USING THE NETWORK TIME PROTOCOL (NTP) TO TRANSMIT INTERNATIONAL ATOMIC TIME (TAI)

Judah Levine
Time and Frequency Division and JILA
National Institute of Standards and Technology
And University of Colorado
Boulder, CO 80305, USA

David Mills
Electrical Engineering Department
University of Delaware
Newark, DE 19716, USA

Abstract

Although Coordinated Universal Time (UTC) is the time scale that is transmitted by almost all time services, this scale is awkward to use in the vicinity of a leap second. Many computer systems cannot represent the epoch corresponding to a positive leap second (23:59:60), and remain synchronized to UTC by stopping the clock at 23:59:59 for 1 extra second whenever a leap second is to be added. This makes it impossible to assign unambiguous time tags to events that happen during this period. In addition, computing the length of a time interval that includes a leap second of either sign is difficult because simply subtracting the two UTC time stamps at the end-points of the interval does not account for the time interval occupied by the leap second itself. To address these issues, we have augmented the Network Time Protocol to allow a client system to reconstruct TAI from UTC and a table of leap seconds. This time scale has no discontinuity during the leap second. Intervals computed using TAI are unaffected by the additional time occupied by the UTC leap second, and the TAI time scale provides an unambiguous time tag to any event -- even one that happens during a UTC leap second. Although our solution is unique to servers that support the Network Time Protocol, it could be adapted to other time services and formats. Such systems could support time tags using either UTC or TAI, and would significantly reduce the problems that result from using UTC alone.

INTRODUCTION

The time services operated by National Metrology Institutes and timing laboratories generally transmit time

signals based on a local realization of Coordinated Universal Time (UTC). These local time scales, which are identified as UTC(lab), are based on the UTC time scale computed by the International Bureau of Weights and Measures (BIPM). Since 1972, the rate of the UTC time scale has been equal to the rate of International Atomic Time (TAI), a time scale that is computed by the BIPM using a worldwide ensemble of cesium clocks and hydrogen masers. The rate of TAI, in turn, is determined by the length of the second, which is defined based on the frequency of a hyperfine transition in the ground state of the cesium atom. In addition to serving as the basis for the definition of time, this transition frequency plays a central role in setting the values for other fundamental constants of physics.

The length of the day computed using the current value for the cesium second is somewhat shorter than the length of a day based on astronomical observations (specifically, the UT1 time scale). The difference is currently about 260 ms (a fractional difference of about 3×10^{-8}); if this difference were left uncorrected, UTC would gain somewhat less than 1 s per year relative to UT1. The rate of this divergence is increasing slowly.

This time divergence is bounded by introducing "leap seconds" into UTC so as to keep the absolute magnitude of the difference between UTC and UT1 less than 0.9 s. (The leap second is not added to TAI, so that the difference between UTC and TAI is exactly an integral number of seconds.) Since the length of a UTC day is significantly less than the UT1 value, small variations in the length of the UT1 day do not affect the sign of the difference. Therefore, leap seconds have always been inserted into UTC, and this is likely to continue for the foreseeable future. However, the definition of the process also supports deleting seconds from UTC, should that ever be necessary.

These leap seconds are usually added at the end of June or December; the most recent one was added to UTC at the end of 1998, and the difference TAI-UTC became 32 s at that time. No additional leap seconds are currently scheduled, although there will almost certainly be one announced during 2001.

THE PROBLEM

The last second of a normal UTC day is 23:59:59, and the first second of the next day is 00:00:00. When a leap second is to be added to UTC, an additional second, whose name is 23:59:60, is inserted between these two seconds. This procedure introduces two difficulties into timekeeping in general and computer-based timekeeping in particular.

The first problem is that most computers keep time internally in units of seconds since some epoch (such as 00:00:00 1 January 1970 UTC). (The conversion to and from other representations of the epoch is handled by the routines that display the internal value.) There is no way of representing 23:59:60 in this system, and most computer systems repeat the time value corresponding to 23:59:59 instead. Although the internal clock remains synchronized to UTC once the leap second is over, this solution effectively stops the clock during the leap second and results in two distinct seconds having the same time stamp.

The second problem is that an interval that includes a leap second is physically longer than a normal equivalent interval, but that difference is not reflected in a computation that simply subtracts the UTC time stamps at the end points of the interval. Measuring the rate of some physical process using the difference in UTC timestamps will therefore show a strange value whenever the interval includes a leap second. (This problem is not unique to leap seconds – leap years introduce the same problem in people's ages.)

THE SOLUTION

Since TAI has the same rate as UTC but has no leap seconds, applications that are affected by the two problems mentioned above can address both of them using timestamps derived from TAI instead of from UTC. Since these two scales always differ by an exactly integral number of seconds, it is a relatively simple matter to compute one from the other once the value of the difference is known.

Although it would be possible to address these problems by switching digital time services to pure TAI, this is not desirable for several reasons. In the first place, the resulting change would disrupt the thousands of systems that are based on the existing UTC-based definition. In the second place, such a change would result in a significant (and ever-increasing) difference between timestamps transmitted by the digital time services and normal civilian time, which will continue to be based on UTC to maintain its coordination with UT1. Therefore, we have implemented a solution which continues to be based on UTC timestamps, but which also contains the integer offset value between UTC and TAI to allow any client process to compute TAI as needed. Our solution is based on the Network Time Protocol (NTP), but the principles could be adapted to other protocols without great difficulty.

The solution we have implemented contains three independent components: (1) publishing the epochs of past and future leap seconds, (2) transmitting this information from servers to their clients, and (3) supporting combined UTC and TAI time stamps within the client itself. We discuss the details of each component in the following sections.

PUBLIC LEAP-SECOND TABLES

All of the public stratum-1 Internet time servers operated by NIST have a publicly-readable file that contains information regarding all known leap seconds since the current system was defined in 1972. The name of the file is leap-seconds.list, and it is in directory "pub". (This filename is a link to a second file, located in the same directory, which contains the leap second data. The name of this other file is leap-seconds.<timeval>, where <timeval> is the time at which the file was last modified in units of seconds since 1900.0. This is done so that users can detect when the file has been modified without actually having to read the file itself or even retrieve it from the NIST server.) You can get the current version of this file using anonymous ftp to any of our time servers. If you use a web browser, the URL would be: *ftp://<timeserver>/pub/leap-seconds.list.* The parameter <timeserver> is the name of any NIST-operated time server. (See our Web page *www.boulder.nist.gov/timefreq/service/time-servers.html* for a complete list.) For example, you could use the URL:

```
ftp://time-b.nist.gov/pub/leap-seconds.list
```

to get the current copy of the table of leap seconds.

The format of the table is described in the file itself. It is a simple text file. The symbol "#" introduces a comment, which continues from that point to the end of the current line. A line starting with #$ is a special

comment line. The parameter on this line gives the last modification date of the file in NTP format (i.e., as a number of seconds since 1900.0). There will always be only one such #$ comment lines. The remainder of the file contains leap second data, one value per line. Two consecutive data lines in the current file are:

```
3076704000      31      # 1 Jul 1997
3124137600      32      # 1 Jan 1999
```

The first value on each line is the epoch at which a new leap second was (or will be) introduced into UTC. This parameter is expressed in seconds since 1900 – the timestamp format used by NTP. The second value gives the difference between TAI and UTC starting at that epoch. The difference specified on each line is in effect until the epoch given on a following line, or into the indefinite future if there is no subsequent entry. (The remaining characters are a comment that gives the civil date corresponding to the epoch specified by the first value.) In each case, the second parameter is to be added to a UTC time stamp to produce the equivalent TAI value. The correction affects only the integer-seconds portion of the NTP time stamp; the value of the seconds fraction is always unchanged.

It is straightforward to compute the Modified Julian Day (MJD) number from an NTP time stamp. To convert, divide the NTP timestamp by the number of seconds in 1 day, and add 15020, the MJD corresponding to 1900.0. Thus 3124137600/86400 + 15020=51179, the MJD corresponding to 1 January 1999. All of these computations can be done using integer arithmetic, so that there are no issues of round-off or truncation.

The International Earth Rotation Service (IERS) in Paris is responsible for announcing leap seconds, and NIST will modify the files on the servers as soon as we receive this notice from the IERS. Leap seconds are normally announced several months in advance, and all of the NIST servers would normally be updated to include the new leap second within one week of our receiving the announcement. It is extremely unlikely that users would have less than one month of advance notice for an upcoming leap second.

TRANSMISSION FORMAT

Version 3 of the NTP time format (see RFC 1305) does not support any additional time parameters beyond the UTC timestamps that are part of the basic protocol. However, the additional space needed to transmit the difference between UTC and TAI is supported by the general format "extension fields" specified for NTP version 4. These extension fields are in addition to (and are added at the end of) an unmodified unauthenticated old-style timestamp packet, so that using them does not break previous versions of the client software that does not expect them or know what to do with them. (However, an association that used the version 3 authentication mechanism will not be compatible with a server that uses the new format to transmit the UTC – TAI offset, since this offset value will be mistaken for part of the old-format authenticator. This shouldn't happen in principle, since a version-3 client does not know how to ask the server for these data, and they are transmitted only if the client asks for them.)

The extension fields defined in NTP version 4 are placed between a version 3 timestamp packet and a terminating authenticator. Each extension field is a multiple of 4 octets (32 bits), and the total length of all extension fields is rounded up to the next multiple of 8 octets (64 bits). Many different types of extension field are defined. The most significant bit of the first octet always specifies whether the packet is a request

(0) or a response (1). The next bit is set in the response to indicate an error condition and is clear otherwise. The remaining bits of this octet specify the version number of the protocol; the format described here has been assigned a version number of 1. The second octet specifies the type of the request and the next 2 specify the length of any associated data. The type that is relevant to this discussion is "TAI leap second table." The details of the format are in [1].

To request the leap second table, a client sends a "TAI leap second table" packet (currently this is type 8) to the server with the response bit cleared (thereby indicating a request for data). The server responds with a table of 32-bit time stamps giving the times in NTP format when a leap second was (or will be) inserted. The table is in reverse chronological order – the time of the most recent leap second is transmitted first. The TAI – UTC difference associated with each of these time stamps is inferred from the transmitted length of the table, using the fact that the offset was 10 seconds when the current leap second system was instituted in 1972 and must be increased by 1 s at the time specified by each subsequent entry. In addition, the response contains a timestamp showing when the table was generated. If the first entry in the response is less than this value then this entry specifies the current TAI – UTC difference. If the first entry is greater than the file timestamp, then the first entry represents a future leap second that has been announced, but whose effective date has not yet arrived. Normally, the current offset is one less than the value inferred from the table length, and the next entry will give the effective date of the current TAI-UTC time difference. If the server is not synchronized to an authenticated source then the timestamp field is 0.

The extension field is terminated with a message authenticator code (MAC), which is designed to guarantee the provenance of the data. The authenticator is a message digest of the entire packet, and is computed using a 32-bit key value. Both symmetric-key and public-key methods can be used to authenticate the messages. The details are in [2]. While the authenticator is optional for a simple NTP time packet, it is required when an extension field is present.

This implementation depends on two assumptions: (1) that leap seconds are always integral values and (2) that all leap seconds will be positive. Both of these are basic to the current definition of UTC, but neither is a fundamental unchanging truth. The first assumption was not true before 1972, for example, and this method cannot be used to represent dates before 1972 for that reason. The second assumption might change if the current definition of the second changed, or if the way that definition is used to construct TAI and UTC were altered. This is not very likely in the short term, but neither the cesium definition nor the current method of calculating UTC will live forever.

CLIENT IMPLEMENTATION

The leap second table changes very slowly, and a client only needs to query a server infrequently – during a cold start and once a month or even less often during steady-state operation. Furthermore, a client could update its internal notion of the current offset value by incrementing it (or, in some future implementation, conceivably decrementing it) each time it executed its special internal leap second code. There are two different implementations, depending on how the client realizes the leap second.

If the client implements the leap second by stopping its clock for one extra second at a time that is equivalent

to 23:59:59, then it must increment the TAI – UTC difference at the start of the second copy of this second. Thus, the internal state of the system would be:

```
UTC Time    System Time      TAI-UTC value
23:59:58    23:59:58         N
23:59:59    23:59:59         N
23:59:60    23:59:59         N+1   (the leap second)
00:00:00    00:00:00         N+1
00:00:01    00:00:01         N+1
```

This is the more common case. On the other hand, if the client implements the leap second by stopping its clock for one extra second at a time that is equivalent to 00:00:00, then it must increment the TAI – UTC difference at the start of the second copy of this duplicated second. The internal state of the system in this case would be:

```
UTC Time    System Time      TAI-UTC value
23:59:58    23:59:58         N
23:59:59    23:59:59         N
23:59:60    00:00:00         N     (the leap second)
00:00:00    00:00:00         N+1
00:00:01    00:00:01         N+1
```

The timestamp in the leap second file that is associated with this event is the value corresponding to the UTC time of 00:00:00 – that is, the first second after the leap second has been inserted. Based on the previous tables, a client process that uses the first method must increment its internal value of TAI-UTC when its internal clock reads one second before the time in the table (for the second time), while a client that uses the second method must increment its internal counter one second after the value of its internal clock reads the time in the table (for the first time). If the internal implementation includes a special flag which is set to indicate that a leap second is currently in progress, then the TAI – UTC difference can be incremented when that flag is set. In each case, the goal is to realize a time scale that does not show any discontinuity (either in time or time interval) across the leap second.

Since maintaining the system clock is a kernel function in all operating systems, the easiest way to implement these strategies is to modify the kernel clock software and to add the additional location that holds the TAI – UTC time difference in the kernel address space. This is usually straightforward in principle, although it requires the source code of the kernel to realize it.

It is also possible to realize this system without modifying the kernel by defining a system-wide overlay to the kernel clock. This is obviously less desirable in principle and has not been done to our knowledge. Another solution (which was used in many older systems) is to implement the time adjustment associated with a leap second by adjusting the effective frequency of the system clock, thereby slewing the system time over some interval in the vicinity of the actual leap second. Since the maximum slew rate supported by most kernels is of order 0.1% or less, a system that implements this method will require several minutes to accommodate the leap second, and its clock will be wrong by a varying fraction of a second during this interval. The only advantage of this implementation is that it produces a system clock that never goes backwards, so that there is no problem with timestamps that violate causality. Since the system clock moves

backwards for only 1 second in the other implementations, this is not a strong enough advantage to outweigh the problem that the clock is wrong for several minutes following the leap second.

The final aspect of the client implementation is the method that is used to provide the TAI – UTC time difference to a user process. In order to maintain backward compatibility with applications that do not use this parameter, most implementations provide a separate system call which provides both the UTC timestamp and the TAI-UTC offset as part of a single structure. It is then a simple matter for any client process to compute the TAI timestamp. A client process that uses UTC timestamps simply ignores the TAI – UTC offset parameter.

SUMMARY AND CONCLUSIONS

We have designed a system that can support simultaneously transmitting both UTC and TAI. A client process can use either or both of these time scales as appropriate – UTC where compatibility with other services or legal traceability is required and TAI where smoothness of time intervals and monotonic time stamps are more important. This dual system is fully compatible with the previous implementation of the Network Time Protocol, and servers that support these new capabilities can simultaneously provide timestamps in the old formats. Although our system is designed around the Network Time Protocol, it can be readily adapted to other formats. In particular, the list of leap seconds is publicly available from all of the NIST time servers, and access to this list does not depend on the Network Time Protocol or on the details of the system used by the client to realize its internal time scale.

Even when the system fully supports the additional machinery for the TAI – UTC time difference, applications that reference the system clock directly without using the TAI correction will continue to operate in UTC as before. While these applications will use timestamps that are fully compatible with other civilian time services, they will have to cope with the discontinuity in time and in time interval in the immediate vicinity of each leap second. In particular, timestamps that are internal to the operating system (the modification time of a file, for example) will continue to be expressed in a time scale derived from UTC.

Finally, we note that the user interface to the clock on most systems is designed so that consecutive requests for the system time will produce responses that are monotonically increasing – even if the increase is only 1 in the least significant bit of the response. This design goal cannot be completely consistent with the methods described above that are used to realize a leap second, although the resulting time offset is almost always too small to be significant.

REFERENCES

[1] David Mills, "Public Key Cryptography for the Network Time Protocol," Internet Draft, Draft-ietf-stime-NTPauth-00.txt, June 2000. Available on the Web at www.eecis.udel,edu/~mills/memos.htm.

[2] David Mills, "Public Key Cryptography for the Network Time Protocol," Electrical Engineering Report 00-5-1, May, 2000, University of Delaware Electrical Engineering Department. Available on the Web at www.eecis.udel.edu/~mills/reports.htm.

Questions and Answers

RICHARD SCHMIDT (USNO): Is it possible that your table could be limited to just two lines? Is it possible that your cache table of leap seconds could be limited to just two lines rather than the full historical table to limit the amount of data that you transfer?

JUDAH LEVINE: There are only 20-odd leap seconds, so the table is going 20 inches long. Generally speaking, you are almost never going to be interested in historical value. You are going to want the current value, and maybe the next value. So, I would guess the table entry is maximum 20-something inches long and, more likely, 2 inches long. And that's it. I think in our lifetime, there's going to be 100 leap seconds, at most. After that, it is going to be somebody else's problem.

ROBERT LUTWAK (Datum): I have a question from the perspective of other manufacturers who are building NTP servers who need to receive advance notice of the leap seconds. It is not unlikely that they may start to rely on this table as a way of getting advance notice of the leap seconds. In which case, maybe instead of just having it be a NIST incorporation in some extension fields, it really needs to be built into the standard of NTP. So when the manufacturers build it into hardware, they can rely it for the future.

LEVINE: I agree with you at the 100% level. That is somewhat beyond my power to do. All I can say is that the extension field concept is built into the standard for NTP. The idea of using the TAI leap second offset is built into Mills's documentation of the standard. The tables are there, and obviously, we encourage other people to do it as well. We're not an enforcement agency.

But yes, it would be nice if this became an official standard, and we can only propose it in order for it to become an official standard. But we are committed to maintaining the system as prescribed. If you look at Mills's documentation, you can see the details of how these are defined. They are fundamental to NTP Version 4 because they are used for other purposes. That is, they are part of the authentication and the digital signature system. So there are many types of extension fields besides this issue of leap seconds. There is a lot of other stuff, there are a lot of other capabilities that you might need an extension field for.

DENNIS McCARTHY (USNO): Two things. One, I would just like to say that I agree that this would go a long way towards helping some problems. But it still doesn't alleviate the basic fundamental issues. So I didn't want people to have a feeling that this solves all the problems and now we can go on.

LEVINE: No, I didn't mean that at all.

McCARTHY: Yes, but I think it does help some of our software users that complain about UTC.

The other thing I was wondering is what about the possibility of having this table actually be maintained by something like IERS or BIPM or something like that where this would be an international conventional table of some sort. You know, you wouldn't have to worry about somebody updating it; it would just be there at the IERS or BIPM.

LEVINE: I think that is a fine idea. I would do everything I could to encourage whoever wanted the table to take the table. It's there, it's available by public ftp from our servers. If anybody thinks it is useful, go for it.

McCARTHY: Yes, we will have an IERS directing board meeting in 2 weeks or something. I was thinking that maybe we could even bring that up for discussion.

LEVINE: That's fine. Now, the only thing that is somewhat special about this table is that its time stamps are in NTP format. And so, if you want to convert them to Julian day numbers, you divide by 86,400 and add 15020, and we all know how to do that. And that's all it would take.

WLODZIMIERZ LEWANDOWSKI (BIPM, France): I would like to add that the BIPM could come to such a table. That could be discussed during the next CCTF meeting, and the dates are now official. It will be held on June 20 and 21, 2001. So I believe that the leap second will again be a subject of CCTF, and such issues that you have raised here could also be discussed.