DRAFT

Aggregating Atomic Clocks for Time-Stamping

T. Saidkhodjaev (UMD), J. Voas (NIST), R. Kuhn (NIST), J. DeFranco (PSU), and P. Laplante (PSU)

Abstract

A timestamp is a critical component in many applications, such as proof of transaction ordering or analyzing algorithm performance. This paper reports on a method called Verified Timestamping (VT) that improves the standard timestamp protocol. VT was developed at the National Institute of Standards and Technology (NIST) for use in algorithms where timestamp accuracy is critical. VT is an aggregation of the outputs from various atomic clocks to create a Timestamping Authority (TsA). The motivation for this research effort included malicious delay issues in Networks of Things [NIST SP 800-183] as well as race conditions associated with the inclusion of new blocks into blockchains. This paper presents the TsA design and the results of VT, which indicate that atomic clock aggregation is not only possible, but a viable means to produce higher integrity timestamps at the ms level of performance. Tests showed that this is sufficient to preserve event ordering, using only a conventional PC with no dedicated connection or specialized hardware.

1.0 Introduction

The term "timestamping" refers to marking the time when a certain event occurred, such as when a message was sent or received. Unfortunately, computer clocks are generally not precise enough for some types of data, such as in a financial transaction where it is essential to determine if a stock purchase occurred during the period of a particular price, for example. In the current environment of high frequency trading, accuracy on the order of µs is needed for audit and market surveillance by regulators (SR FINRA 2016-005). European Commission regulations announced in 2016 for the accuracy of business clocks require clock granularity of one µs or better, with a maximum allowed divergence from Coordinated Universal Time (UTC) of no more than 100 µs (Annex, Directive 2014/65/EU). Similar regulations from the U.S. Financial Industry Regulatory Authority have an even tighter clock synchronization requirement with no more than a 50 µs (SR FINRA 2016-005) divergence. General purpose computer clocks are not a good choice when accurate time is needed as they are known to have poor accuracy, with a possible drift of 5 s to 15 s per day (Lombardi, 2019). The Network Time Protocol (NTP) and atomic clocks are routinely used to synchronize actions and provide more accurate time for internet applications. However, even NTP and atomic clocks are not precise enough for some applications such as those that use blockchain technology where timestamp verification and transaction order is critical (Stavrou & Voas, 2017). In addition, a timestamp that is trusted and verifiable is needed to accurately sequence the blocks for blockchain consensus protocol (Kuhn et al., 2019). The purpose of this project is to propose a general purpose system called Timestamping Authority (TsA), which would be able to provide reliable, real-time, high precision timestamps preserving the order of events regardless of their number and time when they were processed. In this paper we will discuss the TsA motivation and design with blockchain technology as a use case.

2.0 Background

The most accurate and precise time is kept by atomic clocks, which measure the frequency of oscillations of atoms that happens to be very close to a constant number. For most applications, *accuracy* refers to the closeness of measurements to a particular true value (such as a physical location), while *precision* is the closeness of measurements to each other. UTC, as its name implies, involves *coordination* among a set of cooperating systems. In the context of timekeeping, accuracy refers to traceability to UTC, while precision is the degree of synchronization among a set of clocks.

As noted previously, financial regulators require highly accurate timestamps, with UTC traceability. Within financial systems, continuous monitoring of clock synchronization stability is required, with realtime comparisons to UTC references. Global Positioning System (GPS) signals are typically used to provide UTC traceability to a national institution such as NIST, the Research Institutes of Sweden, or others. UTC combines time scales known as International Atomic Time (TAI) and Universal Time (UT1), which rely on a weighted average time from

450 atomic clocks in 70 nations (for TAI) and observations of rotations of the Earth (for UT1). To reduce potential vulnerability to GPS jamming or spoofing, financial institutions may employ specialized high-power encrypted signal transmission to protect systems providing UTC traceability. NTP is commonly used to provide synchronization of clocks within variable-latency packet switched networks. For NTP implementations, atomic clocks are connected to servers that deliver time over the network. These servers in turn are connected to other servers, and so on (Mills, 2003). This tree-like design (Figure 1) is needed to reduce the load on the top-level servers.

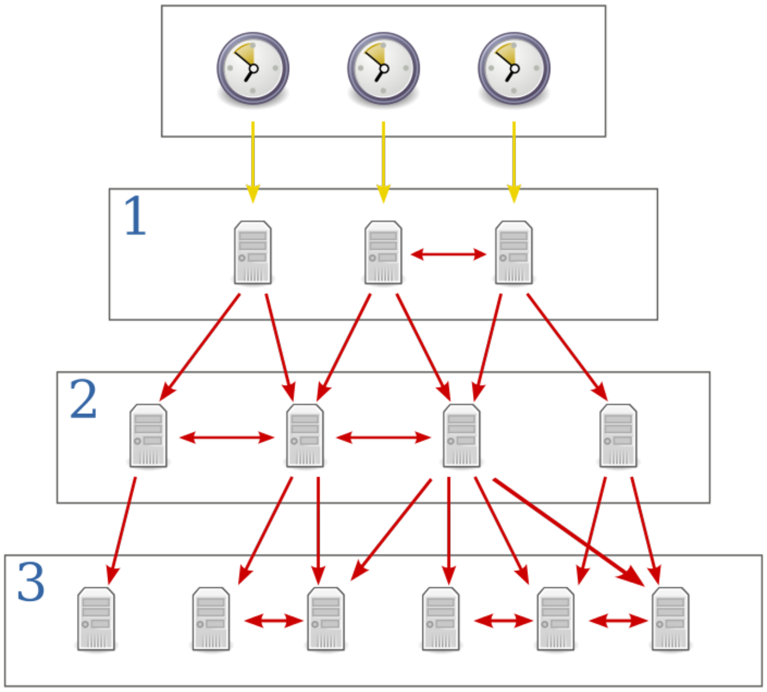

Figure 1: NTP Network Structure.

Yellow arrows indicate a direct connection; red arrows indicate a network connection.

This protocol operates over User Datagram Protocol (UDP), which is a transport layer network protocol. NTP version 3, used in the project, was standardized as Internet Engineering Task Force RFC-1305 (Mills, 1992), and is compatible with the latest version 4, RFC-7822 (Mizrahi & Mayer, 2016). The typical device might use one of the bottom-layer NTP servers to synchronize time. However, using the bottom layer can be insufficient in some situations, since some precision is lost on the way from the top. Improving precision becomes extremely expensive as the required level increases. The cause of this imprecision is due to the asymmetric network routes and network conditions (Mills, 2012).

A blockchain is an example of an application where timestamp accuracy is critical for verification and security. A blockchain is a series of timestamped immutable records that are managed by multiple computer entities. In order to add a new block to the chain, the candidate that wants to insert the block needs to perform a computation (e.g., calculate a hash of the concatenation of the previous block hash and a hash of the current data) and submit the result to the system for verification. The need for accurate timestamps comes into play when there are multiple candidates at the same time. In this case, only the candidate that finishes the hash calculation first will be added. All other candidates must start over with the new last block. This is a time consuming and expensive process. Figure 2 illustrates the addition of a timestamp using the basic methodology where only one candidate will succeed in being added to the chain and all others will have to start the hash calculation over – creating a type of race condition.

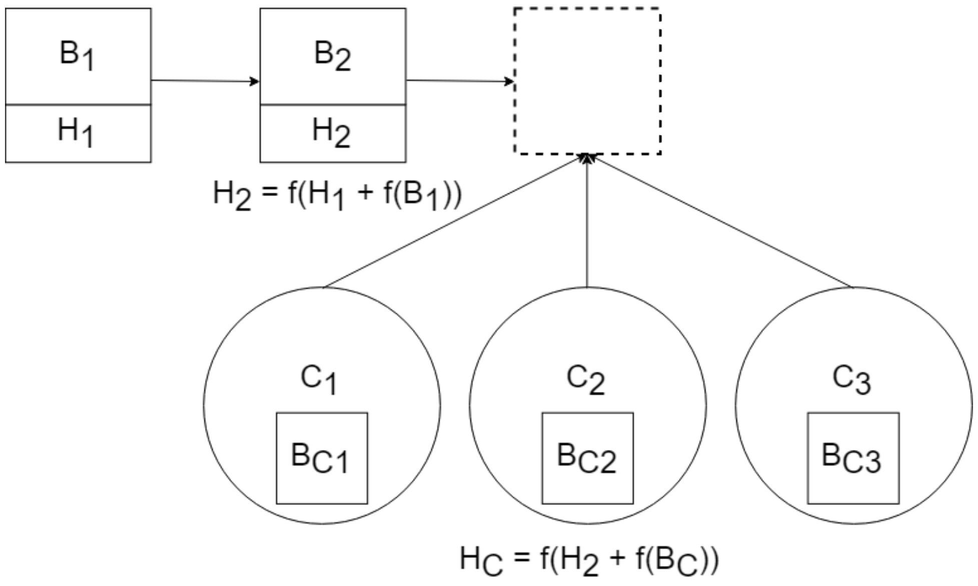

Figure 2: Adding a block to the blockchain with a basic timestamp
(B – block; H – hash; f - hash function; C – candidate)

A possible solution to this problem is integrating a single TsA service into the system as shown in Figure 3.

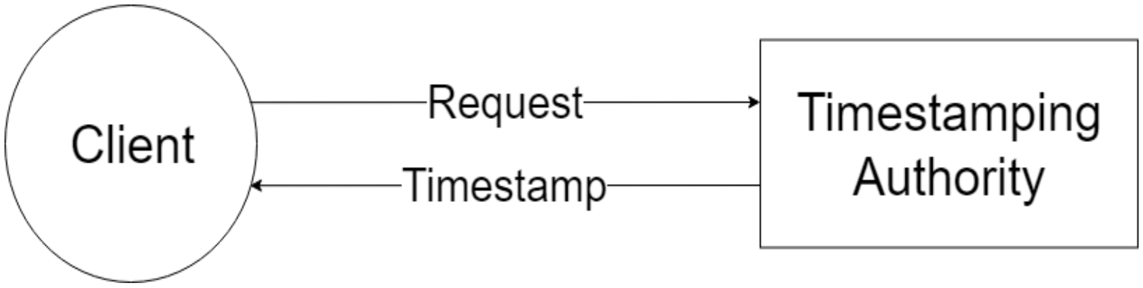

Figure 3: A Timestamping Authority Service

A TsA service could provide an accurate time on demand over a network. However, this does not resolve the situation where events happen simultaneously. As mentioned earlier, transaction order is critical in many applications. Figure 4 shows a single timestamping authority with the requests coming sequentially, thus keeping the accurate order of the requests. However, this design does not scale well and would work only for transactions that occurred geographically close to the TsA.

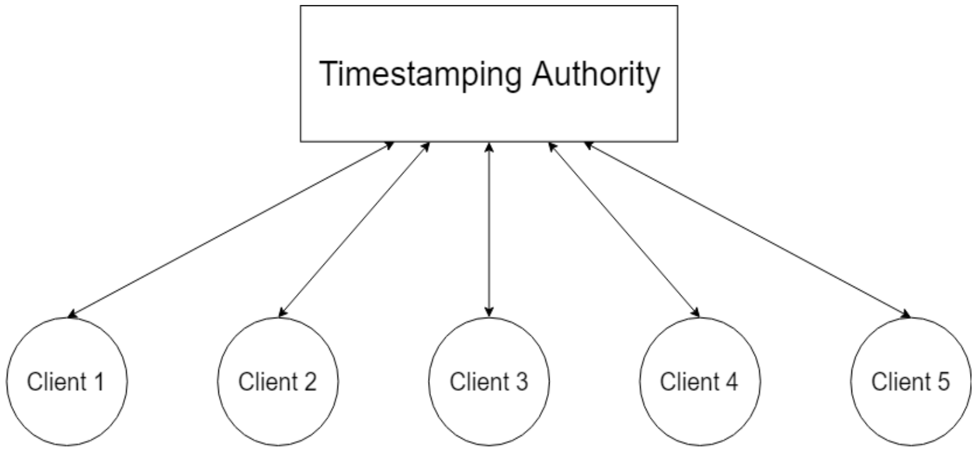

Figure 4: A single timestamping authority

A scalable timestamping solution would use several TsA servers, but then the question of time synchronization between them arises. An open specification called Chainpoint uses blockchain technology and the NIST randomness beacon to tackle this problem. A Chainpoint client stores a hash of the NIST randomness beacon and the timestamp on the blockchain:*Chainpoint (Timestamp + hash(random beacon))* → *blockchain*

The NIST randomness beacon (beacon.nist.gov) produces a purely random 512-bit string every minute. Every random value is then stored in a blockchain and can be verified at any time.

Inserting the random number facilitates proving the authenticity of the timestamp at any minute on the timeline. This approach will both guarantee transaction order and provide verifiability to the timestamps. However, the time resolution of Chainpoint timestamps depends on the randomness beacon, which is one minute, so we need to look for a different approach.

3.0 TsA Design

To ensure accurate time and order, when a time request is made, the timestamp will be determined by using an aggregation from several atomic clocks. Specifically, the atomic clocks that are used as time sources for this project have publicly available NTP servers, which are synchronized to the clocks, thus providing precise time to the clients (see Sect. 3.1 for details of this process). The communication with the client also works over NTP. NTP can provide up to 1 ms precision under ideal network conditions, and the precision deteriorates as asymmetric network routes are introduced (see RFC-1305, Appendix F). When a network route is asymmetric, the time for a request to get to the server is different from the time to get the response. This situation usually happens when long-distance requests are made or when the network condition is unsatisfactory. Since only the nearest atomic clocks are used, to provide high precision, NTP also accounts for latency automatically (see RFC-1305). Figure 5 shows the transition of 4 timepoints in the UDP message that allows the calculation of correct network delay and local clock offset from the server clock.

The testing procedure consists of using one PC, running the Timestamping Authority Server locally, and running from one to three clients also locally. Due to everything running on the same PC, time for the request to get from the client to the server and from the server to the client is negligible, so the precision of the aggregation procedure can be tested. The server starts up, gets time from three atomic clocks, and the clients are run when the server is ready to respond, (i.e. it has calculated an average time from all three clocks). Then, since we may assume that network delays are not an issue, we can collect all the timestamps received by the clients and compare them to the order they were sent. The tests showed that the correct order is preserved when the requests are spaced out by 10 ms or more.

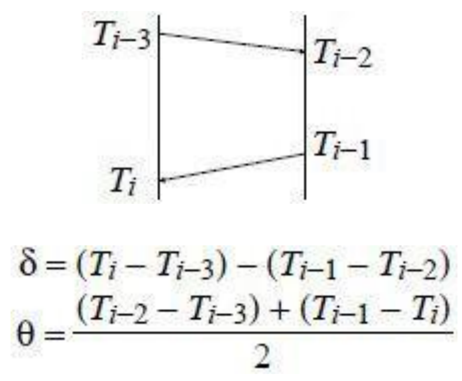
T_{i-3}
T_{i-2}
T_{i-1}
T_i
$\delta = (T_i - T_{i-3}) - (T_{i-1} - T_{i-2})$
$\theta = \frac{(T_{i-2} - T_{i-3}) + (T_{i-1} - T_i)}{2}$

Figure 5: Network Delay calculation from RFC-1305

Figure 6 shows an example time request from Event X. The TsA is retrieving the time from several atomic clocks and returns the precise time to the client.

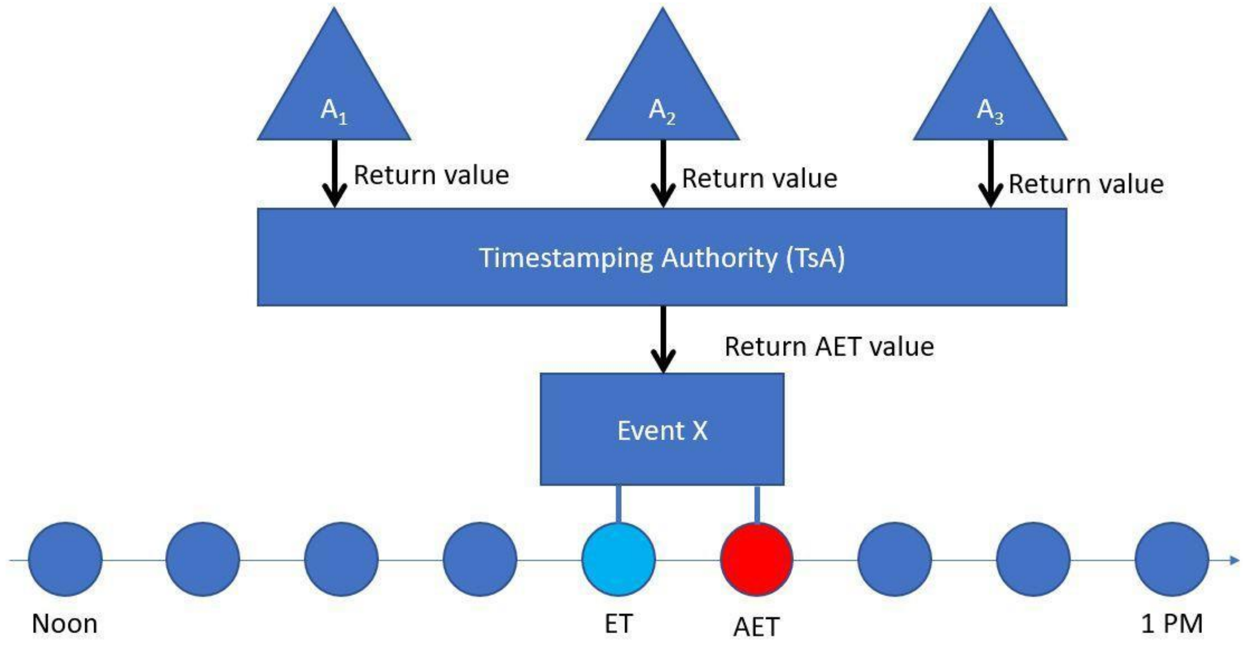

Figure 6: Scheme of TsA operation.

(ET - exact time of the event; AET - approximate exact time; A - atomic clock)

The timestamps are assigned to the candidates and form a queue, giving some time to the first-comer and notifying all other process candidates to wait (Figure 7). That is, the queue is formed among the candidates and each one is given some time to do the hash computation. If the client fails, the opportunity is given to the next candidate. This approach could reduce CPU time spent by the candidates, because they now have an opportunity to wait for the first arrival to finish before continuing their own computations. Timestamps will serve as a proof that the candidate was indeed first to arrive. This approach would be especially useful if there are multiple servers accepting blocks and they need to be coordinated. In this case, they could query the TsA and receive timestamps that could be used across the servers.

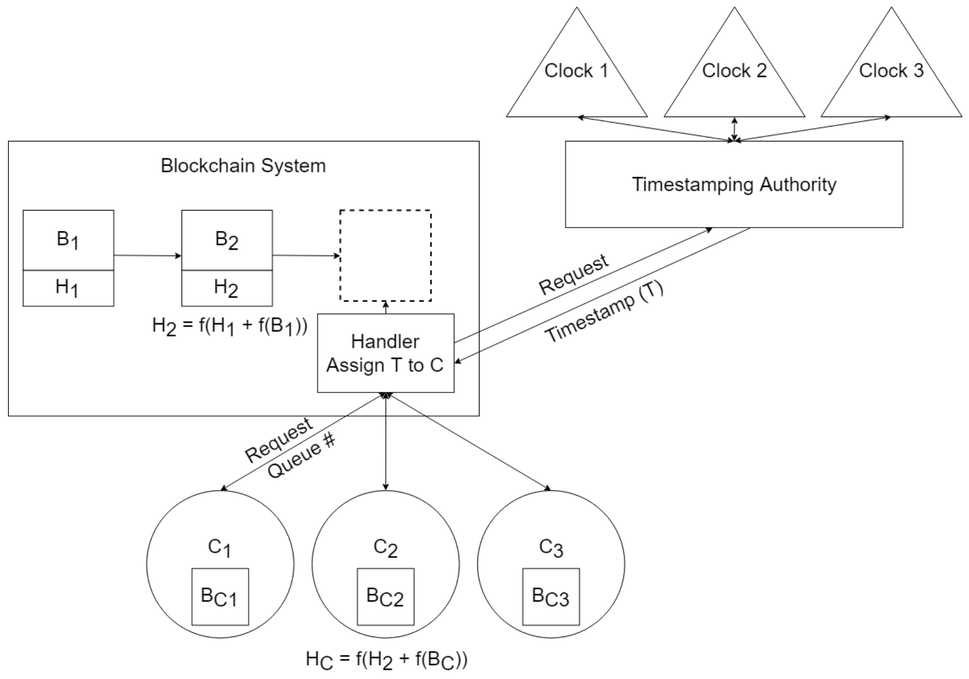

Figure 7: Adding a block to the blockchain with a TsA.

(B – block; H – hash; f - hash function; C – candidate)

It is important to note that the clients are not permitted to use local time due to possible imprecision. Maintaining accurate UTC time is difficult, and TsA does not claim that the time provided is accurate. Instead, the Time Stamping Authority establishes a common timescale for the system that would preserve order of timestamped events. In other words, Time Stamping Authority is precise, but not necessarily accurate. The project described in this paper was designed to demonstrate the effectiveness of an authority by aggregating time from several official reliable atomic clocks and use it to produce the common timescale. The prototype currently aggregates time from three nearby atomic clocks and computes the average of their times. (This is currently implemented as simple averaging, but some form of weighted average may be used in the future.) More effective formulae can be used after the aggregation. Tests with local server and client show that using this approach preserves the order of timestamps.

3.1 Clocks Used and Their Reliability

The International Bureau of Weights and Measures (IBWM) or Bureau International des Poids et Mesures (BIPM) in French, the organization that defines the International System of Units (SI) units, also provides the UTC time standard (https://www.bipm.org). This standard is created based on over 400 participating laboratories measuring time with their atomic clocks and reporting the measured time every 5 days to IBWM, which then calculates the weighted average based on the clock precision and produces UTC. The problem is that it is difficult to transmit clock data in real-time without losing accuracy, hence all calculated time points are in the past. Nevertheless, the data from the atomic clocks spread across the globe suggests that most of them are synchronized up to hundreds of nanoseconds (ns) (https://www.bipm.org/en/bipm-services/timescales/time-ftp/Circular-T.html), which is more than enough for the problem at hand. Within the U.S., the difference between UTC and the two commonly used clocks (NIST and Naval Observatory) is only a few ns. IBWM also publishes the Annual Report on Time Activities (https://www.bipm.org/en/bipm-services/timescales/time-ftp/annual-reports.html), which lists laboratories participating in creation of the time standard, and are known to be reliable. Some of those laboratories have publicly available NTP servers linked to the clocks, which are used in this project as time sources. For the North American region, possible clock servers are the National Institute of Standards and Technology NTP (https://tf.nist.gov/tf-cgi/servers.cgi), the United States Naval Observatory NTP (https://tycho.usno.navy.mil/NTP/) and the National Research Council of Canada NTP (https://nrc.canada.ca/en/certifications-evaluations-standards/canadas-official-time/network-time-protocol-ntp). One problem with using these clocks is that the public servers are vulnerable to distributed denial of service (DDoS) attacks, and therefore all clients making too frequent requests to them are banned (e.g., the NIST clock encourages no more than one request every 4 s). This problem has been solved by maintaining an offset from local time to each of the used clocks, and updating it every 10 s (this interval can be configured). Per Lombardi's estimations, most hardware clocks gain or lose about 5 s to 15 s per day (Lombardi, 2019), so with simple calculations, we see that the hardware clocks can gain or lose at most 2 ms every 10 s. While not perfect, the period of 10 s was chosen as a trade-off to both have acceptable precision, and not get the server banned by the atomic clock NTP servers. Therefore, even if a local clock is imprecise, it is very unlikely that any significant deviation will happen in 10 s.

4.0 Implementation Description

The NTP utilities library from Apache Commons Net was used in the project (https://commons.apache.org/proper/commons-net/). Some code was modified and everything else was used as a library. The project consists of four executables written in Java. The first and primary executable is the server, which can be configured and started via command line, specifying the NTP servers to be used at run time. The server periodically updates local time offsets of the used clocks and services client requests. There is a class *TimeStampingAuthorityServerRunner* with a main method that is compiled to the executable server. There are also two other classes that can be used as library code. One of these (*TimeStampingAuthority*) simply requests time from the clocks, maintains clock offsets and can provide aggregate time via the application programming interface (API), and another one (*TimeStampingAuthorityServer*) also runs an NTP server on a specified port. There is a separate Clock class that contains logic for maintaining the clock offsets and network communication with the atomic clock servers.

The second executable is the client for testing. It can be configured and started via command line, and it logs the time points received from the server to a .txt file. The client makes requests to the server at the specified address with a given frequency. Tests show that an interval of 10 ms or more should preserve events ordering on the TsA side. Even though such precision might not be enough for some applications, it has a benefit of simplicity: no need for advanced equipment or dedicated connections, just one program on any PC. Moreover, we would note that this is a proof-of-concept, so future developments may achieve higher precision.

The third executable takes the log file produced by the client and checks if timestamps the client received are in the ascending order. It can check multiple files at the same time and is also usable from a command line. The fourth executable takes several log files and combines them. This executable was used when several clients were run simultaneously to test how well TsA preserves order of events with its timestamps. All the executables are configurable. VT uses the Maven framework for dependency management and build automation. It also uses Git for tracking changes (https://github.com/usnistgov/blockmatrix/tree/master/TimeStampingAuthority).

Executables can be used on their own, but it is suggested that an interactive development environment (IDE) supporting Maven and Git is used in development, e.g. IntelliJ IDEA or Eclipse (both are free). There are four Maven modules corresponding to four executables in the project. 3 contain just one class with a main method, and one contains several classes mentioned above. There is a directory called "testing" which contains all the executables, a file with the clock data and a script for testing. Maven is configured in such a way that when the "package" command is run, all the modules are assembled into .jar files. When the "verify" command is run after that, the .jar files are moved into the correct directories for convenience. The "out" directory contains all 4 executables bundled with their dependencies as well as a library .jar, containing non-executable server-side classes without dependencies. This library jar can be used in other projects as a dependency, assuming that the Apache Commons Net library is included.

4.1 Installation

This project uses Java Development Kit (JDK) 12.0.1 (the current version as of this writing), but is tested to be backward-compatible with JDK 1.8.0_21.

If used as a .jar, the executables do not have any dependencies except for the Java runtime environment.

If used as Java classes, the Apache Commons Net library needs to be downloaded, which provides implementation of many network protocols, including NTP, along with utilities that make it easier to use these protocols in the program. You can include this library as a dependency in a Maven project using the information on

https://mvnrepository.com/artifact/commons-net/commons-net/3.6 or download the libraries as source code or as binaries on

https://commons.apache.org/proper/commons-net/download_net.cgi. The libraries are open source and are available under the Apache License, Version 2.0.

It is suggested that you use an IDE with support of Maven and Git if you would like to use them in the development of the project, because Maven downloads the dependencies for you and makes building and packaging easier and Git helps revert any changes and keep track of the history of the project. If not, a simple solution would be to rip out the classes of all 4 modules and put them in a new project together. Do not forget to add the Apache Commons Net library as a dependency.

5.0 Limitations

The order-preserving property of the current TsA implementation has been tested only locally. Namely, the procedure was to run the Time Stamping Authority server locally, together with one or more clients that would request timestamps from the server. Since the time on the PC is the same, more tests are needed to determine the reliability of TsA. Namely, clients could be located far from the server, and have their own reliable time source to compare the timestamps with. Based on this kind of testing, the conclusion about order-preservation can be made.

Sometimes several subsequent timestamps get "merged" and have the same value, even though it is known that the events were supposed to have a time interval between them. For example, there could be several timestamps with values 1, 1, 1, 4, 5, while the events were supposed to happen at times 1, 2, 3, 4, 5. It is likely that this happens due to low computing power of the development machine used, and as a consequence of running several processes for actual testing of the program and other processes running in the background. If the events are spaced out with the stated time difference - 10 ms, then the weak order is always preserved. By weak order we mean the sequence of numbers where two numbers can be equal, not strictly increasing.

Implementation factors are a consideration in this merging phenomenon. The prototype was implemented in Java, which sometimes has unpredictable runtime due to garbage collection, and it was not the only program running on the PC, so the operating system (OS) could take away processor time for a fraction of a second, corrupting the timestamps. It is expected that if the implementation had adequate performance and priority is given to the time-stamping server, these anomalies should not be present, and the resulting timestamps should form strictly increasing sequence. Additional testing will be used to evaluate this condition.

1. NTP assumes that the offset from the client clock is to be calculated by the client, so the TsA has no way to know if the client completed the timestamp calculation correctly.
2. When requesting time from the clocks and serving time to the client, only one NTP-request is made, since acceptably low latency network conditions are assumed. It is suggested to make several NTP-requests and take the one with the lowest round trip time to exclude asymmetric route errors.
3. RFC-1305 suggests a formula for combining several clock times to increase accuracy and precision. This could be used instead of a simple average of all the clocks, which is currently used.

6.0 Conclusion

In the real estate community, it is said that property value is based on "location, location, location." But we also know that timing is everything. In sports, the end of a game can be decided in a single second. In financial transactions, it comes down to milli- and microseconds. Here, we have shown that atomic clock aggregation is possible and that it works better if the clocks are somewhat geographically co-located due to latency. We have argued that a Timestamping Authority (TsA) is a feasible approach to creating timestamps of higher integrity.

We plan to continue this research to better understand the impact of latencies on the accuracy of the aggregated clock results.

7.0 References

Lombardi, M., "Computer Time Synchronization," Time and Frequency Division, National Institute of Standards and Technology, https://tf.nist.gov/service/pdf/computertime.pdf, retrieved, 8/17/19.

Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis," RFC 1305, March 1992, https://tools.ietf.org/pdf/rfc1305.pdf, retrieved 8/17/19.

Mizrahi, T., Mayer, M., "Network Time Protocol Version 4 (NTPv4) Extension Fields," Internet Engineering Task Force (IETF), March 2016, https://tools.ietf.org/pdf/rfc7822.pdf, retrieved 8/17/19.

Kuhn, R., Yaga, D., Voas, J., "Rethinking Distributed Ledger Technology," Computer, Feb 2019, pp. 68-72.
Stavrou, A., Voas, J., "Verified Time," Computer, March 2017, pp. 78-82.

Directive 2014/65/EU of the European Parliament and of the Council with regard to regulatory technical standards for the level of accuracy of business clocks
http://ec.europa.eu/finance/securities/docs/isd/mifid/rts/160607-rts-25_en.pdf
Annex: http://ec.europa.eu/finance/securities/docs/isd/mifid/rts/160607-rts-25-annex_en.pdf

SR-FINRA-2016-005. Proposed Rule Change to Reduce the Synchronization Tolerance for Computer Clocks that are Used to Record Events in NMS Securities and OTC Equity Securities
https://www.finra.org/industry/rule-filings/sr-finra-2016-005

Mills, D.L. (2003). A brief history of NTP time: memoirs of an Internet timekeeper. Computer Communication Review, 33, 9-21.

Mills, D.L. (2012). Executive Summary: Computer Network Time Synchronization. https://www.eecis.udel.edu/~mills/exec.html.