Assessing a precise gPTP simulator with IEEE802.1AS hardware measurements

Quentin Bailleul
IRT Saint Exupéry
Toulouse, France
quentin.bailleul@irt-saintexupery.com

Katia Jaffrès-Runser, Jean-Luc Scharbarg
IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3
Toulouse, France
{kjr, jean-luc.scharbarg}@enseeiht.fr

Philippe Cuenot
IRT Saint Exupery, Seconded from Continental Automotive France
Toulouse, France
philippe.cuenot@continental-corporation.com

***Abstract*—TSN (Time Sensitive Networking) standards are arousing growing interest in the critical on-board network community because of their promise of deterministic Ethernet networking. Among these standards IEEE802.1AS allows the synchronization of network devices. This protocol achieves much greater precision than other Ethernet synchronization standard such as NTP or PTP. Thus, it paves the way for new network mechanisms, such as the Time Aware Shaper, and allows the use of applications that are more constrained in terms of synchronization. In order to study the new mechanisms allowing to reach this precision, precise simulations are mandatory. In this paper, we review the different existing tools, extend the most promising one to incorporate the most advanced features of IEEE802.1AS and assess its behavior with respect to measurements. More specifically, we extend an OMNeT++ simulation library, calibrate its results following an extensive measurement campaign of switched Ethernet devices supporting IEEE 802.1AS to assess its performance in terms of precision.**

***Keywords*—IEEE802.1AS, gPTP, TSN, Synchronization, Simulation**

I. Introduction

Critical on-board network architecture cannot cope with the diversification of flows and their constraints. Thus, real-time Ethernet solutions have been designed to bring much higher bandwidth and advanced quality of service policies. Promising solutions are the standards proposed by the IEEE Time Sensitive Networking (TSN) working group. Among these standards, central synchronous shapers like the Time Aware Shaper (TAS) allow the transmission of zero-jitter time-bounded low latency flows. Such shapers rely on a network-wide precise and accurate synchronisation of the devices. The IEEE TSN group has standardized the IEEE 802.1AS synchronization protocol [1] we are investigating in this paper.

The IEEE 802.1AS protocol is a profile of the IEEE 1588 [2] synchronization standard already in use in non-critical systems. IEEE 802.1AS has been designed with the goal of reaching a precision of less than 1 microsecond in a linear network setting where a master clock and an ordinary clock are separated by 7 hops. The precision is defined as the difference between the clock under test and the reference clock. Experimental measurements [3] as well as simulation and worst-case analysis [4] have assessed such precision. Simulation allows the evaluation of this synchronization protocol at a lower cost, provided that simulation assumptions are derived from measurements. To the best of our knowledge, existing simulation libraries have not been compared to real devices supporting IEEE 802.1AS.

In this paper, we first review the already available simulation tools for IEEE802.1AS and select the most advanced open source OMNET++ simulation library. Then, we extend this library to incorporate the most advanced features of IEEE802.1AS and compare its performance to experimental measurements of real IEEE802.1AS devices. From this step, we propose a calibration step of the simulator and assess, after calibration, its precision to validate its behavior in relation to reality. Our contribution is a realistic open source simulation model for IEEE802.1AS that reproduces the behavior of real devices. Thanks to the improvements made to the simulation model and the calibration of the jitter linked to the PHY layer, the drift and the granularity of the clock using measurements on real devices, we achieve a Root Mean Squared Error (RMSE) of approximately 3 ns between the measured and simulated sliding average of the synchronization precision.

This paper is organized as follows. First, we present IEEE 802.1AS and the related work on IEEE802.1AS simulation in Section II. In Section III, we discuss our changes to an existing library. Then in section IV, we present our experimental protocols and discuss our results. And finally, we conclude and present future work in section V.

II. Context

A. IEEE 802.1AS overview

IEEE 802.1AS [1] is a profile of IEEE 1588 Precision Timing Protocol (PTP) [2] for Time Sensitive Networking. Sometimes called generalized Precision Time Protocol (gPTP), this protocol is used to synchronize clocks across a network using the master slave paradigm. Each port is in one of the following three states: master, slave, passive. Master ports

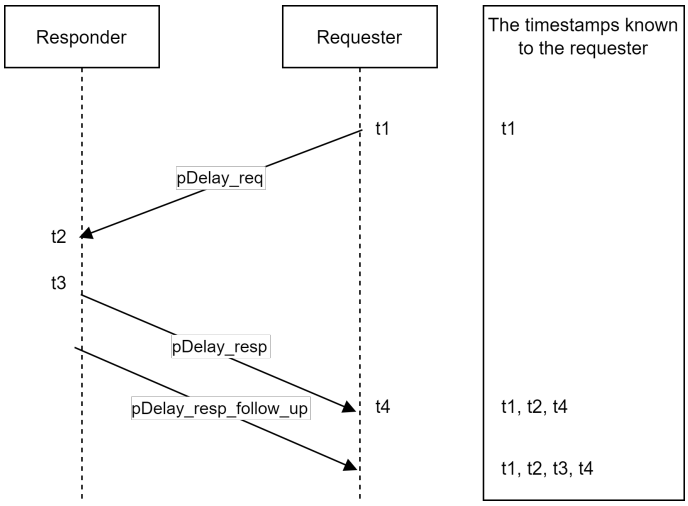

Fig. 1: Peer-to-Peer delay mechanism.

periodically broadcast time synchronization messages. Slave ports process these messages on reception, while passive ports ignore them to avoid loops in the distribution. The time-aware system with all its ports in master state is called Grandmaster and it is the time source of the network. It can be synchronized by an external time source (GPS, NTP, ...) or use its own clock.

To determine port states, IEEE 802.1AS provides two methods. The first one is the external port state configuration. It statically defines the state of the ports for all the devices involved in the synchronization. The second method is the Best Master Clock Algorithm (BMCA). This distributed algorithm is executed on each time-aware system to eventually determine the state of the local ports and to elect the Grandmaster by comparing the information received from each Grandmaster candidate.

Synchronization itself is based on two core mechanisms: *i)* the measurement of the link propagation delay using the Peer-to-Peer delay mechanism and *ii)* the distribution of synchronization information.

The Peer-to-Peer delay mechanism defined in IEEE 802.1AS measures the link propagation delay between two time-aware systems that are separated by one hop. `Pdelay` messages carrying timestamps are exchanged every *Pdelay interval* (1s by default). Measurement of propagation delay needs four timestamps as depicted in Fig. 1. $t1$ is measured when the `Pdelay_req` is issued. $t2$ is obtained upon reception of this message. $t3$ is measured when the `Pdelay_resp` is sent. Finally, $t4$ is measured upon reception of `Pdelay_resp`. Formula (1) calculates the delay of the link, T_{prop}, using the timestamps. Moreover, with the $t3$ and $t4$ timestamps of two consecutive Pdelay procedures, the requester can extract the so-called *neighbor rate ratio* nr of Eq. (2) in order to compensate the relative clock drift in Eq. (1).

$$T_{prop} = \frac{(t2 - t3) + nr.(t4 - t1)}{2} \tag{1}$$

$$nr = \frac{f_{req}}{f_{resp}} = \frac{t3_i - t3_{i-1}}{t4_i - t4_{i-1}} \tag{2}$$

Equation (1) assumes that the link is symetric. Existing asymmetries can be compensated if they can be estimated, typically in a calibration step. To smooth the effects of sources of inaccuracies, some devices include filters for T_{prop}.

The distribution mechanism is based on the transmission of `Sync` and `Follow_Up` messages that allow each time-aware system to synchronize to the Grandmaster clock. Every synchronization interval (typically 125ms), the Grandmaster sends a `Sync` message out of its master ports, followed by a `Follow_Up` message containing t_0, the exact transmission time of the `Sync` message, as pictured in Fig. 2. These two messages are received via the slave ports of the device connected to the Grandmaster. If the receiving device has at least one port in the master state, it forwards both messages to the next time-aware system. The `Follow_Up` message carries t_0, and updated values of the *rate ratio* r and the *correction field* C. These two fields are detailed next.

The rate ratio r allows for logical syntonization of a time-aware system to the Grandmaster rate. It is the product of the neighbor rate ratio calculated by the receiver ports on the path going from the Grandmaster to the time-aware system of interest. It is initialized to 1 by the Grandmaster and is updated on each hop with the equation (3), where i is the receiving node and $i-1$ the sending node.

$$r_i = r_{i-1} * nr \tag{3}$$

The *correction field* C carries the time elapsed in the time-aware systems and on the links on the path between the Grandmaster and the time-aware system preceding the last hop. At hop i, C_i is calculated using the previous correction field C_{i-1}, the previous rate ratio r_{i-1}, the neighbor rate ratio nr, the propagation time undergone during the last hop T_{prop} and the residence time T_{res} of the `Sync` in time aware system as given in (4) :

$$C_i = C_{i-1} + r_{i-1} * T_{prop} + r_{i-1} * nr * T_{res} \tag{4}$$

These operations are repeated at each hop, until the complete set of time-aware systems is reached.

Using the two mechanisms described above, each device can calculate the difference between its local clock and the Grandmaster clock in order to deduce the correction to be applied. Indeed, to deduce the Grandmaster time at the `Sync` reception time, the device adds to t_0, the original time of transmission of the `Sync` by the Grandmaster, the correction field carried in the `Follow_Up` message and the propagation delay of the last hop T_{prop} measured with the Peer-to-Peer delay mechanism.

B. Related Work

Since the first version of 802.1AS in 2011, several simulation tools have been created. Garner et al. [3] proposed a simulation model to evaluate a draft of 802.1AS. In order to reduce its complexity, this discrete event simulation library only models events linked to `Sync`, `Pdelay_Req` and `Pdelay_Resp` messages. It approximates operation in one-step mode. Using this simulation, they showed that the proposed synchronization mechanism can cope with Audio/Video application constraints with a network having up to 7 hops. Lim et al. [5] built a simulation model for IEEE802.1AS using OMNeT ++/INET

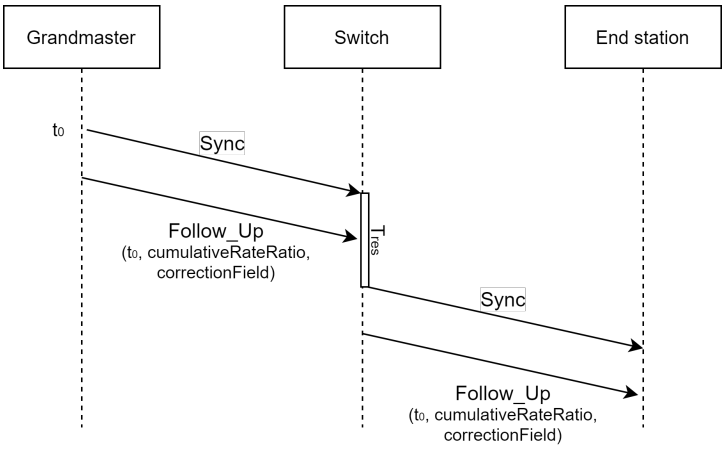

Fig. 2: Synchronization distribution mechanism.

in order to evaluate the performance of the standard on an automotive Ethernet topology. This simulation model uses passive clocks and a static configuration. They highlighted the importance of the choice of filter to smooth out errors in the propagation delay measurement and the improvement in precision when a lower syncInterval is used in the first hop. Gutiérrez et al. [4] developed a simulation library in order to compare probabilistic and worst case precision achievable in a 100 hop network. Their simulation model is based on OMNET ++ / INET with a clock model using a drift that varies over time at a bounded rate. Nevertheless, the neighbor rate ratio is calculated from the perfect neighbor rate ratio and the maximum error allowed by the standard, which makes the Pdelay calculation pessimistic. However, these different simulation libraries are not open source and haven't been assessed by real measurements.

A few works propose open source libraries, e.g. the one proposed by Puttnies et al. [6]. They developed a IEEE802.1AS simulation model, using OMNeT++ [7] with the INET framework [8], containing the core time synchronization and propagation delay measurements operations. The model uses a simple clock model with constant drift. The BMCA is not implemented because they focus on achievable precision. Obtained results have been compared with the simulation results of Lim et al. [5]. Wallner et al [9] also built an open source simulation framework for IEEE1588-2008, also based on OMNeT++/INET, with complex clock models using realistic noises. This very complete simulator allows the use of many PTP mechanisms such as End-to-End or Peer-to-Peer delay measurement, BMCA and transparent clock mechanisms.

The simulation library of Wallner et al. [9] is of higher complexity than the one of Puttnies et al. [6] due to the implementation of many PTP mechanisms that are not part of IEEE802.1AS. As such, we decided to carry out this work on the basis of the library of Puttnies et al. [6] that supports the core functions of IEEE802.1AS by design. In order to compare the simulation accuracy with real measurements, we had to extend this library with new features to better capture all mechanisms of IEEE802.1AS. Resulting extensions are presented next.

III. gPTP Simulation model extensions

In order to improve the simulation accuracy of [6], we had to extend the simulation model to account for clock syntonization on the one hand, and to better capture core platform-dependant features such as clock granularity and jitter of the link delay due to the PHY layer on the other hand.

A. Logical syntonization

As described above, logical syntonization is essential to reach higher levels of synchronization precision. Indeed, it allows to take into account the local drift compared to the Grandmaster one when updating the *correction field*. Without this mechanism, the relative drift between the Grandmaster and the time aware system is not compensated for, causing a wrong estimate of the correction to be applied to the clock. This error is also propagated to the devices on slave ports with the *correction field*. We have added this syntonization step following the IEEE802.1AS standard to the simulator of [6].

B. Towards a more realistic model

To better capture the real behavior of devices supporting IEEE802.1AS, the following sources of imprecision have been added to the simulator.

The first one is the granularity of the clock. This is the step at which the clock ticks, for instance 10ns. Thus, when a duration is measured between two timestamps, e.g. a duration of the propagation delay computation or the residence time of a `Sync` message, the measurement error varies between -10ns and + 10ns. In the simulator, we round the duration to the immediately lower multiple of 10ns to capture clock granularity.

The second source of imprecision is the inaccuracy related to the PHY layer that occurs at the reception device. Despite the hardware timestamping used in devices supporting IEEE 802.1AS, which eliminates software-related jitter, the PHY layer causes an implementation dependent jitter. As shown by Loschmidt et al. in [10], the jitter linked to the PHY layer depends on the protocol used. They show for instance that the jitter is higher with 10Base-T than with 100Base-T. Moreover, a propagation delay asymmetry may exist whose magnitude depends on the initialization of the link PHY layer. These two PHY layer phenomenons have significant impact on the synchronization results since they change the propagation delay statistics and values.

In the following we consider a 100Base-T PHY. Based on [10], a random jitter is added to the reception timestamp of a message. It follows a normal distribution with zero mean and 0.286ns standard deviation. Additionally, a link delay asymmetry may be introduced at link initialization by constant latency. This asymmetry is rooted in the Phased Lock Loop (PLL) system that may lock on dissimilar edges of the signal at the RX interface on both ends of the link. If the PLLs of the two RX interfaces at both ends of the same link lock on dissimilar edges, then the propagation delay in both ways is asymmetric, which causes an error in the estimation of the propagation delay. The 5 possible edges being 8ns apart, the worst asymmetry is therefore of 32ns which causes an error

Protocol Parameters	Value
Sync Interval	0,125s
Pdelay Interval	1s
SyncLocked	True

TABLE I: Protocol parameters

of 16ns when calculating the propagation delay by the time aware system. This error leads to errors in the estimation of the time spent by the `Sync` messages on the link and therefore inaccuracies in the synchronization.

There are other sources of inaccuracy such as PLL or oscillator noise. They can be neglected since their impact on the delay is less than one nanosecond [10].

IV. Experiments

In the previous section, we detailed the changes made to improve the simulation library's realism. This section aims to calibrate and validate the choices made during the implementation and to identify calibration steps to improve simulation accuracy. For this, we will measure and analyse first, on the real TSN switch, the behavior of the switches clock. Second, we challenge the results obtained by the link delay measurement mechanism with the simulated ones. And finally we compare the precision of the synchronization IEEE802.1AS to the simulation results to validate the simulator's accuracy. From the first two steps, we extract calibration steps that can be performed to adjust the simulator to a specific 802.1 PHY technology.

A. Experimental setup

The goal is to calibrate and validate the behavior of the simulation library using real devices. Configuration parameters of IEEE802.1AS are given in Table I. The library is configured with the help of the values determined in the rest of this paper to get as close as possible to the behavior of real device.

Our experimental testbed, pictured in Fig. 3 consists of:

- 4 Fraunhofer IPMS TSN Multiport Switch Core - TSN-SW v0.5.0 on Netleap boards,
- a netTimeLogic PPS analyer,
- 100Base-T Ethernet link.

As described in Fig. 3, the four switches are connected in a chain topology. We capture the progress of the different clocks using the PPS Analyzer. One of our experiments described later requiring greater accuracy required the use of a better quality reference clock as a reference for the PPS analyzer. For this we used a Meinberg microSync HR slaved on GNSS time connected to the reference input of the PPS analyzer with a PPS link. And finally, to configure and retrieve internal switch values, such as the result of the delay propagation calculation, we use the serial port of the switches to communicate with a computer.

Simulations also consider the same topology. Unlike a one-hop topology, this multi-hop topology allows an evaluation of the impact of the *correction field*. The measurements are

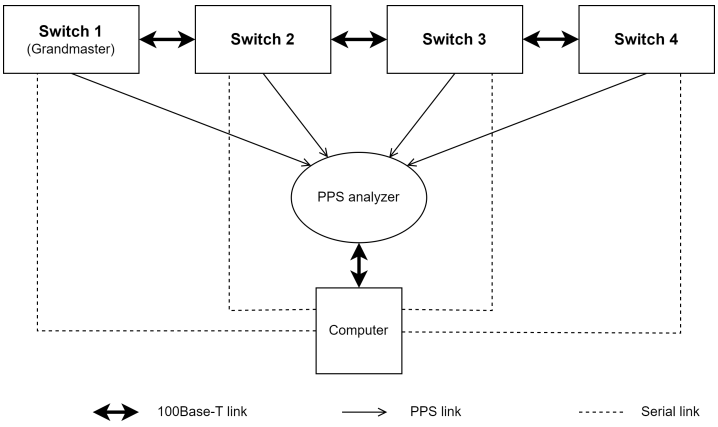

Fig. 3: Experimental measurement topology.

carried out by OMNeT ++ at instants following precise events, e.g. the end of the propagation delay measurement or just before and just after the correction of the clock.

To determine the duration of experimentation and simulation necessary to obtain significant results, we studied the evolution of the mean squared error (MSE) between the normalized distribution of value obtain by the Pdelay mechanism of a given experiment duration compared to a experiment duration of 32h. A duration of 3600s corresponds repeatedly to an MSE of around 10^{-5} % of the number of occurrences as shown in the fig 4. Fig 5 makes it possible to observe the small difference between the distribution of value obtain by the Pdelay mechanism obtained after 1 hour of experience and the distribution obtained after 32 hours. The sources of variability, such as the granularity and the PHY jitter, do not depend on the AS mechanism which is studied, we will therefore use this duration of experimentation for the other mechanisms and not only for the Pdelay mechanism. Thus, for the rest of this study, we take measurements for 3600s.

The following sections present the three experiments that we carried out to compare the operation of the simulator with the operation of real devices in order to calibrate and validate this simulation model.

B. Clock calibration

In [6], Puttnies et al. have chosen to implement a simple constant drift clock. In order to validate this choice and adjust the parameters of the simulator clock deviation in relation to our real devices, we have studied the behavior of the clocks of our switches in freerunning during one hour. To do this, we measured with the help of the PPS analyzer the evolution of the drift for each switch compared to the Meinberg microSync HR slaved on GNSS when the synchronization is deactivated. We use the Meinberg microSync HR on this experiment for its clock quality.

Figure 6 and table II shows the results of this experiment. These measurements were taken 10 minutes after starting the devices, so that the temperature of the electronic components is stable and without attempting to control the room temperature. As indicated in the figure 7, presenting multiple results of the

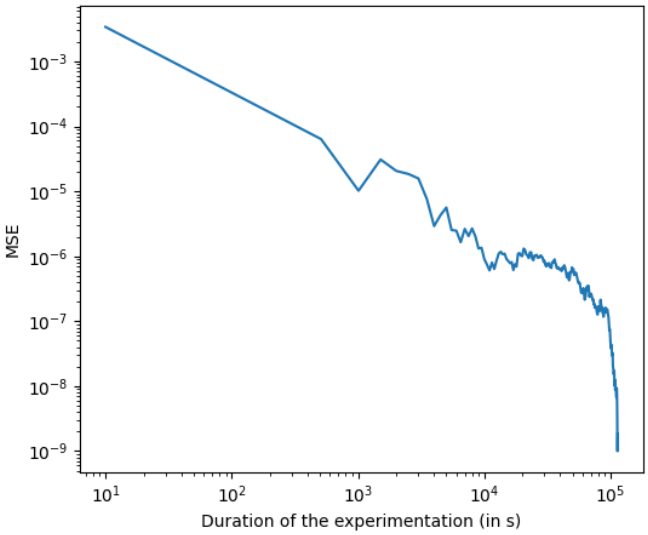

Fig. 4: MSE of the distribution of the results of the Pdelay mechanism according to the duration of the experience compared to a 32 hours experience with the real switches

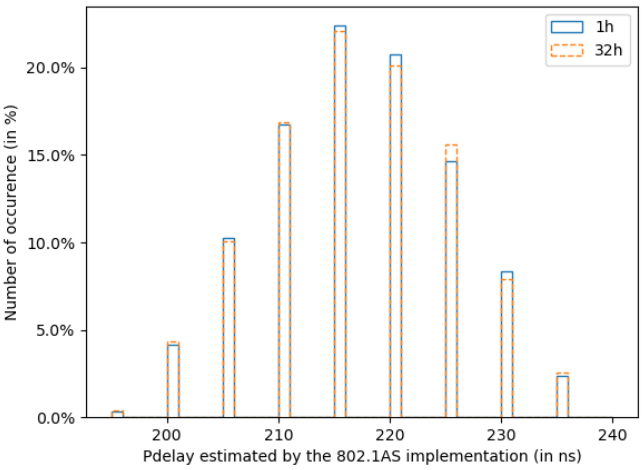

Fig. 5: Pdelay distribution for an 1-hour experience and a 32-hour experience

linear regressions carried out, similar results are obtained for the different repetitions of the measurements during 24h.

Using Fig 6 and the table II, we observe that an implementation of a constant drift clock makes it possible to simulate the behavior of a real clock in freerunning for one hour in an environment where the temperature variations are small. However, by repeating the experiment during 24h, we observe small variations in ppm within a range of 0.009 ppm , as shown in Fig 7 caused by the temperature variation in the room. In order to use up to date ppm value in the simulator, in the rest of our comparison between simulation and reality, we perform a ppm measurement in freeruning compared to the Grandmaster

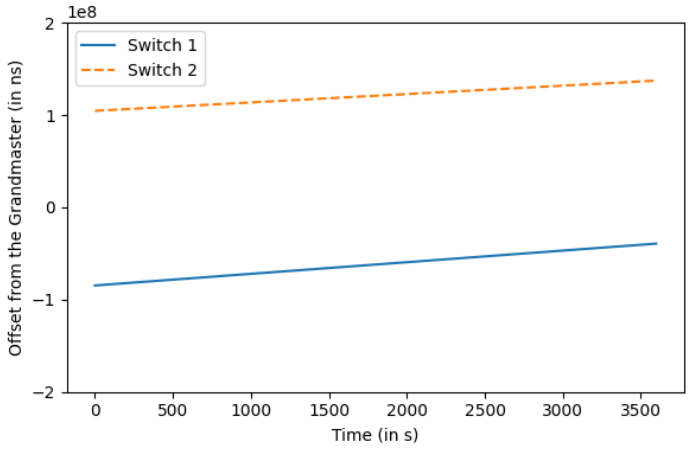

Fig. 6: Grandmaster clock offset of two switches during 3600s

	Switch 1	Switch 2
ppm	12.593	9.094
pvalue	$< 10^{-9}$	$< 10^{-9}$
rvalue	0.99999995	0.99999999

TABLE II: Results of linear regression of the data presented in Fig 6

of the experiment before performing any other measurements on real devices.

C. Canal calibration

In order to adjust and validate the different sources of inaccuracy that we have added to the simulator, we measure the distribution of the values obtained by the Peer-to-Peer delay measurement mechanism without any filter and compare them to the value obtained with the simulator. To configure

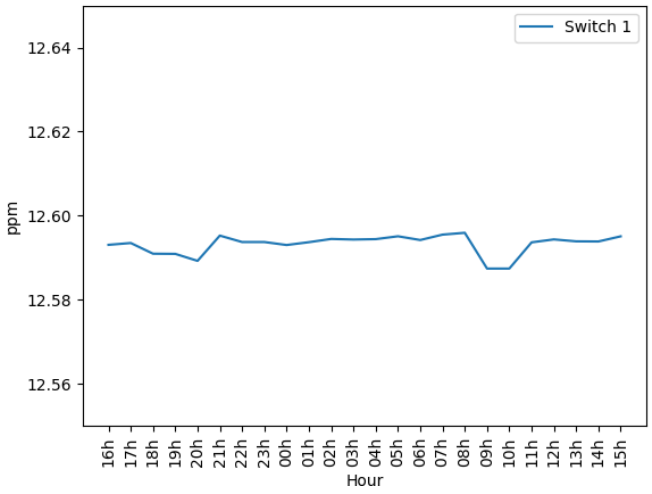

Fig. 7: Result of linear regression of switch 1 clock drift measured for 3600s and repeated for 24 hours.

the simulator, we use the relative drift measured before the experiment and the mean link delay measured during the experiment.

In order to determine the standard deviation to parameterize the normal distribution which adds jitter to the link crossing time in the simulator, we compared the distributions obtained using the simulated and real Peer-to-Peer delay mechanism. Figure 8 shows the MSE obtained between the simulated and measured distribution of the Pdelay as a function of the standard deviation used to parameterize the normal law which causes the PHY jitter when crossing the link. By minizing the MSE, we find a standard deviation of 12.5ns, which is quite different from the 0.286ns measured by Loschmidt et al in [10]. Indeed, our switch embeds a different PHY chip from the one used for their measurements. In addition, our measurement takes place much further in the chain of message transmission. Indeed, our measurement is based on timestamps which takes place between the MAC layer and the PHY layer of the OSI model while their measurement takes place directly at the PHY level using the RX_DV and TX_EN PINs of the MII. Thus other source of jitter can be found between their point of measurement and ours.

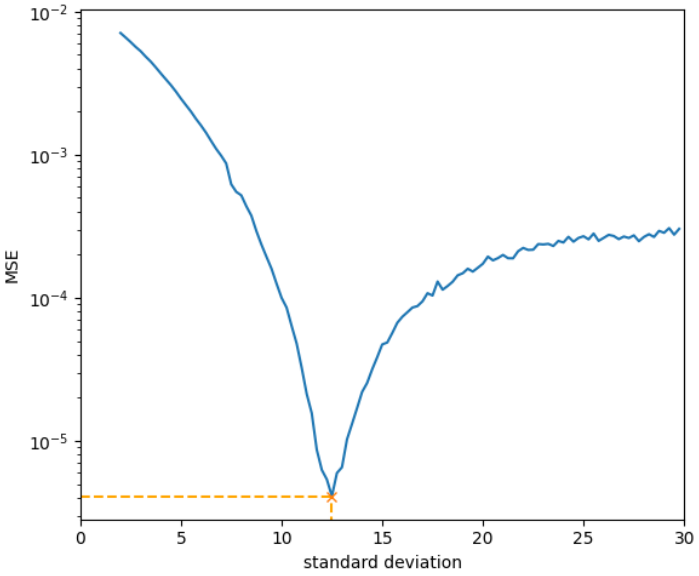

Fig. 8: MSE between the simulated and measured distribution of the Pdelay during 32h according to the standard deviation used to simulate the PHY jitter in the simulator

The figure 9 presents the results obtained where switches are synchronized and exchange Pdelay messages every seconds. We find there the distribution of the different values obtained by the measurement mechanisms of the link delay of one of the switches and of the simulation library obtained during 32h. For this measurement, we use a measurement time much longer than what we have previously determined to be sure that the simulated distribution also covers rare events. The simulation parameters such as mean link delay, granularity and PHY standard deviation were chosen to match the experimental distribution. By analyzing the distribution of the propagation delays obtained by the real device, in Fig 9, we can deduce the granularity of the clocks used by observing the difference between the two consecutive values. Here, the difference being 5ns, the granularity is therefore 10ns, because of the division by two in the formula (1). As can be seen with this figure, our simulation gives a distribution very close to reality although a little more pessimistic than reality but which does not bother us in view of our on-board scope of application.

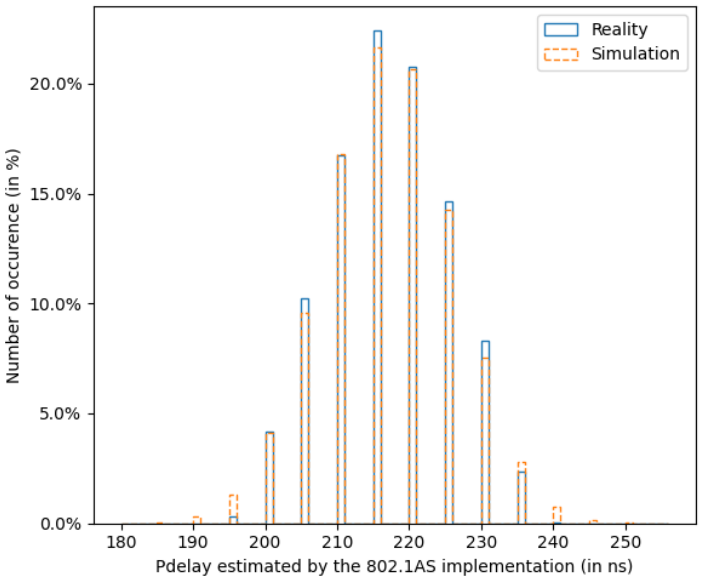

Fig. 9: Distribution of the results obtained by the simulated and real Pdelay mechanisms

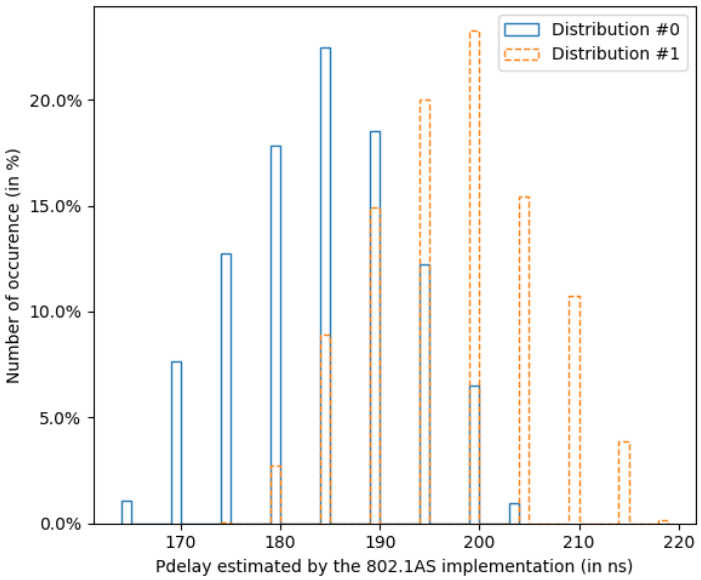

Fig. 10: Distribution of the results obtained during two experiments with the real switches.

When the link between two measurements is reinitialized, a variation of the mean Pdelay, as shown in fig 10, is observed. These two distributions originate from a periodic measurement

of the Pdelay on real devices for which we trigger a link reinitialization every hour. As described by Loschmidt et al. in [11], the difference between the two distributions is a consequence of link delay asymmetry. This asymmetry is induced by the edges on which the PLLs of each physical interface lock during link initialization. If the two PLLs don't lock on the same edge, the various messages of the Pdelay mechanism undergo a different link propagation delay depending on their direction on the link. As such, there is an error in the estimation of the link delay by the Pdelay mechanism since it is based on a symmetrical channel assumption. With the library, we reproduce this behavior by randomly setting the edge on which each interface is locked for each simulation.

D. Validation

In this last part, we compare the synchronisation precision measured with the real TSN switches to the one computed by the OMNeT ++ simulator. This steps validates the different enhancements of the simulation library and the calibration steps proposed in this paper.

1) Bounding the precision.: The measurements are triggered by the PPS analyser every second. The `SYNC` messages are sent about every 125ms. The PPS measurements aren't synchronized with the `SYNC` dates and as such, the measurements may be taken at various times of the synchronization cycle. For instance, if the previous synchronization stage takes place a few nanoseconds before the measurement time, the precision measured using the PPS signal is much better than if the last synchronization stage takes place a few tens of milliseconds before.

Unlike real measurements based on the PPS signal, we can measure in simulation the synchronization precision at specific times which are related to the main protocol execution steps. The worst precision can thus be measured just before the synchronization procedure takes place and the best one just after it. Since we can't compare measurements and simulation at exactly the same dates, we leverage the track the best and the worst synchronization by simulation, and plot the real measurements in the same figure to validate whether measurements lie in between best and worst simulation precision. Figure 11 shows the precision measured on the first hop, as well as the simulated precision before and after sync for 3600s. With this figure, we observe that the simulator allows to bound the result obtained with the real device on the first hop, despite the limitations of the PPS signal. There results are reasonable, even though they don't allow us to judge the accuracy of the bounds obtained with the simulator.

2) Accuracy of simulated precision bounds: By repeating the previous experiment, we observed gaps in the precision measured using the PPS signal as shown in the figures 12, 13 and 14. These figures show the measured precision, as well as the simulated precision before and after synchronization respectively at the first, second and third hops. These PPS gaps represent the situation where we move from measuring precision just after synchronization takes place to measuring precision in reality just before synchronization takes place. Over a campaign of 200 one-hour measurements, we have observes these PPS gaps 19 times.

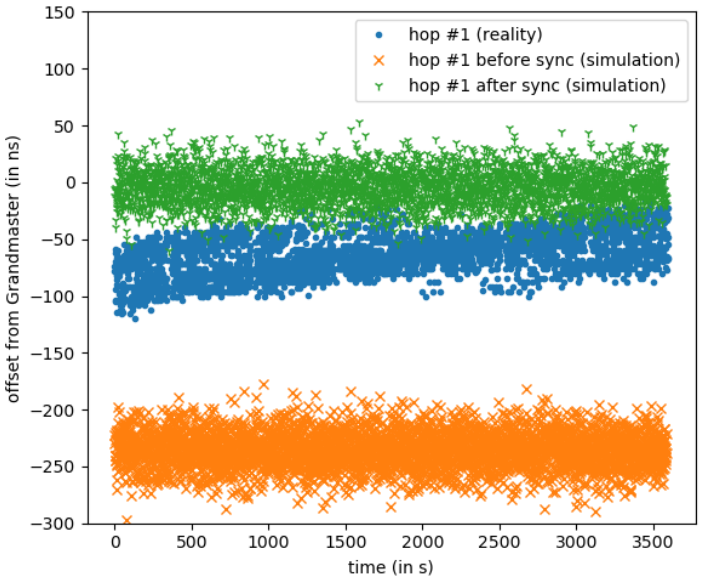

Fig. 11: Offset between the Grandmaster clock and the switch clock at hop #1 in reality and the simulator. The simulator is configured with the granularity, mean link delay and the standard deviation estimated in the previous experiments. For the clock drift, we use the drift measured before each experiment.

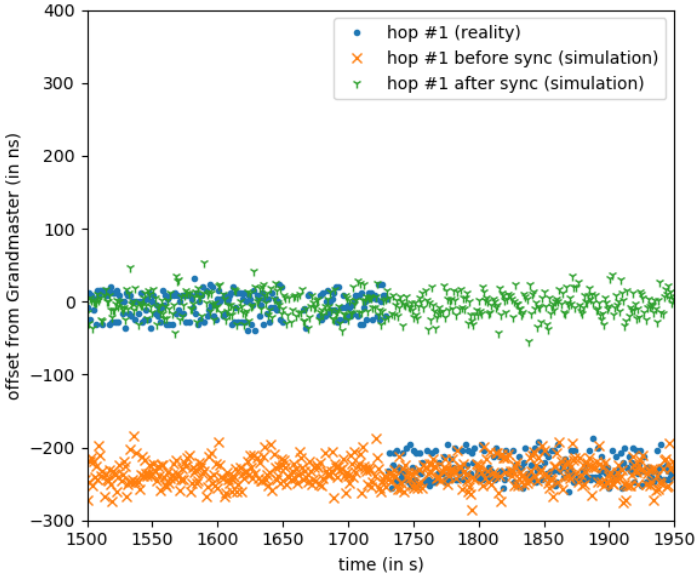

Fig. 12: Identification of a PPS gap in the offset measurement between the Grandmaster clock and the switch clock at hop #1 in reality. Simulated best and worst offset are plotted as well.

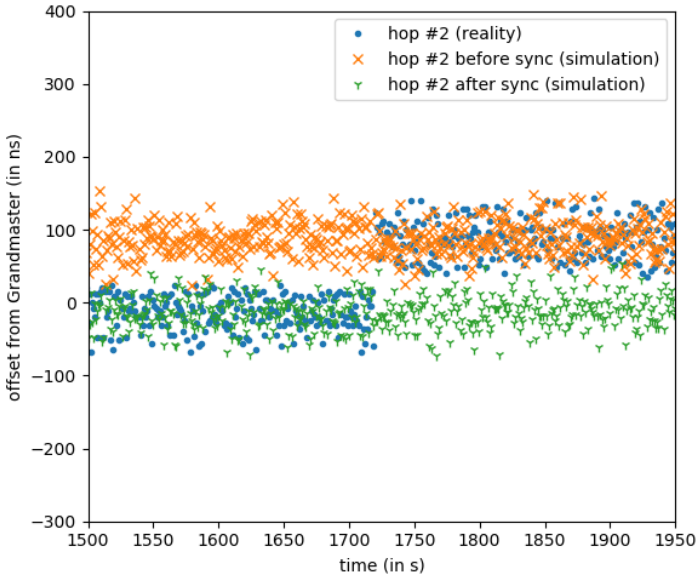

Fig. 13: Identification of a PPS gap in the offset measurement between the Grandmaster clock and the switch clock at hop #2 in reality. Simulated best and worst offset are plotted as well.

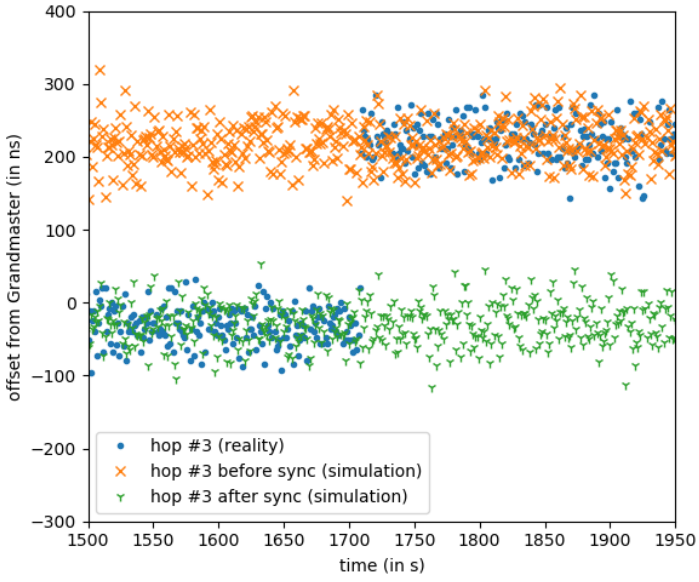

Fig. 14: Identification of a PPS gap in the offset measurement between the Grandmaster clock and the switch clock at hop #3 in reality. Simulated best and worst offset are plotted as well.

	Hop 1	Hop 2	Hop 3
Minimal RMSE (ns)	0.89	1.1	2.1
Maximal RMSE (ns)	7.1	4.8	6.2
Median RMSE (ns)	2.3	2.8	3.9
Mean RMSE (ns)	3.4	2.9	4.2

TABLE III: Result of the RMSE calculation using 10 different simulations

	Switch 2	Switch 3	Switch 4
ppm	-1.850	0.817	1.997

TABLE IV: Clock drift of the different devices measured before the simulator validation experiment

Thus, close to the PPS gap, it's possible to determine when the synchronization cycle takes place. In this situation, it becomes possible to compare the bounds obtained with the simulator with the worst and best precision measured around the PPS gap. With the figures 12, 13 and 14, it can be seen that the simulator makes it possible to precisely limit the synchronization precision reachable with this Ethernet switch. Indeed, we observe the best case, just after the synchronization, and the worst case, just before the synchronization, in a single measurement for the three hops. Furthermore, the measurements and comparisons with the simulation for hop #2 and #3 make it possible to validate the implementation of the calculation of the *correction field* and thus the measurement of the residence time. We also observe that the dispersion of the precision values remains similar despite our pessimistic simulation of the PHY jitter.

These observations are validated by the calculation of the RMSE between the measured and simulated sliding average of the synchronization precision. Sliding average is computed over a window of 150 samples for the 500 samples preceding the PPS gap for each one of the three hops. Due to the progressive shift of the synchronization cycle relatively to the PPS measurement time, we are bound to use a small window. This small window isn't large enough to capture all possible variation in simulation. To compensate for this variability, we perform an RMSE calculation over 10 different simulations. Results are are presented in the table III. As shown in this table, the average RMSE is approximately 3 ns. There is also an increase in the median RMSE as the number of hops increases. This increase in the differences between reality and simulation is mainly caused by the pessimistic estimation of the PHY jitter, introduced during the calibration phase of the delay measurement mechanism.

Using the three figures, we also observe a large variation in the precision before synchronization as a function of the number of hops. This variation is due to the relative drift between the Grandmaster and the device in question. Indeed, before synchronization, the precision error is mainly caused by the drift which has taken effect since the last synchronization, here since 125ms. Before this measurement, we measured the relative drift between the Grandmaster and the different devices and observed a lower drift for switch 3 (hop #2) than for the other devices as shown in the table IV.

V. Conclusion

This paper presents the enhancement of an open source simulation library of the IEEE802.1AS synchronization protocol available here [12]. The integration of logical syntonization and real hardware inaccuracies brings the simulation library closer to reality. Our tests show the fidelity of the simulator after calibration compared to the result obtained with real TSN switches, as evidenced by the MSEs of the order of $4 * 10^{-6}$ % of occurrence between the distribution of values obtained by the simulated and measured link delay measurement mechanisms, as well as the RMSEs of approximately 3 ns between sliding average of the precision measured and simulated. This library now allows to study the precision of the synchronization and its impact on the clients according to parameters such as topology, protocol configuration, filters for measuring propagation delay and clock servo algorithm. In addition, we also propose a method which is repeatable to calibrate the simulator for other switches or end stations.

Future works will investigate the impact of other PHY layers like 1000Base-T and 10Base-T1S on precision using this library. This work will also allow us to focus on the calculation of the worst case precision to optimize the protocol parameters according to the intended use.

Acknowledgment

The authors thank all people and industrial partners involved in the EDEN project. This work is supported by the French Research Agency (ANR) and the partners of IRT Saint-Exupéry Scientific Cooperation Foundation: Airbus Operation, Airbus Defence and Space, CNES, Continental Automotive, INPT/IRIT, ISAE-SUPAERO, ONERA, Safran Electronics and Defense, Thales Alenia Space and Thales Avionics.

References

[1] "IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications," *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421, 2020.

[2] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008)*, pp. 1–499, 2020.

[3] G. M. Garner, A. Gelter, and M. J. Teener, "New simulation and test results for IEEE 802.1 AS timing performance," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2009, pp. 1–7.

[4] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "Synchronization quality of IEEE 802.1 AS in large-scale industrial automation networks," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2017, pp. 273–282.

[5] H.-T. Lim, D. Herrscher, L. Völker, and M. J. Waltl, "IEEE 802.1 AS time synchronization in a switched Ethernet based in-car network," in *2011 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2011, pp. 147–154.

[6] H. Puttnies, P. Danielis, E. Janchivnyambuu, and D. Timmermann, "A Simulation Model of IEEE 802.1 AS gPTP for Clock Synchronization in OMNeT++." in *OMNeT++*, 2018, pp. 63–72.

[7] (2021) Omnet++ simulator version 5.2. [Online]. Available: https://omnetpp.org/

[8] (2021) Inet framework version 3.6.3. [Online]. Available: https://inet.omnetpp.org/

[9] W. Wallner, A. Wasicek, and R. Grosu, "A simulation framework for IEEE 1588," in *2016 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. IEEE, 2016, pp. 1–6.

[10] P. Loschmidt, R. Exel, and G. Gaderer, "Highly accurate timestamping for ethernet-based clock synchronization," *Journal of Computer Networks and Communications*, vol. 2012, 2012.

[11] P. Loschmidt, "On enhanced clock synchronization performance through dedicated ethernet hardware support," Ph.D. dissertation, 2010.

[12] (2021) The simulation library. [Online]. Available: https://gitlab.amd.e-technik.uni-rostock.de/peter.danielis/gptp-implementation