Getting Timestamps Right

Martin Konrad
Control System Engineer
Common Causes of Bad Time Stamps

- Devices/IOCs might use different time servers which are not in sync
  - Synchronize all devices to one time source
  - Block access to other time servers

- IOCs might use different times (e.g. local time vs. UTC)
  - Use the same time on all devices

- Switching between standard time and daylight savings time causes gaps/ambiguities
  - It can be hard/impossible to establish the sequence of events with non-monotonic time
  - Archivers usually drop data that is older than the latest timestamp in the archive
    » One hour of data will get lost!
    » It might not be obvious to the operator which data got lost
  - Avoid daylight savings time

- Leap seconds
Coordinated Universal Time (UTC) is based on very stable atomic clocks.

Rotation speed of the earth varies slightly.

This causes UTC to slowly drift away from the mean solar time (UT1).

If the deviation gets bigger than 0.9 s a leap second is inserted.

Always on June 30th or December 31st.

Next leap second is coming up on June 30th 2015 23:59:60 UTC.

This will be the 26th leap second since 1972.
Leap Seconds II

- https://commons.wikimedia.org/wiki/File:Deviation_of_day_length_from_SI_day.svg
UTC is subject to leap seconds

By definition POSIX timestamps are supposed to be in UTC

However, POSIX timestamps do not have a representation for a leap second (e.g. “2012-06-30 23:59:60” is not a valid POSIX timestamp)

EPICS timestamps suffer from the same problem:

define struct epicsTimeStam {  
  epicsUInt32 secPastEpoch; /* seconds since 0000 Jan 1, 1990 */  
  epicsUInt32 nsec; /* nanoseconds within second */  
} epicsTimeStam;
- PCs running ntpd need to “cheat”
  - ntpd is usually aware of leap seconds
  - Depending on the implementation of the kernel, systems might be able to skip the last second of the day or stretch it to 2 seconds
  - If the kernel does not support this ntpd will correct the time after it is already wrong

- Multiple machines using different ntpd implementations/operating systems might handle leap seconds in a different way
  - Timestamps might be off by up to 1 s!
If you want to calculate time differences between POSIX/EPICS timestamps accurately you need to know how many leap seconds have occurred between your two points in time.

It is impossible to write a self-contained function that calculates time differences accurately:
- Requires leap second data (e.g., from the Internet).
- Leap second data is only known a few months in advance.
  - Needs to be updated during operation.
## Our Solution For FRIB

<table>
<thead>
<tr>
<th>Issue</th>
<th>CST/CDT</th>
<th>CST</th>
<th>UTC</th>
<th>GPS Time</th>
<th>TAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time zone issues</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Gaps</td>
<td>Yes (1h, 1s)</td>
<td>Yes (1s)</td>
<td>Yes (1s)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Duplicate timestamps</td>
<td>Yes (1h, 1s)</td>
<td>Yes (1s)</td>
<td>Yes (1s)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Slowly drifting away from wall clock</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Use International Atomic Time (TAI) to get rid of leap seconds
- We decided to live with the disadvantages
  - TAI is different from wall clock time (operators need to get used to that)
  - The offset to wall clock time changes whenever a leap second occurs
Instead of UTC we will distribute International Atomic Time using NTP/PTP.

Time server does not announce leap seconds.

No further changes required (no need to touch any code) 😊