Network Timing, Weathering the 2016 Leap Second

Yi Cao*, Darryl Veitch*
*FEIT, University of Technology Sydney, Australia, Email: {Yi.Cao,Darryl.Veitch}@uts.edu.au

***Abstract*—We collect high resolution timing packet data from 459 public Stratum-1 NTP servers during the leap second event of Dec. 2016, including all those participating in the NTP Pool Project, using a testbed with GPS and atomic clock synchronized DAG cards. We report in detail on a wide variety of anomalous behaviors found both at the NTP protocol level, and in the detailed timestamp performance of the server clocks themselves, which can last days or even weeks after the event. Overall, only 37.3% of servers had Adequate performance overall.**

I. Introduction

Timekeeping is a vital service provided by computer operating systems. The operating system's *system clock* is software built on local hardware, which is synchronized through timestamp exchange, via the Network Time Protocol (NTP), to a reference timeserver over the network. For scalability, timeservers are organised in a hierarchy, where a *stratum* s timeserver itself synchronizes to a *stratum* $s-1$ server. Anchoring the system are the *stratum-1* timeservers, which have local access to reference hardware such as GPS or atomic clock.

Stratum-1 servers are typically assumed to be reliable and highly accurate. If this were not true, the impact would be potentially significant, since a good proportion of the world's computers are synchronized, ultimately, through accessing public stratum-1 servers. These are hosted typically by institutions such as NIST and USNO (USA) and NMI (Australia), research institutes, companies such as Google and Apple, and some universities.

Network timing distributes *Coordinated Universal Time* (UTC). This is a discontinuous time standard: jumps known as *leap seconds* are inserted (roughly every two years) 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. There are diverse distributed systems that rely on sub-second synchronization for their operation, for which confusion about an entire second could have serious consequences. The leap second of 2012 brought down sites such as Reddit, Yelp (YELP), LinkedIn and FourSquare, and Qantas' entire computer system. During the 2015 leap, some stock exchanges took the precaution of suspending trading. Indeed because of the cost of keeping software 'leap second safe', and the risks of failure, for many years the idea of abolishing them has been given serious consideration.

In this paper we examine the behavior of public stratum-1 servers during the leap second event of end-December 2016. Our goal is to discover how the servers weathered this disruptive event, and to compare against what one would expect, both from the protocol, and clock accuracy viewpoints.

We build on the approach developed in [6] for the end-June 2015 leap second to assess Stratum-1 servers from a remote measurement point. We expand on the earlier work in a number of ways. In terms of scale, we examine more servers (459 compared to 177), over a longer period of time (64 days compared to 22), and at far higher resolution (sampling period of 1 per second compared to 64). The longer time period allows a wider range of extreme behaviours to be captured, and the higher resolution allows the timing behaviour to be sampled closer to the leap second event itself. In terms of protocol analysis, whereas [6] offered a fairly terse account focussed on first and last leap warning times, here we examine the leap header bits comprehensively, relate this to stratum variability, and describe a complex landscape through defining important classes of behavior. In terms of clock analysis, we define two distinct quantities defining the leap behavior, the Time to Jump (ToJ), as well as the 'Time to Expected Behaviour' (TEB) criterion described in [6], use them to define a 'Leap Error Duration' (LED) metric, and link these to the protocol analysis.

Organization: Section II provides background on time standards, leap seconds and NTP, and discusses prior work. Section III describes our testbed, the selection of servers, what the datasets are and how they were collected. Section IV deals with the server clock methodology and performance, whereas Section V describes the protocol aspects. Overall results are given in Section VI, and Section VII concludes.

II. Background

A. Time Standards and Leap Seconds

The primary international time standard is the *Temps Atomique International* (TAI). It is based on combining the 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 $t_{\text{TAI}} = -10\,\text{s}$, best thought of as TAI to which jumps of exactly 1 second have been infrequently applied in order to keep UTC close (within $0.9\,\text{s}$) to UT1. Within UTC, a positive leap second manifests as a downward jump, slowing the clock down with respect to TAI. Negative leap seconds are defined but have never been used.

We focus on the leap second added at the end of December 31, 2016. The leap event was completed at 00:00:00 January 1st UTC, when TAI was $t^*_{\mathrm{TAI}} = 1483228837\,\mathrm{s}$, and $t^*_{\mathrm{UTC}} = t^*_{\mathrm{TAI}} - 37$. For convenience, we plot all timeseries against a timescale "t", which is t_{TAI} with its origin reset to t^*_{TAI}.

The Global Positioning System (GPS) defines its own atomic clock based continuous time scale. It is related to TAI simply as $t_{\mathrm{TAI}} - t_{\mathrm{GPS}} = 19\,\mathrm{s}$. Hence $t^*_{\mathrm{UTC}} = t^*_{\mathrm{GPS}} - 18$, and $t_{\mathrm{UTC}} = t_{\mathrm{GPS}} - 17$ during 2016.

B. 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 [5].

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*, or LI = 10 for a negative leap second. Obsoleted RFCs 1305 (NTP v3) and 4330 (SNTP v4) however, instead state ... *in the last minute of the current day*. As discussed in [6], the language in the RFCs is ambiguous. Common practice in implementations is to assume that warnings are issued in each response packet, beginning during the 24 hours prior to the leap. In this paper we adopt this as a definition of correct behaviour, though we refine it somewhat as described later.

A related issue is when LI should be reset back to the normal *no warning* value of LI = 00. NTPv3 states clearly what one would logically expect: '*... Immediately following insertion the leap bits are reset.*". However, strangely, NTPv4 is silent on this point. We return to this point when we examine warning behavior in detail in Section V-A.

The remaining value is LI = 11 (see Section V-C) which signals that the leap status is unknown because the server clock is *unsynchronized*. It is here that there is a direct connection with the (8 bit) Stratum field of the NTP header. According to the NTPv4 standard, a system clock (at startup or at any other time) should set both LI = 11, and S = 16, which also means unsynchronized. However in transmitted packets, the standard requires that stratum 16 is represented as S = 0. In practice we see a variety of stratum values of our ostensibly Stratum-1 servers, including 0 and even 16, as detailed in Section V-B.

C. 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, configuration problems, and so on. We refer the reader to Malone [6], [2] and references therein.

There have been a number of surveys on the NTP network, in particular those of Mills [3] (1989), Guyton and Schwartz [1] (1994), and Minar [4] (1999), who also surveys prior surveys. While these reviews provide interesting accounts on features of the timing network and its growth, there is very little peer-reviewed work on stratum-1 server behavior, and still less on leap seconds.

Apart from Veitch et al. [6] whose approach we follow, the most substantial work is that of Malone [2]. It provides a longitudinal study covering the leap second events from 2005 to 2015, and forms an interesting reference for the evolution of the NTP landscape over this period. The technical focus is on protocol aspects, in particular the values of the LI bits over time, for a subset of both Stratum-1 and Stratum-2 servers taken from the NTP Pool Project and *ntp.org* (see Section III-B). The study is quite coarse grained, with a probing resolution of one per hour, compared to one per second here. It does not examine the server clocks themselves, and, of course, like [6] does not report on the 2016 leap second. We compare against the findings of Veitch et al. and Malone in more detail in Section V-A.

III. The Experiment:

In this section we describe the experimental infrastructure, servers selection, the experiment itself and its data.

A. Testbed Hardware

Since the servers we will examine are stratum-1 servers, normally the root references for the Internet's timing system, an authoritative evaluation demands a reliable high quality local reference and sound methodology.

Our timing hardware is built around a roof mounted Trimble Acutime$^{\mathrm{TM}}$GG receiver. Via a custom built Timing Distributor system the Pulse Per Second (PPS) signal from the GG is passed at TTL levels to an Stanford Research Systems FS725 atomic clock in order to stabilize it on long time scales.

The PPS output of the atomic clock is fed back to the Distributor, which then provides PPS-only outputs at RS422 levels for DAG cards, and RS232 levels for our local server.

We use two PCs to host NTP clients which connect to the servers under test via a Gigabit Ethernet LAN. For each client, the NTP timing packets are copied by a dedicated fully passive tap. We use IX-TP-CU3 GigEth copper taps from NetOptics, which output each direction of the duplex GigEth link on a separate port.

To receive, store and timestamp the tapped packets we use high performance four-port Endace DAG 7.5G4 capture cards. These GigEth cards lock to the PPS input from the Distributor, and provide hardware timestamping with a 3-sigma accuracy of $100\,\mathrm{ns}$, and a quantization resolution of $7.5\,\mathrm{ns}$.

It is important that the DAG cards not themselves react to the leap second event. To ensure this the GG was configured to output time according to the GPS timescale, which does not follow leap seconds.

B. Server List

We assembled a list of servers from five sources:

Org: the public Stratum-1 URL list maintained at *ntp.org*
Pool: servers participating in the NTP Pool Project
LBL: servers caught in a capture at the LBL border router
Au: the set of Australian public facing servers
Misc: miscellaneous servers that seemed worth including.

From each source we assembled an initial *raw* list of IP addresses of servers which appeared to be Stratum-1. Close to the experiment launch, this was reduced to an *active* list by removing servers which did not respond to NTP client request packets.

For the Org servers, the raw list included all those Stratum-1 IPv4 servers appearing on the webpage on October 24th 2016. In the case of the Pool servers, the complete server list (for all ntppool zones) as at November 16th 2016 was downloaded from the NTP Pool Project website, and filtered for IPv4 Stratum-1 servers. The data set from the Lawrence Berkeley National Laboratory (LBL) mined a an extensive collection of NTP packet traces, covering the period from August 2014 to April 2016, for packets from IPv4 Stratum-1 hosts. The raw Au list includes, we believe, the majority of public servers in the country, to which we added a server in our laboratory, and five from the National Measurement Institute (NMI) which are not publically accessible, but to which we have access. Finally, the Misc list contains servers which appeared at *ntp.org* in the past, some servers suggested by Google, and others used by CAIDA's Ark monitoring network absent from other sources.

The resulting raw and active list sizes from each source are given in Table I. In a small number of cases no useful data was ultimately obtained with respect to the leap second event. The third row of the table shows the number of servers for which we actually have relevant data.

The final list of servers with data, which we call **ListLeap**, contains 459 unique servers. The breakdown according to source is shown in Figure 1. There is a large overlap, of around 45%, between the Org and Pool lists, and given these two sources, the LBL 'dragnet' brought in only an additional 82 servers. For this reason, and because of the popularity of the NTP Pool service which is global in scope, we speculate that **ListLeap** contains a significant percentage of the global public facing Stratum-1 server population.

For load balancing and other reasons described below, the aggregate active server list was split for the purposes of the experiment (see Table II). **List1** was allocated to Client 1 and consists mainly of the active Org servers. The remaining servers form **List2**, and were handled by Client 2.

	Org	Pool	LBL	Au	Misc
raw	325	283	589	15	14
active	202	268	265	14	11
with data	**197**	**258**	**257**	**14**	**10**

TABLE I
THE NUMBER OF SERVERS OBTAINED PER SOURCE.

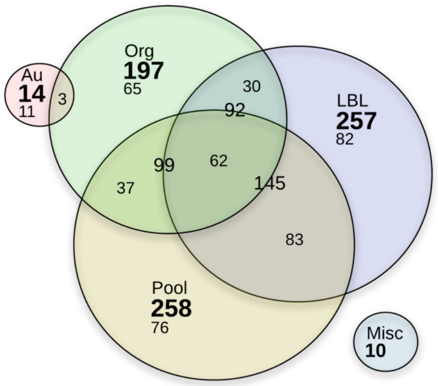

Fig. 1. A breakdown by source of the 459 unique servers in **ListLeap**. Bold denotes source totals.

Compared to the 459 servers studied here, for the 2015 leap second [6] reported on a set of 176 servers, consisting mainly of the Au servers and a subset of 156 Org servers. In that work only Org servers marked as OpenAccess were used, however as this information is often out of date or otherwise inaccurate, in this paper our raw list included the full Org list.

Client	#in-list	#with-data	start	end	days
1	229	223	29 Nov.'16	2 Feb.'17	64
2	250	236	14 Dec.'16	2 Feb.'17	49

TABLE II
THE EXPERIMENT WAS SPLIT OVER TWO CLIENTS.

C. Experiment and Data Sets

The goal of the experiment was to obtain a high resolution view of the response of **ListLeap** servers to the end-December 2016 leap second event. In order to establish a baseline of normal behavior, and also to allow for relevant events which may occur well away from the leap itself, a window of one month to each side was selected. As shown in Table II, Client 2's half of the experiment was launched late, due to hardware problems.

Two events resulted in a loss of data. On Dec. 20 the DAG 2 capture file became corrupted. By combining a backup with the ongoing capture file, the final damage was limited to a gap of around 3 hours for the servers in **List2**. On Dec. 21 power was accidently cut to the floor housing the testbed and the entire experiment had to be restarted, resulting in a loss of around 10 hours of data.

Each client launched independent instances of a request–response exchange deamon to each of the servers in its list in parallel, using a per-server customized polling period. Our default target was to poll every $\tau = 1$ second, a much finer grained sampling than the $\tau = 64$ s used in [6]. Since this is much smaller than the usual minimum value of 16 s for *ntpd* clients, it was possible our traffic would be blocked. We performed calibration runs in order to determine the smallest period out of $\tau = \{1, 4, 16, 64, 256, 1024\}$ seconds accepted by each active server. For the with-data servers the final result was

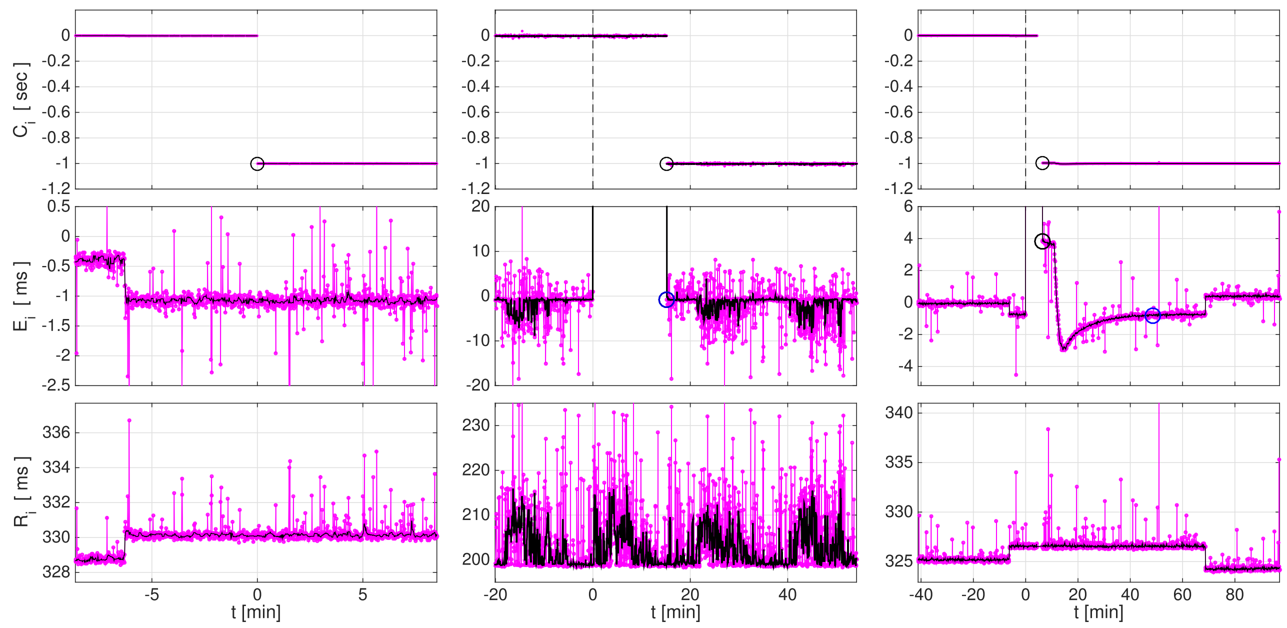

Fig. 2. Introduction to server behavior and the methodology for jump and TEB determination. Left column: Good server (ToJ = TEB = first sample past $t = 0$). Middle: a NotGood server as the leap is delayed (ToJ not at first stamp past $t = 0$), but is however Clean (ToJ = TEB). Right: A NotGood server which is delayed and not clean: post-leap instability resulting in TEB $>$ ToJ. The vertical dashed line marks the leap-second event in the rightmost two plots.

respectively $\{385, 25, 39, 10, 0, 0\}$ using the above periods. This resulted in a total of over 3.68 billion NTP packets collected over both clients.

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* $\{T_{a,i}, T_{b,i}, T_{e,i}, T_{f,i}\}$ of timestamps. Here $T_{b,i}, T_{e,i}$ are the (incoming and outgoing respectively) UTC timestamps 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. The DAG timestamps $T_{a,i}, T_{f,i}$ are on the GPS timescale. We convert them in postprocessing to a continuous 'pre-leap UTC' timescale via $t_{\text{UTC2016}} = t_{\text{GPS}} - 17$ (see Section II-A).

We found that only 5 active servers were running NTPv3 (all of which returned data), the remainder used NTPv4.

IV. Server Clock Behaviour

In this section we examine the behavior of the server clocks before and after the leap second. We look not only in the neighborhood of the leap second, but also over the entire trace to establish a detailed context. We adopt and extend the methodology of [6].

A. 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 $t = t^*_{\text{TAI}}$. The series C_i (resp. E_i) consists of errors in 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 independent of server timestamps and of the leap second event, to judge path conditions independently of server behavior.

We illustrate the methodology through the examples appearing in Figure 2. The server shown in 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 jump position (black circle) at $t = 0.59$s is at the first stamp past $t = 0$, and the previous stamp was at $t = -0.44$, before $t = 0$ as required. We call this observed delay the *Time to Jump* (ToJ). Here ToJ $= 0.59$s, being smaller than the sampling period $\tau = 1$, is consistent with a server that jumped at $t = 0$. The jump value is confirmed by 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 -1.0 ms rather than zero due to path asymmetry, not server error. The level shift event in E_i at around -6 min coincides precisely in position with a shift in R_i, but is smaller in magnitude. It is therefore entirely consistent with a path change which was asymmetric in terms of the outward and inward one-way delays, and does not imply any issue with the server.

The overlayed black curves are the sliding median filtered version of the underlying timeseries. This non-linear filtering preserves level shift transitions, making it very effective in this context, as it allows short term congestion effects to be greatly damped without interfering with jump behavior. We use a window size of $W = 9$ stamps.

The middle column exhibits a server where the jump occurs cleanly, but is 910 s late, and is far from the first stamp past the leap second, since ToJ $=$ 910s is orders of magnitude larger than the sampling period $\tau = 1$. Looking more closely, bursts of variability in E_i of magnitude $\approx$ 5ms can be seen, however these correspond closely with bursts in R_i of twice the magnitude, implying symmetric congestion rather than any leap-induced server anomalies. Hence the initial ToJ $=$ 910s value can be taken as the time at which the server timescale returned to the expected post leap-second behavior (blue circle in E_i). We call this latter delay the Time to Expected Behavior (TEB), and in this case the two are equal. More generally they are related as TEB $\geq$ ToJ.

The right column shows a server where not only is the initial jump late with ToJ $=$ 383.6s, 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 $=$ 48.6 min (blue circle). The R_i plot confirms that this oscillation then recovery event in E_i does not result from path effects but is associated with the leap event at the server, and also that the level shift afterward at around 69min is not a continuation of leap second effects, but is consistent with the level shift seen in R_i at that exact time.

The ToJ value is easy to determine because of the dramatic size, 1 second, of the target event. To do so we use a simple downward jump detection method, thresholded on 0.6s, after first median filtering the signal (recall this will not affect the jump position, nor its size). We define ToJ to be the first such jump looking forward from $t = 0$, and if this occurs at the first sample, we look backward from $t = 0$ to check if it began before the actual leap second. Such *pre-jumps* occured in 9 cases. This approach is robust, and only fails in cases where the jump itself is not uniquely defined, or not defined at all. There were 24 such *ToJ-indeterminant* cases which cover a variety of extreme behaviors including periodic jumps, continuous skew, chaotic jumping of magnitudes ranging from 1 to 10^8 seconds, and eternal 'staircases' where each step is a full second.

The case of TEB is far less straightforward. One reason is the very wide range of behavior one finds, there is no simple rule that can classify all of it. Another important reason is that many servers have recurrent anomalies which can also intersect the leap event. Our philosophy was to attempt to measure server reaction to the leap second itself, rather than to classify the server's timing quality more generally. Thus, if server behavior was apparently unaffected by the leap, then even if its behavior was continuously poor, its leap second behavior would still be classified as good. A third reason is non-local effects. In many cases the immediate neighbourhood of the leap appears very clean (apparently ToJ $=$ TEB), easily passing automated tests over a predefined windows (even quite wide ones), but when looking much further ahead, 'aftershocks' of diverse forms are seen. These can have large amplitudes and continue for long periods, for example some servers reverse their leap and return to the UTC2016 timescale after a few hours, before finally settling down. In these cases we would set ToJ to the first jump, and TEB to just after the final aftershock.

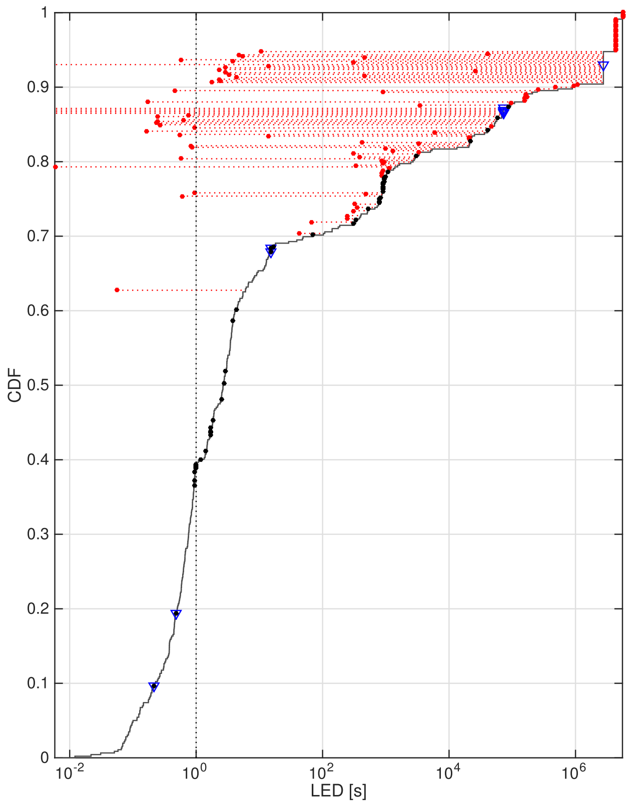

Fig. 3. The CDF of LED, with black dots highlighting servers that are not Good but Clean. To each NotClean server a red line is drawn connecting it to its jump value, thus visualizing $[\text{ToJ}, \text{LED}]$. Pre-jump servers are highlighted with blue triangles, and ToJ-indeterminate servers are red in the top right.

Each server was closely examined using the above approach to determine (ToJ, TEB) values which genuinely reflected a leap-second related jump and subsequent recovery, 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 median filtered E_i following a leap fell below the magnitude of the path noise as revealed by R_i, 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 too difficult to determine. Such servers were then classified as TEB-indeterminant. If a server was ToJ-indeterminant, then is it automatically TEB-indeterminant.

B. Results

From the measured (ToJ, TEB) we define *Leap Error Duration* (LED) as LED $= \max(0, \text{TEB}) - \min(0, \text{ToJ})$. LED is positive even for pre-jump servers, and for others LED $=$ TEB.

The empirical CDF of LED is reported in Figure 3. The range of values is such that a log scale is required. At the

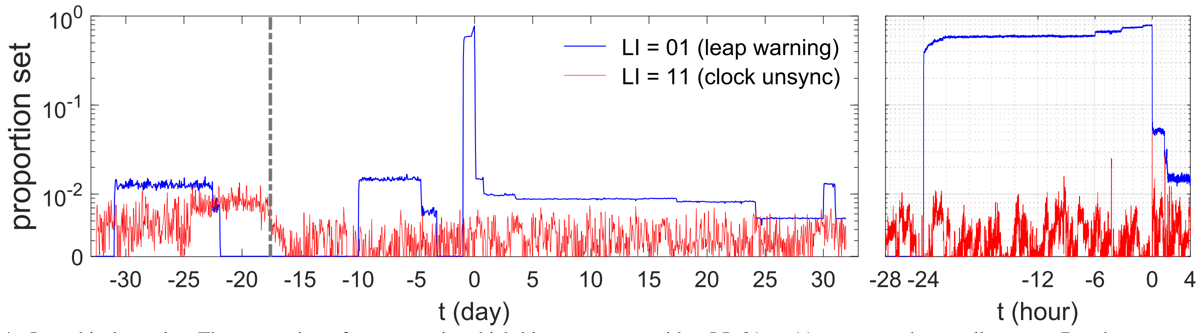

Fig. 4. Leap bit dynamics. The proportion of responses in which bits were set to either LI=01 or 11, aggregated over all servers. Results are aggregated into time bins 1 hour wide in the left plot (covering the entire experiment), and 16s wide in the zoom about the leap second at $t = 0$ on the right. The dashed grey line at -17.5 days shows where Client 2's experiment began.

extremes, for ToJ-indeterminant servers the value is set to the experimental duration (49 or 64 days), and for the other TEB-indeterminant servers the time of the last packet is used.

We partition servers which are not ToJ-indeterminate into 3 natural classes, following the examples from Figure 2 from left to right. We classify a server as Good if both ToJ and TEB fall on the first stamp past the leap second. This corresponds to no evidence of incorrect behavior. A total of 305 servers out of 459 (66.4%) were found to be Good. This definition respects the limitations of sampling and so is not based on the *size* of ToJ. In some cases servers do not respond in the vicinity of the leap second, so a Good server can sometimes have a large ToJ (though in an availability sense this is still not ideal behaviour, even if no erroneous timestamps have actually been observed). We call a jump *Clean* if ToJ = TEB. Clearly Good servers are Clean.

Our second class is servers which are Clean, but not Good. There are 48 (10.0%) of these. They are highlighted by black dots in Figure 3 and only start appearing, not surprising, from around 1 second. There are a wide variety of jump delays exhibited in this category, some very large. The final class are servers which are NotClean, that is ToJ $<$ TEB. There are 82 (17.9%) of these, highlighted in Figure 3 via the red dot and line which visualizes the $[\text{ToJ}, \text{LED}]$ interval for such servers. It is apparent that there are a variety of different sub cases for servers with large TEB, where the jump can be delayed either not at all, slightly, or severely. This class includes the 20 TEB-indeterminant servers that are not ToJ-indeterminant.

Of the 9 pre-jump servers, 4 come from Google servers which implement a slewing strategy over an interval $[-10, 10]$hr. Two other servers where found implementing slewing strategies, but each was considerably delayed.

V. Protocol Behavior

To begin, we wish to gain a general idea of when servers are setting the LI bits, and to what value. Figure 4 provides a view of this aggregated across servers, but preserving the critical temporal dimension. Since response rates vary, we plot the proportion of responses which have bits set, rather than the total number, in each of bin of size 1hr and 16s in the main and zoomed plot respectively (note that Malone [2] plots rather the proportion of *servers* with at least one response set per bin).

In the case of warnings, we see low levels in general except during the 24hr preceeding the leap second, and shortly after it. However there is also some sustained activity of low volume at other unrelated times. In the case of unsync, we see that they are always relatively rare (note the log scale), but not as rare as one would expect, and surprisingly only show a small increase in activity around the leap second. The plot does not show results for LI=10 because, remarkably and happily, no warnings of negative leap seconds were found in the entire data set.

To investigate further we look at the individual server level. We begin by looking at leap warning responses, because these are the most closely related to leap second events, even if they appear far from the (centred) leap second event time $t = 0$. In contrast, unsync responses could occur following any server reboot, and strata can vary for many reasons.

A. *Warnings : LI=01*

Looking at a scatterplot of warning packets across time for all servers, a wide variety of behaviors was seen. The following definitions help to extract useful structure from this picture. Note that the arrival time of a response packet i is taken to be $T_{f,i}$, which is after the timestamp is actually made. In this section we avoid misclassification by ignoring any stamp whose DAG times straddle an interval boundary.

Good
i) all warnings fall within $[-24\text{hr}, 0)$
ii) all responses between first and last warnings are warnings
iii) first warning arrives before $t = -3 \times 1024 = 3072$ s.

Ok
i) all warnings fall after $t = -24\text{hr}$
ii), iii) as for Good.

Extreme (Ex)
Warnings outside $[-24\text{hr}, 0] > 1\%$ of total responses.

None
No warnings at all.

The intent of Good is to capture servers which are both NTPv4 compliant, and behave as one would expect. Condition

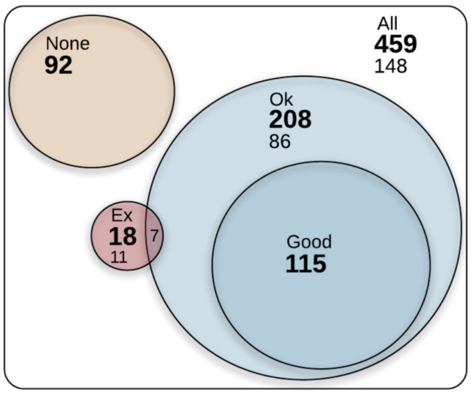

Fig. 5. Relationship of the LI warning classes.

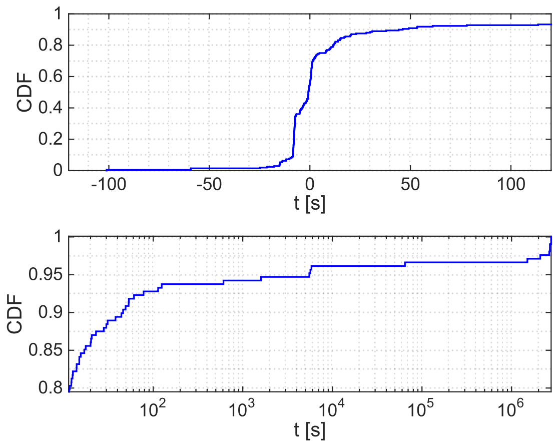

Fig. 7. CDF of Ok warning end times. Top: the body of the distribution including the minimum. Bottom: a logarithmic view of the full tail.

(iii) ensures warnings are sent in time, even for clients with large polling period of $\tau = 1024$s suffering a consecutive pair of lost responses. Because NTPv4 does not disallow warnings after the leap second, Ok relaxes Good to forgive this. The Ex class captures servers sending enormously more warnings than expected. The threshold of 1% was based on inspection of the CDF of the warning to total response ratio, and corresponds to some 54,000 warnings over the trace duration.

The relationship between the classes is shown in Figure 5. We see that 93 of the 208 Ok servers (45%) send warnings after the leap second (compared to under 13% in [6]), but of these only $|\text{Ok} \cap \text{Ex}| = 7$ send an extreme number. Overall, only 208 out of 459 servers (45%) are NTPv4 compliant, and the Good servers form only 25%. Of the 251 servers that are NotOk ($\text{NotOk} = \text{All} \backslash \text{Ok}$), 92 send no warnings at all, 11 are Extreme, and the remainder in general would be Ok, except for a small number of warning sent prior to $t = -24\text{hr}$. These latter would presumably not induce any undesirable effects, and could be included in a more generous definition of a (non-compliant) Ok.

Of the 5 NTPv3 servers, 1 was in Ex, 3 in None, and 1 was Ok. None were Good, the required class for compliance.

To gain more detailed insight, we now look into the temporal behavior within the Ok class. The histogram of Figure 6 shows that most servers begin warnings close to $t = -24\text{hr}$ as expected, but a fair percentage start only later, in particular 4% at the last hour. This is also visible in the zoom on Figure 4.

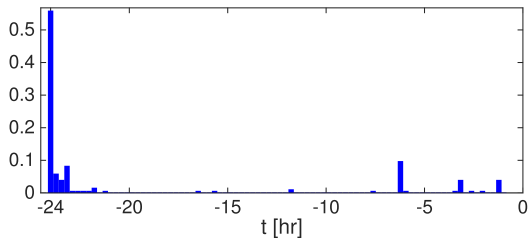

Fig. 6. Histogram of Ok warning start times.

This effect has been noted by Malone [2] and [6], and reflects the variety of implementations in service.

Now consider the times at which the Ok servers ceased their leap warnings. From the CDF tail (lower plot) of Figure 7, 95% had ceased after an hour, but of the remainer, 5 servers still sent a warning by the end of the trace. Two servers sent their last warning right at the end of January, which could be significant in terms of implementation behavior (the experiment was continued into February to allow such observations).

B. *Stratum*

We nominally expect the **ListLeap** servers to be S1 (Stratum 1) whenever they are available. In practice a wide variety of behaviours are seen. The following classes are illuminating and provide background for Section V-C.

Constant
Only one stratum value is ever seen.
Bi
Only two stratum values are ever seen.
Unsync
At least one response is S0 (i.e. $\text{St} = 0$, stratum unsync).
Extreme (Ex)
Servers whose ratio of stratum changes to total responses, outside the interval $[-24\text{hr}, 0]$, is more than 0.1%.

Figure 8 gives the relationships between these classes and their populations, subdivides the Constant class according to stratum, and shows the 11 servers with stratum 16 responses (these should not occur) as triangles. The Constant servers are ideal in that they have no stratum unsync (S0) responses, neither due to background system issues, nor leap second related. The Ex class have very unstable strata, a background effect which has nothing to do with the leap second.

C. *Unsync : LI=11*

The value $\text{LI} = 11$ is set, not in response to leap second events, but if the clock becomes unsynchronized. We do expect

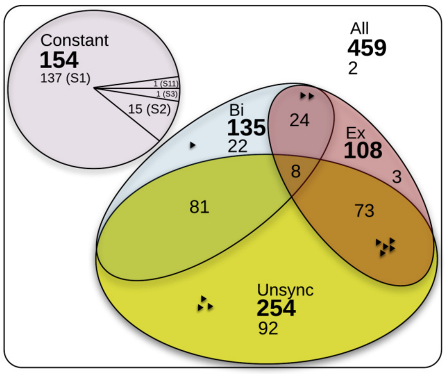

Fig. 8. Relationship between the Stratum classes.

synchronization loss about the leap second in some cases, and this is visible as the modest, thin spike in Figure 4 at $t = 0$. Furthermore, the standard suggests that unsynchronised clocks should result in responses with both LI $= 11$ and Stratum$= 0$ (S0) set. To explore this we define the following classes.

None
No reponses are set to LI unsync.

Extreme (Ex)
Servers whose ratio of LI unsync reponses to total responses, outside the interval $[-24\text{hr}, 0]$, is more than 0.001%.

S0Agree
Servers where each response with LI=11 has Stratum 0 and vice versa (and assuming at least one unsync response).

The Ex class threshold is at an abrupt change point in the empirical CDF of the ratio, and translates to roughly 2 reponses per day. Most servers in the class exhibited extremely high unsync rates, the worst 10 or so of which are essentially responsible for the background rate seen in Figure 4.

The diagram of Figure 9 overlays the Stratum Unsync class over the above LI11 classes. The LI and Stratum indicators of unsync do correspond closely, in particular perfect agreement is seen in $|\text{S0Agree}| + |\text{None} \setminus \text{Unsync}| = 116 + 148 = 264$ cases (57.5%). Furthermore, for each server in Unsync\None, every S0 response has LI11 set, however outside S0Agree there are extra LI11 responses. Perfect disagreement is found in the 6 servers in None$\cap$Unsync (S0 with no matching LI11),

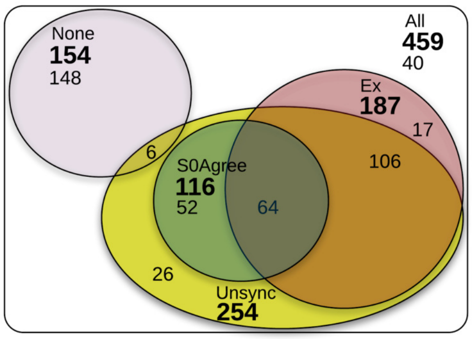

Fig. 9. Relationship between the LI11 classes and stratum Unsync.

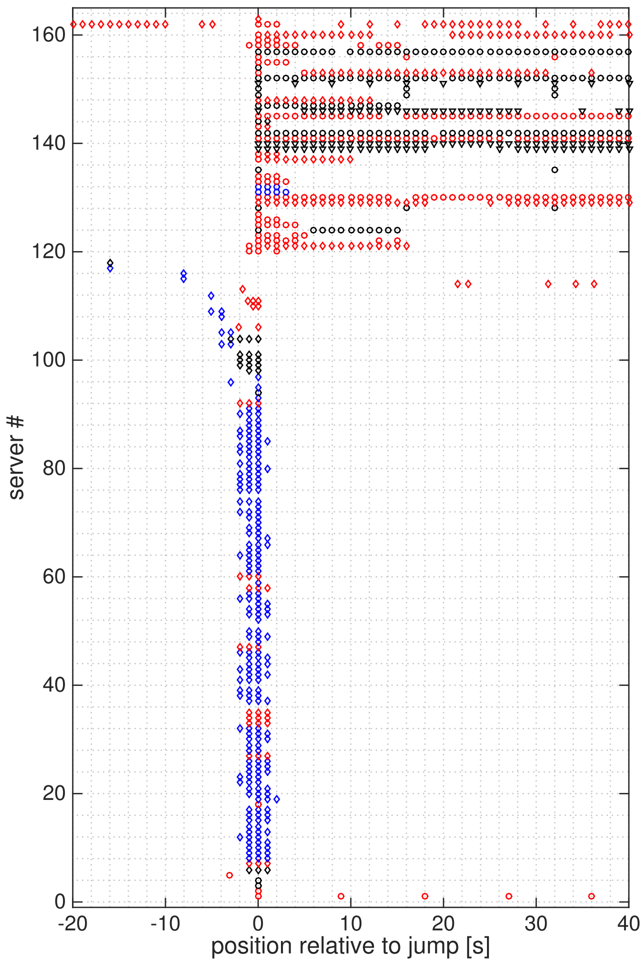

Fig. 10. Jump-centred scatterplot showing unsync responses. Circle: both LI11 and S0 set; Diamond: LI11 only; Triangle: S0 only. Clock classes of (Good, Clean-but-NotGood, NotClean) shown as (blue, black, red).

and the $57 = 40 + 17$ servers in the white area and Ex\Unsync (LI11 with no S0).

We now focus on synchronization loss related to the leap second by restricting to servers with unsync responses (LI11 and/or S0) in a window of $[-1\text{hr}, 1\text{hr}]$ about $t = 0$. There are 163 such, which cut across the S0Agree, Ex, and 'perfect disagree' classes. Figure 10 provides a zoomed scatterplot of such servers showing each unsync response. Since we have observed that the LI11/S0 behavior follows the observed jump point rather than $t = 0$, in this plot the jump points of all servers have been aligned. The servers are ordered according to the (hidden) ToJ value, with the earliest jump at the bottom. The Clock-Good servers (blue) mainly show a small number of LI11 unsync reponses only (no S0, diamonds), both before and after the jump, and do not have members in Ex. The Clean but NotGood servers (black) and the NotClean servers (red) on the other hand, have many more unsync responses, mainly occurring after the jump only (except for servers in Ex), and they are a mixture of S0Agree (circles) and LI11-only behavior. When looking over the full window and beyond,

Server Set	#	Protocol					Clock			Overall	
		LI01		LI11		Stratum	Good	CleanNG	NotClean	Ideal	Adequate
		OK	Good	None	S0Agree	Const(S1)					
ListLeap	459	208	115	154	264	137	305	48	82	**36 (7.8)%**	**171 (37.3)%**
Pool	258	123	76	79	145	61	168	31	49	19 (7.4)%	97 (37.6)%
Org	197	83	33	60	106	54	128	26	39	11 (5.6)%	69 (35.0)%
Au\Lab	13	10	3	7	9	6	13	0	0	2 (15.4)%	10 (76.9)%
NL	66	37	17	25	34	35	**45**	10	9	10 (15.2)%	**32 (48.5)%**
ListLeap\NL	393	171	98	129	230	102	**260**	38	73	26 (6.6)%	**139 (35.4)**%
USNO (US)	15	8	5	8	10	10	9	3	3	4 (26.7)%	6 (40.0)%
NIST (US)	13	10	3	2	4	10	12	0	1	0	10 (76.9)%
VNIIFTRI (RU)	7	2	2	2	3	2	2	0	4	2 (28.6)%	2 (28.6)%
NMI (AU)	5	5	2	5	5	4	5	0	0	2 (40.0)%	5 (100)%
NICT (JP)	4	2	1	2	4	2	2	2	0	1 (25.0)%	1 (25.0)%
SP (SE)	4	0	0	0	0	4	0	4	0	0	0

TABLE III
PERFORMANCE BREAKDOWN PER SERVER SOURCE. CLEANNG MEANS "CLEAN BUT NOT GOOD".

unsync responses continue for longer in the NotClean case, as expected. For each of the NotClean servers the TEB values fall outside the range of the chosen zoom, however we observed that in all cases unsync responses ceased before the TEB value, and well before in all but a few cases.

VI. OVERALL RESULTS

We conclude by combining the results above into an overall evaluation, including a breakdown over server subsets of importance, including the Org and Pool sources, and the 66 National Laboratory (NL) servers contained in **ListLeap**.

The following definitions provide measures of desired behavior integrating across all protocol and clock aspects of direct relevance to leap second performance. They refine and extend the 'Perfect' definition of [6].

Ideal
Clock-Good & LI01-Good & LI11-None & St-Constant(S1).

Adequate
Clock-Good & LI01-Ok \ St-Constant(not S1).

Table III displays the results. Overall, only 37.3% of servers are Adequate compared to the 61% reported in [6] for the analogous 'Perfect' category. As expected, NL servers, at 48.5%, do better than the non-NL servers at 35.4%, but the gap is far less than one might hope. From the clock point of view alone only 45 NL servers (68.2%) are classed as Good, not far above the 56.6% of non-NL servers. The Pool servers outperform Org servers in both the Adequate and Ideal categories, but the difference is marginal. In the lower part of the table the six most populous NLs are given. Here the NMI achieves a perfect Adequate score and the best Ideal rating. In constrast NIST is the 2nd most Adequate, but the least Ideal.

VII. CONCLUSION

We have reported on the behavior of a substantial number of public facing Stratum-1 servers as they passed through the end-December 2016 leap second event and beyond. Using a testbed with authoritative timing, and high resolution probing, we were able to examine and judge the clock performance of these servers as well as their leap second related protocol performance, in a unified way.

Our main finding is that in each of the leap second related criteria: Clock behavior, LI warnings, and unsync state in the vicinity of the clock's jump, large minorities of servers are performing inadequately, or worse. A complex variety of erroneous behaviors are displaying across the dimensions of clock jump position and quality, timely leap warnings (only 45.3% NTPv4 compliant), stable synchronization state, and stratum stability. Overall, only 37.3% of servers have Adequate performance, and only 7.8% could be classed as Ideal over the entire experiment. Even among National Laboratories the performance is not very much better. Moreover, a significant minority of servers have extremely poor behaviour, including 106 servers (23.0%) which either have no clear jump at all (ToJ-indeterminate), else aftershock behavior which extends well beyond the initial jump (Unclean). From the point of view of our methodology their behavior is usually flagrant and easy to detect, however most clients would be unable to distinguish timing problems arising from most such servers, from other effects such as variable path conditions. The case for abolishing leap seconds is supported by these findings.

ACKNOWLEDGMENT

Partially supported by Australian Research Council's Linkage Projects funding scheme #DP170100451. Many thanks to Vern Paxson for his work in providing the LBL dataset, and Ask Hansen at The NTP Pool Project for his help in recovering the list of NTP Pool servers. Thanks also to Greg Dowd and Michael Wouters for their insights.

REFERENCES

[1] J. D. Guyton and M. F. Schwartz. Experiences with a survey tool for discovering network time protocol servers. In *Proc. USENIX Summer Conference*, pages 257–265, 1994.

[2] D. Malone. The Leap Second Behaviour of NTP Servers. In *Proceedings of the Traffic Monitoring and Analysis workshop*. IFIP Digital Library, April 7-8 2016. http://tma.ifip.org/2016/#program.

[3] D. L. Mills. On the accuracy and stablility of clocks synchronized by the network time protocol in the internet system. In *Computer Communication Review*, number 1, pages 65–75, 1989.

[4] N. Minar. A Survey of the NTP Network, 1999. http://xenia.media.mit.edu/ nelson/research/ntp-survey99/ntp-survey99-minar.ps.

[5] NIST. ftp://time.nist.gov/pub/leap-seconds.3629404800.

[6] D. Veitch and K. Vijayalayan. Network Timing and the 2015 Leap Second. In *Proc. of PAM 2016*, Heraklion, Crete, Greece, March 31 - April 1 2016.