Network Timing and the 2015 Leap Second

Darryl Veitch[1] and Kanthaiah Vijayalayan[2]

[1] Faculty of Engineering and IT, University of Technology Sydney, Australia
[2] Melbourne School of Engineering, University of Melbourne, Australia.

Abstract. Using a testbed with reference timestamping, we collected timing data from public Stratum-1 NTP servers during the leap second event of end-June 2015. We found a wide variety of anomalous server-side behaviors, both at the NTP protocol level and in the server clocks themselves, which can last days or even weeks after the event. Out of 176 servers, only 61% had no erroneous behavior related to the leap second event that we could detect.

Keywords: leap second, NTP, Stratum-1 server, network measurement, LI bits, UTC

1 Introduction

Timekeeping is central to network measurement. It is a service typically provided by a computer operating system, whose *system clock* is synchronized, through timestamp exchange over the Network Time Protocol (NTP), to a remote reference. In the timeserver hierarchy, a *Stratum-s* timeserver synchronizes to a *Stratum* $s-1$. Anchoring the system are the *Stratum-1* servers, which have local access to reference hardware such as a GPS receiver or atomic clock. These roots of the timing forest 'hierarchy' can be PCs, or dedicated network appliances.

Network timing distributes *Coordinated Universal Time* (UTC). This is a discontinuous time standard: jumps known as *leap seconds* are inserted (roughly every two years) in order to keep the timescale in step with the solar day. Leap seconds are propagated through the server hierarchy, but it is well known to system administrators and others that this process is far from perfect, and can cause havoc with system clocks and the host systems themselves.

In this paper we examine the behavior of a set of public Stratum-1 servers during the leap second event of end-June 2015. Our objective is to determine which servers perform as expected, both from the server-clock accuracy and protocol points of view, and to characterise the deviations. We find that behavior which is far from ideal is quite common, and there are many examples of extremely poor behavior, for example servers which never incorporate the leap second, or never inform their clients of it. Ideal behavior, as far as we can measure it given the resolution of our dataset, occurs in only 61% of the servers we study. Given their role in the foundation of the timing system, these Stratum-1 findings are bad news for the public timing system as a whole.

Our conclusions are based on measurements using reference timestamps from a GPS synchronized DAG packet capture card, and an analysis methodology capable of disambiguating network events from server behavior. Although it has limitations, most notably the fact that we are only capturing a subset of all public Stratum-1's, we know of no prior study of leap second events which has the detail or precision of what we present here. We intend to make our data available to the community.

The paper is organized as follows. Section **??** provides background on time standards, leap seconds and NTP, and discusses prior work. Section **??** describes our testbed, the selection of servers, what the datasets are and how they were collected. Section **??** outlines the analysis procedure we employed to characterize the nature of the leap second behavior of the servers, even in the face of significant noise. Our results are detailed in Section **??**, and finally, Section **??** discusses the import of our findings.

2 Background

2.1 Time Standards and Leap Seconds

The primary international time standard is the *Temps Atomique International* (TAI). It is based on combining the (relativistically corrected) outputs of high precision atomic clocks in over 300 National Laboratories, including the USA's National Institute for Standards and Technology (NIST), and Australia's National Measurement Institute (NMI). The TAI is a continuous time scale, with each second a standard SI second, and with epoch (origin) at HH:MM:SS = 00:00:10, 1st January 1972. It is best to think of TAI as a real number, in units of seconds, since that epoch. *Universal Time* (UT1) is a descendant of Greenwich Mean Time, a continuous time scale whose unequal seconds allow synchronization to the solar day. Because the Earth's rotation is slowing, UT1 is falling progressively further behind TAI.

The primary time standard used for general timekeeping is *Coordinated Universal Time* (UTC). This is a discontinuous time scale with epoch at $t_{\mathrm{TAI}} = -10\,\mathrm{sec}$, best thought of as TAI to which jumps of exactly 1 second have been infrequently applied in order to keep UTC close (within $0.9\,\mathrm{sec}$) to UT1. Within UTC, a positive leap second manifests as a downward jump, slowing the clock down with respect to TAI. Leap seconds, when needed, are added at the end of the last minute of a month, typically June or December. Negative leap seconds are defined but have never been used.

Realizations of both TAI and UTC are maintained by the *Bureau international des poids et mesures* (BIPM). The timing standards body is the International Telecommunications Union (ITU), but it is the *International Earth Rotation and Reference Systems Service* (IERS) that decides on leap seconds, and announces them months in advance via its biannual "Bulletin C".

In this paper we focus on the leap second added at the end of June 30, 2015. The leap event was completed at 00:00:00 July 1st UTC when TAI was

$t^*_{\rm TAI} = 1435708836\,{\rm sec}$, and $t^*_{\rm UTC} = t^*_{\rm TAI} - 36$. For convenience, we plot all timeseries against a timescale "t", which is $t_{\rm TAI}$ with its origin reset to $t^*_{\rm TAI}$.

2.2 Leap Seconds and NTP

The NTP hierarchy distributes UTC. Stratum-1 servers learn of leap seconds through various mechanisms depending on the reference time source. The most common is GPS, which supports UTC and makes complete leap second information available. A commonly used alternative, used for example by many UNIX operating systems, is to include a 'leap-seconds' text file as part of NTP configuration. This file, which lists leap second event times as well as an expiry date, is maintained by NIST and is available from [**?**].

The main mechanism by which servers of higher strata learn of leap seconds is via their server (or peer). The NTP packet header has a 2-bit *Leap Indicator* (LI) field. RFC 5905 (NTP version 4), specifies that servers set LI = 01 in response packets when a (positive) leap second is scheduled *in the last minute of the current month*. Obsoleted RFCs 1305 (NTP v3) and 4330 (SNTP v4) instead state *.. in the last minute of the current day*. The language in the RFCs is ambiguous, as it confuses describing under what circumstances LI should be set (to 01), with how far in advance to do so. A consistent reading across RFC 5905 and RFC 4330/1305 is that LI should be set in each server response packet during the entire month (5905) or day (4330/1305) of a scheduled leap. Alternatively, there may have been no intention to specify how far in advance to set it. Most informal sources state that warnings are issued in the prior 24hrs in common implementations. We discuss this further below.

The fact that UTC jumps backwards is inherently problematic. It is complex and confusing, may break software reliant on monotonic time, and is fertile ground for bugs. This is exacerbated by the fact that the detail of how time should be kept *during* leap second events is not standardized: approaches vary by operating system, OS version, and NTP version. Note that by 'leap second event' here we mean the entire second just before the jump itself, since over this interval clock software must do something non-standard to account for the leap, including allowing the corresponding minute to have 61 seconds. A useful reference here is [**?**]. As described in [**?**], using the *ntpd* daemon (the incumbent clock synchronization algorithm for client system clocks developed by David Mills [**?**]) with the *-x* option disables the sudden leap second adjustment and so avoids a number of problems, but results in convergence times to post-leap UTC of the order of 10's of hours.

2.3 Prior Work

There are many informal reports available on-line detailing implementation issues with leap seconds, describing bad behavior of client systems, related operating system bugs and configuration problems, and providing recommendations for system administrators. For example contemporary issues in Linux (including recommendations to simply shutdown NTP and restart around leap second

events) can be found in [?,?,?]. RFC 7164 provides an overview of implementation issues in relation to POSIX. A recent paper by Burnicki [?] (see also the related presentation [?]) gives a useful description of how leap seconds are disseminated, and alternative proposals for how to deal with them in end systems, including several 'slewing' variants, where a sudden jump is replaced by a period of modified clock rate. This includes the slewing scheme included in NTP for Windows, and that proposed recently by Google.

There is very little peer-reviewed work on Stratum-1 server behavior, and still less on leap-seconds. The closest work appears to be that of Malone [?], a web page which gives two graphs looking at LI values, of a similar set of servers to ours, about the 2015, 2012, 2008 and 2005 leap second events. While informative, the results are coarse grained, with scant methodological detail, and do not extend to the server clocks themselves.

3 The Experiment: Testbed, Server List, and Data Set

The experiment ran over 22.3 days from June 29 to July 21, 2015. In it a single host ran multiple independent RADclock [?] daemon instances. RADclock is an alternative clock synchronization algorithm with high accuracy and robustness [?]. Each instance emitted an NTP timing request packet every $\tau = 64$ seconds to its chosen server. Packets were timestamped to 200ns precision or better by a DAG3.7GP capture card [?] via a passive tap just outside the host, synchronized to a roof-mounted Trimble Acutime 2000 GPS receiver.

For an NTP packet i which completes its round-trip from the client to server and back, and is successfully matched on return, we obtain a 4-tuple *stamp* of UTC timestamps $\{T_{a,i}, T_{b,i}, T_{e,i}, T_{f,i}\}$. Here $T_{b,i}, T_{e,i}$ are made by the server and are extracted from the returning NTP packet, as are the LI bits and the server Stratum field from the NTP header. For this experiment the configuration was such that the DAG timestamps $T_{a,i}, T_{f,i}$ **ignore the leap second**, and so are on a continuous timescale ('pre-leap UTC') over the experiment.

Our server list is based on the public Stratum-1 url list maintained at *ntp.org*, and contains 176 servers. Of these, 156 were listed as OpenAccess at *ntp.org* at some time between Sep. 2011 and June 2015, and resolved to a unique IP address which responded to NTP requests, during the experiment. To this we added 9 public and 6 private (3 from NMI and 3 in our lab) Australian Stratum-1 servers, and 5 used by CAIDA Ark monitors [?].

4 Methodology

From the timestamp data we estimate, for each server, the time series of round-trip time $R_i = T_{f,i} - T_{a,i}$, *server change*: $C_i = (D_i^{\uparrow} - D_i^{\downarrow})/2$, and *server error* $E_i = C_i + L_i$. Here $D_i^{\uparrow} = T_{b,i} - T_{a,i}$ and $D_i^{\downarrow} = T_{f,i} - T_{e,i}$ are the estimated outgoing and incoming delays, and $L_i = L(T_{e,i})$, where $L(t)$ is a step function rising from 0 to 1 at $t = t^*_{\mathrm{TAI}}$. The series C_i (resp. E_i) consists of errors in

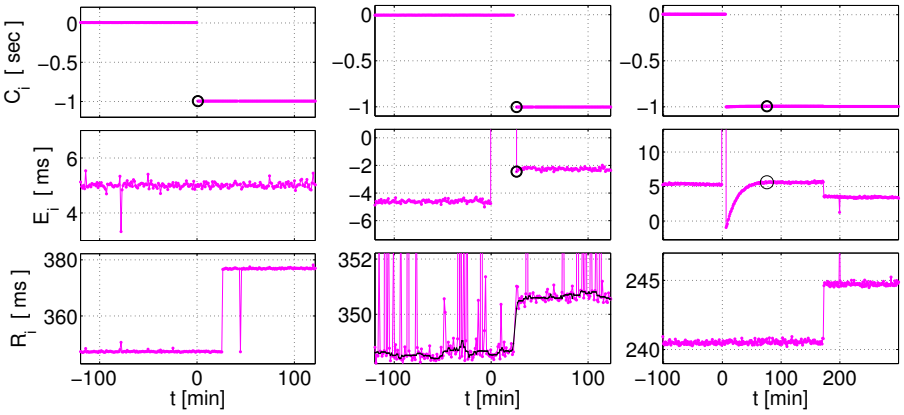

Fig. 1. Introduction to server behavior and the methodology for TEB determination. Left column: Good server; Middle: Bad server due to delayed leap; Right: Bad server with delayed leap plus post-leap instability resulting in a much larger TEB.

the server clock with respect to the DAG timescale (resp. UTC), together with 'noise' due to path routing changes and congestion. We use R_i, which is *entirely independent of server timestamps and of the leap second event*, to judge path conditions independently of server behavior.

We illustrate our methodology through the examples appearing in Figure **??**. The server assigned the left column is well behaved, and so the top plot shows the leap behavior in C_i one would expect. More precisely, the detected leap position (black circle) at $t = 51.7$ sec is at the first stamp past $t = 0$, and the previous stamp was at $t = -13.5$, before $t = 0$ as required. These values were determined through inspecting E_i (middle plot), whose variability is steady about $t = 0$ in a sub-ms band, showing no evidence of perturbations about the leap. Note that E_i is centred about 5.2 ms rather than zero due to path asymmetry, not server error. The level shift event in R_i at around 26 min does not appear in E_i as it results from a symmetric path change.

The middle column exhibits a server where the leap occurs neatly, but is $t = 26.2$ min late. We call this delay the *Time to Expected Behavior* (TEB). After the leap E_i is 2 ms higher than it was before $t = 0$, however inspection of R_i (and its median-filtered version, the black curve) in this zone reveals it to be due to a path change affecting path asymmetry, rather than additional leap-induced server errors. Hence the naive TEB value associated to the main jump is taken as the final value.

The right column shows a server where not only is the initial leap late, but there are additional errors beyond it of a few ms in amplitude (visible in E_i but hidden in C_i) resulting in TEB $= 75.8$ min. The R_i plot confirms that this 'monotonic recovery' event in E_i does not result from path effects but is associated with the leap event at the server.

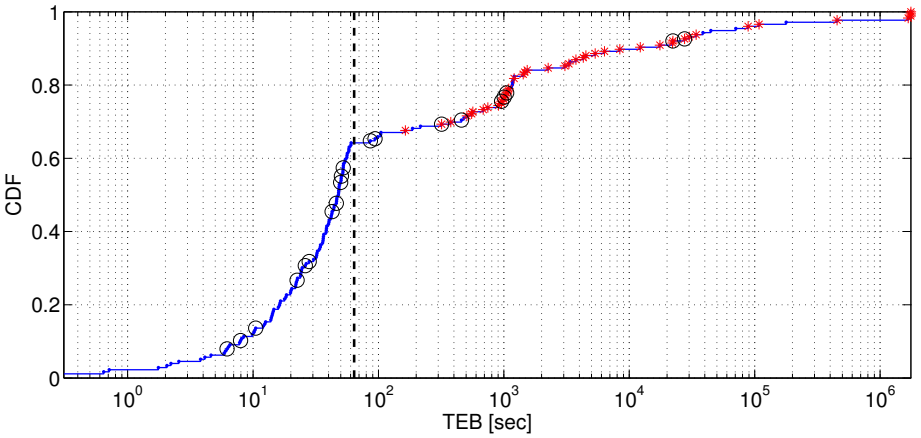

Fig. 2. A cumulative distribution function of TEB across all servers. Red stars denote Bad servers. All servers with TEB < 64 sec (left of dashed line), are Good. The black circles are LI-Bad, that is Bad with respect to LI behavior (see Section **??**.)

Each server was closely examined using the above approach to determine a TEB value which genuinely reflected recovery from the leap second event, rather than any other cause. More precisely, for each server the TEB was set to correspond to the earliest time at which the variations in E_i following a leap fell below the magnitude of the path noise as revealed by R_i. The precise stamp at which this occurs was selected 'by eye' taking into account the degree of short term variability of each time series. In a small number of cases, the nature and/or amplitude of the variability due either to path changes, congestion, or server errors unrelated to the leap event, make the exact value of TEB hard to evaluate, however we are confident that the resulting error is of the order of a few percent even in these cases.

5 Results

5.1 Server Clock Behavior

We label a server clock as 'Good' (else 'Bad'), with respect to its leap second response, when there is no hard evidence of incorrect behavior. That is when E_i is constant for all stamps both before and after $t = 0$, up to the observed variability as calibrated by R_i. The value of this constant reflects path asymmetry, and is close to zero relative to minimum RTT. Since the per-server periodic packet flows have random phase with respect to each other, for such servers we expect to find TEB values uniformly distributed in $[0, \tau]$. A consequence is that actual detection resolutions can be both higher or lower than the (unfortunately low) $\tau/2 = 32$ sec average case. Note that lower values of τ would have run the risk of appearing as an attack on the server.

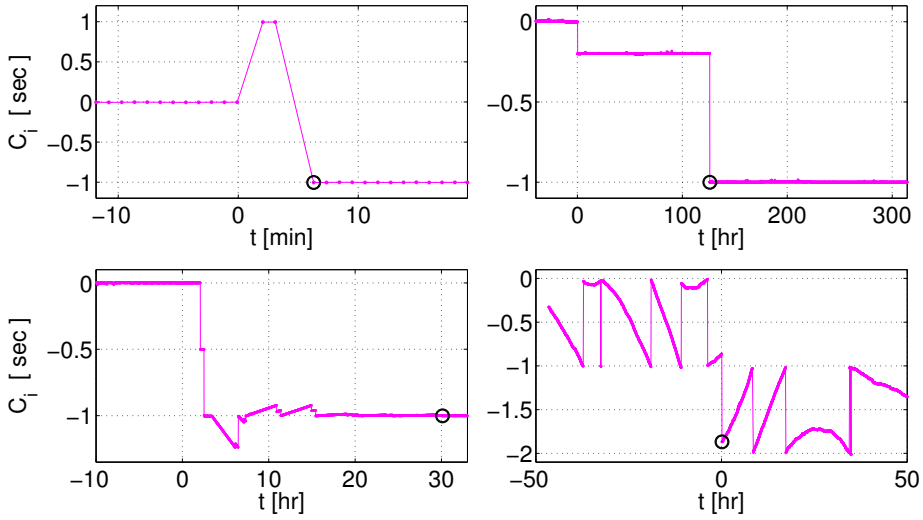

Fig. 3. Examples of Bad servers with more extreme behavior (black circle = TEB). We show C_i plots, as these have a range over 1 second and are thus well suited to contextualize extreme errors. Note that correctness of the displayed TEB values cannot be ascertained from this view. For that we require a zoomed view, such as E_i provides.

Figure **??** records the distribution of TEB values over servers in the form of the empirical $F(x) = \Pr(\text{TEB} \leq x)$. The values require a log scale as they are spread over the entire 22 day experiment duration, including 2 cases where the leap never occured (TEB set to trace duration). A single server jumped early, at $t = -0.164\,\text{sec}$, and could not be plotted. All 112 servers with $\text{TEB} \in [0, \tau]$ (left of the dashed vertical line in plot) are, not surprisingly, Good, and inspection in a linear plot showed they are uniformly distributed as expected, the smallest being $\text{TEB} = 0.13\,\text{sec}$. However, of the 131 Good servers, 19 have larger TEB values because the server did not respond for an interval extending beyond $[0, \tau]$. Although values a little beyond $t = \tau$ could indicate congestion losses or server overload, larger values imply a problem. Thus, for example, whereas $\text{TEB} = 20\,\text{sec}$ for a Good server is consistent with a sampling resolution of $\tau = 64\,\text{sec}$ and does not imply in any way that the server failed to leap for 20 seconds, $\text{TEB} = 100\,\text{sec}$ for a Good server suggests that the server was not behaving ideally (perhaps offline as part of leap second management or failure), over this entire period.

The 45 Bad servers display a wide variety of behaviors. The 'delayed but otherwise clean' leap behavior encountered in the middle column of Figure **??** is found in 15 cases, with a median delay at $\text{TEB} = 22\,\text{min}$. For 5 other servers an initial delay was accompanied by a significant period where the server did not respond, but nothing more complex. The final 25 cases displayed more extreme behavior including multiple jumps, failure to jump, and post-leap instability occuring over periods ranging from hours to weeks. Figure **??** displays the C_i

plots for a number of these. The bottom right plot deserves special mention. It shows a server whose delayed but clean leap at TEB = 9.0 sec occurs in the context of persistent and severe underlying server errors, which in fact are present throughout the entire experiment. A number of both Good and Bad servers display server errors unrelated to their performance during the leap event. A detailed analysis of broader server anomalies is beyond the scope of this paper. In [**?**] we perform such a study based on different and longer datasets from a subset of the servers studied here.

5.2 Protocol Behavior

The most significant fact about the protocol compliance is that 41 of the 176 servers (24%) failed to set the LI bits correctly in **any** of the packets received during the course of the experiment. NTP clients relying on these servers as a reference, and their own clients lower in the hierarchy, would not have received the warning about the impending leap second, and would therefore have failed to insert it themselves, resulting in persistent errors and potentially serious consequences, unless they received word by some other means (such as via a majority of peers, or the leap-seconds file).

Of the 135 servers that did set the LI bits correctly, many did not do so in an ideal manner. Consider first the times at which the servers ceased their leap warnings. A total of 18 servers (13%) continued to set the LI bit after $t = 0$. Of these only 8 had ceased after an hour, two continued for 12 days! and a further 2 for a week. Even a single packet with LI set received after $t = 0$ however has the potential to cause the client to insert an additional leap second at the end of the new month, July in this case. Presumably implementations will attempt to disregard warnings received just after leap events to allow for delays in packet arrival from the server(s), however they may not succeed, in particular if the warnings continue indefinitely.

Now consider the times when the servers *began* to send warnings. As pointed out earlier, it is not clear from the NTP standard when warnings should in fact begin, however 24hrs seems to be commonly used/supposed, in particular in SNTPv4 implementations (RFC 4330) which are still very common. In fact in 6 out of the 135 cases we found that warnings began exactly 6 months in advance! (We know this thanks to a complementary dataset we collected at that time. Interestingly, this happens to correspond to 00:00:00 Jan 1st, the other common time when leap seconds can be scheduled.). These extreme cases aside, the warning start times of the remaining $135 - 6 = 129$ servers do cluster about 24hrs, as shown in the histogram of Figure **??**. We see that in most cases ample warning is given, provided the systems in question are up. There is no evidence of warnings beginning a month in advance as, perhaps, suggested by RFC 5905.

Our results are consistent with the findings of Malone [**?**] for the same 2015 leap second event, who also reports that most leap second warnings begin close to 24hrs in advance, and that around 60% to 80% of servers set the bits to the right value, namely LI= 01, for a positive leap second. He also comments that

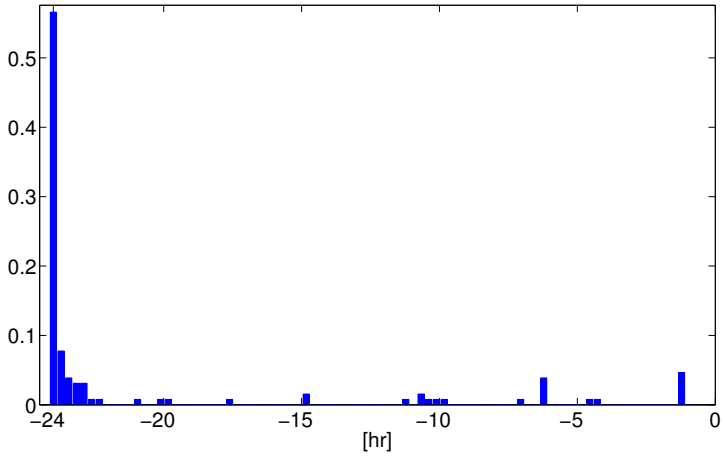

Fig. 4. Histogram of LI warning start times relative to $t = 0$ (excluding the 6 cases where warnings began exactly 6 months in advance).

most servers had ceased their warnings by an hour after the event, in agreement with our findings here.

5.3 Overall Behavior

A natural question to ask is, how many servers are 'perfect'? meaning that there is no evidence of errors of any kind with respect to the expected leap second behavior, neither in the accuracy of the server clock timestamps, nor in the protocol compliance.

To answer this question we must first define more precisely what we mean by protocol compliance. We define the warning start time to be Good if it falls in the interval $[-24 \times 3600, -3072]$ sec. Here we assume that a client is using the maximum polling period to its server of $\tau = 1024$ seconds, and allow for 2 consecutive packet losses ($3 \times 1024 = 3072$) in order to define the last safe time at which a server should begin delivering warnings. In terms of the warning end time, we define behavior to be Good when no warnings are received after $t = 3$ sec, to allow for worst case delay of a packet sent from the server at $t = 0^-$ to the client. Finally, we consider that a server is 'LI-Good' if it sends LI warnings in a way which is Good in each of the above two respects.

In Section **??** we reported that there were 131 Good servers, and using the above definition we find that there are 115 LI-Good servers. The intersection of these, the 'Perfect' servers, is 108 strong, or 61% of the total of 176 in the list. Note that of the Perfect servers, 97 (90%) are among the servers with TEB $<$ 64 sec from Figure **??**. Finally, we should not forget that Perfect should not be taken literally. It actually means that we found no hard evidence of failure in our sampled data. In fact, as explained earlier, a number of Good servers have

suspiciously large TEB values indicative of a server with poor availability, which is not ideal server behavior, and of course our sampling resolution prevents us from detecting servers with errors that were corrected quickly.

Although we have assembled our server list from sources either known to be Stratum-1, or claiming to be through their presence on the list at *ntp.org*, it turns out that not all of them are. In fact a considerable number drop their Stratum-1 status every now and again (the S-varying group), and a small number were never Stratum-1 over the duration of the experiment. Table **??** gives a breakdown, and shows, for each stratum grouping separately, what percentages fall into the Good, LI-Good, and Perfect performance categories. The differences are smaller than one might imagine, with the S1-always servers doing a little better than the others in terms of protocol compliance, but, counterintuitively, worse in terms of clock accuracy. It should be noted however that the samples sizes are small. In particular there is not a significant difference between the two most populous and closely related categories, S1-always and S-varying, in either of the Good and LI-Good groups.

	S1-always	S2-always	S3-always	S-varying	NMI	NIST
size	122	14	3	37	3	10
Good	72%	86%	67%	76%	100%	80%
LI-Good	66%	50%	67%	65%	100%	100%
Perfect	64%	50%	67%	54%	100%	80%

Table 1. Breakdown of different server groups that set LI bits into Good, LI-Good, and Perfect subsets. Four measured stratum-level groups appear on the left, and two National Laboratory groups on the right. The varying stratum category (S-varying) consists of servers which are usually Stratum-1 but sometimes not.

A breakdown of two National Laboratory groups is also given in Table **??** on the right hand side. This is interesting as we expect them to have the highest standards, and in the case of NIST, a large client base. For NMI on the other hand only registered clients are allowed – these are not public servers. Though both NMI and NIST servers showed excellent protocol compliance, 2 of 10 NIST servers in the list fell down on clock performance. Moreover, most of the NIST servers exhibited server anomalies unrelated to the leap second event, whereas none of the NMI servers did. This is in agreement with our findings in [**?**].

5.4 System Dependence

Information on the hardware and software platforms underlying the servers on the list is available at *ntp.org*, however it is incomplete, potentially out of date, and far from uniform. We are in the process of seeking out and contacting the server administrators in order to obtain a more complete picture of variables

such as the nature of the reference timing source, the origin of the server hardware (commercial appliance or commodity), and operating system (proprietary, Linux or BSD), and NTP version and configuration. It is not possible to report on this in detail here, however it seems clear that, in agreement with the observations of Nelson in 1999 [**?**], that by far the most common reference source is GPS. We also find that NTPv4 is more prevalent than NTPv3, and that both commerical and commodity servers are well represented. The administrators and their organisations span a broad range, from National Laboratories in the US, Australia, Sweden, Russia and elsewhere, to time enthusiasts making servers available out of personal interest. Many of the servers on the list participate in the public *ntppool*.

6 Discussion

It is difficult to say how much influence the servers in our list have on public network based timekeeping, in particular since the advent, since late 2013, of NTP amplified reflection attacks [**?**], have caused administrators to block the server query commands that would have made a crawl of their clients and peers possible. Certainly they are only a minority of the total number of publicly accessible servers, given that Minar's 1999 survey discovered 957 Stratum-1 servers [**?**]. On the other hand our list contains several servers from National Laboratories, notably NIST, that can be expected to be well known with sizable client bases, as well as many others which participate in the widely used *ntppool*. We believe that it reasonable to claim that the deficiencies we have detailed in the paper, where only 61% of servers are behaving (as far as we can tell) correctly, and many behave in a very damaging way, can have a considerable impact.

Responding to leap seconds reliably is a complex affair, as it is a function of many interactions of software and hardware of different generations and provenance. The fact that in 2015 there are still so many issues, even in Stratum-1 servers, is a testament to this fact. For this reason it has been debated for some time within the ITU whether leap seconds should be abandoned entirely. In fact the ITU considered this question at the World Radiocommunication Conference (WRC-15) meeting in November 2015 (after this paper was submitted to PAM 2016). The outcome was that more study was needed, and a report on future time scales, including the fate of the leap second, will be considered by WRC-23 in 2023. The approach of this paper can be used as the basis of a broader study of future leap second events leading up to 2023, as well as to track the health of network timing infrastructure more generally.

Acknowledgment

Partially supported by Australian Research Council's Linkage Projects funding scheme #LP120100073, in partnership with Symmetricom (now Microsemi).

References

1. NIST. ftp://time.nist.gov/pub/leap-seconds.3629404800.
2. M. Burnicki, "Technical Aspects of Leap Second Propagation and Evaluation," in *Requirements for UTC and Civil Timekeeping on Earth Colloquium*, vol. 115 of *Science and Technology Series*, Univelt Inc., San Diego, 2015.
3. M. Lichvar, "Five different ways to handle leap seconds with NTP." `http://developerblog.redhat.com/2015/06/01/five-different-ways-handle-leap-seconds-ntp/`.
4. D. L. Mills, *Computer Network Time Synchronization: The Network Time Protocol*. Boca Raton, FL, USA: CRC Press, Inc., 2006.
5. M. Elsins, "HANDLING THE LEAP SECOND – LINUX." `http://www.pythian.com/blog/handling-the-leap-second-linux/`.
6. redhat, "How to clear the Leap Second Insertion flag after it has been received?." `https://access.redhat.com/articles/199563/`.
7. M. Burnicki, "Technical Aspects of Leap Second Propagation and Evaluation," in *FOSDEM*, 2015. `http://nwtime.org/leap-second-resources/`.
8. D. Malone, "Leap Second 2015." `http://www.maths.tcd.ie/~dwmalone/time/leap2015/`.
9. "RADclock Project webpage." `http://www.synclab.org/radclock/`.
10. D. Veitch, J. Ridoux, and S. B. Korada, "Robust Synchronization of Absolute and Difference Clocks over Networks," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 417–430, April 2009.
11. Endace, "Endace Measurement Systems. DAG series PCI and PCI-X cards." `http://www.endace.com/networkMCards.htm`.
12. "Archipelago monitor locations." `http://www.caida.org/projects/ark/locations/`.
13. K. Vijayalayan and D. Veitch, "Rot at the Roots? Examining Public Timing Infrastructure," in *Proc. of IEEE INFOCOM 2016*, (San Francisco, CA, USA), April 10-15 2016.
14. N. Minar, "A Survey of the NTP Network," 1999. `http://alumni.media.mit.edu/~nelson/research/ntp-survey99/html/`
15. J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir, "Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks," in *Proc. IMC 2014*, IMC '14, (New York, NY, USA), pp. 435–448, ACM, 2014.