What time is it? Managing Time in the Internet

Sathiya Kumaran Mani[†], Paul Barford[†], Ramakrishnan Durairajan[#], Joel Sommers[*]
[†]University of Wisconsin - Madison [#]University of Oregon [*]Colgate University

ABSTRACT

In this paper, we report on our investigation of how *current local time* is reported accurately by devices connected to the internet. We describe the basic mechanisms for time management and focus on a critical but unstudied aspect of managing time on connected devices: the time zone database (TZDB). Our longitudinal analysis of the TZDB highlights how internet time has been managed by a loose confederation of contributors over the past 25 years. We drill down on details of the update process, update types and frequency, and anomalies related to TZDB updates. We find that 76% of TZDB updates include changes to the Daylight Saving Time (DST) rules, indicating that DST has a significant influence on internet-based time keeping. We also find that about 20% of updates were published within 15 days or less from the date of effect, indicating the potential for instability in the system. We also consider the security aspects of time management and identify potential vulnerabilities. We conclude with a set of proposals for enhancing TZDB management and reducing vulnerabilities in the system.

CCS CONCEPTS

• **Networks → Time synchronization protocols**;

KEYWORDS

DST; Local Time; Time Zone Database; TZ; TZDB;

1 INTRODUCTION

The modern world is intrinsically tied to the concept of *time*. Day-to-day activities in virtually all aspects of life have some notion of a start time, a duration and an end time, each of which are essential for scheduling and most importantly for coordination between participating entities. As such, time sources that are accurate, consistent and reliable are essential in our society.

ANRW '19, July 22, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6848-3/19/07…$15.00
https://doi.org/10.1145/3340301.3341125

The importance of time has led most governments around the world to set up centralized entities (*e.g.,* the National Institute of Standards and Technology in the US [16]) to specify time standards and to maintain highly precise clocks that serve as reference sources. A natural question to ask is, how do I ensure that my personal clock is coordinated, in a reasonable way, with the time specified on a national reference server? In an age when most people get their time from electronic devices such as computers, smartphones, smartwatches, etc., the answer lies in a set of processes that have been in place and largely unchanged for many years.

In this paper, we investigate how time is coordinated and reported by devices connected to the internet. Our study considers the basic components and processes for time management and focuses specifically on the issue of reporting *current local time*. One manifestation of the importance of current local time is coordinating appointments for people in different locations. How does one know what time it is in a location that is half way around the world? and how are users assured that connected devices, applications running on those devices or cloud services report current local time accurately? To answer those questions, we drill down on a critical, but relatively unstudied aspect of managing time in the internet: the time zone database (TZDB) [14]. The TZDB, operated under the aegis of IANA, is a historical repository that reflects time zones established by governments around the world, Coordinated Universal Time (UTC) offsets for each time zone, and daylight saving time rules. In short, the TZDB is one of the primary mechanisms for connected devices world-wide to report current local time.

The processes that have evolved for maintaining the TZDB consist largely of periodic updates made by a loose confederation of contributors to reflect a new time policy in a local jurisdiction. Our understanding of these processes is enhanced by analysis of the mailing list archive that is specified as a primary mechanism for maintenance and management of the TZDB [31].

Our analysis of the TZDB itself is focused on assessing its stability, integrity, and security. We begin by analyzing how time zone details in the TZDB have changed over the past 26 years. During that period 2,283 updates have been made to the repository, including updates that reflect historical changes in time zones. We find that updates are bursty, but are distributed in a relatively uniform fashion over the years and across the various time zones. While the reasons for

changes vary, the majority of updates reflect changes in daylight saving time (DST) policies. Reasons for DST changes include political elections, local events and religious events. We also find that approximately 20% of updates are modified within 15 days from their date of effect, indicating the potential for societal disruption. One potential catastrophic failure of the TZDB would be either a malicious entity making a change or an unintended update being used by one of the major operating systems. We find several problems related to TZDB updates resulting in errors and software bugs. Some of the problems are caused by uncertain information released by administrative entities and delays in distributing the updates to end users. Finally, we consider the processes for maintaining the TZDB from an adversarial perspective and show that several aspects of update and maintenance are vulnerable to attack.

Our analysis of the TZDB leads us to propose a number of updates to the current system and process that we argue will enhance its integrity and security. First, we recommend a formalization of the process for maintaining the TZDB to ensure that it is sound and secure. Second, we recommend that all updates and the TZDB itself are cryptographically signed to ensure authenticity. Third, we recommend the implementation of an audit process to assure the integrity of the TZDB.

2 BACKGROUND

Time zones originated largely due to the need to standardize *current local time* in order to facilitate coordination of transportation (railway) and communication (telegraph) networks that became commonplace in the late 19^{th} century [19, 27, 43]. Multiple clocks often had to be installed in stations—each calibrated to a given rail company's notion of current local time—in order for customers to be able to make sense of different timetables. The threat of the United States Government intervening to simplify the situation led the rail industry to create *standard rail time* in 1883, a precursor to the time zones in the United States today [19]. Time zones (both industry and government-established) became common within Europe and North America by the end of the 19^{th} century [43]. The arrival of World War I caused the United States to seek to conserve energy through creation of daylight saving time (DST) in 1918, an idea first proposed by New Zealander George Hudson in 1895 and which was being adopted in other parts of the world [19, 43]. A side-effect of establishing DST was that the United States established official time zones within its borders, superseding the railroad industry time zones.

2.1 The Timezone Database

The time zone database (TZDB) project was created by Arthur David Olson in the early 1980s to facilitate timekeeping on computer systems and to provide standard programming APIs to deal with time zones [17, 31]. The database consists of text files (generally one for each continent) with *zone definitions* and *rules*. Zone definitions indicate names of time zones, their offset from the Greenwich Prime Meridian reference, and an indication of a date and time at which the zone ceased to be valid (if applicable)[47]. Current names of time zones typically include the name of a region and the largest city (by population) within the time zone referenced [44]. For example, the following is a historical zone record indicating that local mean time in New York City was offset from Greenwich Mean Time (GMT) by -4:56:02 up until November, 1883[1]:

```
Zone America/New_York -4:56:02 - LMT 1883 Nov 18 12:03:58
```

The most recent release of the TZDB (2019a) specifies 348 time zones in the world.

Rules within the TZDB indicate when the offset for a time zone may change depending on daylight saving time [44, 47]. For example, the current daylight saving rules in effect in the United States are shown below. They indicate when time zones in the US change and by how much, among other details:

```
Rule  US  2007  max  -  Mar  Sun>=8  2:00  1:00  D
Rule  US  2007  max  -  Nov  Sun>=1  2:00  0     S
```

Moreover, the TZDB contains C sourcecode for compiling the database into binary datafiles, as well as reference implementations of C API functions and utility programs that can be used to access information within the TZDB, *e.g.*, the `zdump` tool, which accepts a time zone name and prints the current local time for that zone. On modern UNIX-based systems, database files are typically installed in `/usr/share/zoneinfo` for use by API functions and utilities.

The TZDB was placed explicitly in the public domain by Arthur David Olson in 2009; it is not "owned" by any internet-related authority. However, the *process* by which the TZDB is updated has been specified in RFC 6557 (BCP 175) [31], with current releases of the database being hosted by IANA [14]. RFC 6557 makes explicit that the primary maintainer of the database (currently Paul Eggert of UCLA) is empowered to make any appropriate changes to the database and should consider the views expressed on the TZ mailing list [18]. In practice, it is up to the primary maintainer to reach consensus with mailing list participants about any changes.

When a new release of the database is created, notification is made on the TZ mailing list and files are updated at a canonical location [14]. It is then up to any consumers of the TZDB (*e.g.*, hardware and OS manufacturers, programming library maintainers, etc.) to incorporate the latest versions in their software. The timing of updates is thus critical, since there may be substantial delay between discussion of potential changes to the database to actual incorporation of changes

[1]The dash after the offset indicates that New York City did not observe daylight saving time.

into an OS or software library [30, 31]. For example, in 2015 the Turkish Government did not officially decide to delay an impending daylight saving time change to allow more daylight hours for polls during an election until about three weeks before the change. Although the TZDB was modified soon after the change became official, it took additional time for OS manufacturers to release software updates to account for these changes, *e.g.*, an updated version of Apple's iOS was not released until *three days* before the election. The result was mass confusion over what time it was [30]. While the relatively new time zone distribution service protocol [23] is designed to help reduce update latency between software manufacturers and the installed client base, the extent of its current deployment (if any) is unclear, and it only addresses one aspect of update delay, thus it is not clear how much the distribution protocol would have helped in this instance.

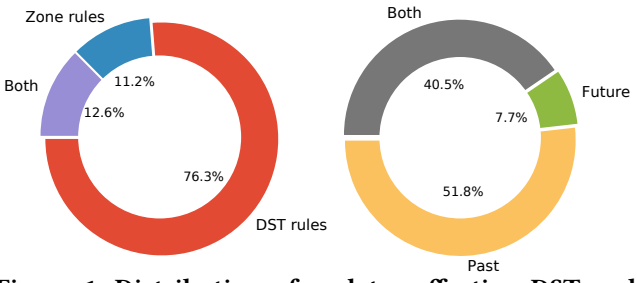

Figure 1: Distribution of updates affecting DST and zone rules (left) and past & future timestamps (right).

It is important to note that the maintainers of the database make no claims regarding accuracy, especially for historical zones and rules [17, 31] before 1970. Correctly handling dates and times in software is notoriously difficult [28], and historical time zone information is included (and continues to be updated) in order to help software developers correctly deal with dates and times in the past. The maintainers of the database also do not make any claim for the database being authoritative, since they rely on some awareness (mediated through the TZ mailing list) of potential adjustments to time zones in the world in order to incorporate any changes. For any changes made to database, comments typically appear in the source files with hyperlinks or other references to justify the changes. A number of other historical notes are also included, making each database file a rich source of information about the evolution of time zones throughout the world [51]. Indeed, the majority of lines of each database source file is made up of comments (*e.g.*, 2331/3487, or 67% of the lines of the `northamerica` source file are exclusively comments in the most recent release of the database, `2019a`).

2.2 Data Used for Analysis

We used two primary data sources for the analyses in this paper. First, we used the TZ database source files from the available 240 releases[2] of the database for analysis [14]. Second, we downloaded the entire TZ mailing list archive for our analysis [18]. The mailing list archives span a time period of 33 years (Nov 1986–May 2019) while the database releases span a time period of 26 years (1993–2019). Each release is named after the year of its release and an alphabetic character serially assigned to releases in that year (*e.g.*, 1995a). The most recent version of the database contains 348 time zone records.

3 TZDB ANALYSIS

3.1 A Maintenance Perspective

To understand the evolution of the TZDB and, specifically, time zones and DST rules around the world, we analyze all releases of the TZDB. Since the mailing list is also the primary source for distributing TZDB releases, we use the email archives to identify the announcement corresponding to each release and extract the release timestamps.

We built a Python-based TZDB parser tool to process the zone and DST rules associated with each time zone. The parser is also capable of detecting the changes in the effective zone and DST rules between consecutive releases. We call such changes *updates* to the TZDB. Using the parser, we identify 2,283 updates to the zone and DST rules across all of 240 TZDB releases over the past 26 years. Our tool also labels 427 updates as "correction updates", which are amendments to previous updates. We discuss correction updates in §3.4.

Categorizing database updates. We take two approaches to characterize the TZDB updates identified by our parser. We begin by identifying updates that change zone rules or DST rules or both. Figure 1 (left) shows that about 76% of TZDB updates make changes to the DST rules highlighting the influence of DST on managing current local time on connected devices.

Next, using the release dates extracted from the announcement emails in the email archive, we identify updates that affect timestamps in the past, the future, or both. For each zone update, we calculate the time ranges the updates affect. Given this time range and the release timestamp, we identify whether the update affects past timestamps, future timestamps, or both *i.e.*, time ranges that straddle the release timestamp. This distribution is shown in Figure 1 (right). The figure shows that *a majority of the release updates affect timestamps in the past, indicating the efforts that go into maintaining the historical accuracy of the database.*

Assessing update timeliness. Since we are able to identify the updates from each release that affect future timestamps, we can calculate the number of days between the date of the release and the affected time range in the future. This helps to characterize the timeliness of the releases and

[2] *In this study we consider only releases that contain time zone data (tzdata) and exclude missing and time zone code-only releases.*

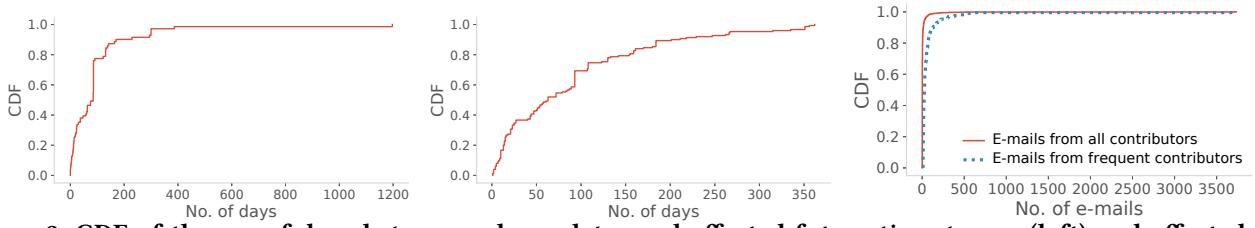

Figure 2: CDF of the no. of days between release dates and affected future timestamps (left) and affected past timestamps (middle) and CDF of the no. of emails sent by each contributor (right).

Figure 2 (left) shows the CDF of this distribution. From the figure, we observe that ~80% of the updates are announced within 100 days from their date of effect. Moreover, about 20% of the updates are announced within 15 days or less from their time of effect, *highlighting the potential for societal disruption due to time lag for incorporation in connected devices.* Similarly, since we see affected time ranges that straddle the release date, we examine those updates to see how far in the past the corrections are made. We remove updates that make corrections beyond a year in the past to eliminate updates that make historic changes. The CDF of the remaining updates shown in Figure 2 (middle) indicates that only 40% of the updates make changes to timestamps within 50 days in the past.

3.2 A Community Perspective

Since the TZDB spans the history of the commercial internet and beyond, obtaining a perspective of the maintenance and administrative activities is crucial for understanding how the loosely-organized group of contributors have maintained this critical asset. Since the mailing list is the primary means of communication among the contributors (§2), we begin by examining the email archives published by IANA [18] to assess behaviors within the community. From the email archives, we calculate the number of unique contributors[3] and the number of emails exchanged by the TZDB community. We find 1,891 unique contributors sent 19,367 emails over the span of 33 years, with an average of 56 messages every month. *The relatively large number of contributors is a potential concern from a management perspective.*

Figure 3 shows the number of unique contributors actively communicating through the mailing list every month. The increasing trend, particularly after the 2012 adoption of TZDB hosting by IANA, suggests a growth of interest and visibility for the database. Similarly, an increasing trend is seen in the number of monthly email messages as shown in Figure 3. We posit that the increasing trends are correlated with the increasing usage of the database, particularly due

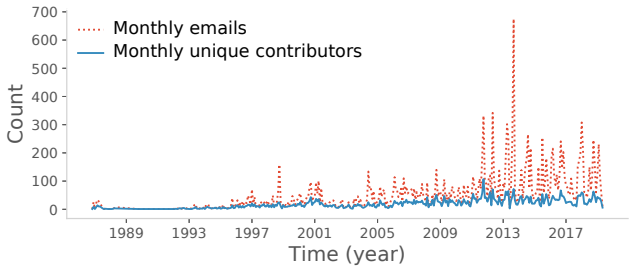

Figure 3: Number of monthly unique contributors and emails sent over the span of 33 years (1986–2019).

to the widespread adoption of mobile/smart devices around the same period [9].

To gain an insight into the effort required to maintain the TZDB, we calculate the CDF of the number of email messages sent by each contributor (Figure 2 - right). Not surprisingly, the top two contributors are the current TZ Coordinator (Eggert) and the founder of the database (Olson). We observe that ~90% of the contributors have sent less than 50 messages (each) throughout the history of TZDB, indicating that *though a large number of contributors participate in reporting errors, making administrative changes to DST and time zone rules from around the world, the database is maintained only by a small clique of contributors.* Figure 2 (right) also shows the CDF only for contributors who have sent more than the average number of messages. We see that even in this subset, about 90% of contributors have sent less than 100 messages.

3.3 A Geo-Political Perspective

While the community and maintenance perspectives provide a baseline for understanding the practical aspects of the TZDB update process, we posit that the *reasons for DST rule changes are often administrative* due to governments changing time zone rules for reasons including elections [30] or large local events such as games [13] or religious reasons [10]. To evaluate this hypothesis, we analyzed rule change frequency. For this analysis, we count the number of rule changes each year for every time zone, the results of which reveal several interesting geo-political events in history, thus providing supporting evidence for our hypothesis.

To perform the rule change frequency analysis, we use the most recent release of the database (*i.e.*, 2019a) and built a parser to count the number of rule changes (both zone and DST rules) each year for every time zone. When we

[3] *Our manual inspections of the archives reveal that the mailing list is mostly used to discuss updates or modifications to time zone and DST rules, instead of queries about using the system. Hence the number of unique email addresses is a reliable estimate of the number of unique contributors participating in maintenance activities.*

generate a histogram, the years with large number of rule changes appear as spikes in the plot. Since we count the rule changes for every time zone, national and regional events that affect multiple time zones are more prominent and easily identifiable. To eliminate noise, we use a simple filtering technique where we use one standard deviation from the mean value as a threshold and consider only spikes that are larger than the threshold value. Moreover, to identify potential historical events that correspond to the spikes in the histogram, we group the contributing time zones by country and look at the history of countries with the highest number of contributions for the year in question.

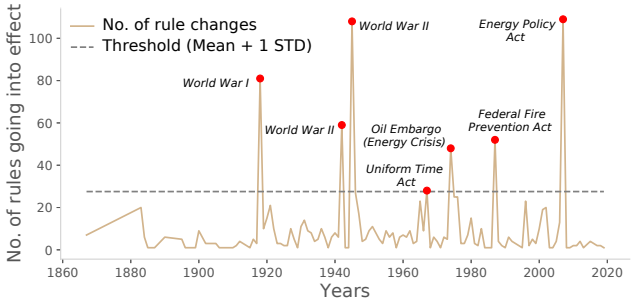

Figure 4: Histogram of rule changes in North America labeled with major geo-political events.

From the plots for North America (Figure 4) and other continents (not shown due to space constraints), we observe time zone updates flagged in our analysis coincident with wars, financial crises, sporting events, etc. In this way, the time zone database provides a unique perspective on historical events. Combining the results from our rule change frequency analysis with information in the database comments to identify interesting world events is part of our ongoing work.

3.4 Problems related to TZDB updates

We describe several anecdotes to illustrate the anomalies and problems related to TZDB updates. To identify such anomalies, we use a combination of techniques involving the use of our parser to label updates that make corrections to previous updates, filtering email archives using key words, and manual inspection of the filtered emails and corresponding comments in the database files. Automating this process using natural language processing algorithms to identify all issues across the data corpus is part of our ongoing efforts.

Correction updates. We find that about 19% of the updates include changes to previous updates, with 26% of such correction updates affecting timestamps before 1970 and the rest of the updates affecting timestamps after 1970. Pre-1970 correction updates highlight the efforts involved in maintaining the historical correctness of the database. While some of the post-1970 correction updates are issued to fix incorrect information as explained below, we find that most correction updates arise largely from uncertainties. For example, the government of Bangladesh announced the adoption of DST without mentioning the end date, prompting the use of a nominal end date in the 2009o release of the database [5]. This information was later changed using a correction update when the end date was officially announced. Such examples highlight the problems that arise due to incomplete information released by administrative entities.

Errors. We find instances of updates that fix incorrect information in the database that were identified and reported by contributors. One example is the instance when a proposed US presidential election year DST policy helped cause a time zone error [1] in 1992. The proposed policy (not yet signed into law) was included in the TZDB with a non-descriptive name ('US/Pacific-New') and resulted in several Unix systems on the US west coast failing to switch out of DST as expected. Similarly in recent years, we see several errors in time zone information for countries such as Chile [2], Pakistan [4], Syria [6] and Jordan [3] that were later fixed by the contributors. These errors highlight the problems with the informal update process.

Software bugs. We also find instances of time zone updates resulting in software bugs in several popular software packages and distributions. For example, an update to introduce negative DST offsets for the time zone 'Europe/-Dublin' in 2018, ended up breaking several packages such as OpenJDK, Joda-Time and ICU (International Components for Unicode) [15]. Similarly, we see updates causing errors in packages such as Qt [12] and user bugs in OS distributions such as Ubuntu [7]. These examples underscore the impact of the TZDB and the errors that might be caused due to incorrect or erroneous updates.

Delayed updates. Another class of problems arise due to delays in distributing TZDB updates to user devices. As mentioned in §2, delays in pushing the updates to end users have resulted in several issues such as the issues with Android and iOS users in Israel [8] and Turkey [30] and has also resulted in incorrect time display in Google search results [11].

4 RECOMMENDATIONS

In this section, we discuss the implications of our analysis of the TZDB in Section 3 and make a number of recommendations for TZDB management and maintenance that are focused on assuring accuracy and enhancing the integrity of the system. It is important to be clear that the intention of this discussion is not to impugn the individuals who have contributed significant time, energy and expertise to the maintenance and upkeep of the TZDB. Rather, we hope to expand the perspective on this very important task in a way that will ultimately improve the system.

We do not provide an implementation since many of our recommendations could be followed using standard open-source tools such as Bugzilla. We also do not discuss methods

for distributing timezone information. A variety of distribution methods are possible and RFC 7808 describes a timezone data distribution service while explicitly noting that it does not address how contributions are made to the TZDB. Hence our recommendations are complementary to RFC 7808. Our recommendations are also intentionally high level—details would need to be fleshed out within the TZDB community.

While it can be argued that the current process for maintaining the TZDB as described in RFC 6557 [31] has "worked", we posit that technology trends such as cloud services, self-driving vehicles, Internet of Things and mobile-edge computing expand the scope of risks of errors or uncertainty in the TZDB. Furthermore, the lack of explicit security measures exposes vulnerabilities in the TZDB to malicious actors seeking to disrupt or deny service. To address these risks, we recommend the following enhancements.

Codification of TZDB update process. As described in RFC 6557, the current process for updates to the TZDB relies entirely on the TZ Coordinator and the TZ mailing list. We believe it is critical to maintain a community-based support and release process that is managed by a Coordinator. This approach has been effective for other large software systems such as Linux kernel development, and is also a good model for the TZDB. We recommend a *formalization* of the process for maintaining the TZDB to ensure that it is sound, that updates are available when required, that updates are appropriately tested, and that consistent documentation is provided. Key aspects of the proposed formalization would include a standard release cycle, standardized documentation of changes, a ticketing system for each step in the process, and results from tests of the TZDB standard codebases.

Secure the TZDB update process. Our examinations of RFC 6557, the TZDB and the TZ mailing list indicate that there is little in terms of formal security measures. Indeed, the entire process relies on the integrity of the contributors, the TZ Coordinator and the repository for the TZDB, associated code and documentation. The implication is that a motivated attacker or *e.g.,* a government entity intent on disenfranchising certain groups may find specific vulnerabilities or utilize the current processes to facilitate malicious or unwarranted updates to be inserted in the TZDB. We posit that there are three general types of threats: *(i)* impersonation of a TZDB contributor or local authority, or the TZ Coordinator; *(ii)* compromise of the TZDB update process; and *(iii)* compromise of the TZDB itself.

As noted above, RFC 6557 states that "Moving forward, the TZ database, supporting code, and any appropriate supporting information SHOULD be cryptographically signed prior to release using well known public keys" [31]. Minimally, we recommend that the update process be enhanced so that only authorized and properly credentialed contributors are allowed to make updates, and that all TZDB updates, the database itself, supporting code and documentation MUST be cryptographically signed. Again, we appeal to the Linux release process in which all releases are signed using Open PGP-compliant signatures which are easily tested by third parties. Recommendations for more comprehensive security measures will be considered in future work.

Audit TZDB updates. To ensure the integrity of the TZDB releases, we recommend the implementation of a standard audit process. This audit would be conducted by a third party separate from the update contributors but in coordination with the TZ Coordinator. The audit would be conducted from the perspective of users in all time zones to ensure that current local time is consistent with the current standard for that area. The results of the audit would be documented and included along with the other documents maintain in the TZDB repository.

5 RELATED WORK

The evolution of clock synchronization fueled the development of protocols for internet-connected devices such as Network Time Protocol (NTP) [38–40, 42, 46] and its variants (*e.g.*, Simple NTP (SNTP) [41], MNTP [36], Precision Time Protocol (PTP) [29], RADClock [52], etc.) as well as protocols for non-traditional networks [20, 22, 32, 33, 37, 50]. Studies have also examined the security implication of protocols [21, 24, 34, 35, 45], their usage [26, 48], and their application to other areas such as latency estimation [25] and event detection [49]. Our work is complementary: while all these aspects of clock synchronization are important and are the focus of prior efforts, to the best of our knowledge, *ours is the first study of current local time and the TZDB.*

6 SUMMARY

In this paper, we examine the process that enables *current local time* reporting on internet-connected devices. The focal point for this process is the TZDB repository that codifies regulations for time zones around the world and is used in all major computing operating systems. We analyze the longitudinal aspects of the TZDB and find that updates are bursty and that the number of updates per month is generally increasing. We find that updates are spread widely over time zones and that most reflect changes to daylight saving time and that many of these are revised relatively soon after they are made. Our detailed analysis of the TZ mailing list and TZDB releases reveals several problems related to corrections and delays in publishing updates, resulting in errors and software bugs.[4] Based on our findings and consideration of RFC 6557 we make three recommendations for enhancing the TZDB maintenance process that are intended to ensure it's accuracy, integrity and security.

[4] *We have setup a public repository with all the data used in our analyses and the code used to generate the figures and results discussed in the paper at: https://github.com/satkum/tzdb_analysis*

ACKNOWLEDGEMENTS

We thank Paul Eggert for his helpful comments and feedback. This work is supported by NSF CNS-1703592, DHS BAA 11-01, AFRL FA8750-12-2-0328. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DHS, AFRL or the U.S. Government.

REFERENCES

[1] 1992. US presidential election year politics help cause time zone bugs. https://mm.icann.org/pipermail/tz/1992-October/009164.html. (1992).
[2] 1995. Chile's time zone data wrong. https://mm.icann.org/pipermail/tz/1995-October/009422.html. (1995).
[3] 2009. Jordan DST out by a day since 2002. https://mm.icann.org/pipermail/tz/2009-April/015519.html. (2009).
[4] 2009. Pakistan data incorrect? https://mm.icann.org/pipermail/tz/2009-October/015845.html. (2009).
[5] 2009. proposed time zone package change for Bangladesh. https://mm.icann.org/pipermail/tz/2009-April/015519.html. (2009).
[6] 2009. Syria end of DST error in tz. https://mm.icann.org/pipermail/tz/2009-October/015900.html. (2009).
[7] 2011. Is this a bug in the upstream or just Ubuntu (10.04LTS). https://mm.icann.org/pipermail/tz/2011-February/016697.html. (2011).
[8] 2013. Cellphone tz updates often not happening in Israel. https://mm.icann.org/pipermail/tz/2013-September/020361.html. (2013).
[9] 2013. How widely used is the tz database? https://mm.icann.org/pipermail/tz/2013-March/018846.html. (2013).
[10] 2014. Morocco DST Interruption during Ramadan 2014. https://mm.icann.org/pipermail/tz/2014-June/020971.html. (2014).
[11] 2016. Google search results displaying wrong time for Egypt. https://mm.icann.org/pipermail/tz/2016-September/024074.html. (2016).
[12] 2016. QTimeZone mishandles tzdata 2016b and later in Russia, Kazakhstan. https://bugreports.qt.io/browse/QTBUG-53071. (2016).
[13] 2018. Australian Time Zone changes for Olympics. https://github.com/eggert/tz/blob/master/australasia#L1351. (2018).
[14] 2018. IANA - Time Zone Database. https://www.iana.org/time-zones. (2018).
[15] 2018. Irish Standard Time vs Irish Summer Time. https://mm.icann.org/pipermail/tz/2018-January/025854.html. (2018).
[16] 2018. National Institute of Standards and Technology. https://www.nist.gov/. (2018).
[17] 2018. Time zone database and code. https://github.com/eggert/tz. (2018).
[18] 2018. tz – Time Zone Database discussion. https://mm.icann.org/mailman/listinfo/tz/. (2018).
[19] I.R. Bartky. 2015. *Selling the True Time: Nineteenth-Century Timekeeping in America.* Stanford University Press.
[20] N. Chirdchoo, W. Soh, and K.C. Chua. 2008. MU-Sync: A Time Synchronization Protocol for Underwater Mobile Networks. In *WuWNeT*.
[21] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir. 2014. Taming the 800 pound gorilla: The rise and decline of NTP DDoS attacks. In *Proceedings of ACM IMC*.
[22] H. Dai and R. Han. 2004. TSync: A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks. *SIGMOBILE CCR* (2004).
[23] M. Douglass and C. Daboo. 2016. Time Zone Data Distribution Service. https://tools.ietf.org/html/rfc7808. (March 2016).
[24] B. Dowling and D. Stebila. 2016. Authenticated Network Time Synchronization.. In *Usenix Security*.
[25] R. Durairajan, S.K. Mani, P. Barford, R. Nowak, and J. Sommers. 2018. TimeWeaver: Opportunistic One Way Delay Measurement via NTP. In *ITC - Teletraffic in a Smart World*.
[26] R. Durairajan, S.K. Mani, J. Sommers, and P. Barford. 2015. Time's Forgotten: Using NTP to Understand Internet Latency. In *ACM HotNets*.
[27] P. Glennie and N. Thrift. 2011. *Shaping the Day: A History of Timekeeping in England and Wales 1300–1800.* Oxford Scholarship Online.
[28] Z. Holman. 2018. UTC is Enough for Everyone, Right? https://zachholman.com/talk/utc-is-enough-for-everyone-right. (May 2018).
[29] IEEE. 2008. IEEE 1588 Precision Time Protocol (PTP), Version 2 Specification. (March 2008).
[30] M. Johnson. 2016. On the Timing of Time Zone Changes. https://codeofmatt.com/2016/04/23/on-the-timing-of-time-zone-changes/. (April 2016).
[31] E. Lear and P. Eggert. 2012. Procedures for Maintaining the Time Zone Database. https://tools.ietf.org/html/rfc6557. (February 2012).
[32] J. Liu, Z. Zhou, Z. Peng, J. Cui, M. Zuba, and L. Fiondella. 2013. Mobi-Sync: Efficient Time Synchronization for Mobile Underwater Sensor Networks. *IEEE TPDS* (2013).
[33] F. Lu, D. Mirza, and C. Schurgers. 2010. D-sync: Doppler-based Time Synchronization for Mobile Underwater Sensor Networks. In *WuWNet*.
[34] A. Malhotra, I.E. Cohen, E. Brakke, and S. Goldberg. 2016. Attacking the Network Time Protocol.. In *NDSS*.
[35] A. Malhotra, M. Van Gundy, M. Varia, H. Kennedy, J. Gardner, and S. Goldberg. 2017. The security of NTPâĂŹs datagram protocol. In *International Conference on Financial Cryptography and Data Security*. Springer, 405–423.
[36] S.K. Mani, R. Durairajan, P. Barford, and J. Sommers. 2016. Mntp: Enhancing time synchronization for mobile devices. In *Proceedings of ACM IMC*.
[37] S.K. Mani, R. Durairajan, P. Barford, and J. Sommers. 2018. An Architecture for IoT Clock Synchronization. In *Proceedings of the 8th International Conference on the Internet of Things*.
[38] D.L. Mills. [n. d.]. Network Time Protocol. https://www.ietf.org/rfc/rfc958.txt. ([n. d.]).
[39] D.L. Mills. [n. d.]. Network Time Protocol (Version 3): Specification, Implementation and Analysis. https://www.ietf.org/rfc/rfc1305.txt. ([n. d.]).
[40] D.L. Mills. 1991. Internet time synchronization: the network time protocol. *IEEE Transactions on communications* 39, 10 (1991), 1482–1493.
[41] D.L. Mills. 1995. Simple Network Time Protocol (SNTP). https://tools.ietf.org/html/rfc1769. (March 1995).
[42] D. Mills, J. Martin, J. Burbank, and W. Kasch. 2010. Network Time Protocol Version 4: Protocol and Algorithms Specification. https://tools.ietf.org/html/rfc5905. (June 2010).
[43] V. Ogle. 2015. *The Global Transformation of Time: 1870–1950.* Harvard University Press.
[44] A.D. Olson. 2009. Theory and pragmatics of the tz code and data. https://data.iana.org/time-zones/theory.html. (May 2009).
[45] S. Park, A. Shaik, R. Borgaonkar, and J. Seifert. 2016. White rabbit in mobile: Effect of unsecured clock source in smartphones. In *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*. ACM.
[46] J. Postel and K. Harrenstien. 1983. Time Protocol. https://tools.ietf.org/html/rfc868. (May 1983).
[47] B. Seymour. 2015. How to Read the tz Database Source Files. https://data.iana.org/time-zones/tz-how-to.html. (October 2015).
[48] J.A. Sherman and J. Levine. 2016. Usage analysis of the NIST internet time service. *Journal of Research of the National Institute of Standards and Technology* 121 (2016), 33.

[49] M. Syamkumar, S.K. Mani, R. Durairajan, P. Barford, and J. Sommers. 2018. Wrinkles in Time: Detecting Internet-wide Events via NTP. In *IFIP Networking*.

[50] A.A. Syed, J.S. Heidemann, et al. 2006. Time Synchronization for High Latency Acoustic Networks.. In *IEEE Infocom*.

[51] J. Udell. 2009. A literary appreciation of the Olson/-Zoneinfo/tz database. https://blog.jonudell.net/2009/10/23/a-literary-appreciation-of-the-olsonzoneinfotz-database/. (October 2009).

[52] D. Veitch, S. Babu, and A. Pàsztor. 2004. Robust Synchronization of Software Clocks across the Internet. In *ACM IMC*.