Leap Second Smearing with NTP

Martin Burnicki

Meinberg Funkuhren
Bad Pyrmont
Germany

martin.burnicki@meinberg.de

2016-11-17

Slightly revised 2023-11-03

Table of Contents

Please read this document carefully if you are going to set up leap second smearing for your NTP servers!

1 Introduction

The NTP software protocol and its reference implementation, *ntpd*, have originally been designed to distribute UTC time as accurately as possible across a network.

Unfortunately, leap seconds are scheduled to be inserted into or deleted from the UTC time scale in irregular intervals to keep the UTC time scale synchronized with the Earth rotation. Deletions haven't yet happened; all past leap seconds were insertions.

Whenever a leap second is to be handled, *ntpd* usually just passes the leap second announcement down to the OS kernel (if the OS supports this) and then **the kernel handles the leap second** automatically **as implemented in the kernel**. NTP servers also pass a leap second warning flag down to their clients via the normal NTP packet exchange, so clients also become aware of an approaching leap second, and can handle the leap second appropriately.

2 The Problem on Unix-like Systems

In most Unix-like systems, when a leap second needs to be inserted, the operating system kernel simply ***steps the system time back by one second*** at the beginning of the leap second, so the last second of the UTC day is repeated and thus ***duplicate timestamps*** can occur.

Unfortunately, the standard function calls used by applications to read the system time don't provide a way to distinguish the second occurrence of the duplicate second from the first one. There are lots of applications which get confused in general if the system time is stepped back, so they also get confused when a leap second is inserted. Thus, many users have been looking for ways to avoid this, and tried to introduce workarounds which may work properly, or not.

So even though these Unix kernels normally ***can*** handle leap seconds, the way they do this is not well-suited for applications.

3 NTP Client for Windows Contains a Workaround

Apart from very recent Windows versions with particular configuration, the Windows system time knows nothing about leap seconds, so since many years the Windows port of *ntpd* provides a workaround where **the Windows system time is slewed by *ntpd*** to compensate the leap second.

So there is no requirement to use a smearing NTP server for Windows clients, but of course it also doesn't hurt if the NTP server sends a smeared time to account for the leap second, as described below.

4 The Leap Smear Approach

Due to the reasons mentioned above, some support for leap smearing has been implemented in *ntpd*. This means, an NTP server adds a certain “smear offset” to the real UTC time which is put into reply packets sent to clients. This “smear offset” starts at 0 and increases over a predefined time interval so that after that interval the leap second offset is compensated. The smear interval should be long enough, e.g. at least several hours, so that NTP clients can easily follow the clock drift caused by the smeared time, and don't lose synchronization.

With this approach, the time an NTP server sends to its clients still matches UTC *before* the leap second, up to the beginning of the smear interval, and again corresponds to UTC *after* the leap second, at the end of the smear interval.

Of course, clients which receive the “smeared” time from an NTP server don't have to (and even ***must not***) care about the leap second anymore. Smearing is just transparent to the clients, and the clients don't even notice there's a leap second.

5 Pros and Cons of the Smearing Approach

The disadvantages of this approach are:

- During the smear interval, the time provided by smearing NTP servers differs significantly from UTC, and thus from the time provided by normal, non-smearing NTP servers. The difference can be up to 1 second, depending on the smear algorithm.

- Therefore, the smeared time received by clients also differs from true UTC, which may also have legal consequences for applications requiring correct legal time which is based on UTC or the local time derived from true UTC, or has to be traceable to UTC, e.g. billing for mobile phone calls which is aligned with certain second boundaries.

However, for applications where it's only important that all computers have the ***same*** time and a temporary offset of up to 1 s to UTC is acceptable, a better approach may be to slew the time in a well defined way, over a certain interval, which is called “smearing the leap second”.

6 The Motivation to Implement Leap Smearing

Here is some historical background for *ntpd*, related to smearing/slewing time.

Up to *ntpd* 4.2.4, if kernel support for leap seconds was not available, or was not enabled, *ntpd* didn't care about the leap second at all. So if *ntpd* was run with -x and thus kernel support wasn't used, *ntpd* saw a sudden 1 s offset after the leap second and normally would have stepped the time by -1 s a few minutes later. However, due to -x *ntpd* did ***not*** step the time but started slewing over a very long period. This could be considered a bug, but certainly this was only an accidental behavior.

However, as we learned in the discussion in NTP bug 2745, this behavior was very much appreciated since indeed the client's system time was never stepped back, and even though the start of the slewing was somewhat undefined and depending on the client's poll interval, the system time was off by 1 second for several minutes before slewing even started.

In *ntpd* 4.2.6 some code was added which lets *ntpd* step the time at UTC midnight to insert a leap second, if kernel support was not used. Unfortunately, this also happened if *ntpd* was started with -x, so the folks who expected that the time was ***never*** stepped when *ntpd* was run with -x found this wasn't true anymore, and again from the discussion in NTP bug 2745 we learn that there were even some folks who patched *ntpd* 4.2.6 to get the 4.2.4 behavior back.

In 4.2.8 the leap second code was rewritten and some enhancements were introduced, but the resulting code still showed the behavior of 4.2.6, i.e. *ntpd* with -x would still step the time. This has been fixed in the current *ntpd* stable code, but this fix is only available with a certain patch level of *ntpd*, e.g. in 4.2.8p8 or later.

So a possible solution for users who were looking for a way to come over the leap second without the time being stepped could have been to check the version of *ntpd* installed on each server in the company. If it's still 4.2.4, be sure to start the client *ntpd* with -x. If it's 4.2.6 or 4.2.8, it won't work anyway except if you had a patched *ntpd* version instead of the original version. So you'd need to upgrade to the current -stable code to be able to run *ntpd* with -x and get the desired result, so you'd still have the requirement to check/update/configure every single machine in the company.

Google's original leap smear approach was a very efficient solution for this. You just have to take care that the company's NTP servers support leap smearing and configure those few servers accordingly. If the smear interval is long enough so that NTP clients can follow the smeared time, it doesn't matter at all which version of *ntpd* is installed on a client machine. It just works, and it even works around clients' kernel bugs related to the leap second.

Since all clients follow the same smeared time, the time ***difference*** between the clients during the smear interval is as small as possible, compared to the -x approach.

The current leap second code in *ntpd* determines the point in system time when the leap second is to be inserted, and given a particular smear interval, it's easy to determine the start point of the smearing. So smearing is finished when the leap second ends, and the client's UTC time is correct again when the next UTC minute / hour/ day after the leap second begins.

The maximum error doesn't exceed what you'd get with the old smearing caused by -x in *ntpd* 4.2.4, so if users could accept the old behavior, they would even accept the smearing at the server side.

In order to affect the local timekeeping as little as possible, the leap smear support currently implemented in *ntpd* does not affect the internal system time at all. Only the timestamps in outgoing reply packets to clients are modified by the smear offset, so this makes sure the basic functionality of *ntpd* is not accidentally broken. Also peer packets exchanged with other NTP servers are based on the real UTC system time, as usual.

The leap smear implementation has been submitted to the official NTP code repository, and the changes can be tracked via NTP bug 2855.

7 Using ntpd’s Leap Second Smearing

- Leap Second Smearing must ***not*** be used for public servers, e.g. servers provided by metrology institutes, or servers participating in the NTP pool project. There would be a high risk that NTP clients get the time from a mixture of smearing and non-smearing NTP servers which could result in undefined client behavior. Instead, leap second smearing should only be configured on time servers providing dedicated clients with time, if all those clients can accept smeared time.

- Leap Second Smearing is only conditionally compiled in, i.e. if the *./configure* script from the NTP source code package is run with the *--enable-leap-smear* parameter before the executables are built.

- Even if *ntpd* has been compiled with leap smearing support, leap smearing is only done if explicitly configured, i.e. if a *leapsmearinterval* is specified in the ntpd’s configuration file.

- The leap smear interval should be several hours, up to 1 day (86400 s). If the interval is too short, the applied smear offset increases too fast over time, so NTP clients might not accept this. 86400 s should be a good choice.

- If several NTP servers should be set up for leap smearing, the ***same*** smear interval should be configured on each server, and it should me made sure that all servers use the same algorithm to compute the smear offset.

- Smearing NTP servers don't send a leap second warning flag to its clients. Since the leap second is applied gradually, the clients don't even notice there's a leap second being inserted, and thus no log message or similar related to the leap second will be visible on the clients.

- Since clients don't (and *must not*) become aware of the leap second at all, clients getting the time from a smearing NTP server ***must not be configured to use a leap second file***. If they had a leap second file, they would handle two leap seconds at the same time: the smeared one from the server, plus another one inserted by themselves due to the information from the leap second file. As a result ***there would be a time step***, and thus a wrong 1 s offset which would be corrected a few minutes later by another time step.

- Clients must ***not*** be configured to poll both smearing and non-smearing NTP servers at the same time. During the smear interval they would get different times from different servers and wouldn't know which server(s) to accept.

8 Setting Up a Smearing NTP Server

If an NTP server running *ntpd* should do the leap smearing, the leap smear interval (in seconds) needs to be specified in the NTP configuration file *ntp.conf*, e.g.:

```
leapsmearinterval 86400
```

Please keep in mind the leap smear interval should be as large as possible, at least several hours, since otherwise clients may not be able to follow the drift caused by the smeared time.

When *ntpd* starts and a smear interval has been specified, a log message is generated accordingly, e.g.:

```
ntpd[31120]: config: leap smear interval 86400 s
```

While *ntpd* is running with a leap smear interval specified, the command *ntpq -c rv* reports the smear status, e.g.:

```
#  ntpq -c rv
associd=0 status=4419 leap_add_sec, sync_uhf_radio, 1 event, leap_armed,
version="ntpd 4.2.8p3-RC1@1.3349-o Mon Jun 22 14:24:09 UTC 2015 (26)",
processor="i586", system="Linux/3.7.1", leap=01, stratum=1,
precision=-18, rootdelay=0.000, rootdisp=1.075, refid=MRS,
reftime=d93dab96.09666671  Tue, Jun 30 2015 23:58:14.036,
clock=d93dab9b.3386a8d5  Tue, Jun 30 2015 23:58:19.201, peer=2335, tc=3,
mintc=3, offset=-0.097015, frequency=44.627, sys_jitter=0.003815,
clk_jitter=0.451, clk_wander=0.035, tai=35, leapsec=201507010000,
expire=201512280000, leapsmearinterval=86400, leapsmearoffset=-932.087
```

In the example above *leapsmearinterval* reports the configured leap smear interval all the time, while the *leapsmearoffset* value is 0 outside the smear interval and increases from 0 to -1000 ms over the interval, while smearing is in progress. So this can be used to monitor if and how the time sent to clients is smeared.

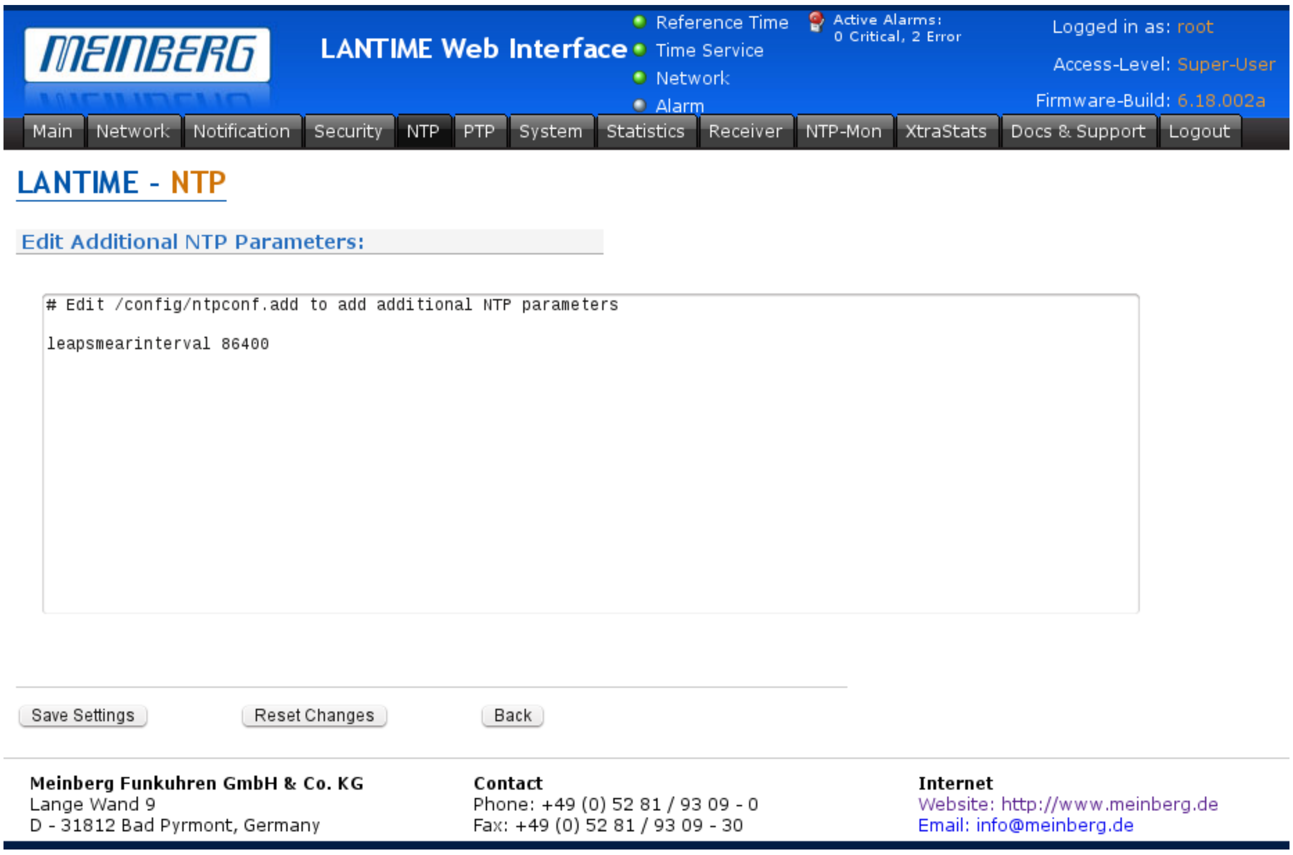
9 Leap Smearing With a Meinberg LANTIME NTP Server

If you are operating a Meinberg LANTIME NTP server, you need to be sure the installed firmware version and version of *ntpd* are current enough to support leap smearing. LANTIME firmware versions v6.18.002a and all newer/later versions support this.

If your LANTIME runs an older firmware, please contact Meinberg support for an update.

If the firmware supports leap second smearing, you can configure the smear interval via the LANTIME's web interface. Just go the “NTP” tab, select “NTP configuration”, and then click on “Edit Additional NTP Parameters”.

This opens a text box where you can enter additional lines to be appended to the *ntp.conf* file, so you can add the “leapsmearinterval” directive as shown below:

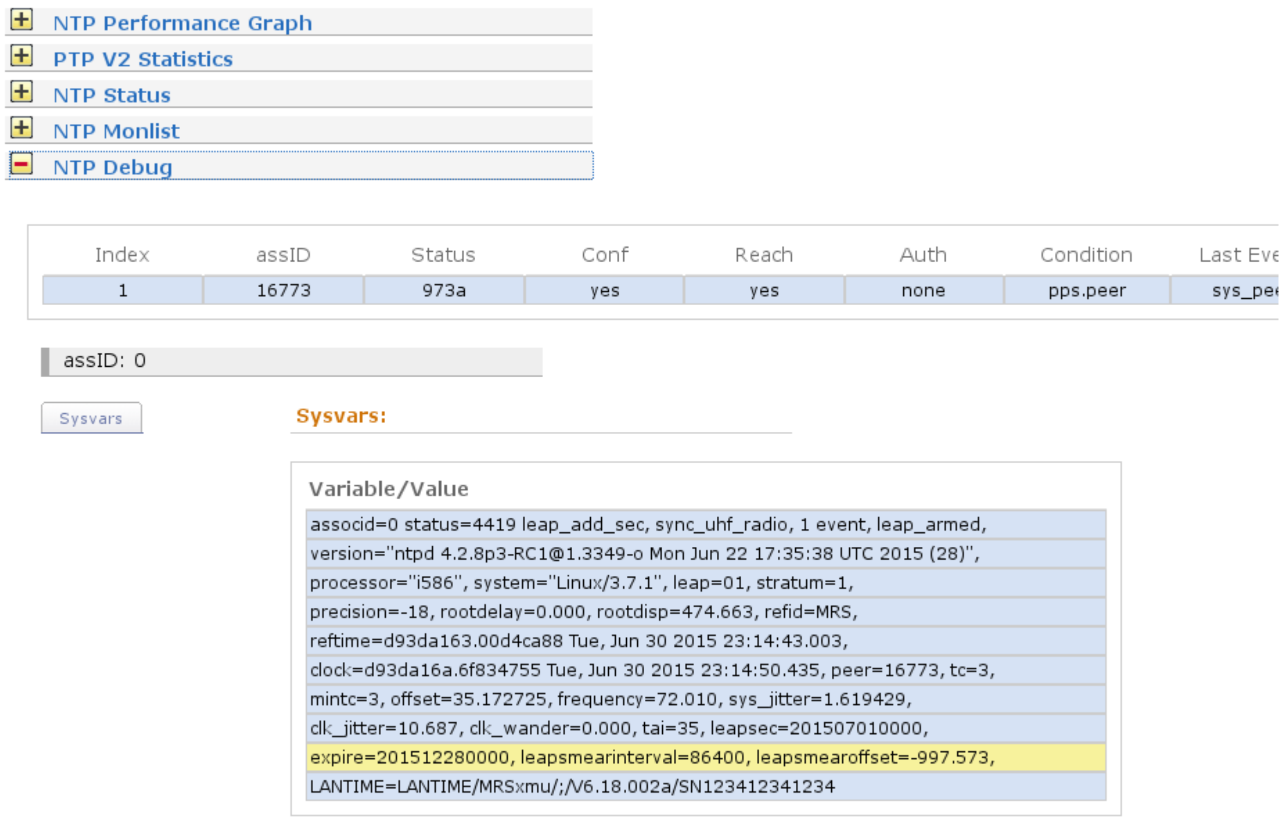


```
# Edit /config/ntpconf.add to add additional NTP parameters

leapsmearinterval 86400
```

Once the configuration has been saved, the NTP daemon is restarted.

While the NTP service is running on the LANTIME, the result of the *ntpq -c rv* command can be reviewed on the *Statistics* tab in the web interface, if you expand the entry *NTP Debug:*



Index	assID	Status	Conf	Reach	Auth	Condition	Last Eve
1	16773	973a	yes	yes	none	pps.peer	sys_pe

assID: 0

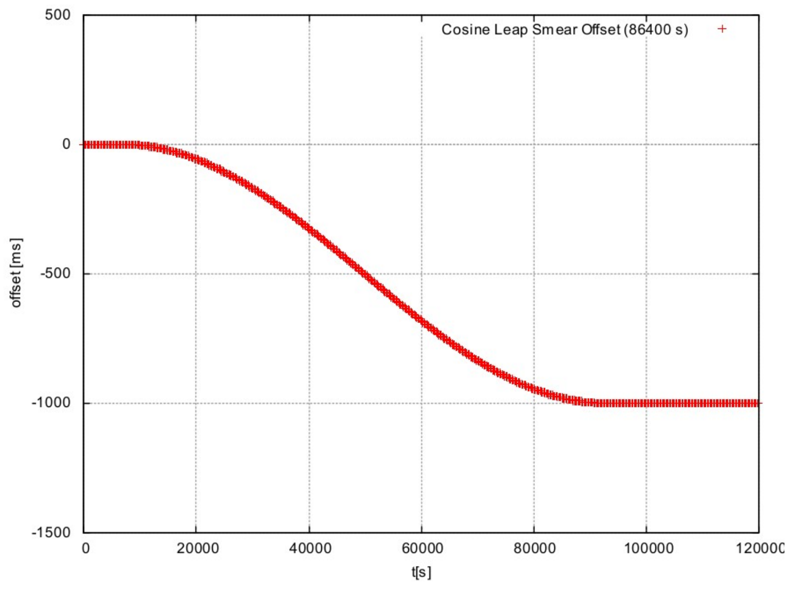
Sysvars:

```
Variable/Value
associd=0 status=4419 leap_add_sec, sync_uhf_radio, 1 event, leap_armed,
version="ntpd 4.2.8p3-RC1@1.3349-o Mon Jun 22 17:35:38 UTC 2015 (28)",
processor="i586", system="Linux/3.7.1", leap=01, stratum=1,
precision=-18, rootdelay=0.000, rootdisp=474.663, refid=MRS,
reftime=d93da163.00d4ca88 Tue, Jun 30 2015 23:14:43.003,
clock=d93da16a.6f834755 Tue, Jun 30 2015 23:14:50.435, peer=16773, tc=3,
mintc=3, offset=35.172725, frequency=72.010, sys_jitter=1.619429,
clk_jitter=10.687, clk_wander=0.000, tai=35, leapsec=201507010000,
expire=201512280000, leapsmearinterval=86400, leapsmearoffset=-997.573,
LANTIME=LANTIME/MRSxmu/;/V6.18.002a/SN123412341234
```

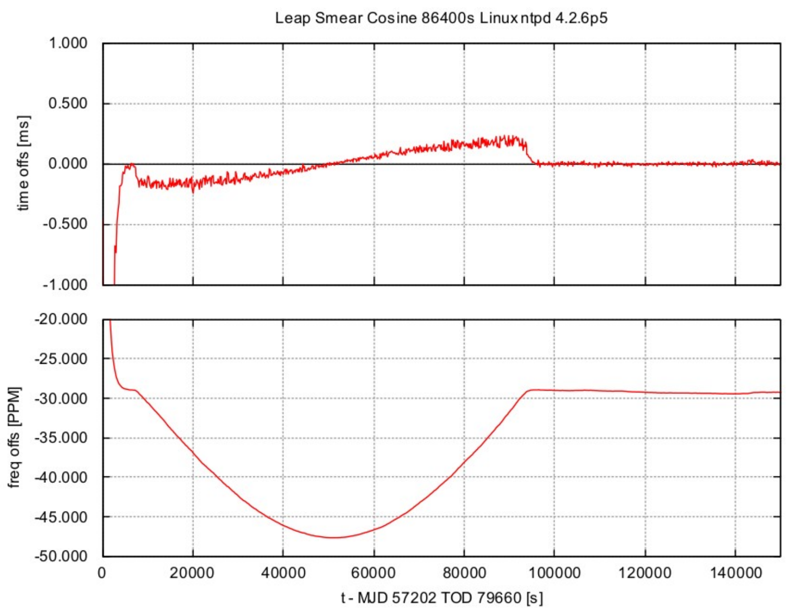
In the example above, you can see that the leap smear interval is one day (86400 s), the current time is 45 minutes before the leap second event, and the current smear offset sent to the clients is -997.573 ms, i.e. most of the leap second has already been smeared.

10 Leap Smear Test Results

Here is the result of a test program which queries the time from a smearing NTP server to show how the smear offset increases over a 86400 s interval:

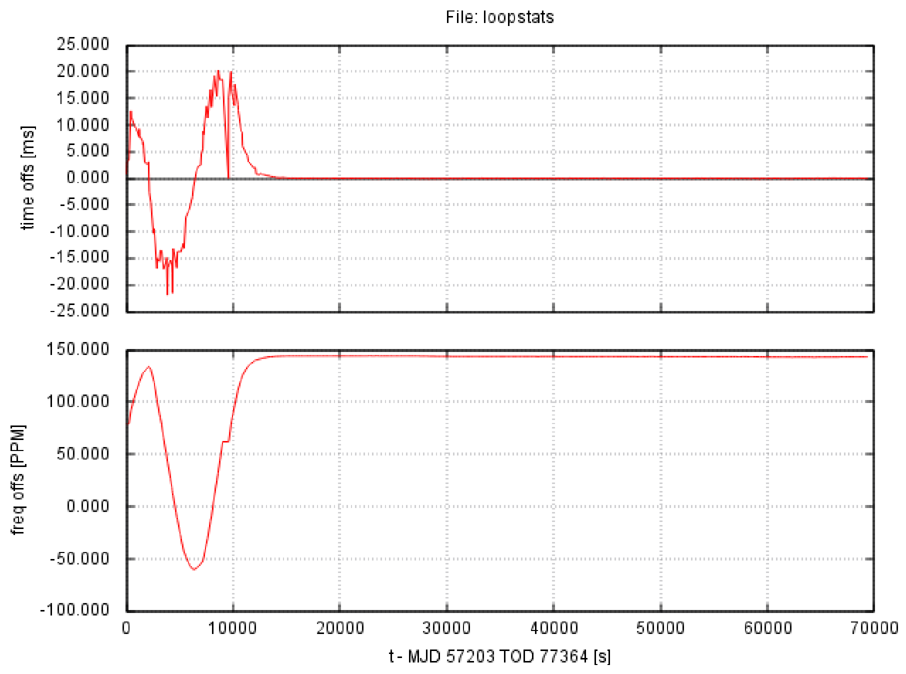


And here are is the *loopstats* plot of a client *ntpd* synchronizing to that server:



As can be seen above, to the client the smearing looks like a clock drift which changes over time. However, the client follows the server time with only 250 us maximum offset caused by the varying drift.

In the next graph the smear interval was only 2 hours:



It can be seen that the time offset swing was much larger, up to 20 ms. The client was unable to follow the server's smear anyway, and thus stepped the time to compensate the offset.

So a much better approach is to use a smear interval as long as possible, up to one day (86400 s).

More test results also from different smearing approaches which are not natively supported by *ntpd* can be found in an associated page of the **Meinberg Knowledge Base**:

NTP Leap Smearing Test Results
https://kb.meinbergglobal.com/kb/time_sync/ntp/leap_second_smearing/ntp_leap_smearing_test_results