# Time synchronization in internets

**Johannes Franken**
**<jfranken@jfranken.de>**

When you try to compare some files' timestamps or the contents of logfiles, which were generated on different computers, you'll appreciate their time being synchronous. It's even better, if it matches the international atomic time.

On this page I give an overview over the tools and protocols commonly used for synchronizing times.

# Contents

# The daytime protocol

In 1983 the daytime protocol was specified as RFC867 . Today the daytime-server is built into inetd, listening on port 13 and telling the local time in plain text:

```
$ netcat hamster 13
Tue Sep  3 19:04:14 2002
```

Unfortunately it does not bind to a date format, so you have to agree upon a format before utilizing its service. Some implemetations even lack a timezone, which leads to problems when entering DST. Since the minimal unit is a second, the displayed time can differ from the real time as much as one second plus network delay.

# The time protocol

The time protocol was published as RFC868 , in order to simplify automated processings of times. Today the time-server is built into inetd, listening on port 37 and telling the number of seconds that have passed since July 1st 1900 (0am, GMT). The deviants are the same as for the the daytime protocol.

Here's how the query works in pratice:

```
$ netcat hamster 37 | od -t u1
0000000 193  31 128 156
0000004
```

The time server delivers a 4 byte word. Transforming to a number:

```
$ echo '156 + 128*256 + 31*256*256 + 193*256*256*256' | bc
3240067228
```

Skipping 70 years (due to input range limitations of gnu date)

```
$ echo 3240067228 -2208988800 | bc
1031078428
```

So what date is now, 1031078428 seconds after 1970?

```
$ date  -d "1970-01-01 0:00:1031078428"
Tue Sep  3 19:40:28 CEST 2002
```

If you've got a newer (after 2003?) implementation of GNU date, use this syntax instead:

```
$ date -d "19700101 + 1031078428 seconds"
Tue Sep  3 19:40:28 CEST 2002
```

# rdate

The most used tool for copying times over the time protocol is **`rdate`**, For example,

```
$ rdate -p hamster
Wed Sep  4 19:46:13 2002
```

shows you the time at the computer **`hamster`**. If you run it as root and without the **`-p`** parameter, It will adopt that time to your local machine. Of course, you can have cron call this regularly.

**More about:**
see rdate(8) manpage

It's bit of a problem with rdate, that it can easily spread false system times.

# netdate

Better use the **`netdate`** tool, which first compares the times of of **several** time servers and then adopts the one from the first of the largest group of hosts whose times agree with each other within a certain limit.

```
$ netdate tcp hamster gate hera
hamster -0.738          Wed Sep  4 20:35:16.000
gate -0.742             Wed Sep  4 20:35:16.000
hera -1.479             Wed Sep  4 20:35:16.000
hamster -0.481          Wed Sep  4 20:35:17.000
```

**More about:**
see netdate(8) manpage

# localtimed

Because time-servers send GMT-time, clients need to add the hours for their timezones and DST (if applicable) themselves to get the local time.

Unfortunately there still are some time-clients out (e.g. embedded in card readers), which interpret the transmitted time as local time. To work around this problem, I wrote a *localtime-demon*, which sends the server's local time (incl. timezone/DST).

Download: localtimed [1 kB]

# The network time protocol (NTP)

The NTP-protocol was invented by Professor Dr. David L. Mills in 1985. Today it is widely used in an revised version 3 ( RFC1305 ). Today, there also is version 4, which adds support for IPv6.

A NTP-client receives the time (incl. milliseconds) from several timeservers. After some inspection and the use of smart, mathematical algorithms, it can choose the best server and strip the round-trip-time. Thus, the time calculated is normally less than 50ms off the timeserver's.

## ntpdate

The **`ntpdate`** program adopts the time from the ntp-server specified at the command line.

```
$ /etc/init.d/ntp stop
Stopping NTP server: ntpd.
$ ntpdate -b gate
8 Sep 18:48:58 ntpdate[26171]: step time server 192.168.42.1 offset -0.000031 sec
```

If you do not pass the **`-b`** parameter, it will accelerate or decelerate the lokal clock until it's correct, in order to avoid time steps. This way one run of **`ntpdate`** can take hours. So, if you want to run **`ntpdate`** on system startup (e.g. from **`/etc/init.d/ntpdate`**), be sure to set the **`-b`** parameter.

BTW: ntpdate cannot change the system time, while there is a ntpd process active on the same host. If ntpdate aborts with the following message:

```
8 Sep 18:50:42 ntpdate[4671]: the NTP socket is in use, exiting
```

you most likely forgot to turn off **`ntpd`**.

**More about:**
see ntpdate(1) manpage

## (x)ntpd

The (x)ntpd [ (Experimental) Network Time Protocol Demon ] is for permanent synchronization. Due to its mathematical design features, it gives you precision in the range of nanoseconds while using minimal network bandwith. It will

1. query several ntp-servers or other devices (e.g. DCF-77 or GPS receivers) in special intervals,
2. deduct any delays caused by the network,
3. sort its time sources by reliability,
4. correct minor deviations by ascending or descending the local clock, in order to avoid time steps.
5. adjust the local clock's speed permanently, so that it can stay synchronous even without a network if the room temperature's constant.
6. provide its time to the network over ntp, selectably
   - on request (unicast),
   - as regular brodcasting in your local network segment, or
   - as multicast stream to a number of subscribers
7. offer mutual authentification to prevent damages caused by unauthorized servers or clients.

Normally you've got one timeserver in your LAN, which synchronizes its time with public Internet timeservers and provides it to all of your local clients.

Here's the configfile **`/etc/ntp.conf`** for all clients:

```
driftfile /var/lib/ntp/ntp.drift
server 192.168.42.1 burst
```

Explanations:

- The driftfile stores cognitions about the local clock's inaccuracies, so that they're still available after restarting ntpd or rebooting the machine.
- The host **`192.168.42.1`** serves as your timeserver in unicast-mode.
- It will be queried in **`burst`**-mode, which vitally speeds up the measurement of reliabilities and network delays. As a consequence thereof the network load for the first 30 seconds and after each network connection loss will be 8 times the usual, but, well, on a 100Mbit/s LAN you won't notice this.

and here's the configfile **`/etc/ntp.conf`** for the server:

```
driftfile /var/lib/ntp/ntp.drift
server ntp2.ptb.de minpoll 4 maxpoll 10
server xlink1.xlink.net minpoll 4 maxpoll 10
server willow.fernuni-hagen.de minpoll 4 maxpoll 10
server ws-lei1.win-ip.dfn.de minpoll 4 maxpoll 10
server tuminfo1.informatik.tu-muenchen.de minpoll 4 maxpoll 10
server NTP.HEH.Uni-Oldenburg.DE minpoll 4 maxpoll 10
server ntps2.gwdg.de minpoll 4 maxpoll 10
server ntp.rz.tu-harburg.de minpoll 4 maxpoll 10
server ntp.nml.csiro.au minpoll 4 maxpoll 10
server ntp0.fau.de minpoll 4 maxpoll 10
server clock.tl.fukuoka-u.ac.jp minpoll 4 maxpoll 10
server goodtime.ijs.si minpoll 4 maxpoll 10
server tick.usno.navy.mil minpoll 4 maxpoll 10
server time-nw.nist.gov minpoll 4 maxpoll 10
```

Explanations:

- the **`minpoll-`**-parameter tells ntpd the time interval to query a server at the first time and after each network failure. Possible values are:

| Value | Interval |
|---|---|
| 4 | 16 seonds |
| 5 | 32 seonds |
| 6 | 64 seonds |
| ... | ... |
| 17 | 36.4 hours |

- the **`maxpoll`**-parameter tells ntpd the time interval to query the servers when the connection's been fluent for some time.

**More about:**
see ntpd(1) manpage

# ntpq

The **`ntpq`** command supplies you with a shell for status queries about a given timeserver. If you don't provide a timeserver on the command line, it will use **`localhost`**.

```
$ ntpq
ntpq> help
Commands available:
addvars        associations   authenticate   cl             clearvars
clocklist      clockvar       cooked         cv             debug
delay          exit           help           host           hostnames
keyid          keytype        lassociations  lopeers        lpassociations
lpeers         mreadlist      mreadvar       mrl            mrv
ntpversion     opeers         passociations  passwd         peers
poll           pstatus        quit           raw            readlist
readvar        rl             rmvars         rv             showvars
timeout        version        writelist      writevar
ntpq> quit
```

The most interesting of those commands is **`peers`**. You can also reach it directly (that is, without using the shell), if you specify the **`-p`**-parameter on the command line.

```
$ ntpq -pn gate
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*192.53.103.104  .PTB.            1 u  519 1024  377   44.966    0.762   0.689
-193.141.40.1    192.53.103.104   2 u   80 1024  377   53.898   -4.443   8.575
+132.176.114.23  192.53.103.104   2 u   83 1024  377   74.690    0.338   0.024
#193.174.75.162  192.76.176.253   3 u   72 1024  377   46.868    1.128   0.142
#131.159.0.1     131.159.0.76     3 u   84 1024  377   61.069   -3.552   0.441
-134.106.148.1   131.188.3.222    2 u   77 1024  377  129.458  -11.106  11.891
-134.76.98.232   192.53.103.103   2 u   71 1024  367   46.386    1.660   1.055
 134.28.202.15   0.0.0.0          0 u    - 1024    0    0.000    0.000 4000.00
+130.155.98.1    .ATOM.           1 u   81 1024  371  348.563    1.065   0.051
 131.188.3.220   .GPS.            1 u  12h 1024    0    0.000    0.000 4000.00
-133.100.11.8    .GPS.            1 u  133 1024  357  385.072   -7.701   0.799
-193.2.4.2       .GPS.            1 u 1106 1024  376   83.967   -3.170   0.057
-192.5.41.40     .PSC.            1 u   27 1024  377  157.448   11.293  10.819
-131.107.1.10    .ACTS.           1 u   78 1024  377  280.552  -26.444  37.160
```

Explanations: The table shows one row for each server configured. The meaning of the columns is as follows:

<table>
<tr><th>Column</th><th>Explanation</th></tr>
<tr><td>The first character</td><td>tells the quality of the server:
<table>
<tr><td>*</td><td>The best source. Its time is currently taken as reference.</td></tr>
<tr><td>+<br>#<br>-<br>.</td><td>acceptable qualities, sorted descending</td></tr>
<tr><td>x</td><td>falseticker</td></tr>
<tr><td>(Leerzeichen)</td><td>no answer or depending on the local host</td></tr>
</table></td></tr>
<tr><td>remote</td><td>the IP address or (if ntpq was called without the -n parameter) the hostname of the server.</td></tr>
<tr><td>refid</td><td>the IP address or (if ntpq was called without the -n parameter) the hostname of the server.</td></tr>
<tr><td>st</td><td>the stratum, that is, how many hops the server is away from a primary time source (e.g. an atomic clock)</td></tr>
<tr><td>t</td><td>the connection type:<br>l=local,<br>u=unicast,<br>m=multicast,<br>b=broadcast</td></tr>
<tr><td>when</td><td>how many seconds ago the server was queried the last time</td></tr>
<tr><td>poll</td><td>the interval (seconds) to query the server.</td></tr>
<tr><td>reach</td><td>the reachability of the server in octal digits, from 0 (never) to 377 (always).</td></tr>
<tr><td>delay,<br>offset,<br>jitter</td><td>some statistical values (in milliseconds). The less, the better.</td></tr>
</table>

**More about:**
see ntpq(1) manpage

# ntpdc

The **ntpdc** tool is for remote configuration of a running ntp server.

**More about:**
see ntpdc(1) manpage

# Further readings

- Network Time Synchronization Project

# The simple network time protocol (SNTP)

In 1996, Dr. Mills explained, that everybody, who does not need the precision of NTP, can limit the NTP-protocol to single server requests. He called this method "Simple Network Time Protocol" (SNTP) and documented it in RFC2030 .

## net time, w32time

Windows (2000, 2003 and XP) come with two SNTP-clients:

- **`w32time`** is a system service, which automatically copies the time in given intervals.
- **`net time`** is a program to manually request the time and to configure w32time.

| Command | Effect |
|---|---|
| **`net time /setsntp:server1,server2,...`** | to select the timeservers |
| **`net time /querysntp`** | displays the selected timeservers |
| **`net stop w32time`**<br>**`net start w32time`** | stopps and starts the regular queries to the time-servers. |

# The SMB-protocol

## Under LanManager (net time)

OS/2, MS-DOS, Windows 95, 98, NT3 and NT4 come with the **NET** program, with with you can copy the time from windows-hosts, that have server services (netbios-ssn, TCP-Port 139) enabled, over the SMB-protocol.

```
C:\PROGRA~1> net time \\mausi /set /yes
```

**Attention:** Starting with Windows 2000, the **NET**-command changed from SMB- to the SNT-Protocol.

## Under Linux (nettime)

Since samba 3.0, the **net**-command is part of the samba distribution.

If you are looking for a standalone version of **NET TIME** , you will find a solution in **nettime**:

```
$ nettime //mausi
Current system time set to Wed Sep  4 21:07:18 2002
```

Here is a statically linked binary for Linux:
nettime2.bz2 [285 kB]
and here its source code as samba 2.0.7 module:
nettime.c [8 kB]

# Other ways

## The Transmission Control Protocol (TCP)

Linux places the kernel livetime (5 bytes, in hundreth of a second) in the options at the end of any tcp header.

```
$ tcpdump  host mausi & telnet mausi 22
[1] 25912
tcpdump: listening on eth0
12:16:15.513290 hamster.jfranken.de.ssh > gate.jfranken.de.2156:
P 3146288727:3146288759(32) ack 3154615717 win 8576
<nop,nop,timestamp 104273773 49599748> (DF) [tos 0x10]
^C
$ bc
scale=5
104273773/100/60/60/24
12.06872

0.06872*24
1.64928

0.64928*60
38.95680
quit
$ uptime
12:22:27 up 12 days,  1:38,  4 users,  load average: 0.00, 0.00, 0.00
```

## The Internet Control Message Protocol (ICMP)

Professor David L. Mills described the Internet Clock Service (see RFC778 ) in 1981. It would transmit the time (without the date) in milliseconds since midnight via ICMP packets of type 14 (see RFC792 ). For this purpose, W. Richard Stevens wrote the **icmptime** program, which compares the local time to a remote host's. It's available at ftp://ftp.uu.net/published/books/stevens.tcpipv1.tar.Z

## The internet Protocol (IP)

RFC781 from the year 1981 describes, that one could already do that in the IP header. As far as I know, there hasn't been a practical implementation of this service.