Supplementary Material II

This document is written as a part of supplementary materials for the paper titled “Clock Offset Estimation for Systems with Asymmetric Packet Delays” submitted for publication in IEEE/ACM Transactions on Networking.

This document provides a list of recent studies related to PTP that were not covered or were intentionally omitted in the manuscript. For Case I and Case II, we used the same citation number as the revised manuscript. For the other case, we used temporary numbers starting with R because they were not cited in the manuscript.

Selected recent studies related to the IEEE 1588 PTP (2017–present):

Category I (Standard)

[18] “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” *IEEE Standard 1588-2019.*

Description. The latest IEEE 1588-2019 standard [18] is a minor revision of the IEEE 1588-2008 standard [7]. The latest standard newly includes the optional specifications for asymmetry calibration and timestamp correction. The calibration feature has been added for certain media and provides a nearly constant relation between the transmissions in both directions. Additionally, the standard continues to adopt PTPv2-enabled devices. A brief description of the new features is provided on the web page linked to the address below.

http://sagroups.ieee.org/1588/wp-content/uploads/sites/144/2020/04/IEEE-1588-2019-changes.pdf

Category II (Studies on mitigating the error that stems from the asymmetry of packet delays)

[19] H. Puttnies, et. al., ”PTP-LP: Using Linear Programming to Increase the Delay Robustness of IEEE 1588 PTP,” *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018.

Description. The study presents the concept of PTP-LP, which uses linear programming (LP) with sets of PTP timestamps to explore the linear upper and lower bounds of a slave clock $C(t)$. The value of the slave clock is subsequently estimated by averaging the explored upper and lower bounds.

PTP-LP assumes that the two respective functions defined by the two sets of PTP timestamps $(t_1, C(t_2))$ and $(C(t_3), t_4)$ are linear. However, in the real world, the functions are not often linear because of random packet delays and clock skew. In practical scenarios where the packet delay and clock skew are randomly time-varying, the explored bounds cannot be applied properly and the performance of PTP-LP can degrade.

[20] Y. Geng, et al., ”Exploiting a natural network effect for scalable, fine-grained clock synchronization,” *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018.

Description. Y. Geng, et al. proposed an approach called HUYGENS. For fine-grained clock synchronization, the HUYGENS has the following characteristics:

1. Exchanges packets every 500 µs or 4 ms.
2. Rejects the impure packets that experience queuing delays, random jitters, and timestamping noise.
3. Uses a support vector machine with pure packets to estimate the clock offset and clock skew.
4. Constructs a reference spanning tree (RST) from the network graph.
5. Performs an algebraic loop correction by exploiting a network effect on the RST.

The HUYGENS can be applied in situations where it can filter out packets that do not suffer from queuing delays, random jitters, and timestamping noise. Moreover, this approach can only be used in applications such as data centers that allow for huge amounts of packet exchange for clock synchronization.

[21] M. Goodarzi, et al., "Synchronization in 5G networks: a hybrid Bayesian approach toward clock offset/skew estimation and its impact on localization," *EURASIP Journal of Wireless Communications and Networking*, vol. 91, 2021.

Description. M. Goodarzi, et al. proposed an approach to mitigate synchronization errors resulting from the uncertainty in timestamping. By investigating asymmetric transmission and reception delays, the authors constructed a pairwise clock synchronization and insisted that this is suitable for the edge of a network, in which fast and frequent synchronization is required to maintain a low relative time error. The authors also argued that another synchronization based on belief propagation can be used for the synchronization of backhaul nodes.

The pairwise clock synchronization reported in this study is designed under the assumption that the other components of the packet delays from master to slave and vice versa are symmetric. For example, their estimation assumes that the queuing delays of the forward and backward paths are equal.

Category III (Efforts to implement the PTP synchronization in wireless environments)

[R1] P. Chen and Z. Yang, "Understanding Precision Time Protocol in Today's Wi-Fi Networks: A Measurement Study," *2021 USENIX Annual Technical Conference (ATC)*, pp. 597-601, 2021.

Description. P. Chen and Z. Yang argued that there are restrictions on purchasing or using a commercial, off-the-shell (COTS) WNIC with the publicly available hardware timestamping interface for PTP. They proposed a method to make use of a hardware function for local positioning to emulate the PTP clock.

[R2] A. Garg, et. al., "Wireless Precision Time Protocol," *IEEE Communications Letters*, vol. 22, no. 4, pp. 812-815, 2018.

Description. A. Garg, et. al. proposed WPTP (Wireless PTP), a PTP implementation design using the characteristics of Wi-Fi broadcasting. The main goal of WPTP is to reduce the number of packets and convergence time of PTP in wireless environments. WPTP neither improves nor worsens the synchronization accuracy of PTP.

[R3] D. Shrestha, et al., ”Precise Clock Synchronization in High Performance Wireless Communication for Time Sensitive Networking,” *IEEE Access*, vol. 6, pp. 8944-8953, 2018.

Description. D. Shrestha, et al. proposed a wireless PTP which incorporates a clock skew estimation. Their approach was designed under two assumptions: i) the packet delays of *sync* and *follow_up* messages are the same and ii) the *follow_up* message can include its own accurate Tx timestamp of the message. However, as mentioned in our paper, the first assumption is difficult to satisfy in practice. Moreover, *follow_up* cannot contain its own accurate Tx timestamp because the exact Tx time is determined after the message leaves the node.