Enabling TSN over IEEE 802.11: Low-overhead Time Synchronization for Wi-Fi Clients

Jetmir Haxhibeqiri*, Xianjun Jiao*, Muhammad Aslam*, Ingrid Moerman* and Jeroen Hoebeke*
* IDLab, Ghent University – imec, Ghent, Belgium
Email: [firstname.lastname]@ugent.be

***Abstract*—The ever-increasing need for real-time communication of factory processes in one hand, and the offered flexibility of wireless communication on the other, is pushing Time Sensitive Networking (TSN) evolution towards the wireless networks. By definition, wireless networks are non-deterministic due to their random channel access mechanism. In order to introduce TSN vision to the wireless world, such randomness needs to be controlled. In this paper, we implement a low-overhead beacon-based time synchronization mechanism that offers synchronization accuracy of as low as 10 μs, on average, for low beacon interval. Such accuracy is achieved by implementing a follow-up beacon packet, in addition to beacon mechanism itself, in order to account for any delays in sending beacons. The synchronization mechanism is tested in different intra and inter access point communication settings.**

***Index Terms*—TSN, time synchronization, WiFi, beacon based**

I. Introduction

Traditionally, factory devices have been using field buses to enable critical communication between them, thus, fulfilling the reliability, data rate, and time requirements. Over time, such communication has been replaced by the Real-Time Ethernet (RTE) that is based on the IEEE 802.3 standard. IEEE 802.3 standard by its CSMA/CD channel access mechanism is non-deterministic by definition, violating the requirement of real-time communication due to its random introduced delays. Consequently, to provide determinism and real-time communication, IEEE 802.3-based legacy networks have employed a set of Time-Sensitive Network (TSN) standards.

Despite its benefits of offering mobility and reducing deployment costs, wireless communication is merely used for non-critical none-time sensitive industrial communication. The ever-increasing need for real-time communication and accurate control of industrial processes is pushing TSN evolution further to wireless networks as well. To support TSN over wireless networks, low-overhead highly-accurate time synchronization methods need to be extended beyond wired network [1].

The most widely used wireless standard is IEEE 802.11, which, similarly to IEEE 802.3, is non-deterministic due to its channel access mechanism that is based on CSMA/CA. Consequently, IEEE 802.11 can not be used for critical factory communication as is. Several time-division multiple access (TDMA) mechanisms have been proposed in the past [2], [3], to enable determinism and reliability in IEEE 802.11. Some of them were concentrated in the downlink only, leaving aside the need for determinism in uplink too. Nevertheless, the determinism should be ensured for end-to-end communication, employing wired-wireless interaction.

The first step towards end-to-end TSN is a low-overhead and accurate distribution of time information to all entities in the network. Especially the overhead needs to be minimized in the wireless part of the network. To this end, in this paper, we will show a low-overhead beacon-based synchronization algorithm. We will use openwifi platform [4], [5] to implement a beacon follow-up mechanism that increases the accuracy.

The paper is organized as follows. First of all, in section II we will give an overview of related research work. In section III we will discuss the time synchronization mechanism and its implementation based on beacons. Section IV will describe the measurement methodology for synchronization accuracy measurements. In section V we will show the results of time synchronization and scheduling accuracy. Finally, section VI concludes the paper.

II. Related Works

We will focus the discussion in this section on two aspects: studies that are related to extending the time synchronization boundaries to wireless networks and studies related to wired-wireless solutions that enable inter-segment end-to-end deterministic communication. The latter one depends heavily on the time synchronization accuracy to achieve end-to-end determinism.

Precise Time Protocol (PTP) [6] is used in the wired networks to offer sub-μsecond time accuracy between nodes. Its accuracy depends heavily on the way how timestamping is done on the node itself. Its accuracy decreases when the software timestamping is used. PTP relies on the estimation of the time-of-arrival (ToA) of the synchronization packet. To this end, a small variation of the channel impulse response (CIR) produces variation of the ToA, introducing thus synchronization errors in the PTP. In [7] authors offer an enhanced timestamping method for increased synchronization accuracy under channel variation conditions for the wireless link. The enhanced timestamping method uses the CIR to accurately estimate the ToA in line-of-sight (LOS) and none-LOS (NLOS) scenarios [7]. While the synchronization accuracy is increased compared to standard PTP, the solution requires changes at the hardware level, which in many cases would not be feasible as the client nodes are already deployed in the network. Contrary,

we try to limit the changes at the client side only to software changes to surpass PTP synchronization accuracy over wireless links. In [8] authors implement the aforementioned enhanced method and validate it in an SDR implementation of both access point (AP) and client side.

In [9] and [10] authors propose to synchronize the timing of AP and client to their respective time synchronization function (TSF) timers. By default, as the TSF timers of the AP and the client are synchronized, the timing of AP and client will be synchronized too. However, the problem with such a solution is that the reference time is the TSF timer itself, and not the actual time of the AP. The main limitation is that it provides internal synchronization in a wireless network with no possibility to extend the synchronization to the wired network or to other APs.

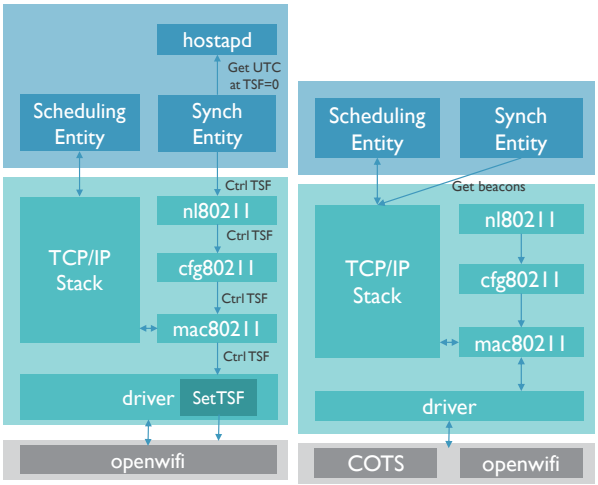
Authors in [2] offer TDMA MAC protocol on top of 802.11 hardware with the possibility of μsecond time synchronization between nodes. The synchronization method is based on three-way handshaking between nodes to determine the pairwise clock offset as well as round-trip delay. Such an approach is not standardized and is applicable only for mesh networks, leaving open questions regarding its scalability for the managed modes where a single AP will serve several end clients.

In [11] presents a hybrid wired/wireless centralized architecture approach for ensuring determinism for wireless industrial applications. They propose a node architecture for deterministic AP that organizes traffic from and to end-nodes using a TDMA-based MAC on top of IEEE 802.11 physical layer. The solution is evaluated on the network simulator. Similarly, [3] presents a hybrid usage of TDMA and CSMA on wireless links. The TDMA is achieved only in downlink by changing the power management flag for certain clients at the AP. Still, this approach will not require tight synchronization between nodes at the expense of achieving only downlink determinism.

Contrary to PTP-based synchronization mechanisms in [7], [8], where delay request-response is used, the presented solution in this paper uses beacon-based mechanism for time synchronization that keeps the overhead limited in the wireless channel. In addition to research presented in [9], [10], we extend the time synchronization to inter-AP scenarios by controlling AP TSF from the wired network.

III. Synchronization Mechanism and its Implementation

In IEEE 802.11 beacons are used in both mesh as well as managed typologies to offer possibilities for the new devices to discover already present networks. Each beacon contains parameter information, such as SSID, supported data rates, operational channel, traffic indication map (TIM) for power saving mode clients, etc. In addition to this, beacons contain also the TSF that is a 64-bit 1 μsecond counter that is used to synchronize the timings between WiFi devices. In the managed mode beacons are transmitted by APs to advertise the presence of the network. WiFi clients and AP are synchronized based on TSF by letting each client updating its TSF counter based on the TSF counter received on the last beacon from the AP. Such synchronization is used only by the WiFi cards themselves to correctly calculate the time, like e.g. when the network allocation vector (NAV) time ends. Consequently, such synchronization is unaware of the time in the host itself and does not change the host time itself.



(a) Access point (AP) (b) Client node

Fig. 1. The wireless node architecture

In the 2012 standard amendments [12], a new information element, time advertisement (TA) tag, was introduced in the beacon packet to relate the TSF counter with the time of the host itself. The time advertisement can be encoded in two ways, either as the Coordinated Universal Time (UTC) time when the TSF counter was zero or as a timestamp offset to the TSF. Though the mechanism exists, the standard does not mention how such information can be used by the other hosts to synchronize their times, neither how the host or AP can control the TSF counter themselves.

To implement the beacon-based time synchronization the following components are needed: the protocol stack that supports the TA tag on the beacon, the component that will send beacon follow-up packets (when TSF is not set by the PHY itself), a controlling entity (a SYNCH-APP) that will initiate the control of the TSF counter by the driver, and a processing entity (a client SYNCH-APP) that will process received beacons and follow-up packets to control the time of the host.

Software Defined Radio (SDR) platforms are feasible as they offer higher flexibility in both MAC and PHY layer parameters that users can control, in contrast to commercial chips. Thus, for our APs we used openwifi platform [4], where we implemented the mechanism to control the TSF timer and send the follow-up beacon packets.

The AP node architecture is shown in Figure 1a. The synchronization application entity is implemented using Click modular router [13], while hostapd[1] is used for the AP. The synchronization application periodically initiates the control

[1]https://w1.fi/hostapd/

of the TSF counter by the driver level. The actual TSF counter is calculated at the driver level (nearer to the card itself) to avoid any stack delay impact in the calculated TSF value. To determine the current TSF timer, the synchronization application entity needs to know the exact UTC when the TSF counter was initialized. For this, we implemented an additional control message in the hostapd itself to get such information. Consequently, the synchronization application will use hostapd control interface to get the UTC when the TSF counter was initialized and pass such information to the driver level. The driver itself will calculate the actual TSF counter, based on current UTC, and set it.

The beacon is *timestamped* with the TSF counter at the driver level. Consequently, the beacon TSF counter is not the actual TSF counter when the beacon is transmitted. Even-though the transmission of the beacon packet does not follow the random channel access mechanism, it can be delayed due to the clear channel assessment (CCA) mechanism when the channel is found busy and by the PCF inter-frame space (PIFS) time. Thus, relying only on the beacon to synchronize, the error will be up to PIFS time in the best case, not accounting for other errors introduced by busy channel under high loads. Moreover, the synchronization error between two clients from two different APs can be as high as twice of PIFS time. Thus, a follow-up beacon packet is implemented that will inform the host for the actual TSF counter when the beacon left the AP radio. Once the beacon is transmitted, the actual TSF timer is collected by the driver level when the transmission interrupt happens generating the beacon follow-up packet. The beacon follow-up packet will contain only two fields as payload: the TSF counter and the beacon sequence number to relate the follow-up packet with the beacon packet. The synchronization entity can choose the periodicity of follow-up beacons too, reducing the overhead further. On the client side, the synchronization application entity at the user-level will process beacons and follow-up packets to adjust the host system time. For the client side, commercial chips can be used as there is no need to control any of the low-level parameters as in AP. The client node architecture is shown in Figure 1b. The synchronization application will rely on the provided received timestamp for beacons by the underlying layers.

Let T_0 be the actual time when the TSF timer was initialized, TSF_1 be the timestamp of the beacon, and TSF_2 be the timestamp when the beacon will start to be transmitted. TSF_2 will be sent to the receiver side using the follow-up beacon. The receiver side will save the timestamp when the beacon was received, TS_1 and when the follow-up beacon was received TS_2. The time elapsed from the point when the beacon was timestamped until its reception at the receiver will be:

$$T_{elapsed} = TSF_2 - TSF_1 + TX_{time} \tag{1}$$

where TX_{time} is the time it takes to transmit the beacon packet and can be calculated at the receiver side based on the data rate used and the length of the packet. As all the values mentioned are known to the receiver after receiving

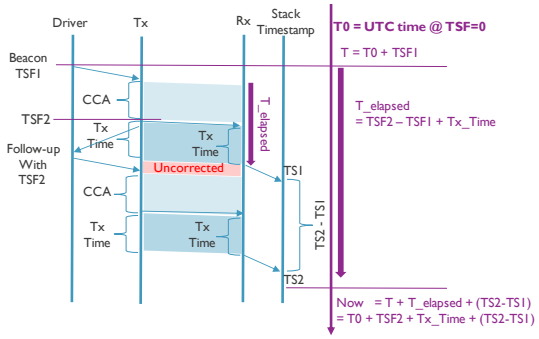


Fig. 2. Synchronization mechanism time diagram

the follow-up beacon, the receiver can calculate the current time by the following formula:

$$\begin{aligned} T_{now} &= T_0 - T_{elapsed} + TS_2 - TS_1 \\ &= T_0 + TSF_2 + TX_{time} + TS_2 - TS_1 \end{aligned} \tag{2}$$

In Figure 2 the time diagram of synchronization mechanism is shown. As shown in the figure there might be an uncorrected time difference between clients and the AP due to delays in the timestamping at the receiver side, however, this delay will be negligible and will not accumulate over time.

In case when multiple APs are used in the network, they need to be synchronized by the wired network. To achieve network-wide time synchronization, it is assumed that one of the synchronization possibilities in the wired network is used [14], [15].

IV. Time Synchronization Measurement Setup

With regard to validating the accuracy of synchronization mechanism, a measurement methodology that is independent of the synchronization mechanism itself has to be designed. Thus, we make use of a parallel path to validate the synchronization accuracy.

In PTP Linux implementation[2] the synchronization mechanism itself reports the time offsets to the master clock. In beacon-based synchronization application, attention is given also to the time offsets between two different wireless devices at a certain time. Moreover, as beacon intervals are comparably smaller than re-synchronization periods in PTP, reporting offsets every beacon interval will be quite an overhead. Thus, we use a parallel wired path to periodically measure the time difference between client nodes to validate the mechanism.

The measuring setup is shown in Figure 3. All nodes under test (AP and clients) will be connected to a wired test network. In the wired test-network, a single node will send broadcast packets periodically. Such packets will be received at all other nodes under test and the receiving timestamp will be logged by each node. Then, the time difference between logged information will show the time offset between respective nodes. Measurements do not dependent on any processing stack delay of the transmitting node, while on the receiving node they are

[2]http://linuxptp.sourceforge.net/

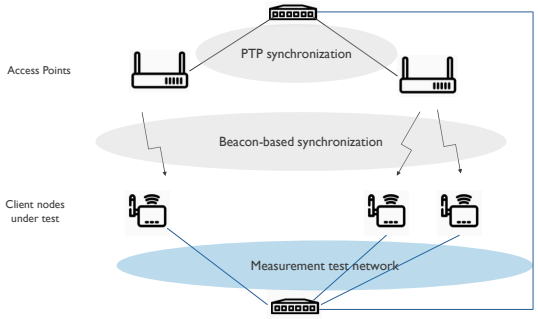
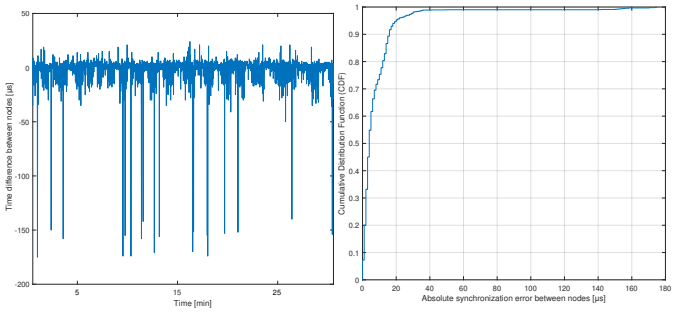
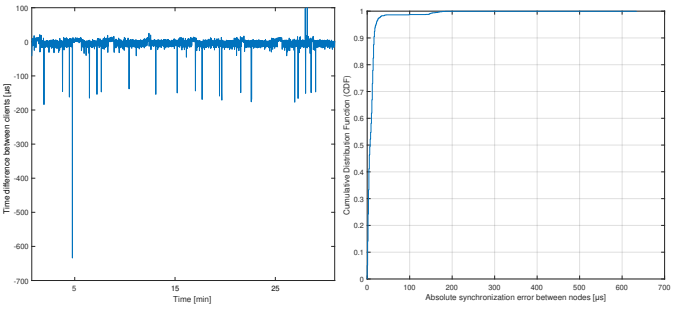


Fig. 3. Measurement test setup



(a) Time difference between clients (b) CDF of absolute synchronization error

Fig. 4. Synchronization error between clients when TSF is uncontrolled



(a) Time difference between clients (b) CDF of absolute synchronization error

Fig. 5. Synchronization error between clients when TSF is controlled by PTP

timestamped at the driver level (the lowest possible point in the stack). For inter-AP synchronization measurements, two APs are synchronized using PTP over the wired network, and the time difference between clients connected to different APs will be evaluated similarly.

The time difference between devices is calculated as $\delta_{ij} = Rx_i - Rx_j; i \neq j$, where i and j are two different client nodes in the network. The standard deviation, σ_{ij}, represents the time difference jitter between nodes, while μ_{ij} represents the mean value of the difference and pr_{90} represents the 90^{th} percentile of the difference.

V. Results

In the following, the time synchronization mechanism is validated in different intra and inter-AP settings, against the time difference between clients over time, its cumulative distribution function (CDF), and its mean and standard deviation.

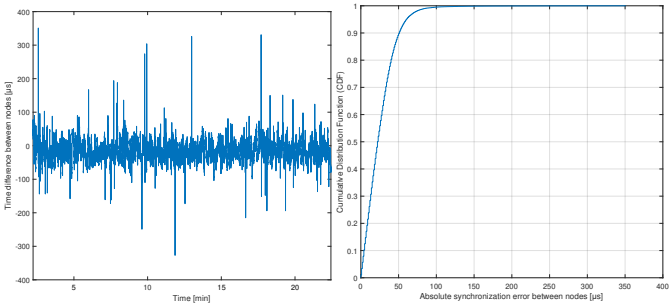
Experimental tests were performed using openwifi APs, with beacon follow-up packets enabled, and clients using COTS hardware. In case when APs were using COTS hardware, beacon follow-up packets can not be used as well as the TSF can not be controlled from the AP due to driver limitation. Consequently, in such a case only clients from single AP can be synchronized, while the wired-segment of the network can not control the TSF counter of the AP. We benchmark the results achieved by the proposed mechanism with the results achieved with PTP over wireless link.

1) openwifi AP case: In this scenario, single opnewifi AP and commercial clients were used to validate synchronization measurements. First, the TSF counter was not controlled by the synchronization entity but left to run in its own (Figure 4) while in the second case the TSF counter was controlled (Figure 5). In the latter case, the time of the AP host was also controlled by a PTP master in the wired network. The synchronization error does not change in both cases (Figure 4 and Figure 5, being smaller than $\sim$17 μs in 90% of cases. However, in the second case the standard deviation is smaller due to the control of the TSF from the user-space, with σ 16.58 μs compared to 22.18 μs of the previous case.

2) PTP over WiFi case: We benchmark the synchronization accuracy of the proposed time synchronization mechanism by running PTP over WiFi. Clients and AP are synchronized using PTP over WiFi using PTPd [3] implementation. To the best of knowledge of the authors, there are not any WiFi cards that support hardware time stamping. The time-stamping of the packets is done above the driver in the stack, decreasing thus the accuracy of the synchronization mechanism. In this setup, the AP was set to ptpd master while the clients were the ptpd slaves. We used the same parallel measurement method, to determine the time offset between client nodes. As shown in Figure 6b, the synchronization error between two nodes associated with the same AP will be smaller than 50 μs in 90% of the cases, with σ of 20.12 μs. It can be seen that the beacon based synchronization mechanism with follow-up beacon support offers 50% lower time offset between WiFi clients than PTPd over WiFi (using software timestamping).

3) Beacon interval: can impact the synchronization accuracy between clients connected to the same AP. Larger the beacon interval, longer the time between subsequent synchronization moments for the clients. To asses such impact we performed 30 minutes long measurements for each beacon interval setting using the measurement setup shown in section IV using openwifi AP. In Figure 7 statistics of synchronization error between nodes for different beacon intervals are shown. It is seen that by increasing the beacon interval the standard deviation of the synchronization error is increased, however, the average value in any of the cases was smaller than 20 us.

[3]https://manpages.debian.org/testing/ptpd/ptpd.8.en.html



(a) Time difference between clients (b) CDF of absolute synchronization error

Fig. 6. Synchronization error between clients when PTPd over WiFi is used.

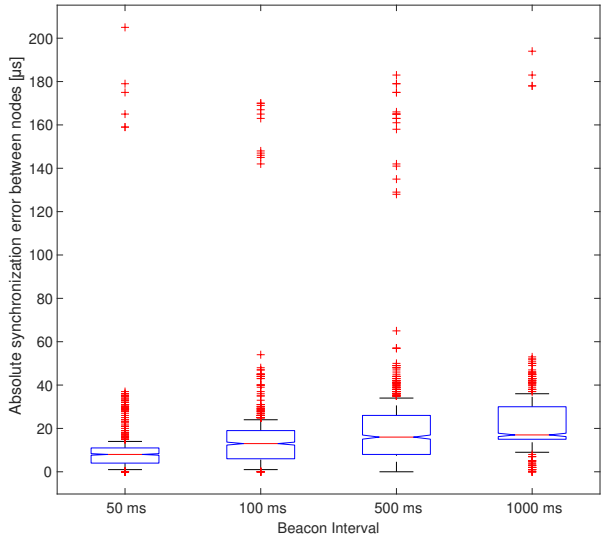


Fig. 7. Impact of the beacon interval on the synchronization error statistics.

The box plot shows 25^{th} and 75^{th} percentile, with outliers greater than 90^{th} and lower than 10^{th} percentile, respectively. The case of beacon interval of 1 second had the highest 90^{th} percentile, 34.5 us. However, the normal setting for the beacon interval is 100 ms. This shows that even under low-quality link conditions, when the client will miss some beacons, up to a second, still it will remain time-synchronized with a synchronization error lower than 21 μs, on average. Table I summarizes statistics for each case.

Another factor that impacts the synchronization accuracy is the load in the wireless network. Higher the load in the network, the higher the possibility for the beacons to be delayed due to busy channel conditions. On the other hand, higher the node load, slower the processing of the packets by the user-level synchronization application. Similarly, we did tests using the same setup with a single openwifi AP, with beacon interval of 100 ms. In the case of *no load* there was no traffic between any of the client nodes under test connected to the AP. In the case of *network load* there was a single Iperf stream between two clients connected to the AP, that were different from the client nodes under test. In this case, the

TABLE I
IMPACT OF BEACON INTERVAL ON STATISTICS OF ABSOLUTE SYNCHRONIZATION ERROR

Stats	**Beacon Interval [ms]**			
	50	***100***	***500***	***1000***
σ [us]	15.54	18.03	23.21	26.06
μ [us]	10.21	15.04	20.12	21.03
90% [us]	19	24	35	34.5

generated traffic in the network will impact only the beacon and follow-up packets and sending time, but will not increase the load on the clients under test itself. In the case of *node load* the Iperf stream was running between both clients under test. In such a case, clients that need to synchronize were generating/receiving traffic that will increase the processing load. In both cases, the required bandwidth was set to 20 Mbps. For each of the 3 scenarios, two different measurements of 20 minutes were conducted at different times. As can be seen in Figure 8 the synchronization accuracy is not affected by the network load. Under network load conditions beacons might be delayed, however, such an impact is compensated by the follow-up beacon. Under the node load condition the packets can be delayed by the driver and the timestamping at the receiver side might be not correct. This is compensated with the relative time difference between beacon processing time and follow-up beacon packet processing time that will filter out such delays or any possible batch processing effect on the client side.

4) Inter-AP synchronization: The mechanism to control the TSF counter in the AP offers the possibility to have end-to-end time synchronization between wired and wireless network segments. To this end, the AP system time will sub-ordinate to a master clock in the wired network using one of the synchronization possibilities [14], [15], while the TSF counter will sub-ordinate to the system time of the AP. Thus, AP will behave like a wired-wireless time synchronization bridge. Hereby, the time reference for all the clients in the wireless network will be the same and will be fed by the time master from the wired network segment.

To check the synchronization accuracy in such cases, we perform the measurements for time offset between wireless clients that are connected to two different APs. Both APs were openwifi APs, while one client under test was connected to the first AP, while two other clients under test were connected to the second AP. Both APs used the same beacon interval of 100 ms. Figure 9 shows the statistics of synchronization error in the inter-AP case. The mean synchronization error is 16.2 μs for inter-AP case compared to 14.7 μs in intra-AP case. Similarly, standard deviation, σ, in inter-AP case is increased to 13.3 μs compared to 11.11 μs in case of intra-AP. This small increase in absolute synchronization error statistics in inter-AP case is related to the accuracy of the synchronization between APs themselves. As the beacon-based synchronization always uses an absolute time reference (the UTC when TSF counter was 0), the AP application needs to start at the time when the

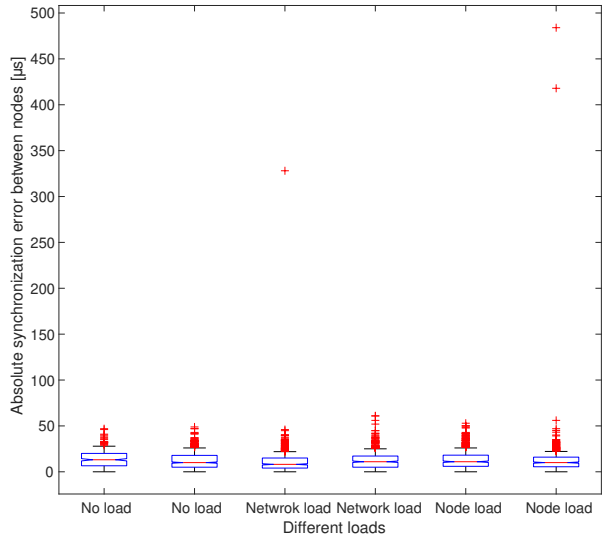


Fig. 8. Impact of the network and node load on the synchronization error.

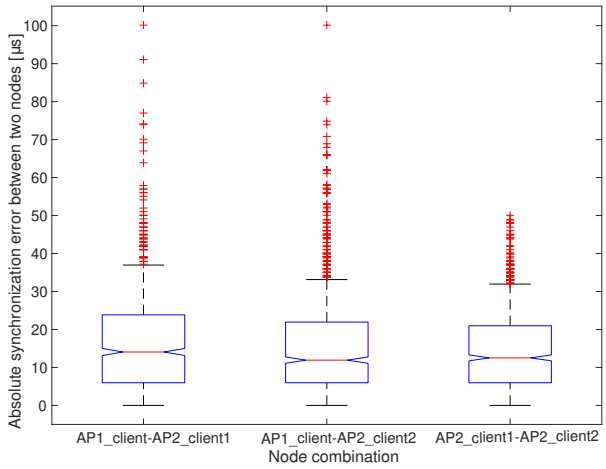


Fig. 9. Statistics of absolute synchronization error for inter-AP case.

AP system times are already synchronized. If there will be a fixed small offset between the absolute time on the first and second APs, then such fixed offset will be reflected to the synchronization error between inter-AP clients too.

VI. Conclusions

In this paper a beacon-based time synchronization mechanisms that extends the time synchronization to wireless segment of the networks is shown. The node architecture for the AP and wireless end nodes is designed, with no low level changes at the end node side. The time synchronization mechanism is validated in both intra and inter AP communication settings and it is benchmarked to the PTP performance over wireless link. It is shown that the in case of intra-AP setting, the synchronization performance depends on the synchronization accuracy of the APs as well. The impact of the beacon interval as well as the load on the wireless link is validated to. Further improvements can be done in order to decrease the overhead and increase the synchronization accuracy. This includes the ability of the physical layer to timestamp the beacon packet itself, without the need for the follow-up packet. Future research can be done on how such beacons can be used for traffic scheduling in wireless network.

Acknowledgment

This research research was partially funded by the Flemish FWO SBO S003921N VERI-END.com (Verifiable and elastic end-to-end communication infrastructures for private professional environments) project and from the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” program, and from the FWO-Flanders, under grant agreement G055619N.

References

[1] D. Cavalcanti, J. Perez-Ramirez, M. M. Rashid, J. Fang, M. Galeev, and K. B. Stanton, “Extending accurate time distribution and timeliness capabilities over the air to enable future wireless industrial automation systems,” *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1132–1152, 2019.

[2] P. Djukic and P. Mohapatra, “Soft-tdmac: A software tdma-based mac over commodity 802.11 hardware,” in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 1836–1844.

[3] S. Zehl, A. Zubow, and A. Wolisz, “hmac: Enabling hybrid tdma/csma on ieee 802.11 hardware,” *arXiv preprint arXiv:1611.05376*, 2016.

[4] X. Jiao, W. Liu, and M. Mehari. (2019) open-source ieee802.11/wi-fi baseband chip/fpga design. [Online]. Available: https://github.com/open-sdr/openwifi

[5] Orca project. [Online]. Available: https://www.orca-project.eu/

[6] “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, 2008.

[7] Ó. Seijo, J. A. López-Fernández, H.-P. Bernhard, and I. Val, “Enhanced timestamping method for sub-nanosecond time synchronization in ieee 802.11 over wlan standard conditions,” *IEEE Transactions on Industrial Informatics*, 2020.

[8] O. Seijo, I. Val, J. A. Lopez-Fernandez, and M. Velez, “Ieee 1588 clock synchronization performance over time-varying wireless channels,” in *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. IEEE, 2018, pp. 1–6.

[9] A. Mahmood, R. Exel, and T. Sauter, “Performance of ieee 802.11’s timing advertisement against synctsf for wireless clock synchronization,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 370–379, 2016.

[10] A. Mahmood, R. Exel, and T. Bigler, “On clock synchronization over wireless lan using timing advertisement mechanism and tsf timers,” in *2014 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. IEEE, 2014, pp. 42–46.

[11] Z. Fernández, Ó. Seijo, M. Mendicute, and I. Val, “Analysis and evaluation of a wired/wireless hybrid architecture for distributed control systems with mobility requirements,” *IEEE Access*, vol. 7, pp. 95 915–95 931, 2019.

[12] “Ieee 802.11, draft standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications,” *Specifications*, 2012.

[13] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, “The click modular router,” *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000.

[14] “Ieee approved draft standard for a precision clock synchronization protocol for networked measurement and control systems,” *Specifications*, 2019.

[15] “Ieee standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications in bridged local area networks,” *Specifications*, 2010.