearson Studium, November 2007.