NTP — Überblick

H. Zuleger, March 30, Corona 2nd

Inhaltsverzeichnis

Einführung

Zeitsynchronisation ist vergleichbar mit einer Armbanduhr die man zwar zum Ablesen der Zeit verwendet, die aber häufig falsch geht und man daher gern mal den Bus oder auch das Mittagessen verpasst.

Eine Möglichkeit ist es, in regelmäßigen Abständen die Armbanduhr nachzustellen, d.h. sie mit einer Referenzzeit zu synchronisieren. Einmal am Tag (z.B. Morgens) gleicht man daher die Uhr mit der Kirchturmsuhr ab und korrigiert entsprechend. Dieses Verfahren entspricht ungefähr der Art wie *SNTP* (**S**imple **N**etwork **T**ime **P**rotocol) arbeitet.

Was bei der Armbanduhr noch gut funktionieren mag, stellt bei Rechneruhren ein Problem dar, verletzt das Nachstellen der Uhr, doch grundlegende Eigenschaften der Zeit.[1] Die Rechneruhr zurückzustellen erzeugt u.U. Zeitstempel in der falschen Reihenfolge, was nicht nur irritierend ist, sondern schwerwiegende Folgen haben kann. Stellt man die Uhr sprunghaft vor, „verliert“ man Zeitpunkte und damit u.U. regelmässig über `cron` gesteuerte Tätigkeiten.

Man kann das Problem minimieren indem man genauere Armbanduhren verwendet oder die Zeitspannen zwischen dem Abgleich verkürzt.
Um jedoch das grundlegende Problem der Zeitsprünge zu lösen, bleibt nur die Frequenz der Uhr anzupassen. Wir können die Zeitdauer einer Sekunde ändern, und damit die Uhr schneller oder langsamer laufen lassen. Damit verletzen wir zwar das Gesetz der "Gleichförmigkeit", aber das gilt ja seit Einstein sowieso nur noch eingeschränkt.[2]

Bei dem genannten Beispiel geht es übrigends **nicht** um die Sicherstellung, dass die Kirchturmsuhr die „korrekte“ Zeit anzeigt, sondern lediglich darum, dass Alle die diese Uhr als Referenz benutzen, sich auf die gleiche Uhrzeit beziehen. Schließlich ging es ja nur darum die Mahlzeiten nicht zu verpassen.

Bei NTP ist das zwar anders: Hier wird in der Regel auf die „Koordinerte Weltzeit“ (UTC) synchronisiert. Aber sonst trifft’s ungefähr die Arbeitsweise und Aufgabe von dem **N**etwork **T**ime **P**rotocol.

Zeitsynchronisierung Historisch

Bei uns auf dem Dorf läuten jeden Tag drei mal die Glocken:

- Morgens um 7:00 Uhr
- Mittags um 12:00 Uhr und
- Abends um 17:00 Uhr.

Für die (früher seltenen) Nutzer:innen von Taschenuhren sind dies drei mal täglich Zeitbroadcasts zum Stellen der eigenen Uhr. Für die Feldarbeiter:innen sind es Hinweise zu den Mahlzeiten nach Hause zu kommen.

Das obige Kirchturmsverfahren überträgt übrigends *keine* Zeitstempel, d.h. die aktuelle Uhrzeit wird *nicht* übermittelt. Das ist bei anderen Kirchtürmen anders: Hier wird häufig zu jeder Stunde (manchmal auch jede Viertelstunde) das Zeitsignal (Anzahl der Tagesstunden) übertragen. Dabei können, je nach Ausstattung im Turm, sogar unterschiedliche akustische Signale genutzt werden. Die Viertelstunden mit einem, zwei, drei, resp. vier hellen Tönen, und zur vollen Stunde dann anschließend die Stundenzahl mit einer tiefen Glocke.

Abbildung 1. Clock Gun

In Edinburgh wird täglich um 13:00 Uhr GMT eine Kanone (http://oneoclockgun.1oclockgun.org/index.html) abgefeuert. Sinn ist ebenfalls die akustische Zeitsynchronisation (wobei man dabei die Signallaufzeit nicht ausser Acht lassen sollte).

Abbildung 2. Time Ball

Der Kanonenschuss wurde als akustische Unterstützung einer zuvor bereits installierten Methode über einen „Time Ball“ eingeführt. Seit 1861 bewegt sich täglich ein Ball (http://timeball.1oclockgun.org/index.html) zur Spitze des Nelson Monuments am Calton Hill, der pünktlich um ein Uhr wieder herunterfällt. Ähnliche Instrumente zur Zeitsynchronisation findet man auch z.B. in Greenwich.

In Berlin wurden ab 1891 ca. 30 sogennannte Urania Säulen aufgebaut, deren eingebaute Uhr auf ca. eine Sekunde genau ging. (Siehe hierzu [3]).

Ab 1933 wurde in Deutschland die sogenannte „Normalzeit“ (https://uhr.ptb.de/) durch eine Quarzuhr, und seit 1969 über eine Atomuhr bei der Physikalisch Technischen Bundesanstalt (https://www.ptb.de/cms/) (PTB) in Braunschweig erzeugt und bereitgestellt. Hierfür kommt ab 1959 ein Langwellensender in Mainflingen zum Einsatz, der ganz Europa abdeckt und mit einer genauen Zeitquelle versorgt. Dieser Sender wird heute noch von den weit verbreiteten Funkuhren verwendet. Manche NTP Zeitserver nutzen ebenfalls einen Funkempfänger um auf diese offizielle Zeitquelle zuzugreifen.

Grundlagen

NTP ist ein **Protokoll** zur „Übermittlung“ der Zeit über ein (IP)Netz: **N**etwork **T**ime **P**rotocol.
Besser: Ein Protokoll zur **Synchronisation** von (Rechner) Uhren über ein (IP)Netz.
NTP ist insbesondere **keine Zeitquelle** d.h. es ist selbst keine Uhr! Als Zeitquelle dient typischerweise

- eine Atomuhr :-)
- ein Radioempfänger (DCF77 in Deutschland)
- ein GPS Empfänger (Weltweit)

- oder auch eine eingebaute (Quarz)Uhr.

Steht all dies nicht zur Verfügung, kann man als Referenz auch eine entfernte Uhr über NTP ansprechen. Genau dafür ist es ja entwickelt worden.

Die Genauigkeit der Zeitquelle ist nur wichtig wenn es um eine absolute Zeit geht. Dies muss nicht in allen Fällen gegeben sein, allerdings wird im Regelfall eine Synchronisation der Rechneruhr mit der Koordinierten Weltzeit (UTC) angestrebt.

Bei Unix Systemen ist die interne Rechneruhr immer auf UTC eingestellt. Lediglich in der Anzeige wird diese auf die Lokalzeit umgerechnet.

Protokoll Übersicht

NTP ist ein leichtgewichtiges Protokoll. Per Design soll ein Server viele tausende Clients bedienen können, weshalb viel Wert darauf gelegt wurde, dass auf Serverseite kein „State“ vorgehalten werden muss.

Als Protokoll kommt **UDP** mit Port 123 zum Einsatz. Unterstützt wird sowohl IPv4 als auch IPv6. Ein NTP Datagramm ist lediglich 48 Byte groß. Übertragen werden mehrere Zeitstempel, die jeweils eine absolute Zeitangabe (UTC) die als Anzahl Sekunden seit 1.1.1900 sowie Sekundenbruchteile enthalten.

NTP versucht Paketlaufzeiten bei der Übertragung „herauszurechnen“. Dies funktioniert nicht sehr gut bei unterschiedlichen Laufzeiten in Hin- und Rückrichtung (asymmetrische Bandbreiten).

Ein NTP Server kann (meist über eine serielle Verbindung) auf externe Zeitquelle zugreifen. Originärer Einsatzzweck ist aber die Synchronisation mit anderen Uhren über ein Datennetz. Jedes NTP Instanz ist Client und Server zugleich.

Anwendungsbereiche

Warum ist eine Zeitsynchronisation überhaupt notwendig? Welche Genauigkeit ist gefordert? Wird eine absolute Zeit benötigt oder geht es lediglich um Zeitgleichheit?

Eine sehr unvollständige Auflistung einiger typischer Anwendungen, die unterschiedliche Anforderungen an die Synchronizität von Rechneruhren stellt:

1. Zertifikatsprüfung o.ä.
   - Genauigkeit im Minuten Bereich ist ausreichend
   - Absolute Zeitangabe notwendig
2. Auswertung von Logdateien, Firewalls, etc.
   - Genauigkeit im Sekunden resp. Subsekunden Bereich
   - Je nach Anwendung auch mal einstellige Millisekunden
   - Absolute Zeit i.d. Regel erwünscht
3. Abgleich und Auswertung von Traces auf verschiedenen Maschinen
   - Genauigkeit im Millisekunden Bereich
   - Absolute Zeit nicht notwendig
4. Zeitsynchronisation für Funk- und Netzwerkprotokolle
   - Time sensitive Networking (https://en.wikipedia.org/wiki/Time-Sensitive_Networking)
   - Mobilfunk / Finanz Transaktionen
   - Sub Microseconds
   - Absolute Zeit nicht zwingend notwendig

Der letzte Punkt zeigt Anwendungsbereiche die bisher durch NTP nicht abgedeckt werden konnten. Hierfür wurden anderen Protokolle wie z.B. das Precision Time Protocol (PTP) verwendet. NTP zieht aber aktuell nach und hat seine Genauigkeit verbessert.

Implementierungen

Früher gab es im wesentlichen die Referenzimplementierung (`ntpd`) von David L. Mills. In den letzten Jahren hat sich jedoch eine größere Vielfalt an Zeit–Synchronisations Programmen entwickelt.

- `ntpd` : Der Klassiker von Mills
- Die OpenBSD Variante `OpenNTPD`
- `NTPsec` : Fork und designierter Nachfolger von `ntpd`
  Implementiert die neue Authentifizierungsvariante NTS
- `chrony` : NTP und NTS Implementierung
- Simple Network Time Protocol (`sntp`)
  Einfacher Client ohne State

Funktionsweise

- NTP ist keine Zeitquelle, sondern ein Zeit Übertragungs- und Synchronisations-Protokoll
- Ein NTP Server synchronisiert sich entweder
  - mit einer externen Zeitquelle, oder
  - mit einem anderen NTP Server
- Eine externe Zeitquelle wird als Stratum 0 bezeichnet
- Ein NTP Server der auf eine Stratum 0 Quelle zurückgreift ist selber ein Stratum 1 Server.
- Jeder NTP–Zeitserver hat immer ein Stratum der um eins größer ist als der Stratum des Servers den er als Referenz nutzt. Jede NTP („Client“) Instanz ist gleichzeitig auch Server.
- Synchronisation der Rechneruhr wird erreicht durch
  - minimales Verändern der Zeitdauer einer Sekunde (Frequenzanpassung)
  - Dies ist ein lange dauernder Vorgang: Veränderung ist typischerweise kleiner als 0.5ms / Sekunde
  - Das Ziel dabei: Vermeiden von Zeitsprüngen!
  - Insbesondere Rückwärtssprünge sind ein Problem
  - Zeit ist ein kontinuierlich, gleichmäßig, fortschreitender Prozess
- Die Zeitübertragung erfolgt in Sekunden und Sekundenbruchteile (jeweils als 32 Bit Wert) seit 1.1.1900 (Epoch 0)
- Als Übertragungsprotokoll wird UDP mit IPv4 oder IPv6 als Transport verwendet
- Die Größe eines Basis Datagramms (also ohne Authentisierung und Extension Felder) beträgt lediglich 12 Langwörter (32Bit), d.h. inklusive UDP Header gerade mal 56 Byte.
- Ein Synchronisationsabgleich besteht aus je einem Datagramm vom Client zum Server und zurück.
- Die Polling Frequenz wird zwischen 64 Sekunden und 1024 Sekunden angepasst. [4]
- Die Zeitstempel beziehen sich auf die „Koordinierte Weltzeit“ (Universal Time Coordinated)

Zeitzonen

Die Erde ist in unterschiedliche Zeitzonen aufgeteilt. Diese richten sich nach den Längengeraden auf der Erde. Der Null–Meridian (https://www.heise.de/hintergrund/Zahlen-bitte-Der-Nullmeridian-oder-Irgendwo-muss-die-Weltkugel-ja-anfangen-7302816.html) (Längengrad 0), der östlich von London durch den kleinen Ort Greenwich verläuft bildet den Referenzpunkt für die Zeitzonen. Früher wurde diese als Greenwich Mean Time (GMT) bezeichnet.

Deutschland liegt in der Mitteleuropäischen Zeitzone (https://www.timeanddate.de/zeitzonen/deutschland) (Central European Time) die eine Stunde östlich von Großbritanien liegt. Zeitlich liegen wir daher „vor“ England.

Bei einer weltweiten Vernetzung von Computern ist es sinnvoll alle Rechneruhren auf die gleiche Zeit zu synchronisieren. Alle Unix Rechneruhren sind daher intern auf UTC eingestellt. In der Anzeige von `date` oder `ls` (und bei anderen Programmen die eine absolute Zeit anzeigen) findet dann die Umrechnung in die jeweilige lokale Zeitzone statt.

Dies erlaubt nicht nur eine Zeitanzeige nach eigenen Präferenzen sondern verhindert auch die Notwendigkeit der Umstellung der Rechneruhr auf Sommer- resp. Winterzeit.

Bei Unix Systemen wird die angezeigte Zeit durch die Umgebungsvariable `TZ` gesteuert. Die Ausgabe von `date` erfolgt immer in der lokal eingestellten Zeitzone. Die Universalzeit (also die interne Uhr) wird über die Option `-u` angezeigt.

```
$ date
Sat  6 Mar 23:08:53 CET 2021
$ date -u
Sat  6 Mar 22:08:57 UTC 2021
```

Über das Setzen der Umgebungsvariable kann aber auch jede andere Zeitzone zur Anzeige gebracht werden, inklusive korrekter Sommerzeit.

```
$ TZ=:America/Chicago date
Sat  6 Mar 16:09:12 CST 2021
        # Ein Blick in die Zukunft
$ date -d "2021-3-27 16:00"
Sa 27. Mär 16:00:00 CET 2021
$ date -d "2021-3-28 16:00"
So 28. Mär 16:00:00 CEST 2021
```

Die Koordinierte Weltzeit im Verhältnis zur Erdzeit

Die Koordinierte Weltzeit basiert auf der internationalen Atomzeit (TAI) die durch weltweit verteilte Atomuhren ermittelt wird.

Allerdings schafft es unsere Erde nicht sich mit der Zeit dieser Atomuhren zu synchronisieren.[5] Die Erde läuft etwas unrund (mal schneller und aktuell etwas langsamer), d.h. die Sonnenzeit (UT1) weicht aktuell um 37 Sekunden von der Atomzeit ab. Den Physikern um die TAI ist das eher egal sie halten an „ihrer“ Zeit fest.
Für die koordinierte Weltzeit hat man jedoch entschieden diese an die Erdzeit anzupassen. Und so werden in unregelmäßigen Abständen Schaltsekunden (https://www.timeanddate.de/zeitzonen/schaltsekunden) eingefügt oder weggelassen.

NTP unterstützt dies. Die letzte derart eingefügte Sekunde war am 31.12.2016 23:59:60 (UTC). Die nächste geplante Schaltsekunde könnte am 31. Dezember 2021 eingefügt werden, allerdings gibt die Erde aktuell gerade etwas gas (https://www.telegraph.co.uk/news/2021/01/04/earth-spinning-faster-now-time-past-half-century/). Ursprünglich war die Schaltsekunde bereits für Juni 2021 (https://datacenter.iers.org/data/latestVersion/16_BULLETIN_C16.txt) geplant. Ob der 31. Dezember 2021 gehalten wird ist noch ungewiss.

NTP Modi

NTP unterscheidet unterschiedliche Arbeitsweisen bzw. Modi. Diese werden in den NTP-Datenpaketen durch Ziffern unterschieden.

- Modus 1+2 für gegenseitige Zeitsynchronisation (Peers, also „gleich gute“ Zeitquellen)
- Modus 3+4 für Synchronisation mit einer „besseren“ Referenz Zeitquelle (Client-Server))
- Modus 5 für Broadcast/Multicast Übertragung (1:n)
- Modus 6 zur Abfrage und Konfiguration des NTP-Prozesses (`ntpq`)

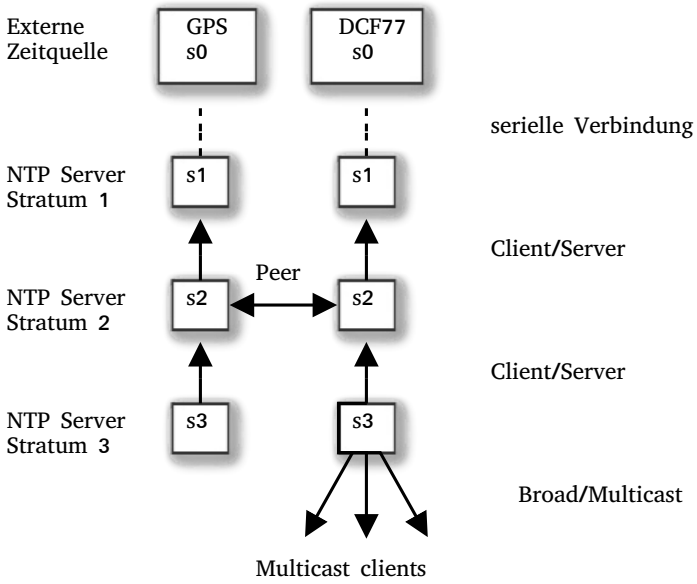


Abbildung 3. Die NTP Verfahren zur Zeitsynchronisation

Eine NTP Instanz synchronisiert sich immer entweder mit einer externen Zeitquelle oder mit anderen NTP Instanzen.

Referenzquellen

Eine externe Zeitquelle gilt aus Sicht des NTP-Protokolls als sogenannte Stratum 0. Darunter versteht man eine präzise Zeitquelle, meist eine Atomuhr. Diese liefert bereits die UTC Zeit, oder sie wird durch den NTP Server in UTC umgerechnet. Entsprechende Zeitquellen können sein:

- GPS/GLONASS/Galileo Empfänger (Jeder Satellit enthält eine Atomuhr)
- Radio Clock (z.B. DCF77 Mainflingen (https://de.wikipedia.org/wiki/DCF77) bei Seligenstadt/Mainhausen geo:50.01556,9.01083)
- Eine stationäre Atom Uhr (z.B. die der PTB (https://uhr.ptb.de/) in Braunschweig) erreichbar z.B. über eine Telefonleitung
- In dem RFC zu NTPv4 werden noch andere mögliche Referenzquellen benannt

Tabelle 1. Referenzuhren (IDs) (Auszug aus RFC5905)

ID	Clock Source
GOES	Geosynchronous Orbit Environment Satellite
GPS	Global Position System
GAL	Galileo Positioning System
PPS	Generic pulse-per-second

IRIG	Inter-Range Instrumentation Group
DCF	LF Radio DCF77 Mainflingen, DE 77.5 kHz
LORC	MF Radio LORAN C station, 100 kHz
NIST	NIST telephone modem
PTB	European telephone modem (PTB Atomuhr/Braunschweig)

NTP-Hierarchien

- Client/Server Synchronisierungen erzeugen eine Hierarchie.
- Peersynchronisationen finden auf der gleichen Ebene statt.

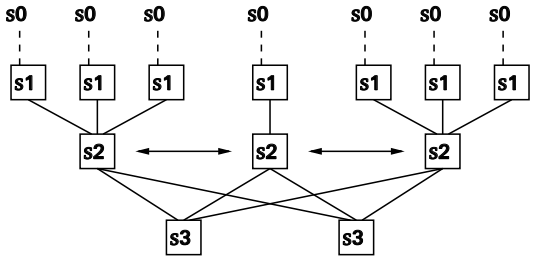


Abbildung 4. Hierarchie von NTP Servern

Das Peer Verfahren wird meines Erachtens in der Praxis kaum mehr angewendet und von *NTPsec* auch nicht mehr unterstützt.

Gleiches gilt übrigends auch für Broad- bzw. Multicast Server. Im folgenden wird daher primär auf die Arbeitsweise bei Client-Server Assoziationen eingegangen. Multicast Setups werden im Abschnitt [_autokonfiguration] beschrieben.

Zeitermittlung

Für eine Client-Server Verbindung gilt folgende Vorgehensweise

- Die Kommunikation ist vom Client initiiert
  - Die Adresse des Servers wird typischerweise in der Client Konfiguration hinterlegt (Alternativen sind möglich siehe Abschnitt [_autokonfiguration]).
  - Server operieren stateless (!)
  - Clients halten (in der Regel) „Assoziationen“ zu mehreren Servern ([_redundanz])
- Clients versuchen ihre Uhr mit der des „besten“ Servers zu synchronisieren
  Welcher der beste Server ist wird durch mathematische Berechnungen ermittelt
- Im Protokoll werden mehrere Zeitstempel übertragen
  - Alle Zeitpunkte (Timestamps) sind 32-Bit Sekunden plus 32-Bit Sekunden-Bruchteile
  - Startpunkt für Epoch 0 ist der 1.1.1900
  - Erster Rollover (zu Epoch 1) am 7. Feb 2036
  - Einfügen einer Schaltsekunde wird angezeigt

Das Protokoll versucht grundlegend zwei Werte zu ermitteln:

a. Den Offset zur Referenzquelle

b. Das Delay bei der Übermittlung über das IP-Netz

Hierfür werden vier Zeitwerte herangezogen

- **Ot**: Absende Zeitpunkt des Clientpaketes (Originate Timestamp)
- **Rt**: Zeitpunkt des Empfangs beim Server (Receive Timestamp)
- **Tt**: Absende Zeitpunkt des Serverpaketes (Transmit Timestamp)
- **Dt**: Zeitpunkt des Empfangs beim Client (Destination Timestamp)

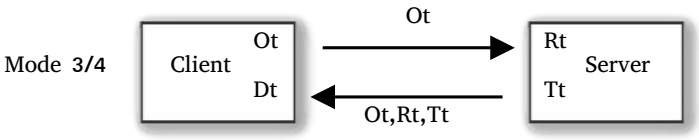


Abbildung 5. Übertragene Zeitwerte bei der Client-Server Kommunikation

Die **Delay** Berechnung ist dann ` delta = (Dt - Ot) - (Tt - Rt) `.

Und der **Offset** entsprechend ` theta = ( (Rt - Ot) + (Tt - Dt)) / 2 `.

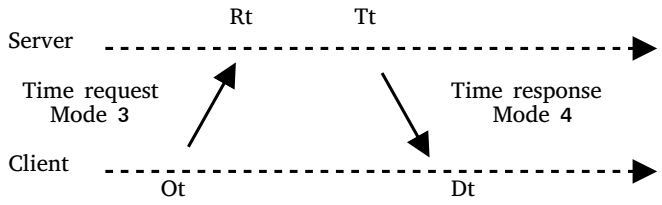


Abbildung 6. Zeitleiste bei der Client-Server Kommunikation

Danach wird mit viel (mir unbekannter) Mathematik berechnet wer die „beste“ Quelle ist. Zu dieser synchronisiert sich der Client. Gleichzeitig dient der Algorithmus zum Erkennen sogenannter *false ticker*, also Uhren die mit ihrer Zeit „weit“ ausserhalb dessen liegen was NTP als sinnvoll erachtet.

Kontrolle

Ein laufender NTP Prozess kann über das Kommando `ntpq` abgefragt werden [6] um den Status der Synchronisation zu zeigen.

Beispiel 1. Abfrage eines NTP Server mit mehreren Assoziationen

```
$ ntpq -p -w
     remote        refid         st t when poll reach   delay   offset  jitter
==============================================================================
 2.debian.pool.ntp.org
                 .POOL.          16 p    -   64    0    0.000    0.000   0.001
-mux.hznet.de    130.149.17.8     2 u  903 1024  377   20.483   -0.348   0.184
-electrode.felixc.at
                 253.10.71.116    3 u  771 1024  377   20.103   -1.450   0.326
-2a02:8106:19::5 .PPS.            1 u 1247 1024  376   36.074    0.575   3.243
+2a01:4f9:c010:157::1
                 95.216.154.135   3 u  461 1024  377   40.143   -1.191   0.514
*a.chl.la        131.188.3.222    2 u  679 1024  377   18.646   -1.179   0.664
-time.cloudflare.com
                 10.21.8.19       3 u   40 1024  377   26.660   -2.585   0.576
-ts5.sct.de      192.53.103.108   2 u  492 1024  377   20.449   -0.580   0.492
-ntp01.fra-pool.fastether.net
                 237.17.204.95    2 u  342 1024  377   15.557    1.066   0.344
+mail.morbitzer.de
                 79.133.44.136    2 u  121 1024  377   18.237   -0.488   0.453
```

Das Zeichen am Anfang der Zeile zeigt den Status des Servers:

`*` ⇒ Syncronised Peer
`#` ⇒ Backup
`+` ⇒ Candidate Peer
`-` ⇒ False Ticker
Leerzeichen ⇒ Undefined

Die Spalte *remote* enthält den Rechnernamen (reverse DNS Lookup) oder die IP-Adresse (Option `-n`). Durch die Option `-w` wird sowohl der Rechnername als auch eine IPv6 Adresse vollständig ausgeschrieben. Die Information der nächsten Spalten stehen dann u.U. eine Zeile tiefer.

In *refid* steht die IPv4-Adresse oder der 32 Bit Hash der IPv6 Adresse des „Upstream“ Zeitservers. Ist der Stratum Wert 1 (nächste Spalte), enthält die Spalte eine vier Zeichen lange Bezeichnung der Zeitquelle (`.PPS.`). Ist der Stratum Wert 16, können dort auch andere Angaben stehen wie z.B. `.POOL.`, `.NTS.` oder ähnliches.

Unter der Überschrift *st* ist der Stratum Wert des Servers aufgeführt und *t* ist der Typ der Assoziation: `u` steht für Unicast, `b` entsprechend *broad/multicast*, `p` meint *Pool* Definition. Bei erfolgreicher NTS Authentifizierung steht hier eine Ziffer mit der Anzahl der verbleibenden Cookies.

Die Spalte *pool* gibt die Rate (in Zweierpotenzen) der Anfragen an diesen Server an. Der Wert startet bei 64 und der aktuelle Maximalwert ist 1024. In der Spalte davor findet sich der Sekundenzähler seit der letzten Anfrage. Server werden also in einem Intervall von ungefähr eine bis 17 Minuten angefragt.

Ist der Server nicht erreichbar wird das Abfrageintervall verlängert. Die Erreichbarkeit der letzten acht Anfragen wird in einem ein Byte großen Schieberegister *reach* festgehalten (Achtung: In der Anzeige oktale Darstellung).[7]

Detailliertere Angaben einer Gegenstelle werden durch die Abfrage der internen Variablen angezeigt. Hierzu benötigt man die sog. „Assoziation ID“, die man über ein weiteres Kommando ermitteln kann:

Beispiel 2. Detaillierter Status

```
$ ntpq -c asso
ind assid status  conf reach auth condition  last_event cnt
===========================================================
  1 30064  8811   yes  none  none    reject    mobilize  1
  2 30065  f314   yes   yes   ok    outlier   reachable  1
  3 30066  1314    no   yes  none   outlier   reachable  1
  4 30067  132a    no   yes  none   outlier    sys_peer  2
  5 30068  147a    no   yes  none candidate    sys_peer  7
  6 30069  16fa    no   yes  none  sys.peer    sys_peer 15
  7 30070  135a    no   yes  none   outlier    sys_peer  5
  8 30071  136a    no   yes  none   outlier    sys_peer  6
  9 30072  13fa    no   yes  none   outlier    sys_peer 15
 10 30073  14fa    no   yes  none candidate    sys_peer 15
```

```
$ ntpq -c "rv 30065"
associd=30065 status=f314 conf, authenb, auth, reach, sel_outlier, 1 event, reachable,
srcadr=mux.hznet.de, srcport=123,
dstadr=2003:a:b45:6310:d8b0:4f87:1718:7210, dstport=123, leap=00,
stratum=2, precision=-24, rootdelay=19.104, rootdisp=30.029,
refid=130.149.17.8,
reftime=e3fc9eb3.dd359c3f  Wed, Mar 17 2021 16:28:51.864,
rec=e3fca227.d9191d08  Wed, Mar 17 2021 16:43:35.848, reach=377,
unreach=0, hmode=3, pmode=4, hpoll=10, ppoll=10, headway=49, flash=00 ok,
keyid=11, offset=-0.580, delay=20.350, dispersion=15.498, jitter=0.376,
xleave=0.202,
filtdelay=    20.35   20.52   20.48   20.34   20.53   20.54   20.55   20.57,
filtoffset=   -0.58   -0.52   -0.35   -0.21   -0.19   -0.10   -0.13   -0.13,
filtdisp=      0.00   16.11   32.16   48.35   63.93   79.80   95.70  111.50
```

Das Kommando `ntpq` ist Teil des klassischen *ntpd* Paketes. Es wird in leicht veränderter Form auch mit der neueren *ntpsec* Implementierung mitgeliefert. Alle folgenden Ausgaben sind mit der neueren Variante erstellt.

Chrony besitzt ein eigenes Kommando zur Abfrage des NTP-Server Status.

Redundanz

Bei sehr vielen Internet Diensten spielt das Thema Redundanz eine Rolle. Die Frage ist, was passiert bei Ausfall des Dienstes. Häufig wird dann ein zweites System, oft in georedundantem Aufbau, inklusiv Methoden der Ausfallerkennung und Umschaltung auf den Backup Standort installiert.

Bei NTP stellt sich eine ähnliche Frage, allerdings sieht die Beantwortung anders aus. Dies hat mit der Arbeitsweise von NTP zu tun. Das Protokoll bringt von sich aus bereits sehr viele Mechanismen mit um die Qualität einer vielzahl von Zeitquellen zu ermitteln, und sich dann mit der Besten zu synchronisieren. Dies schliest auch eine Resynchronsisation zu einem anderen Server ein, falls eine Quelle „unzuverlässiger“ werden sollte oder gar komplett ausfällt. Insofern bringt NTP alle Voraussetzungen für Redundanzen mit benötigt jedoch eine entsprechende Anzahl Zeitquellen (siehe Kasten [Uhrvertrauen]).

- Eine Server Assoziation ⇒ Blindes Folgen
  Bei Ausfall „Freerun“ der lokalen Uhr.
- Zwei Server ⇒ Welcher ist der Beste?
  Eine Synchronisation ist schwierig, insbesondere wenn beide Server abweichende Zeitvorstellungen haben.
- Drei Server ⇒ Synchronisation mit dem „Besten“
  Wer auch immer das ist, und wie auch immer dies ermittelt wird.

Ein redundantes NTP Setup ist erst ab vier Zeitquellen gegeben!

U(h)r Vertrauen

Wer ein körperliches Unwohlsein empfindet, welches nicht selbst-erklärend oder selbst-verflüchtigend ist, konsultiert in der Regel die Ärztin des Vertrauens. Im Allgemeinen traut man der Diagnose und leitet entsprechende Maßnahmen ein.

Manchmal reicht das Vertrauen jedoch nicht aus. Eine zweite Meinung muss her! Diese nutzt jedoch nur bedingt etwas: Kommt die zweite Koryphäe zur gleichen Einschätzung ist alles ok. Was jedoch wenn die Diagnosen stark voneinander abweichen?
Um eine dritte Einschätzung kommt man dann nicht herum.

Bezogen auf unser „Uhr“ Vertrauen:
Entweder einen oder drei Server konfigurieren. Und wer auf Ausfallsicherheit Wert legt braucht mindestens vier.

(Auto)Konfiguration

Woher bekommt ein NTP Daemon seine Zeitquellen? Wie kann man eine größere Anzahl an Zeitquellen konfigurieren?

Theoretisch gibt es inzwischen mehrere Verfahren:

a. Spezifizierung der Gegenstellen über das `server` bzw. `peer` - Kommando
b. Neuerdings kann entsprechendes auch über ein `pool` Kommando erreicht werden
c. Broadcast/Multicast Gruppen (z.B. die Well-Known IPv6 Multicast Gruppe `ff05::101`)
d. Manycast

Einige dieser Verfahren sind eher als Auslaufmodelle zu betrachten.

- NTP-Peers haben sich nicht wirklich durchgesetzt und die gegenseitige Synchronisation gilt als angreifbar. Ohne Authentisierung sollte eine Peer-Konfiguration nicht verwendet werden. Da man die Gegenstelle kennt, lässt sich jedoch eine Shared Secret basierte Authentifizierung mit leidlichem Aufwand implementieren. Trotzdem haben peer-Konfigurationen in der Praxis kaum eine Bedeutung.
- Broadcast/Multicast
  Auch hier ist die Quelle nicht vertrauenswürdig und darf daher ohne Authentisierung nicht genutzt werden. Darüber hinaus geht ein Multicast Client erst einmal in einen Client/Server Betrieb um die Qualität der Quelle zu ermittelt. Erst dann wird erneut in den Multicast Modus geschaltet (Sehr irritierend).
  Insgesamt kann man sagen, dass sich dieses Verfahren nicht durchgesetzt hat. Der große Vorteil, dass man es mit einer Standardkonfiguration auf Clientseite nutzen kann (Default Multicast NTP Gruppe) wird durch die Notwendigkeit einer Authentifizierung ad absurdum geführt.
- Manycast
  Ein vergleichsweiser neuer Ansatz, bei dem der Client eine Anfrage an eine Well-Know Multicast Gruppe sendet und darüber die IP-Adressen von lokal erreichbaren NTP Servern lernt. Zu diesen wird dann eine klassische Client-Server Verbindung aufgebaut. Ohne Authentifizierung hat auch diese Verfahren Schwachstellen. Insgesamt ein sehr interessanter Ansatz, aber in der Praxis m.E. nicht weit verbreitet.
  Ein Manycast Setup wird im Abschnitt [_authentifizierung] gezeigt.

Bleiben zwei Verfahren übrig, die im folgenden genauer beleuchtet werden.

Server Assoziationen

- Bei Server Assoziationen wird die Gegenstelle im Client über das `server` Kommando konfiguriert
- Typischerweise über einen DNS Namen (FQDN)
- Die Dual Stack Namensauflösung kann mit der Option `-4` / `-6` auf jeweils ein IP-Protokoll beschränkt werden
- Im Ergebnis resultiert aus einer Server Konfigurationszeile genau eine Assoziation.
  Dies passiert auch dann, wenn mehrere IP-Adressen hinter dem Namen hinterlegt sind.
- Server Assoziationen sind permanent, d.h. sie bleiben bestehen, selbst wenn der Server nicht antwortet

Vorteile

- Die Anzahl der Gegenstellen ist definierbar
- Gewisser Schutz gegen Angriffe (Angreifer muss Server IP spoofen)

Nachteile

- Gegenstelle muss bekannt sein
- Die Gegenstelle bleibt konfiguriert, selbst wenn sie aus- bzw. wegfallen sollte.
- Konfiguration häufig über DHCP, aber wer traut schon DHCP? [8]

In den Anfangszeiten des Internet war es nicht einfach einen NTP Server zur Synchronisation zu bekommen. Es gab zwar Listen mit Servern, oft sogar Stratum 1, aber häufig musste (oder sagen wir sollte) die Nutzung angekündigt werden. Obwohl NTP auf Serverseite eher „lightweight“ zu nennen ist, stand der schnell wachsenden Anzahl an Clients nur vergleichsweise wenige Server gegenüber.

Und auch die Verwaltung in Form einer Liste war ja eher unpraktisch. In dieser Zeit ist das NTP Server Pool Projekt (https://www.pool.ntp.org/en/) gestartet. Durch die Möglichkeit mit relativ geringem Hardwareaufwand auf die Atomuhr eines GPS Satelliten zuzugreifen, konnte das Ziel, eine größere Anzahl von öffentlich erreichbaren NTP Server zu etablieren erreicht werden. All diese Server sind unter der Domain `pool.ntp.org` ansprechbar.

Heute gibt es so viele Server (4378 Stand 2021-03-23), dass man begonnen hat sie nach geographischen Regionen aufzuteilen. So sollte man in Deutschland die Server im Pool `de.pool.ntp.org` nutzen.

Obwohl mehr als zwei Drittel der "deutschen" Server IPv6 beherrschen, sind diese ausschließlich im Pool `2.de.pool.ntp.org` zu finden. Für den globalen Pool gilt entsprechendes.

Hinter jedem DNS Namen sind immer vier IP-Adressen hinterlegt. Die Idee war, vier `server` zu Konfigurieren und jedem den Pool Namen zu geben.

```
server  de.pool.ntp.org
server  de.pool.ntp.org
server  de.pool.ntp.org
server  de.pool.ntp.org
```

Der DNS Resolver rotiert in der Regel die zurückgelieferten Namen, sodass jeweils unterschiedliche IP-Adressen als Server genutzt werden. Damit ist man aber letztlich von einer Funktionalität des DNS Resolver abhängig. Besser ist es in diesem Fall die vier Subdomains 0 bis 4 zu nutzen:

```
server  0.de.pool.ntp.org iburst
server  1.de.pool.ntp.org iburst
server  2.de.pool.ntp.org iburst
server  3.de.pool.ntp.org iburst
```

So ist gesichert, dass jede Server Zeile auf einen eindeutigen NTP Server verweist. Allerdings bleibt ein Problem: Fällt einer der Server aus, oder schlimmer noch zwei, findet kein Ersatz statt.

Beides sind Gründe die oben gezeigte Konfiguration heute nicht mehr in dieser Form durchzuführen. Wer den NTP Server-Pool verwenden möchte, sollte unbedingt eine Pool-Konfiguration (siehe [_pool_assoziationen]) nutzen.

Pool Assoziationen

Die Einführung des public NTP-Servers Pools führt zu einem anderen Problem. Man nutzt einen Dienst dessen SLA unbekannt ist. Jederzeit kann ein Server des NTP-Pools heruntergefahren, oder sogar komplett abgeschaltet werden, ohne dass der Betreiber dies den Poolverwaltern mitteilen müsste.

Eine `server` Angabe ist jedoch die Konfiguration einer festen Assoziation, d.h. der NTP Client versucht dauerhaft den angegeben Server zu erreichen. Ausgefallene, oder ausser Betrieb genommene Server können nur durch eine Neukonfiguration ersetzt werden.

Aus dieser Situation wurde das Pool-Kommando entwickelt. Zum einen werden über *eine* Pool-Konfigzeile *mehrere* Client-Server (Mode 3/4) Verbindungen aufgebaut, da hier **alle** zurückgelieferten IP-Adressen verwendet werden. Die oben aufgelisteten *vier* `server` Zeilen lassen sich damit durch *eine* `pool` Konfigzeile ersetzen, da im Pool zu einem DNS-Namen immer vier Adressen geliefert werden.
Zum anderen sind *pool* Konfigurationen nicht statisch. Fällt ein Server dauerhaft aus, kann er durch einen anderen Server ersetzt werden, da es sich bei der Pool Konfiguration eben nicht um eine **dedizierte** Verbindung zu genau einem Server handelt.

Die Pool Konfiguration wird als *mobilize* Assoziation in der Liste der Server Assioziation (als Stratum 16) mitgeführt und regelmässig erneuert.

Vorteile

- Standard Konfiguration für alle Clients
- Gewisser Schutz gegen Angriffe durch große Anzahl Server
- Einfache Konfiguration (Jeder DNS-Poolname liefert vier Adressen/Server)
- Server Anzahl selbstregulierend (`tos maxserver` *n*)

Nachteile

- Gegenstelle unbekannt (Vertrauenswürdig?)
- Nur ein Subpool (`2.pool.ntp.org`) unterstützt IPv6
- Bisher ohne Authentifizierung (*pool.ntp.org* könnte Serverzertifikate ausstellen?)

Mit einer Pool Konfiguration erreicht man eine selbstreparierende NTP Server Konfig.

Authentifizierung

NTP ist ein stateless UDP Protokoll, dass sich prinzipbedingt leicht angreifen lässt. Insbesondere ist ein Address-Spoofing vergleichsweise einfach möglich, womit sich ein Angreifer als legitimer Server ausgeben könnte.

Weiter oben wurde gezeigt, dass eine große Anzahl Serverassoziationen für die Arbeitsweise von NTP sinnvoll sind, und die eingebaute Fehlererkennung ein „Verstellen“ der Uhr durch einen Angreifer wirkungsvoll verhindern kann. Soll ein derartiger Angriff erfolgreich sein, müssten schon sehr viele Server Antworten gefälscht werden. Dies ist praktisch nur durchführbar wenn man netztechnisch sehr nah am Client ist.

Bei einer kleineren Anzahl von Zeitquellen und insbesondere wenn diese nicht manuell konfiguriert werden (DHCP, Multi- oder Manycast) ist eine Authentifizierung auf jeden Fall angeraten. Gleiches gilt für Peer Verbindungen.

Verfahren

Lange Zeit standen lediglich zwei Verfahren zur Authentifizierung zur Verfügung

a. Ein symmetrisches Verfahren mittels Shared Secret (MD5, SHA1)
b. Seit 2010 ein asymmetrisches Verfahren namens AutoKey (https://tools.ietf.org/html/rfc5906)

Letzteres ist schwer zu konfigurieren und wurde spätestens 2019 aufgrund von Sicherheitsproblemen in den NTP Best Current Practices (https://tools.ietf.org/html/rfc8633) als deprecated eingestuft.

Ende 2020 ist dann der RFC für „Network Time Security (NTS) für NTP“ (https://tools.ietf.org/html/rfc8915) verabschiedet worden. Dabei handelt es sich um ein skalierbares, Public-Key basiertes Verfahren zur Absicherung von NTP, welches bereits von einigen NTP-Implementierung unterstützt wird.

Symmetrische Authentifizierung

- Einfach in der Anwendung
- Gut für Peer Verbindungen
- Leider nur MD5 und SHA1 verbreitet (chrony auch AES256 o.ä.)
- Kein automatisierter Schlüsseltausch (Manuelle Konfig)
- Nicht skalierbar

Die Schlüssel werden in einer separaten Schlüsseldatei verwaltet. So können die Zugriffsrechte restriktiver gehalten werden. In der NTP Konfiguration wird auf die Datei über `key` *`pfadname`* verwiesen und die aktuell verwendeten Schlüssel als `trustedkey` markiert.

Authentifizierung in der ntp.conf

```
$ grep -e "^key" -e "^trustedkey" /etc/ntp.conf
keys /etc/ntp.keys      ❶
trustedkey 11 17        ❷
trustedkey (20 ... 40)  ❸
```

❶ Laden der Datei mit den Schlüsseldefinitionen

❷ Schlüssel mit Nummer 11 und 17 erlauben

❸ Schlüssel mit den Nummern 20 bis 40 erlauben

Das Format der Schlüsseldatei ist einfach: Pro Zeile ein Schlüssel. Die Zeile beeinhaltet die Schlüsselnummer, den Schlüsselalgorithmus (`MD5` , `SHA1`) und das Shared Secret.

Die ntp.keys Schlüsseldatei

```
$ cat /etc/ntp.keys
1 MD5 hgjfdshjfsahfh
11 SHA1 fhfhflhfhjeöjfioru83h
12 SHA1 ....................
13 SHA1 -----
```

Konfiguration & Status eines NTP Client mit Authentifizierung

```
$ grep -e "^server" -e "^keys" -e "^trustedkey" /etc/ntp.conf
keys /etc/ntp.keys
trustedkey 11
server mux.hznet.de iburst key 11

$ ntpq -p -w
     remote           refid      st t when poll reach   delay   offset  jitter
 ==============================================================================
 2.debian.pool.ntp.org
                 .POOL.          16 p    -   64    0    0.000    0.000   0.001
-mux.hznet.de    85.10.240.253    3 u  132  256  377   20.249    1.394   0.810
+ntp1.m-online.net
                 212.18.1.106     2 u  162  256  377   20.417    1.503   0.712
-2a02:8106:19::3 .PPS.            1 u   16  512  253   35.003   -1.877   1.878
+bo.leptonics.com
                 78.196.167.192   3 u   78  256  377   19.919    1.079   0.354
-ntp.sebi.org    217.31.202.100   2 u   67  256  377   18.211    0.765   0.652
-dns02.xstream-labs.com
                 131.188.3.220    2 u  132  256  377   20.451    1.731   0.778
-any.time.nl     85.199.214.99    2 u   46  256  377   28.257   -2.052   3.400
*193.158.22.13   .MBGh.           1 u   25  256  377   20.964    1.188   0.341
+ptbtime1.ptb.de .PTB.            1 u  189  256  377   21.377    1.117   0.446
```

Authentisierte Manycast Konfiguration

Auf Serverseite muss das listening auf Multicastadresse eingeschaltet werden. Für ein Manycast Setup ist zwingend eine Authentifizierung vorgesehen. Im folgenden Beispiel per Shared Secret.

Beispiel 3. Konfiguration des Manycast Servers

```
$ grep -e "^interface" -e "^manycast" /etc/ntp.conf
interface ignore all
interface listen ::1
interface listen 2003:a:b45:6300:ba27:ebff:fe3b:c24a
manycastserver ff05::101 key 27         # be a server for manycast clients
```

Die Einschränkung auf *eine* IPv6 Adresse ist bei Systemen die temporäre Adressen verwenden sinnvoll, da der Rechner sonst mit **allen** Adressen eine Antwort liefert :-(

Beispiel 4. Listening IPv6 NTP Ports auf Serverseite

```
$ netstat -6 -nau | grep 123
udp6       0      0 ff05::101:123           :::*
udp6       0      0 2003:a:b45:6300:ba2:123 :::*
udp6       0      0 ::1:123                 :::*
udp6       0      0 :::123                  :::*
```

Beispiel 5. Konfiguration und Status des Manycast Client

```
$ grep "^manycast" /etc/ntp.conf
manycastclient ff05::101 key 17
```

```
$ ntpq -c peer -w
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 ff05::101       .ACST.          16 a    -   64    0    0.000    0.000   0.000
*2003:a:b45:6300:ba27:ebff:fe3b:c24a
                 124.56.169.243   3 u   21   64   77    0.838   -0.492   0.561
```

```
$ ntpq -c asso
ind assid status  conf reach auth condition  last_event cnt
===========================================================
  1 49106  c811   yes  none   yes    reject    mobilize  1
  2 49107  761a    no   yes   ok   sys.peer    sys_peer  1
$ ntpq -c "rv 49107"
associd=49107 status=761a authenb, auth, reach, sel_sys.peer, 1 event, sys_peer,
srcadr=2003:a:b45:6300:ba27:ebff:fe3b:c24a, srcport=123,
dstadr=2003:a:b45:6300:cd2e:ce3e:3d8d:89b6, dstport=123, leap=00,
stratum=3, precision=-18, rootdelay=38.376, rootdisp=10.376,
refid=124.56.169.243,
reftime=e3ff7c02.54a4a4b3  Fri, Mar 19 2021 20:37:38.330,
rec=e3ff7c33.aa7bd4e9  Fri, Mar 19 2021 20:38:27.665, reach=077,
unreach=1, hmode=3, pmode=4, hpoll=6, ppoll=6, headway=0, flash=00 ok,
keyid=17, offset=-0.492, delay=0.838, dispersion=191.150, jitter=0.561,
xleave=0.156,
filtdelay=     0.90    0.89    1.01    1.00    0.84    0.84    0.00    0.00,
filtoffset=   -0.28   -0.15   -1.20   -1.34   -0.92   -0.49    0.00    0.00,
filtdisp=      0.00    0.96    1.95    2.91    3.90    4.86 16000.0 16000.0
```

Der Client sendet regelmäßig normale Zeitanfragen an alle gelernten Server. Zusätzlich werden jedoch weiterhin Anfragen an die Multicast Adresse gesendet um nach neuen Servern Ausschau zu halten.

Beispiel 6. Paketdump des Manycast Clients

```
$ tcpdump  -n port 123
20:41:43.664669 IP6 2003:a:b45:6300:cd2e:ce3e:3d8d:89b6.123 > 2003:a:b45:6300:ba27:ebff:fe3b:c24a.123: NTPv4, Client,
length 72
20:41:43.665894 IP6 2003:a:b45:6300:ba27:ebff:fe3b:c24a.123 > 2003:a:b45:6300:cd2e:ce3e:3d8d:89b6.123: NTPv4, Server,
length 72
20:41:48.664594 IP6 2003:a:b45:6300:cd2e:ce3e:3d8d:89b6.123 > ff05::101.123: NTPv4, Client, length 72
20:41:48.666095 IP6 2003:a:b45:6300:ba27:ebff:fe3b:c24a.123 > 2003:a:b45:6300:cd2e:ce3e:3d8d:89b6.123: NTPv4, Server,
length 72
```

Public Key Authentifizierung (NTS)

NTS (**N**etwork **T**ime **S**ecurity) ist seit September 2020 als RFC8195 (https://tools.ietf.org/html/rfc8915) veröffentlicht. [9] Es erlaubt eine skalierbare, authentifizierte Zeitsynchronisation mit unterstützenden Servern.

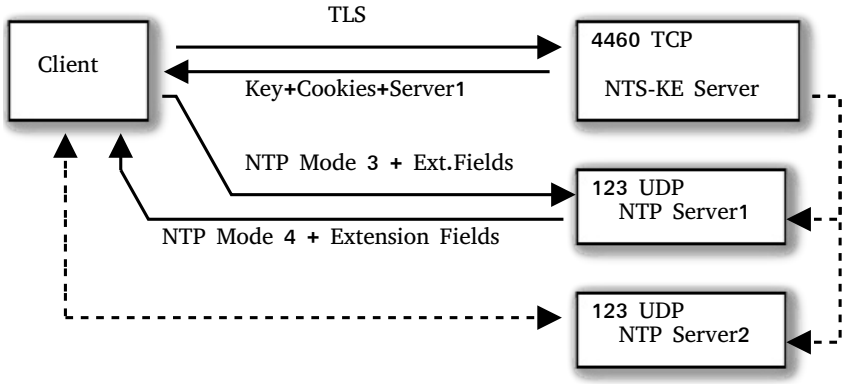
Die größte Herausforderung bei der Entwicklung einer Authentifizierung war die gute Skalierbarkeit von NTP zu erhalten. Daher sollte NTP nach wie vor UDP als Transport Protokoll verwenden und der Server keinen Status für die Clients vorhalten müssen.

Realisiert wurde dies indem NTS auf zwei Phasen aufgeteilt wurde:

1. Schlüsseltausch über TLS (NTS Key Establishment Protocol, NTS-KE)
2. NTP wie bisher über UDP allerdings mit einem Extension Header für die Authentifizierung.

In Phase eins, die idealerweise nur einmal durchlaufen wird, baut der Client eine TLS Verbindung über TCP auf dem Standardport 4460 zu dem NTP Authentifizierungsserver auf. Dies muss nicht der eigentliche NTP-Zeitserver sein! Innerhalb der Verbindung wird der Server anhand eines X.509 Zertifikat verifiziert und es findet ein Schlüsselaustausch statt. Zusätzlich übergibt der Server (bis zu) 8 Cookies, in denen alle relevanten Parameter für diese Clientverbindung enthalten sind.

Optional wird die Adresse und der Port (Standard 123) des danach zu verwendenden Zeitservers übermittelt, falls dieser von dem Authentifizierungsserver abweichen sollte.
Anschliessend wird die Verbindung beendet und der Server behält keinerlei Informationen über diesen Client.



In der Phase zwei sendet der Client NTP Mode 3 Pakete zu dem Zeitserver. Diese sind ergänzt um *Extension Fields*. In einem der Felder sendet der Client eines der gespeicherten Cookies mit. Der Server bekommt über das Cookie alle Informationen zu der vormals ausgehandelten TLS Session übermittelt, und kann damit das Antwortpaket authentifizieren.

Da ein Cookie lediglich einmal verwendet werden darf, und damit für jede Anfrage des Clients ein neues Cookie benötigt wird, sendet der Server als Zusatz zu dem normalen Antwortpaket ein weiteres verschlüsseltes Cookie mit. In der Regel wird also ein Client immer 8 Cookies für die Kommunikation mit einem Server vorrätig haben.
Der aktuelle Stand der vorhandenen Cookies wird in der Ausgabe von `ntpq` in der Spalte *t* angezeigt.

Liste der NTP Assoziationen mit einem authentifizierten Server

```
$ ntpq -c peer
     remote                refid      st t when poll reach   delay   offset   jitter
==============================================================================
 2.debian.pool.ntp.or .POOL.          16 p    -  256    0   0.0000   0.0000   0.0019
-mux.hznet.de         192.53.103.100   2 8  115  256  377  20.5639   1.5614   0.1471
-2003:a:87f:c37c::2   .DCFp.           1 u  271  512  377  26.5506  -1.6948   0.8982
+ernie.gerger-net.de  213.172.97.14    3 u   28  256  377  15.1308   0.6937   0.1568
-server1.sim720.co.uk 193.190.230.65   2 u  276  512  377  20.3368  -1.1382   0.2362
+rag.9t4.net          131.188.3.221    2 u   23  256  377  20.2224  -1.1097   0.2761
*2003:a:47f:abe4::2   .DCFp.           1 u   13  256  377  25.1666  -0.1475   0.5984
+time.cloudflare.com  10.21.8.19       3 u  154  256  377  26.9004  -0.2814   0.2202
```

NTS Client

Die clientseitige Konfiguration ist simpel: Das Schlüsselwort `nts` sorgt bei der Serverkonfiguration dafür, dass der Client den Server unter TCP 4460 anspricht. Evtl. muss noch das Verzeichnis für die Prüfung der Root-Zertifikate über das `ca file|dir` Kommando spezifiziert werden.

Beispiel 7. Konfiguration und Logfile Auszug eines NTS Clients beim Start

```
$ grep "^server" /etc/ntpsec/ntp.conf
server www.hznet.de iburst nts
```

```
$ tail -f /var/log/ntpsec/ntp.log
ntpd[22997]: NTSc: DNS lookup of www.hznet.de took 0.002 sec
ntpd[22997]: NTSc: connecting to www.hznet.de:4460 => [2a01:4f8:150:92a9::2]:4460
ntpd[22997]: NTSc: set cert host: www.hznet.de
ntpd[22997]: NTSc: Using TLSv1.3, TLS_AES_256_GCM_SHA384 (256)
ntpd[22997]: NTSc: certificate subject name: /CN=www.hznet.de
ntpd[22997]: NTSc: certificate issuer name: /C=US/O=Let's Encrypt/CN=R3
ntpd[22997]: NTSc: certificate is valid.
ntpd[22997]: NTSc: Good ALPN from www.hznet.de
ntpd[22997]: NTSc: read 880 bytes
ntpd[22997]: NTSc: Got 8 cookies, length 104, aead=15.
ntpd[22997]: NTSc: NTS-KE req to www.hznet.de took 0.101 sec, OK
ntpd[22997]: DNS: dns_check: processing www.hznet.de, 1, 21901
ntpd[22997]: DNS: Server taking: 2a01:4f8:150:92a9::2
ntpd[22997]: DNS: dns_take_status: www.hznet.de=>good, 0
```

NTS Server

Für einen NTS Server ist ebenfalls nicht viel zu konfigurieren:

```
nts cert /etc/ntpsec/fullchain.pem      ❶
nts key /etc/ntpsec/privkey.pem         ❷
nts enable mintls TLS1.3                ❸
```

❶ Die Datei mit dem Serverzertifikat und der Trust Chain bis zum Root Zertifikat.

❷ Der private Schlüssel des Server Zertifikats. Diese Datei muss gegen unberechtigten Zugriff geschützt sein.

❸ Damit wird die NTS-KEY Funktionalität eingeschaltet und der Server lauscht auf dem TCP Port 4460

```
$ netstat -nat | grep 4460
tcp        0      0 0.0.0.0:4460            0.0.0.0:*               LISTEN
tcp6       0      0 :::4460                 :::*                    LISTEN
```

Genaueres zum NTS Protokoll findet sich in [NTSsetup].

Referenzen

NTS

NTS Published as Standard
Johannes Weber
Oktober 2020
https://weberblog.net/nts-published-as-standard/ (https://weberblog.net/nts-published-as-standard/)

NTS Setup

Setting up NTS-Secured NTP with NTPsec
Johannes Weber
Dezember 2019
https://weberblog.net/setting-up-nts-secured-ntp-with-ntpsec/ (https://weberblog.net/setting-up-nts-secured-ntp-with-ntpsec/)

RFC5905

Network Time Protocol Version 4: Protocol and Algorithms Specification
David Mills, Jim Martin, Jack Burbank, William Kasch
June 2010
https://tools.ietf.org/html/rfc5905 (https://tools.ietf.org/html/rfc5905)

RFC7822

Network Time Protocol Version 4 (NTPv4) Extension Fields
Tal Mizrahi, Danny Mayer
March 2016
https://tools.ietf.org/html/rfc7822 (https://tools.ietf.org/html/rfc7822)

RFC8633

Network Time Protocol Best Current Practices
Dennis Reilly, Harlan Stenn, Dieter Sibold
July 2019
https://tools.ietf.org/html/rfc8633 (https://tools.ietf.org/html/rfc8633)

RFC8915

Network Time Security for the Network Time Protocol
Danile Franke, Dieter Sibold, Kristof Teichel, Marcus Dansarie, Ragnar Sundblad
September 2020
https://tools.ietf.org/html/rfc8915 (https://tools.ietf.org/html/rfc8915)

NTP

NTP Wikipedia
https://en.wikipedia.org/wiki/Network_Time_Protocol (https://en.wikipedia.org/wiki/Network_Time_Protocol)

NTPsec

The NTPsec Program
https://www.ntpsec.org/ (https://www.ntpsec.org/)

NTP Pool

The NTP Pool Project
https://www.ntppool.org/de/ (https://www.ntppool.org/de/)

---

1. Konstant, gleichfömig, fortschreitend
2. Bewegte Uhren gehen langsamer
3. The PTB Chronicle (https://www.ptb.de/cms/fileadmin/internet/publikationen/masstaebe/heft_12/mst12ThePtrChronicle_Normalzeit.pdf)
4. Selbst bei einer Poll Rate von 64 Sekunden und IPv6 als Transportprotokoll liegt die Bitrate eines Clients gerade mal bei 12 Bit/s
5. Vermutlich gibt es keine NTP Implementierung dafür
6. *Chrony* bringt sein eigenes Kontrollprogamm `chronyc` mit
7. Wer Schwierigkeiten bei der Umrechnung hat möge sich mit `echo "2o 8i reachwert p" | dc` behelfen
8. Es ist schon merkwürdig wie gelassen wir uns in einem fremden Netz eine IP-Adresse, einen default Router sowie einen DNS Resolver unterschieben lassen. Ob dann noch NTP dazu kommt ist fast schon nachrangig. Tatsächlich sollte gelten: „Don’t trust DHCP“
9. Siehe auch NTS Published as Standard (https://weberblog.net/nts-published-as-standard/) von Johannes Weber.