Distributing Deterministic, Accurate Time for Tightly Coordinated Network and Software Applications: IEEE 802.1AS, the TSN profile of PTP

Kevin B. Stanton

Abstract

Real-time cooperation among agents requires sharing of certain temporal information. That information might take the form of "*take action X right now!*" or it might better be achieved in some cases by establishing an a priori agreement by which each action should be taken and when, together with a shared understanding of what time it is. Distributed applications increasingly rely on this latter synchronized time approach rather than explicit "*right now!*" communication for cooperative sense and control tasks. Such a capability is foundational to the field of cyber physical systems (CPS) where computation and communication interact with the physical world in real time. While multiple historical approaches exist for time-synchronized (and timely) communication among sensing, computing, and actuating devices, the addition of robust temporal performance characteristics to standard Ethernet through the work of the Time Sensitive Networking (TSN) Task Group of IEEE 802.1 is bringing together the benefits of standard IT networking with excellent worst case latency and synchronization performance that was previously available only to these specialized and sometimes proprietary networks. Such advances will be important in next-generation applications as diverse as CPS/real-time distributed control, smart factories, sensor fusion, autonomous vehicles, and even cloud applications [1]. As we show, precision network time synchronization can dramatically enhance the operation of these applications and is a requirement of certain new network latency enhancements as well.

Introduction

Accurate time synchronization has been a discipline of scientific and engineering interest since the days of Newton's pendulum clock and John Harrison's marine chronometer. The last decade has brought enormous improvements in the accuracy with which time may be synchronized to devices over standard and readily available networking equipment, resulting in even more applications of accurate time, feeding the need for further advancements in time accuracy, robustness, and security. Together, the Time Sensitive Networking (TSN) standards provide accurate time, deterministic and low latency, redundancy, and robustness, all of which are required in a large array of applications, including industrial, automotive, energy, and audio and/or video. After briefly examining general concepts of time synchronization, we provide an overview of the uses of synchronized time by software and other applications operating at the edges of a network in various markets, describe unique challenges in securing and accessing time in software applications (even virtualized), and outline some of the application requirements that motivated certain design decisions for IEEE 802.1AS [2], the TSN profile of IEEE 1588 [3].

We begin with an analogy of the use of time synchronization to cause the delay experienced by certain time-critical information to be deterministic and small within the network itself. Imagine driving through a tight grid of busy city streets during rush hour without ever having to stop your car or slow down. This would be a dream come true for commuters with an arrival deadline and ideal for the communication of data with its own deadline. In both instances, worst case transit time or latency is a key figure of merit. Obviously, very tight timing of stoplights would be required to achieve such a feat, and precise timing of transmission is likewise required within a network to consistently achieve the best case latency in the worst case. Note that time synchronization was not achieved through deterministic latency. Rather, deterministic latency was achieved through time synchronization and the application of a global schedule, which are analogs for two TSN standards, IEEE 802.1AS and 802.1Qbv, respectively.

A standard method of synchronizing time with a deterministic sub-microsecond error, IEEE 802.1AS, is included as a foundational member of the standards developed by the TSN Task Group of IEEE 802.1 (heretofore referred to as TSN), a profile of the Precision Time Protocol (PTP) defined by IEEE 1588. While time synchronization can facilitate network features that thus provide robust and low latency guarantees, time synchronization more generally provides massive benefits to a broad class of applications, as we will see.

As deployment of PTP continues to expand, we can look forward to increasingly flexible and productive manufacturing, and higher temporal

Digital Object Identifier:
10.1109/MCOMSTD.2018.1700086

The author is with Intel Corporation.

2471-2825/18/$25.00 © 2018 IEEE

data fidelity. We can only hope that city traffic planners take a cue from the masters in traffic engineering within the IEEE 802.1 and Internet Engineering Task Force (IETF) communities.

Of the many approaches to time synchronization, the TSN Task Group of IEEE 802.1 chose to adopt the IEEE Std. 1588-2008 family of protocols (collectively called PTP) in order to reuse the now ubiquitous Ethernet hardware support–all codified into the IEEE 802.1-2011 profile of PTP and under revision as of 2018.

For better or for worse, the IEEE 1588 standard defines a number of optional, even mathematically redundant protocols and parameters that must be selected among to form an interoperable set. Such formal selections are formally known as a "PTP Profile" and are typically published by a standards development organization. As of this writing, in addition to the TSN profile published in IEEE 802.1AS-2011 and currently under revision, there are three profiles defined by the current revision of IEEE 1588 (up from two in the previous version), three more defined by the International Telecommunication Union (ITU), one each defined by the International Electrotechnical Committee (IEC) and IEEE for power/energy systems, one by the Audio Engineering Society (AES) for audio, one by the Society of Motion Picture and Television Engineers (SMPTE) for video, and yet another by the IETF. Enhancements in the IEEE 1588 revision currently specify mechanisms for devices to unambiguously associate a PTP message with a particular profile. Previously it was possible to do so only for the IEEE 802.1AS profile, which had a dedicated bit in the PTP header. The author hopes that in the future the proliferation of new profiles will cease and, indeed, that the number of profiles deployed in a particular networking environment will tend toward one.

Synchronizing Time

Imagine that you are given the task of telling the people at the Royal Observatory in Greenwich what time it is according to the Big Ben clock in London, and all you have at your disposal is a scooter. Your scooter has only two speeds: Stop (0 km/h) and Go (10 km/h). You could simply look at Big Ben, write the time on your hand, ride your scooter to Greenwich, and loudly announce the time (one-way time transfer). There are several problems with this approach. First, the announced time will be incorrect by your time of travel. The obvious solution is to turn around and travel immediately back to Big Ben and note the elapsed time. If the trip takes the same amount of time in both directions, the time announced in Greenwich was incorrect by half of the round-trip travel time. You would want to correct for this at subsequent announcements (two-way time transfer). Second, even riding a scooter at a constant speed, your travel time will vary in each direction and on each subsequent trip because of the unpredictability of stoplights at intersections encountered. We call this "delay variation." The solution to this problem is one of the innovations introduced by PTP. If you were to use a stopwatch to measure the sum of the time spent waiting at each intersection for a red light to turn green, you could remove the time error otherwise contributed by stopping and waiting. This is illustrated in Fig. 1.

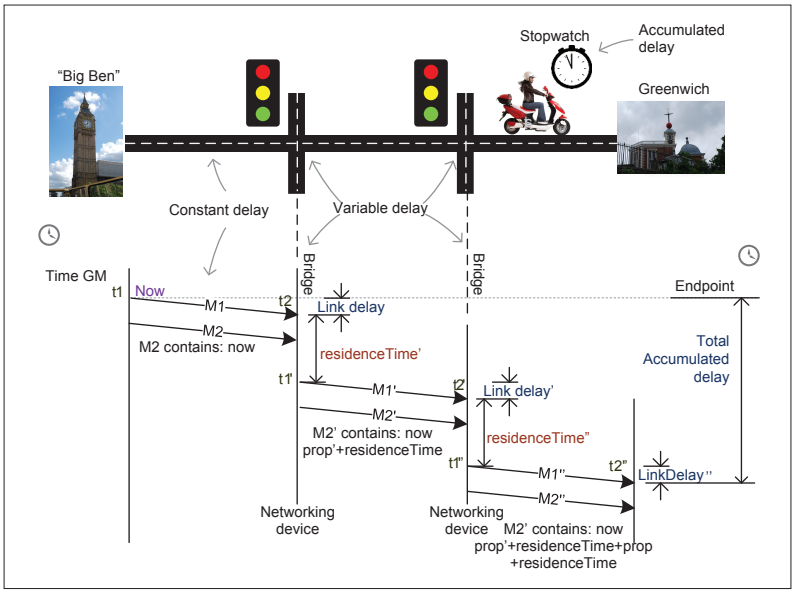

Figure 1. Time propagation with delay compensation.

This analogy, which employs two-way time-transfer, applies directly to the Generalized Precision Time Protocol (gPTP) defined by IEEE 802.1AS. Note that we use "gPTP" when referring to this specific PTP profile, and "PTP" when referring to the broader set of protocols defined in IEEE 1588 of which gPTP is a profile.

Through ANNOUNCE messages and the Best Master Clock Algorithm,[1] one device may be automatically selected to be the source of time (the Grand Master or GM). The GM periodically sends a network packet containing the time[2] with the timing information eventually propagating across the network.

The timing information passes through each networking element measure (like a stopwatch); how long it delayed that timing packet[3] is a quantity labeled *residenceTime* in Fig. 1. The magnitude of this delay is not material as long as the delay is measured for each timing packet by each networking element (switch, router, AP, etc.) and accumulated downstream from the GM to the ultimate recipients of the timing packet, along with the measured delay of propagation across the upstream link, a quantity labeled *link delay* in Fig. 1. Like the scooter, the speed of communication between intersections is constant (speed of light in this case, a considerable improvement over 10 km/h). Also like our scooter analogy, two-way time transfer is used to compute the one-way transit time, although with gPTP this measurement is performed between every pair of interconnected network elements.

Obviously, the more accurately the round-trips and the individual network element delays are measured, the more accurate the end result. gPTP requires that these timestamps are measured with accuracy of no worse than 40 ns (for Ethernet), with most commercial silicon capable of achieving substantially better accuracy. An interesting consequence of performing these measurements in hardware is that time accuracy is independent of network traffic, CPU interrupt latency, and network bandwidth or cable length.

[1] ANNOUNCE messages are used to communicate information used to select the GM.

[2] Actually, a SYNC message is used to take the requisite measurements, leaving a trail of delay measurements like breadcrumbs, with a FOLLOWUP message coming behind and carrying the original GM transmit time and cumulative delay (it is hard to transmit a SYNC message and later modify it to contain the time of transmission, although some PTP profiles specify this so-called one-step operation).

[3] For other profiles of IEEE 1588 the residence time delay may be measured by only some of the network elements or may be measured by none of the network elements, resulting in substantially degraded timing accuracy.

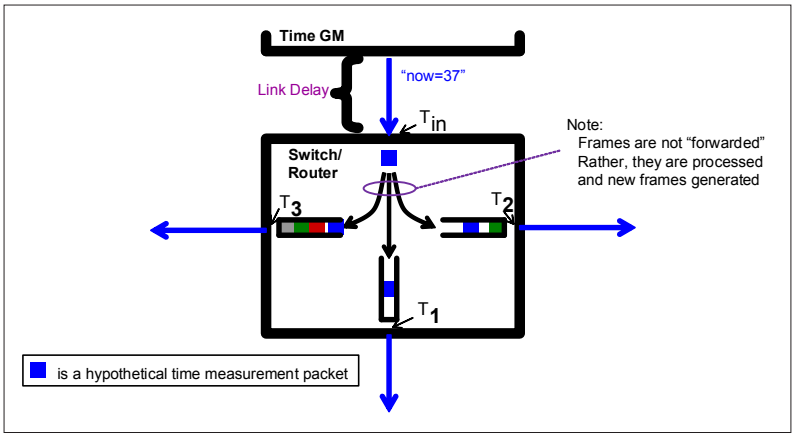

FIGURE 2. Bridge computation of residence time.

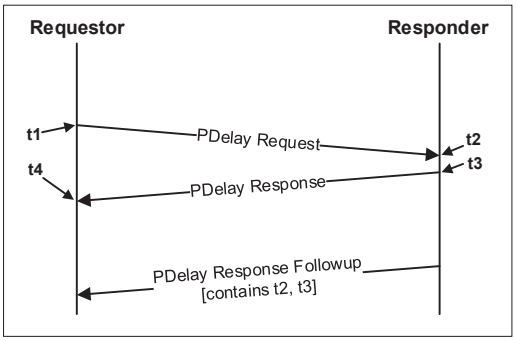

FIGURE 3. Measuring link delay with the PDelay protocol.

One important benefit realized along paths where network devices participate in the time-transfer protocol, as gPTP demands, is that the worst case time accuracy can be determined ahead of time, rather than requiring probabilistic analysis based on statistics of the packet delay variation [4]. The natural trade-off for achieving deterministic worst case time accuracy per hop is that the switch typically needs firmware (and hardware timestamping) in order to participate explicitly in the protocol and to measure and propagate the delays. The GM time is modified by the contributed delay introduced while propagating that timing information downstream, computed by adding the link delay and adding the difference between the egress-port-specific exit time (shown as T_1, T_2, T_3) and the arrival time (T_{in}), as shown in Fig. 2.

Before going further we make a few comments on time and frequency. Time, as we use it in everyday human experience, is essentially a count of a series of repeating events such as the swinging of a pendulum (i.e., a second) or the rotation of the Earth (i.e., a day). Similarly, an electronic device might use a quartz crystal as its frequency reference, or one might employ the extremely stable electromagnetic oscillation of a laser (e.g., an optical clock). The bane of timekeeping is that the ticktocks of any two clocks always differ in their count rate or frequency. To make matters worse, the frequency of every clock varies over time. Even if time were synchronized EXACTLY at the beginning, and each device contained a high-quality frequency reference, without a method of measuring and synchronizing them the clocks would become unsynchronized almost immediately and continue to drift. For this reason, a protocol such as PTP must run periodically, first synchronizing the time and then continuing to compensate for differences in frequency references and changes in those references. gPTP defaults to synchronizing time eight times per second.

Time Transfer over Wired/Ethernet Links

The gPTP profile employs the PTP-defined PDelay mechanism, both to measure the round-trip time between devices (including, e.g., the CAT-5 wire delay) and to measure and communicate their frequency difference, useful for computationally correcting residence time durations (Fig. 3). While it would be theoretically possible to synchronize time with just the PDelay round-trip message exchange, PTP requires use of a SYNC message to trigger transmit and receive timestamps. SYNC is followed by a FOLLOWUP message in the gPTP profile, which carries, among other things, the time at which the previous SYNC message was sent, as seen in Fig. 1. Using all of this information, each network device can compute and propagate the current time away from the GM toward the devices that need the time.

For completeness we briefly list several other options that can be specified in a profile of IEEE 1588:

One-Step: Start transmitting the SYNC message (or other event message), but quickly modify that message with the transmission time during transmission but before the message finishes transmission, eliminating the need for a FOLLOWUP message.

End-to-end: Instead of measuring the hop-by-hop link delay using PDelay, every slave sends a DELAYREQUEST frame to the GM, which then responds to each of those slaves with another message. Each PTP-aware network device in the path corrects any introduced delay and jitter. Where such support is not present, worst case error is difficult or impossible to determine using this method, but it allows best effort time synchronization through networking components that lack PTP protocol support.

IPv4/IPv6: Multiple encapsulations of PTP messages are defined.

Time Transfer over Wireless/Wi-Fi Links

It is important to note that while 802.1AS defines the gPTP profile of PTP for Ethernet, it also extends the PTP time semantics to IEEE 802.11 links, also known popularly as Wi-Fi, as well as other media. Unlike modern Ethernet links, which are full-duplex, Wi-Fi links are notorious for introducing substantial (millisecond) jitter in frame transfer latency due to frame retransmission, dynamic link rate changes, and other bandwidth enhancements such as frame aggregation. All of these can introduce milliseconds of delay variation and thus uncertainty when transferring time over even a single wireless link. For such 802.11 wireless links, gPTP defines methods to transfer time using the Timing Measurement (TM, defined in IEEE 802.11-2012) or Fine Timing Measurement (FTM, defined in IEEE 802.11-2016) protocols,

which were designed to perform round-trip measurements at such a low level in the protocol stack that they are below retransmission and aggregation, and immune to rate changes. See Fig. 4 for an illustration of the TM protocol as employed by gPTP. Implementations of FTM, defined for round-trip time measurement for the purpose of indoor location estimation (where 3 ns inaccuracy corresponds to about 1 m of uncertainty), has the possibility of further reducing the time error resulting from multiple arrival paths within the wireless channel, identifying and timestamping even a very weak line-of-sight arrival of the electromagnetic wave-front from the sender such that single-digit nanosecond time accuracy might be achievable across a single Wi-Fi link.

gPTP defines transfer of time consistently from a single GM over heterogeneous networks consisting of both wired and wireless networks of various types, and the draft of the revision of IEEE 1588 includes provisions for IEEE 802.1 to provide such expansion of PTP to this and other IEEE 802 media in a standard way.

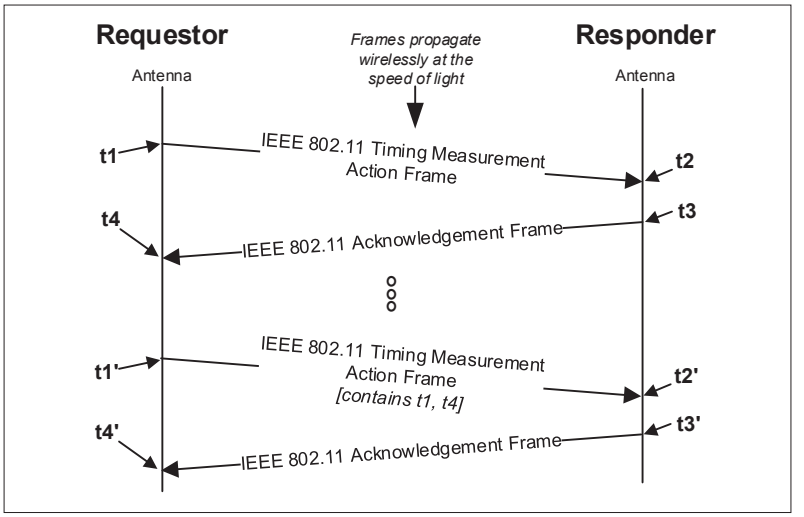

Figure 4. The IEEE 802.11 Timing Measurement protocol.

Applications of Accurately Synchronized Time

General Principles

We now turn our attention to some of the varied uses of synchronized time with primary focus on those for which gPTP was developed. Use of time can be broadly categorized as follows:

1. Timestamping events (software events or hardware events).
2. Schedule events (software events or hardware events).
3. Measure elapsed time from one event to another event.
4. Measure the frequency of an input signal with respect to the PTP reference.
5. Output a signal with a certain frequency with respect to the PTP time reference.

In the following sections we describe use of one or more of the above for specific purposes. The author predicts that as the deployment of these standards continues to increase, additional applications of predictably accurate time synchronization will continue to expand.

Ordering Events

For all of these, it is important for the system to be aware of the inaccuracy of time transfer both between and within the compute system. As an example, if gPTP transfers time to two systems with a maximum error of 50 ns, and those two systems timestamp two hardware events, *a* and *b*, and the order of these events must be determined later, the conclusion could be that *a* precedes *b* or that *b* precedes *a* or that it is impossible to tell (*a* and *b* occurred within the uncertainty band).

Distributed Sampling

When sampling some physical quantity or phenomenon at more than one location or from more than one perspective, it is often necessary to coordinate the sample events, with relative simultaneity being a special case. Using explicitly synchronized time to maintain temporal alignment of such events has the distinct advantage of decoupling sampling simultaneity from network latency. For example, if time were synchronized among a set of camera, LIDAR, and radar sensors within an automobile, it would be possible for a sensor fusion device to schedule aperture open and close times of the various sensors (e.g., for them all to be simultaneous), reducing the search space when performing fusion-based object recognition [5]. In the case where parallax of multiple cameras is exploited to compute distance to an object, there are distinct advantages to initiating photon exposure in a coordinated manner such that each line of the camera captures photons reflecting off of an object of interest during the same time interval, lest the distance be estimated incorrectly due to relative motion between the sampling intervals of the respective cameras.

While not a likely use of the gPTP profile, other applications of PTP include distributed sensor arrays like those used in particle physics and other scientific experiments, notably the CERN Large Hadron Collider, which employs advanced techniques to synchronize time to below 0.2 ns over multiple network hops and kilometers of fiber [6].

Industrial Control or Cyber-Physical Systems

Similarly, for industrial automation (e.g., motor or drive control as part of a pick-and-place robot or other motion control within an assembly line), it is important to take velocity, position, voltage, or current measurements along the full set of joints, servos, or other actuators in a way that is well coordinated, with relative simultaneity being the simplest policy. Accurate synchronization among networked sensing and/or actuating nodes (sometimes connected in networks of daisy-chains of devices, each with a three-port Ethernet switch) is thus critical for industrial manufacturing efficiency and product quality. Increased velocity and precision drive the need for increased sample rates, decreased and deterministic network latency, and improved time accuracy as the sensors and actuators perform their actions at the scheduled time. Requirements for time accuracy in the deep sub-microseconds among the distributed actuators and sensors is not unusual. Operating system (OS) scheduling of a compute task within a CPU with knowledge of *when* the last sensor data of a cycle

Accurate time is heavily used by a broad spectrum of applications as seen above, but is also required for proper operation by network devices for some of the TSN standards, most notably the scheduled transmission defined in the IEEE 802.1Qbv standard.

will arrive from across the network (thanks to TSN) might further improve the achievable cycle time, allowing deployment of a larger spectrum of compute systems in industrial environments.

Audio/Video

Professional audio and video (AV) applications require both accurate timestamping of isochronous streams of data and regeneration of a common audio or video clock frequency elsewhere in the network, whether at a mixer, audio/video processor, microphone/camera, or renderer (speaker or display). Sometimes referred to as the Audio Video Bridging (AVB) standards [7], these predecessors of the published TSN standards (including IEEE 802.1AS-2011, gPTP) have been deployed broadly in so-called professional audio/video installations. Microsecond-level time accuracy or better is a common requirement for arrays of speakers or microphones. gPTP is broadly deployed in environments including broadcast studios, theme parks, conference halls, auditoriums, and recording studios. An example is the Apple Mac™ since the Maverics OS implements gPTP for just this purpose [8].

Distributed, Coordinated Computation

A large number of additional applications include cloud application monitoring, synchronizing fifth generation (5G) base stations and other processing nodes in telecommunications, and telemetry of the power grid, especially where micro-grids present new challenges when reattaching themselves to the main power grid following a power outage or fault. Of increasing interest recently is the need to timestamp financial transactions with a prescribed worst case accuracy with respect to UTC in order to meet regulatory requirements [9].

The TSN/802.1Qbv Application of gPTP

Accurate time is heavily used by a broad spectrum of applications as seen above, but is also required for proper operation by network devices for some of the TSN standards, most notably the scheduled transmission defined in the IEEE 802.1Qbv standard. In this sense, network equipment itself, in addition to distributing accurate time, also implements an "application" (802.1Qbv) that relies on accurate time synchronization, as described below.

Returning to our first scenario where we want time-sensitive traffic to encounter only green lights (and putting aside interrupt-driven green lights for emergency vehicles), we could achieve this goal for a large number of vehicles optimally if we know when each vehicle will depart, where it is destined, and the route it will take. With this information a schedule that is optimal with respect to some cost function can be computed and a schedule sent to every traffic intersection where it is used to determine which directions see green lights when, and which see red. We would also want the possibility to perform an orderly update of the individual schedules during operation.

The 802.1Qbv standard similarly defines the schedule behavior of the transmit selection algorithm, providing the possibility of congestion-free routing of time-sensitive traffic.

Fortunately, for a large class of real-time applications, including many cyber-physical systems (CPSs), much of the time-sensitive network traffic from sensors or to actuators is predictable and periodic, whether 1 cycle/s or 32,000 cycles/s, making a fixed schedule feasible.

Other Considerations

Security of Time

Whether from benign causes or with malicious intent, a number of things can go wrong when transferring time. First, the time data contained in the gPTP frames themselves might be changed, spoofed, or replayed, or the GM itself could be compromised. Second, time transfer suffers from a malady that is unique among most communications: an asymmetrical delay applied to gPTP frames corrupts the timing information carried by those frames, since predictable and small time error depends on the properties of constant and symmetrical delay between networking nodes. An IETF draft [10] lays out the threats model for PTP time protocols in packet-switched networks. Some of these threats are discussed and dealt with in a dedicated security annex of the IEEE 1588 revision.

Use of Time by Software in Compute Nodes

As discussed above, application software might need to know the time "now," with respect to PTP, such that software or hardware events may be measured or affected within the compute system or in a node across the network, or a duration can be measured. The former is typically achieved through calls to the POSIX time functions, which return either monotonic time or real time. Linux, for example, can discipline the value returned in both instances to the frequency and time of an incoming PTP clock, thanks to the PTP Hardware Clock (PHC) driver and infrastructure [11]. This allows for immediate and accurate access to "now" with respect to PTP by software applications. Other applications of time within a compute node are application-dependent, but broadly supported by software stacks within Linux; for example, the ALSA stack for audio allows application logic to request the relationship between the real-time clock and the position of an audio stream (e.g., via the `snd_pcm_htimestamp()` function). Similar functions are available for generating pulse-per-second (PPS) hardware outputs, accessing timestamps from sensor data, or camera inputs.

Referring to Fig. 5, a software call to retrieve the current PTP time typically results in a read of the low-latency CPU counter, accessible through the ReaD Time Stamp Counter (RDTSC) instruction within just a handful of nanoseconds if running on an x86 CPU. The resulting value is then scaled by the Linux OS to nominal nanoseconds (1) is PTP-frequency-corrected (2), and a large constant is added based on the epoch (3), typically nanoseconds since 1970 (ignoring leap seconds). The scaling factor can be adjusted regularly by the PHC infrastructure based on the regularly measured relationship between the CPU counter and the corresponding PTP time. All of these quantities are available via the POSIX `clock_gettime()` function of Linux associated with the clock identifiers (1) `CLOCK_MONOTONIC_RAW`, (2) `CLOCK_MONOTONIC`, and (3) `CLOCK_REALTIME`. The multiplier in (2) is adjusted regularly to

drive to zero the difference between PTP and (3) based on periodic cross-timestamps (4) (Fig. 5).

Once the GM time is known to an Ethernet interface of a computing system, it must make its way within the system to where it is needed elsewhere in the system such as the CPU, sensing circuit, frequency generator, camera trigger, or voltage regulator; or the time might even need to be propagated out through another Ethernet interface via another PTP profile. This has been referred to as the "last inch problem" [12]. While time transfer within the system is typically outside the scope of the aforementioned IEEE standards, there are accepted methods of achieving good accuracy in such transfers, such as using as a PPS hardware signal routed between chips or time transfer from the Ethernet controller to the compute system-on-chip (SoC) using the Precision Time Measurement (PTM) protocol of PCIe [13].

One interesting challenge for timekeeping within a compute node is the handling of virtualization [14]. In some cases, a virtual machine (VM) may want to observe time passing only when it is actively running. A hypervisor may maintain this fiction for the VM by effectively stopping time while that VM is not operating. While this mode of operation can be useful for characterizing the performance of an application running in that VM, it causes huge problems when that VM attempts to interact with an external time source or draw a conclusion regarding dynamics in the physical world where the virtual timeline contains gaps when viewed from a physical timeline; thus, a timeline associated with the physical world must be maintained for a VM that has that requirement.

In systems where the temporal application logic is implemented substantially in hardware (where any software is not time-aware), the Ethernet PTP hardware counter itself might be both synchronized in hardware to the GM, and used directly by digital logic to schedule hardware events and timestamp the sampling of input stimulus or thresholds.

Enhancements in gPTP beyond PTP

The gPTP profile, as described above, adds support for 802 standards beyond the 802.3 Ethernet standard, most notably 802.11 (commonly referred to as Wi-Fi) links. Beyond these additions, gPTP was designed to:

- Support very fast lock times, including where reconfiguration of the clock tree is necessary
- Support seamless change from one GM to another
- Permit very inexpensive implementation in network equipment while meeting the needs of the distributed applications
- Support centralized configuration of clock tree and multiple redundant GMs
- Predictable time inaccuracy

We next provide additional context for these five enhancements.

1. Lock time is defined as the time required for a device to achieve the requisite time accuracy with respect to the GM and for devices to achieve the requisite time accuracy with respect to one another. Fast lock times are difficult to achieve if the network equipment participating in the protocol maintains its own notion of the time by performing long-term averaging of the received GM time. gPTP addresses this challenge by requiring every device to explicitly measure the frequency offset between its local frequency reference and that of each of its neighbors. The pair-wise-measured frequency offsets are computed and known at both ends of every link. Thus, if a new GM suddenly appears or the network time tree is reconfigured, the end-to-end frequency offset between any node and the GM may be computed through accumulation of the stepwise frequency offsets as it is passed explicitly in a special rate_ratio field. This frequency offset need not be measured indirectly over long periods of time in a chain of cascaded PLLs. In this way, time stability after reconfiguration may be achieved with gPTP in just a few seconds rather than tens of seconds or minutes.

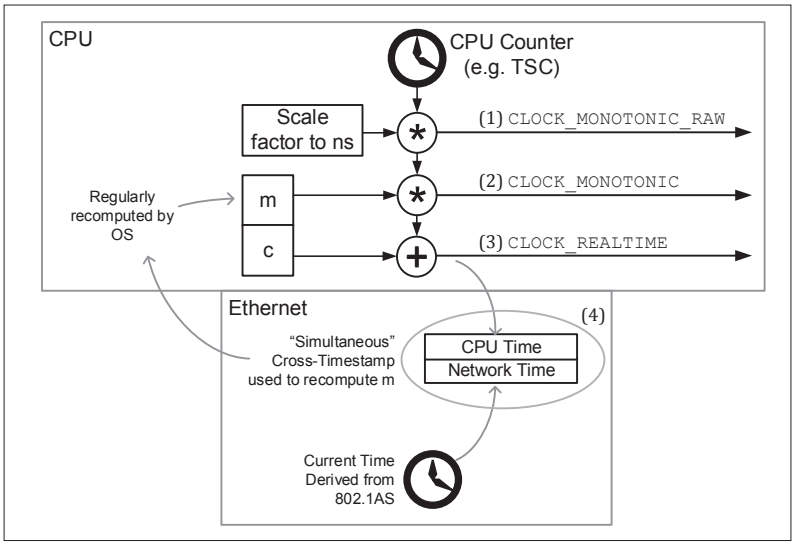

Figure 5. Fast software access to current PTP time.

2. Seamless change from one GM to another is achieved by requiring a new GM (where possible) to first observe the current GM in order to measure its own time and frequency offset from the active GM. It then announces itself, along with its time and frequency differences with respect to the previous GM. This allows other devices to more rapidly perform any necessary time extrapolations.

3. Simplified hardware implementation by network equipment with scalability to multiple time domains simultaneously from multiple GMs per port is achieved by allowing a bridge or router to use a single local free-running clock for all PTP packet timestamping operations (adjustable hardware clocks are typically more complex and require hardware to be replicated for each domain of each GM operating through each device), and applying corrections to the time computationally as it is propagated to other ports using a linearity transformation rather than physically adjusting the frequency of the local reference clock. Note: Some applications of time, including IEEE 802.1Qbv, benefit from a dedicated application clock that is frequency disciplined to match the appropriate domain of the selected GM for the purpose of applying the transmit schedule to queues containing time-sensitive frames.

4. Centralized configuration of the network clock tree is added in the revision of IEEE 802.1AS.

Further work is needed to further enhance and standardize Byzantine fault tolerance, time transfer over increasingly heterogeneous networks including 5G, vetting of the new security recommendations for time transfer by the various profiles of IEEE 1588 (including gPTP), and further interoperation or coexistence of the various current, legacy, and emerging profiles of IEEE 1588.

This allows port and state configuration in the style popularized by software defined networking (SDN) techniques. Master/slave states of each port can be set by centralized management, and redundant GMs and domains can even be configured, each with a set of potentially overlapping trees spanning the slave nodes. The old adage says that the person with one watch believes they know what time it is, but the person with two watches is never sure. These mechanisms make it possible to configure several GMs to synchronize to a common time source and subsequently propagate that time over maximally disjoint paths. Building on such mechanisms, even Byzantine failures may be detected and/or mitigated.

5. Predictable time accuracy on the order of sub-microseconds is achievable (together with the above requirements) only if all devices are participating in the protocol. Thus, gPTP defines methods whereby network equipment unaware of gPTP is identified and the clock tree is terminated. If proper care were taken, other profiles of 1588 that provide predictable accuracy could theoretically be used with TSN standards such as 802.1Qbv, also assuming that all relevant TSN nodes were designed to support that PTP profile.

Compliance and Multi-Vendor Interoperability

The TSN standards define what the externally observable behavior shall be in order to be compliant with those standards, and they do this extremely well. But the standards typically do not, and should not, make market-specific determinations that these features from these standards should be implemented by equipment manufacturers in this timeframe for this and that markets. However, it is natural for an open-membership, non-profit corporation to provide a legal world-wide forum for making these decisions among equipment manufacturers, with resulting certification and testing programs, detailed test plans, an ecosystem of testing tools, open source software, and documented "best practices." Avnu Alliance [15] has filled this role for nearly a decade, with robust certification tests defined for the 802.1AS and other TSN standards for the professional A/V, industrial, automotive, and consumer electronics markets.

Conclusion

The fact that gPTP is intended to both provide accurate time to end stations and be used by certain TSN standards to also deliver robust, deterministic, and low-latency data transfer services to those same end stations is a testament to the broad utility of robust, accurate time synchronization. Even if the TSN standards dedicated to reducing latency for time-sensitive data did not make use of gPTP for time synchronization, other applications, particularly of the end stations participating in the CPS or other distributed application logic, continue to drive the need for more accuracy, increased robustness, and protection against malicious agents within the network and among the end nodes. Just as all successful standards continue to expand in capability, there is need to continue enhancing PTP and related standards, as well as OS and software support for precision time within end stations that execute the time-coordinated application logic. In particular, further work is needed to further enhance and standardize Byzantine fault tolerance, time transfer over increasingly heterogeneous networks including 5G, vetting of the new security recommendations for time transfer by the various profiles of IEEE 1588 (including gPTP), and further interoperation or coexistence of the various current, legacy, and emerging profiles of IEEE 1588.

Acknowledgments

The author would like to recognize the fastidious and tireless work of Geoffrey Garner, Editor of the IEEE 802.1AS standard. Geoff is also acknowledged for his substantial contributions to IEEE 1588 and the timing community generally over many decades.

References

[1] J. E. C. B. D. B. L. G. B. I. E. A. L. a. K. S. M. Weiss, "Time-Aware Applications, Computers, and Communication Systems (TAACCS)," NIST Tech. Note 1867, Feb. 2015.
[2] IEEE Std. 802.1AS-2011, "IEEE Standard for Local and Metropolitan Area Networks — Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks."
[3] IEEE Std. 1588–2008, "IEEE Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems."
[4] G. M. G. a. H. Ryu, "Synchronization of Audio/Video Bridging Networks Using IEEE 802.1AS," *IEEE Commun. Mag.*, vol. 49, no. 2, Feb. 2011, pp. 140–47.
[5] T. H. M. F. T. S. a. K. D. A. Westenberger, "Temporal Synchronization in Multi-Sensor Fusion for Future Driver Assistance Systems," *IEEE Int'l. Symp. Precision Clock Synchronization for Measurement, Control and Commun.*, Munich, Germany, 2011.
[6] T. W. J. S. a. P. A. M. Lipinski, "White Rabbit: A PTP Application for Robust Sub-Nanosecond Synchronization," *IEEE Int'l. Symp. Precision Clock Synchronization for Measurement, Control and Commun.*, Munich, Germany, 2011.
[7] M. D. J. T. *et al.*, "Heterogeneous Networks for Audio and Video: Using IEEE 802.1 Audio Video Bridging," *Proc. IEEE*, vol. 101, no. 11, Nov. 2013, pp. 2339–54.
[8] "AudioVideoBridging.framework/AVB; https://developer.apple.com/library/content/documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemFrameworks/SystemFrameworks.html.
[9] EC, "RTS 25 on Level of Accuracy of Business Clocks," Brussels, 3316 final, ANNEX 1, 7.6.2016 2016.
[10] T. Mizrahi, "Security Requirements of Time Protocols in Packet Switched Networks," IETF RFC 7384, 2014.
[11] R. C. a. C. Marinescu, "Design and Implementation of a PTP Clock Infrastructure for the Linux Kernel," *IEEE Int'l. Symp. Precision Clock Synchronization for Measurement, Control and Communication*, Portsmouth, NH, 2010.
[12] K. B. S. J. C. Eidson, "Timing in Cyber-Physical Systems: The Last Inch Problem," *IEEE Proc. Int'l. Symp. Precision Clock Synchronization for Measurement, Control, and Commun.*, Oct. 2015, pp. 19–24.
[13] "Precision Time Measurement (PTM)," PCI SIG, 2013.
[14] A. S. *et al.*, "Time in Cyber-Physical Systems," *2016 Int'l. Conf. Hardware/Software Codesign and System Synthesis*, Pittsburgh, PA, 2016.
[15] http://www.avnu.org.

Biography

Kevin B. Stanton, Ph.D. (kevin.b.stanton@intel.com) is a senior principal engineer at Intel, where he is responsible for time-related standards, technologies, and their applications. He was a co-author of the published 802.1AS standard, and drove expansion of PTP support to 802.11 wireless links. He is also Chairman of the Avnu Alliance, and believes that accurate time will one day be made available to every important computing device in the galaxy.