(Preprint) AAS 13-504

TECHNICAL ASPECTS OF LEAP SECOND PROPAGATION AND EVALUATION

Martin Burnicki*

Leap seconds are scheduled by the International Earth Rotation Service (IERS) whenever the difference between true Earth rotation and the UTC time scale reaches a certain limit. Whenever a leap second has been scheduled by the IERS, a warning must be disseminated to time keeping devices so that clocks become aware of the scheduled leap second early enough to be able to handle the leap second properly. There are different ways to propagate leap second warnings, and different ways to handle leap seconds, and thus there are a number of pitfalls causing unexpected results and potential malfunctioning.

INTRODUCTION

Leap second warnings can be propagated by different timing signals, protocols, *etc*. For example, the GPS satellites transmit a specific point in time when a leap second is to be inserted or deleted, but other timing signals may just provide a leap second warning flag which is set during a certain interval before the leap event, where the warning interval depends on the specification of the protocol.

Also, there are different implementations as to how leap seconds are handled, which especially affect the sequence of timestamps across the leap second event. The clock can be stepped at the beginning or end of the leap second, can be slowed down or even stopped during a leap second insertion, or time readings can be slewed across a leap second. This makes it difficult to compare time stamps which have been taken from different clocks or different systems during a leap second.

Last, but not least, there are implementations of time keeping software which don't always work correctly, *e.g.*, invalid leap second warnings are generated, leap seconds are not handled at all, or severe bugs occur due to side effects of the leap second handling.

LEAP SECOND HANDLING

There is no exact specification for how leap seconds are to be handled by clocks providing time with resolution below 1 second. If a leap second is deleted then one second can simply be skipped, *i.e.*, time could just be stepped forward by 1 second. If a leap second is inserted, which is usually the case, a rule is required for how time should be incremented during the inserted leap second.

* Senior Software Engineer, Meinberg Funkuhren GmbH & Co. KG, Lange Wand 9, D-31812 Bad Pyrmont, Germany.

Enumerating Seconds over a Leap Second

Most articles about leap seconds only refer to the way leap seconds are counted for human readable date and time. If a leap second is deleted then the 59th second of the last minute before UTC midnight is just skipped, and an inserted leap second is simply counted as second "60" before midnight. This behavior is perfectly acceptable for wall clocks and displays with only second resolution:

Table 1. Enumeration of Seconds Over a Leap Second

2012-06-30 23:59:57
2012-06-30 23:59:58
2012-06-30 23:59:59
2012-06-30 23:59:60←leap second
2012-07-01 00:00:00
2012-07-01 00:00:01
2012-07-01 00:00:02

As can be seen, if the times in Table 1 are normalized, then 60 seconds yield 1 minute, which is carried over to the minutes field. Thus the minute count reaches 60 and rolls over to 0 with a carry over to the hours and so on, so the resulting normalized time is exactly the same as the first second of the next day. This is exactly what happens with computer clocks, *etc*., which simply count the number of seconds since a given epoch. If they run on UTC as proposed by the POSIX standard, the second count is the same at the beginning and end of the inserted leap second.

So how should we distinguish between both, and how should we enumerate time stamps taken in short intervals during an inserted leap second? There are different approaches to doing this, and each approach has its own advantages and disadvantages. Usually the exact behavior is implemented in the operating system kernel.

Step Time Back at the End of the Leap Second

If time is simply stepped back at the end of an inserted leap second as shown in Figure 1, then time is not monotonic, and thus duplicate time stamps occur *after* the leap second, *i.e*., at the *beginning* of the next UTC day. As a result, there can be *later* time stamps assigned to events which occurred *earlier*, which can heavily mess up applications using time stamps to order the sequence of events or transactions.

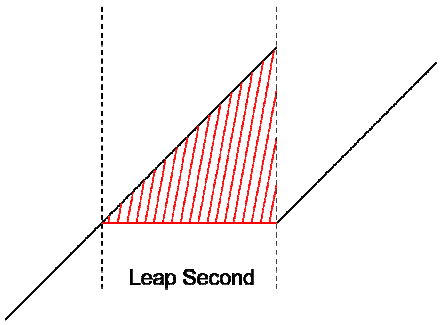

Figure 1. Step Time at the End of the Leap Second.

Step Time Back at the Beginning of the Leap Second

If time is simply stepped back at the beginning of an inserted leap second as shown in Figure 2, then time is also not monotonic. The difference from the previous case is that duplicate time stamps occur *during* the leap second, *i.e.*, at the *end* of the UTC day. Similarly there can be *later* time stamps assigned to events which occurred *earlier*, which can cause the same confusion as the previous case.

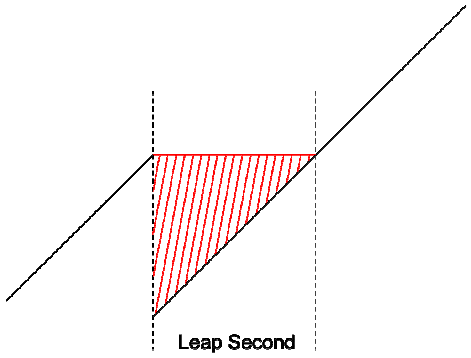

Figure 2. Step Time at the Beginning of the Leap Second.

Stop Time Counting for One Second

If time display is just put on hold for one second as shown in Figure 3, then time stamps are all the same during the inserted leap second. This means that time does not increase strictly monotonically, and time stamps can't be used to order events.

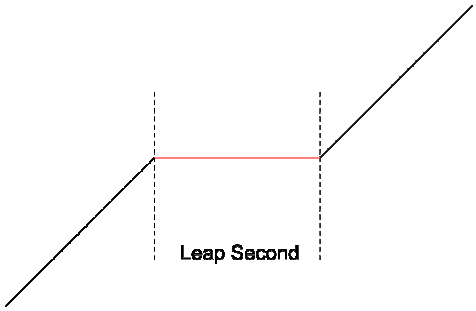

Figure 3. Stop Time During Leap Second.

Increment Time Least-Significant Bit (LSB) on Request

A modified approach which guarantees strictly monotonic time stamps has been proposed by David L. Mills, the inventor of the Network Time Protocol (NTP), who suggested stopping the clock during an inserted leap second, but incrementing the fractions of time stamps by the smallest possible time increment whenever the time is read by an application.[1] This is shown in Figure 4, with an enlarged scale to demonstrate the behavior.

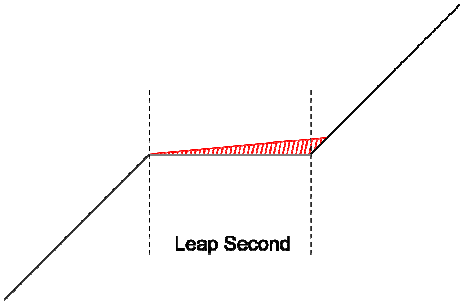

Figure 4. Small Time Increments During Leap Second.

Slewing Time Half Speed over Two Seconds

Some operating systems are not aware of leap seconds and thus are not prepared to handle them. In such case it may be possible to slew the system time over the leap second. For example, the Windows version of the NTP reference implementation slows the system clock down to half the nominal speed for 2 seconds, as shown in Figure 5, so after 2 seconds the system time is again aligned to UTC. This is not optimal, but at least it is a workaround which yields less error than no intervention, and one second after the leap second the system time is correct again.

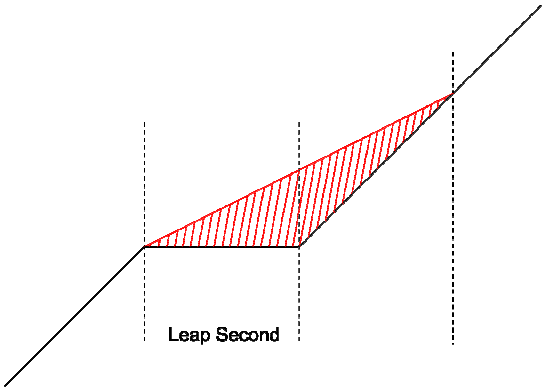

Figure 5. Slew Time over 2 Seconds.

UTC-SLS Proposal

A paper published by Markus G. Kuhn suggests applying a leap second insertion over the last 1000 seconds of a UTC day that ends with a leap second.[*] This approach, called *UTC with Smeared Leap Seconds* (UTC-SLS), should be implemented in the OS kernel so the OS can handle the leap second event by itself if it receives a leap second warning early enough.

This sounds like a good solution to avoid problems over the leap second insertion interval, but the disadvantage is that time differs from true UTC during the smear interval. There can also be potential degradations in accuracy in time synchronization networks if some nodes use UTC-SLS while others use standard UTC.

[*] Kuhn, Markus G., "UTC with Smoothed Leap Seconds." URL: http://www.cl.cam.ac.uk/~mgk25/time/utc-sls/

Google Leap Smear

Due to problems Google observed with the leap second in 2008, Google decided to implement an own solution called *Google Leap Smear*.[*] This is basically similar to the UTC-SLS approach, but needed to be implemented on Google's private NTP servers only.

Those NTP servers send a "modulated" time to their clients by gradually adding a couple of milliseconds to every update, varying over a time window before the moment when the leap second actually happens. For the clients, this looks like a phase adjustment similar to the time corrections required due to temperature drift, but at the end of an inserted leap second they have already gained the extra second.

The main advantage of this approach is that it only needs to be implemented on the NTP servers, which saves a lot of work if there is a huge number of clients, but the disadvantage is that time drifts away from true UTC during a longer time interval before the leap second.

In a closed network where all machines get their time from private servers, all clients should keep the same time if they run standard NTP, and if all NTP servers use Leap Smear. The only limitation is that time is off UTC during the smear interval, which may be acceptable if the applications running on the client machines can tolerate this.

PROTOCOLS AND SYSTEMS DEALING WITH LEAP SECONDS

After a leap second has been scheduled by the IERS, a leap second warning has to be disseminated and propagated to all devices which are expected to follow true UTC. There are different techniques to propagate such information: via radio signals, computer network, and other communication channels. Each communication channel can support one or more protocols, and it also depends on the protocol and data format used, if and how a leap second warning can be propagated.

GPS Satellite System

GPS time is a linear time scale, and thus differs from UTC. It has a fixed offset to TAI. Thus the navigational message sent by the satellites includes UTC correction parameters, which in turn include information on an upcoming leap second. Since GPS time is counted as week numbers and second-of-week, leap second information contains a week number, the number of the day of that week at the end of which a leap second is scheduled, and the UTC offset before and after the specified leap second event time. This data format is very flexible since it:

- specifies an exact point in time for the leap second event,
- can handle both positive and negative leap seconds, and
- can even specify the handling of several leap seconds at the same time.

The data set sent by the satellites is usually updated a few days after a leap second is scheduled by the IERS, so GPS receivers can become aware of this event about 6 months before it actually happens, with the exact time when it is going to happen.

A limitation is that the full GPS week number counts from 0 to 1023 before it rolls over to 0 again, so the GPS epoch needs to be known to compute the full calendar date for a leap second event. Also, the week number included in the UTC parameter set is a truncated 8 bit number of

[*] Google Blog, "Time, technology and leaping seconds." URL: http://googleblog.blogspot.de/2011/09/time-technology-and-leaping-seconds.html

the 1024 week number only. This is enough to specify a week within a ±127 week interval relative to the current GPS week, but it is ambiguous if no leap second is scheduled for more than 127 weeks. This can cause faulty leap second warnings be output by GPS receivers if this ambiguity is not taken into account by the GPS receiver firmware.

An interesting question is how a GPS receiver can propagate its knowledge of a pending leap second to the clock which needs to be alerted. This obviously depends on the communication mechanism and protocol.

German Long Wave Transmitter DCF-77

The long wave transmitter DCF-77 is located in Germany and disseminates the legal time for Germany. In can also be received, and is actually widely used in the European countries around Germany. The long wave signal transports up to 59 data bits per minute including the current local date and time, and one of these bits is an announcement bit for a positive leap second.

The problem is that the leap second warning bit is only transmitted 59 minutes before the leap second event actually occurs. This interval may be sufficient for radio clocks which evaluate the time code directly, but this interval is too short if a DCF-77 radio clock is, *e.g.*, used as a reference clock for an NTP daemon, where clients poll the server in intervals up to 1024 seconds, or even longer.

The AM signal used by the long wave transmitter is susceptible to electric noise, so a potential problem is that the leap second is missed and reception is not possible during the 1 hour interval where the leap second warning is transmitted.

NIST Time Services

The U.S. National Institute of Standards and Technology (NIST) provides time services via telephone modem services (ACTS) and long wave transmitters (WWVB, WWVH) which also transmit a leap second warning flag. I have not found a reference telling how long before the leap second event the announcement flag is set, but some references say the flags announce a leap second for the end of the month, which lets me assume the warning is transmitted for an interval longer than just a few minutes or hours.

IRIG Time Codes

The first IRIG time codes were introduced around 1960, *i.e.*, long before leap seconds have been invented. Thus standard IRIG codes, like the popular B-122 code, don't support transportation of a leap second warning. Even if the transmitted time code counts seconds up to 60 when a leap second is inserted, this may be sufficient for immediate evaluation, *e.g.*, in displays, but when the “60” is detected by a time code receiver it is much too late for applications like NTP using such receiver as reference clock (or, *refclock*) to propagate a leap second properly to the time consumers.[*]

IEEE Time Codes

IRIG time codes specify a set of reserved bits which can be used in an application-specific way. The IEEE 1344 standard from 1995 determines how these reserved bits can be assigned to transport supplemental information in a well-known way. This supplemental information contains a time qualifier code, a year number, UTC offset, as well as daylight-saving time (DST) and leap second status.

[*] A *reference clock* is a special device that supplies the current time very accurately.

IEEE standard C37.118 from 2005 is a revised version of IEEE 1344 which is basically identical, except for the UTC offset which is transported with reversed sign. Both codes provide the same leap second information: a leap second pending flag, and another bit telling if the pending leap second is negative, or positive.

Unfortunately the IEEE standards also determine that the leap indicator flags must not be set earlier than 59 seconds before the leap second event, which is much too short, *e.g.*, for usage with NTP.

Serial Time Strings

Serial Time String can be used to transfer time and status information, *e.g.*, from a radio clock or GPS receiver to a time synchronization software, *e.g.*, an NTP daemon. Very few time string formats provide a field for a leap second announcement, but many other formats including the popular NMEA sentences used with GPS modules don't support this. So if an NTP daemon gets the time from a GPS receiver, *e.g.*, via the NMEA protocol, the NTP daemon does not receive a leap second announcement even though the GPS receiver knows about the upcoming leap second about 6 months in advance.

NIST Leap Second File

The U.S. NIST makes a leap second file available on their web site.[*] This is just a text file containing a list of historic and current leap seconds, *e.g.*:

```
2272060800    10    # 1 Jan 1972
2287785600    11    # 1 Jul 1972
                …
3124137600    32    # 1 Jan 1999
3345062400    33    # 1 Jan 2006
3439756800    34    # 1 Jan 2009
3550089600    35    # 1 Jul 2012
```

In addition, there are some comments and a file expiration date, so applications using this file can determine when the file is outdated and should be upgraded. This file can optionally be evaluated by the standard NTP daemon to determine when a leap second is pending even if there is no reference clock providing a leap second warning, or if there's a reference clock which provides no leap second warning at all, or if the leap second warning appears too late. The advantage of this file is that it also provides the total number of leap seconds, so it can be used to convert TAI to UTC and vice versa.

Olson TZ Data Base

The Olson tz database is a widely used set of tables to determine the local time offset to UTC for a given date and time in a given country or time zone. The database also keeps track of changes over the years, so it can also be used for historic dates and times.

Besides providing information about the local time zones it also provides a file with a list of leap seconds which is used by the associate library functions to convert system time to civil time if the so called “right” zone tables have been configured for local time conversion. However, as far as I know this is not widely used, yet, at least not in default installations of Linux, FreeBSD, and Solaris.

[*] ftp://time.nist.gov/pub/, ftp://tycho.usno.navy.mil/pub/ntp/

PTP/IEEE1588 Network Protocol

The Precision Time Protocol (PTP) supports 2 different flags to announce a positive or negative leap second. By default the protocol itself works with TAI, so the UTC to TAI offset is also included. The leap second announcement flags are set about 12 hours before UTC midnight, which is usually sufficient for time synchronization concepts.

A potential source for problems is that there are two independent flags for both types of leap seconds. In a faulty implementation a PTP grandmaster could erroneously set both flags at the same time, and it depends on the client implementation how it acts if both announcements are received at the same time: handle the first or second flag, or ignore the invalid state, provided that such a check has been implemented.

The open source implementation of ptpd originally didn't evaluate leap second announcements at all. This was only detected and fixed a few weeks before the last leap second event at the end of June, 2012. Currently the open source ptpd runs only on Linux and FreeBSD, so it just has to pass a leap second announcement down to the kernel who handles the leap second.

Network Time Protocol (NTP)

The Network Time Protocol (NTP) also supports announcement of a negative or positive leap second. However, this is coded in a way that only one type of leap second can be announced at the same time, which avoids potential errors as with PTP.

For the reference implementation, the exact behavior of leap second handling varies with the software version, *e.g.*, the time interval at which a leap second warning is accepted prior to the event, or the conditions under which a leap second warning is accepted if there are several potential sources for a leap second announcement (like one or more reference clocks, one or more upstream NTP servers, and optionally the NIST leap second file).

Previous NTP versions accepted a leap second announcement even if only one of several upstream NTP servers sent it. This has led to problems in the past if, *e.g.*, a single upstream server provided a faulty leap second warning. To avoid this potential problem, the current stable implementation of NTP accepts leap seconds announcements from upstream servers only if a majority of the configured upstream servers sends the announcement.

If the NTP program runs on a Unix version which supports kernel discipline, then the NTP daemon program (ntpd) just passes a leap second down to the kernel which handles the leap second. On systems which don't support kernel discipline, leap seconds were not handled at all by earlier NTP versions. Special support for the Windows port of NTP was introduced in version 4.2.4, but only the current release version 4.2.6 steps the time back if a leap second needs to be inserted on other systems which don't support the kernel discipline.

NTP can evaluate the NIST leap second file, but the way to configure this has changed between v4.2.4 and v4.2.6, so care must be taken if NTP is upgraded across these versions and the configuration file is left unchanged.

Unix Kernels

Unix kernels often implement the kernel clock implementation proposed by Dave Mills, or variations of it, and usually can also handle a leap second if they receive an announcement early enough. The associated programming interface can be used by time synchronization programs like ntpd or ptpd to discipline the system time, to pass leap second announcements to the kernel, *etc*.

KNOWN BUGS

There are some common bugs which caused malfunctioning on related systems, or even let the affected systems stop immediately, so the services running on those systems became unexpectedly unavailable.

Linux Kernel Deadlock

While holding a kernel lock, the Linux kernel's leap second handling code tried to log a message about the leap second insertion, which in turn tried to acquire the same kernel lock. On busy kernels this could lead to a deadlock, so the machine completely stopped working when the leap second was inserted.[*,†] Affected kernels were about 2.6.22 through 2.6.26.6.

Leap Second Caused High CPU Load on Linux Systems

After the leap second on June 30, 2012, CPU load increased to 100 % continuously which caused significantly increased overall power consumption in data centers.[‡, §, **] Affected kernels versions are around 2.6.32.

Windows

On Windows there is only an indirect error, namely that Windows doesn't account for leap seconds at all, which may cause malfunctions in applications if they suddenly observe a one second offset to other systems which have accounted for the leap second. Current NTP versions have a workaround which slews the Windows system time across a leap second insertion so that the resulting error is minimized.

NTP Reference Implementation

NTP versions up to 4.2.4 evaluated the leap second only if kernel support was available, otherwise the system time was stepped by one second about 15 minutes after the leap second event. A workaround for Windows has been submitted for v4.2.4, and correct handling for other systems without kernel support was implemented by Dave Mills in 4.2.6.[††]

Another bug in ntpd caused a leap second loop. After a real leap second event, the leap second warning bit was propagated back and forth between NTP servers, so a kind of loopback prevented the leap second warning from being cleared. Thus some NTP servers inserted another leap second at the end of every next month until the loop was manually broken up, *e.g.*, by restarting the affected NTP daemons.[‡‡]

---

[*] Linux Kernel Mailing List, "LKML Bug: Status/Summary of slashdot leap-second crash on new years 2008-2009."
URL: https://lkml.org/lkml/2009/1/2/373

[†] RedHat Bug 479765, "Leap second message can hang the kerne.l"
URL: https://bugzilla.redhat.com/show_bug.cgi?id=479765

[‡] Linux Kernel Mailing List, "Potential fix for leapsecond caused futex related load spikes."
URL: https://lkml.org/lkml/2012/7/1/27

[§] RHEL6, "Potential fix for leapsecond caused futex related load spikes."
URL: https://bugzilla.redhat.com/show_bug.cgi?id=836803

[**] "Leap second bug in Linux wastes electricity."
URL: http://www.h-online.com/open/news/item/Leap-second-bug-in-Linux-wastes-electricity-1631462.html

[††] NTP bug 508: "ntpd doesn't handle leap seconds gracefully on systems without kernel support for leap seconds."
URL: https://bugs.ntp.org/show_bug.cgi?id=508

[‡‡] NTP Bug 2246: "sys_leap is sticky." URL: https://bugs.ntp.org/show_bug.cgi?id=2246

Unfortunately, in NTP v4.2.6 support for negative leap seconds was removed, even though it had already been available in earlier versions, and even though incoming support for negative leap seconds is available via the protocol, and outgoing support for negative leap seconds is available by the kernel interface.

Invalid Leap Second Warning Issued by 3rd Party GPS Receivers

In July 2005 the GPS satellites started to broadcast a leap second warning for the end of December 2005. A faulty receiver assumed the announcement was for end of September and provided a leap second warning to NTP servers which used these devices as reference clock.

As a first consequence, invalid leap second warnings were propagated on the network, and handled by the NTP clients which received this leap second warning.

As a second consequence, the acceptance of leap second warnings from upstream servers was modified in the NTP reference implementation, such that it now only accepts leap warnings if a majority of upstream servers send the same leap second warning.

AVOIDING ERRORS IN SYSTEM DESIGNS

In complex systems, it is important to make sure that a leap second warning is propagated early enough to all components which need to care about the leap second. Usually the announcement is propagated in several steps from one node to another, and several types of data connection might be used between the individual nodes. If one data connection does not meet the requirements for the whole system, the whole system may not work correctly when a leap second is to be handled.

For example, in an NTP network the clients may have polling intervals of 1024 seconds or even longer, so their upstream server has to know about the pending leap second even earlier. Even if the reference clock is a GPS receiver which knows 6 months in advance that there's an upcoming leap second, the GPS receiver is unable to pass the leap second warning to ntpd if a serial time string format is used which doesn't support this.

Many GPS receivers are also capable of generating an IRIG or IEEE time code signal, but of course the GPS device can only propagate a leap second warning via the time code if the selected time code supports this, and if the time code receivers also understand the IEEE extensions. As mentioned above, this may not be early enough for NTP applications. Providing ntpd with a NIST leap second file if the available transport mechanisms are not appropriate can significantly improve the situation, but only if the leap second file is updated in regular intervals.

Of course also careful software design and implementation is required to avoid problems with leap second handling. For example, if time is synchronized between remote systems, then any time differences determined across a leap second should be discarded to prevent control loops from being messed up just because different nodes increment time in different ways during a leap second, which can result in temporary large offsets.

POSSIBILITIES FOR FUTURE IMPROVEMENTS

We have seen that there is an ambiguity of time stamps if time is simply stepped back by 1 second to insert a leap second. This problem with UTC is similar to the problem with local time at the end of a daylight saving time interval. In both cases it would be helpful to get some status information together with the time stamp, *i.e.*, if a local time stamp is DST or not in case of a

DST changeover, and similarly a flag telling that a leap second is in progress. The Network Time Foundation has scheduled a project for *Google Summer of Code 2013*[*] where possibilities for a new general time stamp format and associated API calls are to be developed and tested.[†]

A potential limitation here is that there are many applications, *e.g.*, high speed traders, which try to read timestamps at a very high rate, as fast as possible, and as accurately as possible. However, additional computations and comparisons to determine the level of the status flags might extend the execution time required to read the time and status flags. Also, locking mechanisms might be required to ensure that the returned time stamp and status flags are always consistent.

An interesting approach proposed by Steve Allen is to run the computer system clock on GPS time and use the "right" zone tables with a modified leap second table to convert to correct civil time including leap seconds.[2] Since GPS time is a linear time scale and the tz leap second file also provides historic leap second data, it is always possible to convert a system time stamp unambiguously to a different time scale which possibly observes leap seconds.

However, GPS is a time scale which is specific to the GPS satellite system. Since the time difference from GPS to TAI is constant it would in my opinion make more sense to use TAI instead of GPS for the basic linear time scale. There are two advantages of this approach:

- the tz leap second file could be used without being truncated
- interaction would be easier with PTP, which already uses TAI by default

It would be advantageous if there was an API call available to determine if the kernel clock runs on TAI or UTC, so time synchronization software like ntpd or ptpd could detect this at runtime and pass time adjustment data to the kernel clock accordingly.

Care must be taken if this approach is used on embedded systems where the firmware is not updated regularly, and thus the time zone and leap second files are usually not updated. There should possibly be a safe protocol to update the internal tz leap second table in a similar way ntpd does with the NIST leap second file via the NTP autokey protocol, or, at least the current number of accumulated leaps seconds should be forwarded as is done by the PTP protocol.

Finally, from the user's point of view it would be a good idea to add some expiration date to the tz leap second file, or even line up the NIST and tz leap second file formats so that they could alternatively be used by applications.

CONCLUSION

Every hardware and software component involved in the propagation and handling of leap seconds has its own specific properties, features, and limitations. For a complex system it is important to put things together in a way that the overall system still works as expected in case of a leap second.

Future improvements are possible to reduce the number of potential errors. Attempts should be made to make enhancements compatible with existing standards (use the TAI scale instead of GPS time) and existing applications (ntpd, ptpd, *etc*.), and reduce the number potentially redundant information in different formats (NIST vs. tz leap second files).

---

[*] Google Summer of Code 2013. URL: http://www.google-melange.com/gsoc/homepage/google/gsoc2013

[†] Network Time Foundation, "New Timestamp Format and API." Google Summer of Code 2013.
URL: http://wiki.nwtime.org/Main/GSoCProjectIdeas

REFERENCES

[1] Mills, D.L., "A kernel model for precision timekeeping." *Network Working Group Report RFC-1589*, University of Delaware, March 1994.

[2] Steven L. Allen, "Timekeeping System Implementations: Options for the Pontifex Maximus." *Decoupling Civil Timekeeping from Earth Rotation—A Colloquium Exploring Implications of Redefining UTC*. American Astronautical Society Science and Technology Series, Vol. 113, Univelt, Inc., San Diego, 2011. pp. 325-33.