Time Synchronization in Time-Sensitive Networking

Stefan Waldhauser, Benedikt Jaeger*, Max Helm*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
E-Mail: stefan.waldhauser@tum.de, jaeger@net.in.tum.de, helm@net.in.tum.de

***Abstract*—Time-Sensitive Networking (TSN) is an update to the existing Institute of Electrical and Electronics Engineers (IEEE) Ethernet standard to meet real-time requirements of modern test, measurement, and control systems. TSN uses the Precision Time Protocol (PTP) to synchronize the device clocks in the system to a reference time. This common sense of time is fundamental to the working of TSN. This paper presents the principles and operation of PTP and compares it to the Network Time Protocol (NTP).**

***Index Terms*—clock, network time protocol (NTP), precision time protocol (PTP), synchronization, time, time-sensitive networking (TSN)**

1. Introduction

In real-time systems, the correctness of a task does not only rely on the logical correctness of its result but also that the result meets some deadline [1]. A typical example of a real-time system is a control system in industrial automation that has to integrate multiple sensor readings and then initiate an action in response. This requires a deterministic underlying network.

Time-Sensitive Networking (TSN) is a set of standards that represent an ongoing effort of the Institute of Electrical and Electronics Engineers (IEEE) 802.1 TSN Task Group [2] to extend and adapt existing Ethernet standards on the data link layer to meet real-time requirements of modern test, measurement and control systems. Advantages over traditional Ethernet include guaranteed bounded latency for time-critical data while transmitting time-critical data and best-effort data on the same network [3].

The foundation of TSN is establishing a shared sense of time in all networked devices in the system. Only then they are able to perform time-sensitive tasks in unison at the right point in time. This shared sense of time is achieved through the use of the Precision Time Protocol (PTP) [3]. PTP is a message based time transfer protocol that enables clocks in the nodes of a packet-based network to synchronize (phase and absolute time) and syntonize (frequency) with sub-microsecond accuracy to a reference time source. PTP was first described in the IEEE 1588-2002 standard. The standard was later revised in IEEE 1588-2008 [4]. The revised protocol is commonly referred to as PTP Version 2 and is used in TSN together with the profiles IEEE 802.1AS and IEEE 802.1ASRev [5].

The paper starts with a description of PTP Version 2 in Chapter 2. The chapter begins with a brief overview of the two fundamental phases of the protocol: the best master clock algorithm and message based time synchronization. Then the various PTP device types and message types are discussed. The two phases are described in more detail in the rest of the chapter.

Next, in Chapter 3, PTP Version 2 is compared to another message based time synchronization protocol called Network Time Protocol (NTP) Version 4. Various advantages and disadvantages are discussed.

Finally, Chapter 4 concludes the paper.

2. Precision Time Protocol

An IEEE 1588 system is a distributed network of PTP enabled devices and possibly non-PTP devices. Non-PTP devices mainly include conventional switches and routers. An example PTP network can be seen in Fig. 1.

The operation of PTP can be conceptually divided into a two-stage process [6]. In the first stage, the PTP devices self-organize logically into a synchronization hierarchy tree using the Best Master Clock Algorithm (BMCA). The devices are continuously exchanging quality properties of their internal clock with each other. The PTP device with the highest quality clock in the system eventually assumes the role of grandmaster (GM) and provides the reference time for the whole system. The subnet scope in which all clocks synchronize to the GM is called PTP domain. The BMCA is explained further in Section 2.3.

In the second stage, time information continuously flows downstream from the GM between pairs of PTP ports with one port in the master state serving time information and the other in the slave state receiving time information. Eventually, the system reaches an equilibrium where all clocks are synchronized to the GM of the system. Time synchronization between master and slave is initiated by the master port, which periodically sends synchronization messages to its slave. These messages are timestamped by the master at transmission and by the slave at arrival. A slave now has two timestamps, the sending time according to the clock of the master, and the receiving time according to its clock. As the message takes some time to travel through the network, the slave also needs to know the network delay to calculate the offset to the master [6].

PTP supports two mechanisms to calculate this delay: End-to-End (E2E) and Peer-to-Peer (P2P). The E2E mechanism requires the slave to measure the total delay between itself and the master (thus end-to-end). The P2P mechanism, on the other hand, requires each device (including switches and routers) on the path between master and slave to measure the delay between itself and its

doi: 10.2313/NET-2020-04-1_10

direct neighbor (peer). The total network delay between master and slave is the sum of the peer delays along the path. Technically, E2E can be used in the same domain as P2P as long as the two are not mixed along the same messaging path. Thus, between master and slave, all nodes must either use E2E or P2P [7]. The two mechanisms are discussed in more detail in Section 2.5.

2.1. PTP Device Types

The standard defines five PTP device types: ordinary clocks, boundary clocks, end-to-end transparent clocks, peer-to-peer transparent clocks, and management nodes [4].

An ordinary clock (OC) is an end-device (as opposed to a switch or router) with a single PTP capable port and an internal local clock. It can either assume the role of slave (leave node) or GM (root node) in the synchronization hierarchy.

The main source of error in PTP is asymmetry in the network delay between master and slave. Asymmetric network delay means that sending a message from master to slave takes a different amount of time than the other way around. The most significant sources of asymmetric network delay are different processing and queueing delays in ordinary switches and routers, different data transmission speeds, error differences in the generation of timestamps, and messages taking different routes through the network [6].

IEEE 1588-2008 defines two types of PTP enabled switches and routers to deal with the asymmetry problem: Boundary clocks (BC) and transparent clocks (TC).

A BC has multiple PTP capable ports and one internal clock shared by all ports. If the BC is selected as the GM of the system, then all ports switch to the master state. Otherwise, the BC selects the best clock seen by all of its ports. The corresponding port then switches to the slave state, allowing the internal clock to synchronize. The other ports switch to the master state, serving time information based on the now synchronized internal clock. By terminating and then restarting the time distribution, each BC creates a branch point (internal node) in the synchronization tree. This allows the BC to effectively remove the adverse effects of its processing and queuing delays.

Like a BC, a TC has multiple PTP capable ports, and one shared internal clock. Eliminating asymmetry is achieved by timestamping the entrance and exit of PTP messages that pass through the device. The time the message spent inside the device, called residence time, is calculated by subtracting the entrance timestamp from the exit timestamp. The TC then adds the residence time to a correction field in the PTP message before passing it along. The slave can then remove the accumulated queuing and processing delays by using the correction field value in the offset calculation.

Transparent clocks exist in variants supporting either the P2P delay mechanism or the E2E delay mechanism. Only a single delay mechanism is allowed in the link between master and slave. A boundary clock with ports supporting each of the two mechanisms can be used to connect regions using the different mechanisms. See 1 for an example.

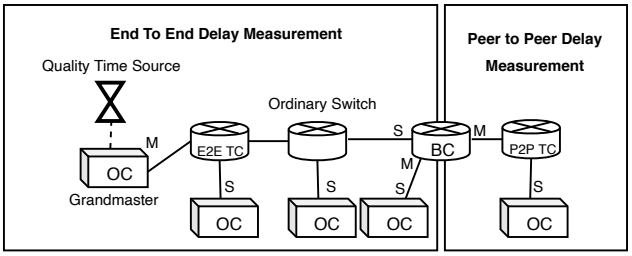

Figure 1: Example PTP Domain (Adapted from [8])

Management Nodes do not take part in the time synchronization but can be used to read and write various PTP properties of other nodes via *Management* messages [6].

2.2. PTP Message Types

IEEE 1588 defines two groups of PTP messages [4]: (1) Event messages, which require an accurate timestamp both at sending and receiving because PTP uses these as timing events. (2) General messages, which are being used to transmit information. In contrast to event messages, sending and receiving of general messages does not produce a timestamp.

The event message are *Sync*, *Delay_Req*, *Pdelay_Req* and *Pdelay_Resp*. These are used in the time synchronization process to transfer timestamps and correction information between master and slave. The general messages are *Announce*, *Follow_Up*, *Delay_Resp*, *Pdelay_Resp_Follow_Up*, *Management* and *Signaling*. *Announce* messages are used in the BMCA to exchange clock quality information. *Management* messages are used to configure PTP devices. *Signaling* messages are used by PTP clocks to communicate in special settings, such as unicast environments.

IEEE 1588 clocks can either support the one-step or two-step messaging mechanism. When sending *Sync* or *Pdelay_Resp* messages, clocks need to tell the receiver the sending timestamp. They are either capable of including this timestamp in the *Sync* and *Pdelay_Resp* themselves (one-step-clock), or they need to send a second follow-up message containing it (two-step-clock). *Follow_Up* and *Pdelay_Resp_Follow_Up* messages are used for this [9].

As mentioned before, any delay asymmetry causes a loss in accuracy. 'Artificial' network delay is created if the timestamps that are generated on the path from master to slave have a different error than those generated on the path from slave to master. This happens for example if software timestamping is used because the operating system and protocol stack packet processing delay fluctuates. Therefore it is recommended to use devices with PTP enabled NICs (Network Interface Cards) in the network. These specialized NICs have a clock, which is used to timestamp the received and transmitted PTP messages as close to the physical layer as possible [10]. Timestamps generated via hardware support have a constant low error and therefore improve synchronization accuracy.

PTP usually is implemented using multicast communication, but it can also be configured for unicast messaging. The PTP standard does not require any specific transport protocol, but most commonly, UDP is used. The well known UDP ports for PTP traffic are 319 (Event Message) and 320 (General Message) [11].

doi: 10.2313/NET-2020-04-1_10

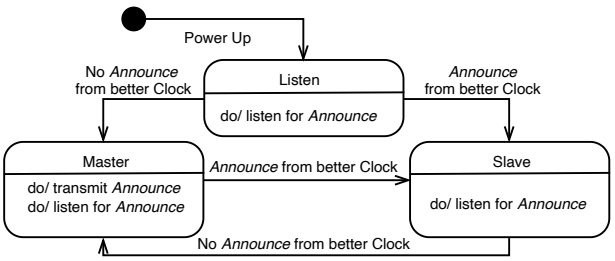

Figure 2: Simplified PTP State Machine of an Ordinary Clock (Adapted from [12])

2.3. Best Master Clock Algorithm

IEEE 1588 is an administration-free system that can deal with events like system restarts, failure of a clock, or changes in network topology automatically. This is achieved via the BMCA, which runs continuously in OCs and BCs in a domain [4].

The basics of the BMCA can be explained using the simplified state diagram of an OC in Fig. 2. All OCs listen for defined intervals to *Announce* messages, which are sent in a specific frequency by ports in the master state to the PTP multicast address. These messages contain attributes about the sending clock. At the end of each listening interval, an OC has either received an *Announce* message from a better clock or not.

The attribute comparison algorithm uses the following criteria in order of precedence to determine if an *Announce* message from a better clock has been received [12] [4]:

1) *priority1*: This is a user configurable field. It is the first parameter to be considered by the BMCA. Therefore an administrator can manually set up a clock quality hierarchy.
2) *clockClass*: This field generally described the quality of a clock. A clock connected to a GPS receiver has a higher class than a free-running clock.
3) *clockAccuracy*: This field describes the accuracy of the clock. The value is picked from defined accuracy levels in the standard, for example, 25 ns to 100 ns.
4) *offsetScaledLogVariance*: This field describes the stability of the clocks oscillator.
5) *priority2*: This is a user configurable field. It can be used to manually rank clocks of equal quality.
6) *clockIdentity*: This field is usually set to the Ethernet MAC address. It is a unique number that is used to break ties.

If a message from a better clock has been received, a master OC switches to the slave state. If no such message has been received, a slave OC switches to the master state and starts transmitting *Announce* messages. Freshly rebooted OCs are in a special listening state and can either switch to the master or slave state [12].

The process in BCs is similar, but these devices have to compare all of the *Announce* messages received on all the ports, to determine if they become a GM (all ports in master state) or just a branching point (one port in slave state and the others in master state) [12].

Eventually, only a single clock assumes the role of GM in the domain.

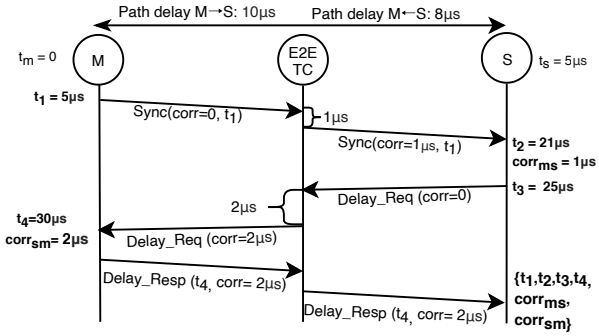

Figure 3: E2E Synchronization (One-Step-Clocks) (Adapted from [4])

It is important to note that the BMCA never stops running. This allows the system to react to certain events by dynamically changing the synchronization hierarchy. For example, if the current GM gets disconnected from the network, a new GM is determined automatically.

2.4. Syntonization

Syntonization in this context means frequency locking two clocks by agreeing on the length of a second. Syntonized clocks, therefore, are running at the same rate. This paper does not discuss the details of syntonization using PTP, but it is important to note that any port in the slave state and any TC syntonizes to the GM [4].

2.5. Synchronization

Time synchronization implies phase-locking two clocks and making them agree on the same time of day. Phase locking means that incrementing the time does not only happen at the same rate in both clocks but also at the same time. Agreeing on the time of day means synchronizing the 'wristwatch time' – year, month, day, hour, minute, seconds and so on in a given timezone. Any OC or BC with a port in the slave state synchronizes to its respective master in the hierarchy [4].

2.5.1. E2E Synchronization. Fig. 3 shows the message exchange to synchronize a one-step slave clock and a one-step master clock with an E2E transparent clock between them. In the example, there exist two sources of delay asymmetry: (1) A difference of 1 µs in the TC processing time. The negative effects of this asymmetry can be automatically removed. (2) A difference of 2 µs that has its origin in transmission speed or path length differences. PTP can not automatically remove the influence of this asymmetry. However, if measured manually, PTP can be configured to account for it [6].

As seen in the example, the slave collects four timestamps and two correction values during the message exchange:

- t_1: *Sync* sending timestamp in master time. In a two-step clock this timestamp is contained in a separate *Follow_Up* message and not in the *Sync* message itself.
- $corr_{ms}$, $corr_{sm}$: Each TC on the path adds the residence time to the correction field in the *Sync* or *Delay_Req* message

doi: 10.2313/NET-2020-04-1_10

- t_2: *Sync* receiving timestamp in slave time
- t_3: *Delay_Req* sending timestamp in slave time
- t_4: *Delay_Req* receiving timestamp in master time

The fundamental assumption of all synchronization protocols that are based on the exchange of timing information via networks with unknown propagation delays is a symmetric network delay between master and slave [6].

Under this assumption the slave is able to calculate the network delay d between itself and the master by dividing the corrected round-trip delay by two:

$$d = \frac{[(t_4 - t_1) - (t_3 - t_2)] - corr_{ms} - corr_{sm}}{2} \tag{1}$$

This assumption is critical since it is not possible to determine one-way delays with an unknown clock offset.

In the example:

$$d = \frac{(30\,\mu s - 5\,\mu s) - (25\,\mu s - 21\,\mu s) - 1\,\mu s - 2\,\mu s}{2} = 9\,\mu s$$

The slave can now calculate the offset o from the master by subtracting from t_2 (slave time): t_1 (master time), the network delay, and the TC correction factor. The result represents the part of the timestamp difference that originates from the slave and master clock divergence.

$$o = t_2 - t_1 - d - corr_{ms} = 21\,\mu s - 5\,\mu s - 9\,\mu s - 1\,\mu s = 6\,\mu s \tag{2}$$

The actual offset is 5 µs, so there is an error of 1 µs. This error occurs because the above assumption was wrong: There is uncorrected asymmetry in the delay between master and slave of 2 µs. However, the example demonstrated that PTP is successfully able to remove the amount of asymmetry stemming from queue effects in ordinary switches and routers by replacing them with TCs.

In general for the error e:

$$e = \frac{ND_{ms} - ND_{sm}}{2} = \frac{10\,\mu s - 8\,\mu s}{2} = 1\,\mu s \tag{3}$$

The maximum possible error due to asymmetry in the network is, therefore, half of the round-trip delay.

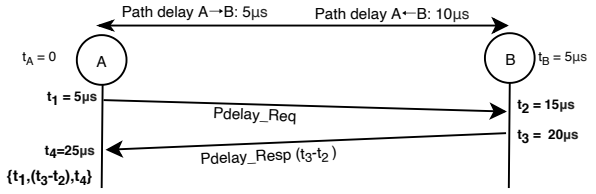

Figure 4: P2P Delay Measurement (One-Step-Clocks) (Adapted from [4])

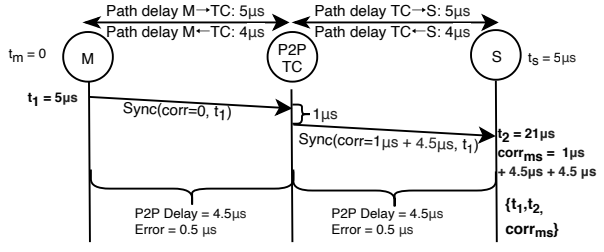

Figure 5: P2P Synchronization (Adapted from [4])

2.5.2. P2P Synchronization. A link between master and slave that is set up to use P2P synchronization calculates the network delay differently. Periodically two directly connected clocks independent of their state perform a message exchange to measure the network delay between them. An example is shown in Fig. 4.

Four timestamps are generated that are used to calculate the network delay:

$$\begin{aligned} d &= \frac{[(t_4 - t_1) - (t_3 - t_2)]}{2} \\ &= \frac{(25\,\mu s - 5\,\mu s) - (20\,\mu s - 15\,\mu s)}{2} = 7.5\,\mu s \end{aligned} \tag{4}$$

The error is again half of the network delay asymmetry between clock A and clock B. This peer delay is measured for both directions. This is important because during the lifetime of the system, the master-slave states of A and B can change.

Fig. 5 shows an example of time synchronization between master and slave using the P2P mechanism with the same delay values as in the E2E case. The timestamps t_1 and t_2 are still created by sending a *Sync* message from master to slave, but the network delay is calculated differently.

Each clock on the link that receives the *Sync* message adds the peer delay value to the correction field. In addition, the TCs add the residence time to the correction field as usual. The correction field, therefore, always represents the network delay from the master until the current node.

The slave adds the final peer delay to the correction field and can now calculate the offset to the master:

$$\begin{aligned} o &= (t_2 - t_1) - corr_{ms} \\ &= (21\,\mu s - 5\,\mu s) - 10\,\mu s = 6\,\mu s \end{aligned} \tag{5}$$

The error is the same as in the E2E example because the total error is just the sum of all errors made during the peer network delay calculation.

Even though no higher precision can be achieved using the P2P mechanism, there are several other factors to consider [7]:

- Ordinary switches and routers do not respond correctly to *Pdelay_Req* messages, in case such devices are used in the network, the E2E mechanism has to be used.
- As the master does only need to respond to *Pdelay_Req* messages from its direct neighbors and not to *Delay_Req* messages from all the slaves that sync to it, a P2P system scales much better. The load on a master that a lot of slaves sync to is dramatically reduced.
- As no *Delay_Req* messages are used, there is no risk of the *Sync* and *Delay_Req* message taking different paths in the network. Thus the risk for delay asymmetry is reduced.

3. Related Work

PTP was designed for usage in local industrial automation and measurement networks where specialized devices like BCs and TCs can be used as switches and routers. Another protocol called Network Time Protocol (NTP), on the other hand, is the workhorse for synchronizing system clocks of devices over the Internet to a common timebase

doi: 10.2313/NET-2020-04-1_10

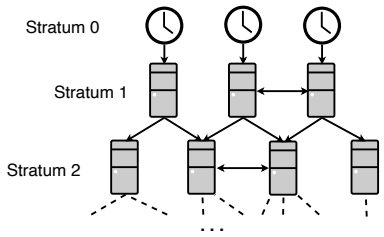

Figure 6: Example NTP Hierarchy

(usually UTC). NTP time synchronization is used, for example, in general-purpose workstations and servers. It is one of the oldest (the first version was released in 1985) protocols still in use today and is currently in its fourth major version. NTP uses UDP on port 123 [13].

Similar to PTP, an NTP network (for example, the global Internet) is hierarchically organized into primary servers, which are directly connected to a reference clock, secondary servers, and clients. In NTP, there also exists the concept of a stratum which represents the logical distance of a server/client to a reference clock. Primary servers have a stratum value of 1 and secondary servers values between 2 and 15. If a server has a stratum value of 16, it means that it is not yet synchronized. A server in stratum n is a synchronization client to a server in stratum $n-1$. In real-world configurations, stratum levels above 4 are rare [14]. Fig. 6 illustrates the hierarchical strata model of NTP. To increase robustness, two NTP servers in the same stratum can also synchronize with each other as peers. If a server loses connectivity to its upstream NTP server, it can receive time information from its peers.

Time synchronization of an NTP client is established through periodic request/reply exchange with one or more NTP servers they are authorized to access. As in PTP, the offset of the client clock to the server clock is calculated from the four timestamps generated during the exchange. The critical source of error is again the delay asymmetry between the two messaging directions.

A key advantage of NTP over PTP is that in NTP, a client polls typically many servers for time synchronization. In case of disagreements between the sources, the most extensive collection of agreeing servers is used to produce a combined reference time, thereby declaring other servers as faulty or not trustworthy [15]. PTP slaves, on the other hand, are trusting a single time source blindly [16]. Slaves can only assume that the calculated offset to the master is correct, as they are not capable of comparing it to some value from other sources. This means that if the GM has some error that causes it so send the wrong time information in *Sync* messages but that does not affect the clock quality presented in *Announce* messages, slaves change their clock to the wrong time. Several researchers have proposed protocol modifications to increase PTP robustness [17], including giving slaves the ability to check the calculated offset against time information from multiple NTP servers [18].

Another area where NTP has an advantage over PTP is security. NTP supports authentication with symmetric keys or public/private certificate pairs to allow clients to verify the authenticity and integrity of received messages. The standard IEEE 1588-2008, on the other hand, does not include any fully defined security model [19]. Security was not a priority in the development of PTP due to the typical use case under consideration at the time, i.e. time synchronization in closed local area networks (LANs) [20]. This means that in PTP, it is not possible to verify the authenticity and integrity of the critical *Announce* and *Sync* messages. This allows a malicious actor to influence the BMCA or time synchronization. Security researchers have shown that it is possible to cause a major disturbance in PTP synchronization via an *Announce* message Denial of Service attack on a slave. They were even capable of taking control of the whole PTP domain by creating an evil grandmaster, that claims better quality than other alternatives [21].

What PTP lacks in terms of robustness and security, it makes up for in accuracy. Typical accuracy expectations of PTP are in the order of 100 ns [22] while the typical values for NTP accuracy over the Internet range from 5 ms to 100 ms [23] if there is considerable delay asymmetry, such as when one direction is via satellite and the other via broadband.

One might ask why PTP accuracy and NTP accuracy differ so much when the protocols use an almost identical message exchange to calculate the clock offset. The difference in typically achieved synchronization accuracy has its origin in the vastly different networking environments the two protocols are used in.

PTP is primarily used in lightly loaded high-speed LANs. In these networks, overhead is of little concern, and update intervals of a few seconds or less can be used. Clocks lose their synchronicity over time because of changes in the physical environment (primarily temperature and barometric pressure) that affect the oscillator [24]. High-frequency update intervals allow clocks to re-synchronize faster. NTP requires long update intervals of one minute to several hours to minimize load on the typically heavily used network [22] [24]. NTP also operates in wide area networks (WAN), where differences in network speeds and routing paths are common sources of delay asymmetry. Furthermore, a significant amount of delay asymmetry can be removed from a PTP network by using only clocks that support hardware timestamping and connecting them exclusively via TCs or BCs. While hardware timestamping in clients and servers is rare but possible, NTP supports no mechanism for removing variations in queuing time in switches and routers [25].

Theoretically, a new version of NTP that uses the same delay asymmetry reduction strategies as PTP could be developed. If this were done, NTP could reach the same levels of accuracy and precision as PTP [22]. However, the current research focus is on improving PTP, rather than developing a more precise NTP.

IEEE 1588-2019 (PTP Version 2.1) is currently in the works. This new version addresses some of the robustness and security issues of PTP by enabling message and source integrity checking [26]. The next protocol version also allows sub-nanosecond accuracy and picoseconds precision of synchronization by incorporating the White Rabbit extension [27], which was developed at CERN, into the standard as a new configuration profile [28]. IEEE 1588-2019 is likely going to be released in early 2020 [29].

doi: 10.2313/NET-2020-04-1_10

4. Conclusion

Choosing PTP as the time synchronization protocol for the important TSN effort, established PTP as the most important protocol for synchronizing clocks in real-time networks. PTP achieves high accuracy not by a novel way of calculating the offset of a clock, but through hardware timestamping and the usage of specialized network infrastructure devices. PTP currently lags behind NTP in the areas of robustness and security. Substantial changes to the protocol are needed to improve the protocol in these areas. It will be interesting to see if the next version IEEE 1588-2019 makes it possible to get both accuracy and security at the same time.

References

[1] K. G. Shin and P. Ramanathan, "Real-Time Computing: A New Discipline of Computer Science and Engineering," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 6–24, Jan 1994.

[2] IEEE 802.1 TSN Task Group. [Online]. Available: https://1.ieee802.org/tsn/

[3] A. Weder, "Whitepaper: Time Sensitive Networking," Tech. Rep. [Online]. Available: https://www.ipms.fraunhofer.de/de/press-media/whitepaper-download/TIME-SENSITIVE-NETWORKING-An-Introduction-to-TSN.html

[4] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, July 2008.

[5] "Whitepaper: Time Sensitive Networking," Tech. Rep. [Online]. Available: https://www.cisco.com/c/dam/en/us/solutions/collateral/industry-solutions/white-paper-c11-738950.pdf

[6] J. Eidson, *Measurement, Control, and Communication Using IEEE 1588*, ser. Advances in Industrial Control. Springer London, 2006.

[7] End-to-End Versus Peer-to-Peer. [Online]. Available: https://blog.meinbergglobal.com/2013/09/19/end-end-versus-peer-peer/

[8] The IEEE 1588 Default Profile. [Online]. Available: https://blog.meinbergglobal.com/2014/01/09/ieee-1588-default-profile/

[9] One-step or Two-step? [Online]. Available: https://blog.meinbergglobal.com/2013/10/28/one-step-two-step/

[10] PTP's Secret Weapon: Hardware Timestamping. [Online]. Available: https://www.corvil.com/blog/2016/ptp-s-secret-weapon-hardware-timestamping

[11] Protocols/ptp - The Wireshark Wiki. [Online]. Available: https://wiki.wireshark.org/Protocols/ptp

[12] What Makes a Master the Best? [Online]. Available: https://blog.meinbergglobal.com/2013/11/14/makes-master-best/

[13] NTP - The Wireshark Wiki. [Online]. Available: https://wiki.wireshark.org/NTP

[14] Sun Blueprint: Using NTP to Control and Synchronize System Clocks - Part I: Introduction to NTP. [Online]. Available: http://www-it.desy.de/common/documentation/cd-docs/sun/blueprints/0701/NTP.pdf

[15] Combining PTP with NTP to Get the Best of Both Worlds. [Online]. Available: https://www.redhat.com/en/blog/combining-ptp-ntp-get-best-both-worlds

[16] P. V. Estrela and L. Bonebakker, "Challenges deploying PTPv2 in a global financial company," in *2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings*, Sep. 2012, pp. 1–6.

[17] M. Dalmas, H. Rachadel, G. Silvano, and C. Dutra, "Improving PTP robustness to the byzantine failure," in *2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, Oct 2015, pp. 111–114.

[18] P. V. Estrela, S. Neusüß, and W. Owczarek, "Using a multi-source NTP watchdog to increase the robustness of PTPv2 in financial industry networks," in *2014 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, Sep. 2014, pp. 87–92.

[19] RFC 7384 - Security Requirements of Time Protocols in Packet Switched Networks. [Online]. Available: https://tools.ietf.org/html/rfc7384

[20] K. O'Donoghue, D. Sibold, and S. Fries, "New security mechanisms for network time synchronization protocols," in *2017 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, Aug 2017, pp. 1–6.

[21] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. Wojciak, and S. Guendert, "Impact of Cyberattacks on Precision Time Protocol," *IEEE Transactions on Instrumentation and Measurement*, pp. 1–1, 2019.

[22] IEEE 1588 Precision Time Protocol (PTP). [Online]. Available: https://www.eecis.udel.edu/~mills/ptp.html

[23] How does it work? [Online]. Available: http://www.ntp.org/ntpfaq/NTP-s-algo.htm

[24] D. Mills, *Computer Network Time Synchronization: The Network Time Protocol on Earth and in Space, Second Edition*. CRC Press, 2017.

[25] NTP vs PTP: Network Timing Smackdown! [Online]. Available: https://blog.meinbergglobal.com/2013/11/22/ntp-vs-ptp-network-timing-smackdown/

[26] What's coming In the Next Edition of IEEE 1588? [Online]. Available: https://blog.meinbergglobal.com/2017/09/24/whats-coming-next-edition-ieee-1588/

[27] M. Lipiński, T. Włostowski, J. Serrano, and P. Alvarez, "White rabbit: a PTP application for robust sub-nanosecond synchronization," in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Sep. 2011, pp. 25–30.

[28] White Rabbit Official CERN website. [Online]. Available: http://white-rabbit.web.cern.ch/Default.htm

[29] iMeet Central. [Online]. Available: https://ieee-sa.imeetcentral.com/1588public/