Worst-case synchronization precision of IEEE802.1AS

Quentin Bailleul, Philippe Cuenot
IRT Saint Exupéry
Toulouse, France
{first name}.{surname}@irt-saintexupery.com

Katia Jaffrès-Runser, Jean-Luc Scharbarg
IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3
Toulouse, France
{first name}.{surname}@enseeiht.fr

***Abstract*—Time Sensitive Networking (TSN) is gaining interest in the critical embedded networking community thanks to the various Quality of Service (QoS) mechanisms that can be rolled out to offer different levels of determinism to flows in a switched Ethernet network. Among these standards, limited jitter flows can benefit from the Time Aware Shaper (TAS) which requires the deployment of the IEEE802.1AS synchronization. Indeed, TAS assumes a global time with a bounded drift between any two nodes. In this paper, we derive a refined and general mathematical model that offers upper and lower bounds on the worst-case precision that can be applied to different Ethernet technologies. Results for 100Base-T and 1000Base-T technologies are given. Both simulations and empirical measurements validate the almost two times closer upper bound we obtain compared to the state-of-the-art model.**

***Index Terms*—IEEE802.1AS, gPTP, TSN, Synchronization**

I. Introduction

Sharing on-board networks between critical flows and less/non-critical ones is a popular trend [1] [2], since it simplifies network architecture and limits resource over-provisioning. Time Sensitive Networking (TSN) is proposed by IEEE in this context. It is a set of standards which bring an Ethernet based solution with various Quality-of-Service mechanisms. Of particular interest is the Time Aware Shaper (TAS) [3] that offers scheduled transmissions to flows requiring a limited jitter service. It relies on a network wide synchronization of all devices (end systems and switches).

In TSN, synchronization is standardized by IEEE802.1AS [4] which is a profile of the IEEE1588 [5] synchronization standard designed for non-critical systems. The main goal of IEEE802.1AS is to offer a sub-microsecond precision in a 7-hop switched network. It is based on timestamps, which might be inaccurate. Work has been devoted to the evaluation of this inaccuracy. Loschmidt et al. [6] [7] study the sources of timestamp inaccuracy when using PTP, highlighting inaccuracies caused by the physical layer. In [8] [9], Garner et al. use simulations and measurements on a 7-hop network in order to verify compliance with various audio/video application constraints. Lim et al. [10] show that the synchronization is hardly impacted by the network load using simulation. Gutiérrez et al. [11] study the achievable precision with IEEE 802.1AS. Using simulation, they derive the probability of meeting a maximum precision constraint as a function of the number of hops. In this simulator, they introduce some sources of inaccuracy described by Loschmidt et al. [6] [7]. Additionally, they propose an analytical model to derive an upper bound on the worst-case precision of IEEE 802.1AS for 100Base-T networks by accounting for inaccuracy sources such as clock drift, clock granularity and the physical jitter described by Loschmidt et al. Theoretical results are given for 100-hop network. In [12], Puttnies et al. develop a IEEE802.1AS simulation model using OMNeT++/ INET framework, containing the core time synchronization. In previous work [13], we extended this model by adding realistic inaccuracy sources described in [6] [7] and calibrated the simulator to make it representative of the IEEE802.1AS hardware that we use.

In this paper, we extend the model of [11]. We introduce a more realistic, yet still general, model to bound the precision of IEEE802.1AS. We model more accurately the physical layer communication delay variations using the work of Loschmidt et al. [7]. An important contribution of our work is the definition of a communication model that accounts for both physical jitter and link asymmetries, that we illustrate in the result section for both 100Base-T and 1000Base-T technologies.

We propose as well a finer modelling of the other sources of inaccuracy and of the more precise two-step mode of IEEE802.1AS. We challenge our model and the model of [11] with thorough fine-grained simulations and measurement campaigns on network architectures representative of automobile [10], satellite [2] and airplane [14] networks. Our model being analytical, it scales to any network size. We show that our bound is two times less pessimistic than the state-of-the-art model of [11]. Experiments on a 1-hop platform show that our bound is 56ns higher in absolute value compared to the worst precision measured during 200 1-hour experiments. And finally, we show that 1000Base-T gives 41% smaller precision bound than 100Base-T on our platform.

II. IEEE 802.1AS overview

IEEE802.1AS is a IEEE1588 Precision Timing Protocol (PTP) profile for Time Sensitive Networking (TSN). It synchronizes time-aware systems clocks across a network using the master slave paradigm. Such a network is depicted in Figure 1. The Grandmaster (GM) broadcasts synchronization information on its Master ports (M). Each device receiving the information on a slave port (S) forwards it to its master ports, the passive ones (P) ignore them to avoid cyclic dependencies.

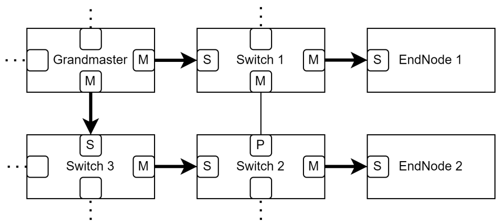

Fig. 1. IEEE802.1AS network architecture.

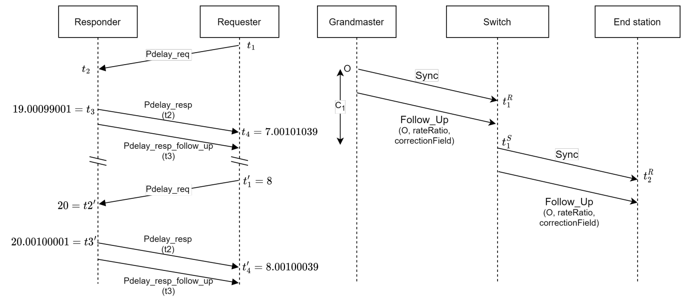

Fig. 2. Left : Two consecutive Peer-to-Peer delay exchanges. We get $nr = 1.00002$ and $D = 200$ns; Right : Synchronization distribution mechanism.

The selection of the Grandmaster and the synchronization tree (defined by the type of each port) can be chosen dynamically, using the Best Master Clock Algorithm (BMCA), or statically.

Synchronization relies on: *i)* the measurement of the link propagation delay with the Peer-to-Peer delay mechanism and *ii)* the distribution of synchronization informations.

The peer-to-peer delay mechanism uses three messages that are exchanged periodically (every second by default) between a requester and a responder. All the ports of a time-aware system are requesters, but also responders to respond to the request of the neighbor time-aware system. As depicted in Figure 2 - Left, four hardware timestamps are needed: $i)$ t_1 is measured when the `Pdelay_req` is issued ; $ii)$ t_2 is obtained upon reception of this message ; $iii)$ t_3 is measured when the `Pdelay_resp` is sent ; $iv)$ t_4 is measured upon reception of `Pdelay_resp`. In this paper we consider the two-step mode where t_3 is sent in a separate `Pdelay_resp_follow_up` message since it offers a higher precision. The standard also proposes an alternate one-step mode, where t_3 is transmitted in the `Pdelay_resp` message. The propagation delay of the link D, called the *Pdelay*, is given by:

$$D = \frac{nr \times (t_4 - t_1) - (t_3 - t_2)}{2} \tag{1}$$

nr is the *neighborRateRatio*. It compensates the relative clock drift and is defined using t_3 and t_4 timestamps from two consecutive *Pdelay* procedures as illustrated in Figure 2 - Left:

$$nr = \frac{f_{req}}{f_{resp}} = \frac{t'_3 - t_3}{t'_4 - t_4} \tag{2}$$

Eq. (1) assumes that the propagation time is symmetric. Existing asymmetries can be compensated if they can be estimated.

The distribution of synchronization information relies on the transmission of two messages. Every *syncInterval* (125ms by default), the Grandmaster sends a `Sync` message out of its master ports, followed by a `Follow_Up` message (two-step mode) containing O, the exact transmission time of the `Sync` message (a.k.a. the *preciseOriginTimestamp* in the norm), as pictured in Figure 2 - Right. `Sync` and `Follow_Up` are received via the slave ports of the time-aware system connected to the Grandmaster. If the receiving device has ports in the master state, it directly forwards the `Sync`. Next, it updates the `Follow_Up` message that carries the *preciseOriginTimestamp* O, the *rateRatio* r and the *correctionField* C, then sends it to the children time-aware systems.

The *rateRatio* r_i allows for logical syntonization of a time-aware system i to the Grandmaster rate. It is set to 1 by the Grandmaster and is updated on each hop with $r_i = r_{i-1} \times nr_i$, where i is the receiving node and $i-1$ the sending node.

The *correctionField* 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 *rateRatio* r_{i-1}, its current *neighborRateRatio* nr, its current value of D_i and the residence time $t_i^S - t_i^R$ of the `Sync` in its buffer:

$$C_i = C_{i-1} + D_i \times r_{i-1} + (t_i^S - t_i^R) \times r_{i-1} \times nr \tag{3}$$

At each `Sync + Follow_Up` reception, a time-aware system i calculates the difference between its local time and the estimated Grandmaster time GM_i to update its clock correction value that can be positive or negative. The Grandmaster time GM_i is estimated by system i:

$$GM_i(t) = O + C_{i-1} + D_i + (t - t_i^R) \tag{4}$$

where O is the *preciseOriginTimestamp*, C_{i-1} the *correctionField* transported by the `Follow_Up`, D_i the previous hop *Pdelay* retrieved by the peer-to-peer delay procedure and $(t - t_i^R)$ the time elapsed since the reception of the last `Sync`.

III. Modeling sources of inaccuracies

We now introduce a generic system model that we use for the formal development of the worst-case precision bound of Section IV. This model captures the sources of synchronisation inaccuracy due to the timing behavior of the network and the time-aware systems. They are related to $i)$ the physical inaccuracy of clocks like drift and granularity and $ii)$ the communication delay variability induced by the physical layer implementation of the network interface card.

A. Clock model

1) Clock drift ρ: Oscillators are imperfect: their oscillation frequency does not stay constant over time. This frequency variation, called drift rate, is measured in *parts per million (ppm)* defined by the number of seconds the local clock deviates in a million seconds of the reference time. The accuracy of an oscillator is characterized by a bound on this drift rate. For instance, an oscillator characterized with +10ppm (resp. -10ppm), may run up to 10μs faster (resp. slower) with respect to a perfect time every second. Practically, the drift varies over time due to aging or external conditions, such as temperature.

Drift is maximized when the clock undergoes a constant drift rate given by its oscillator upper bound (10ppm for instance). A clock can therefore be modeled with Eq. (5) where t_i is the time on device i, t_p the perfect time, ρ_i the bound on the drift rate of i oscillator and I the interval since the last synchronisation.

$$t_i = t_p + \rho_i \times I \tag{5}$$

This drift can be mitigated by periodic re-synchronization using IEEE802.1AS for instance. However, re-synchronization is prone to multiple inaccuracies that we model in the following.

2) Clock granularity G: The granularity G is the duration between two increments of the clock counter. Thus, each timestamp measured in the IEEE802.1AS protocol undergoes an error between 0 and G. Since synchronization mechanisms rely on measuring delays (i.e. differences of timestamps), any delay undergoes an error between $-G$ and G.

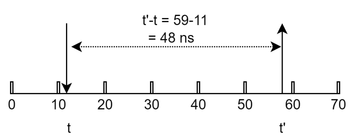

Fig. 3. Impact of clock granularity on a duration measurement.

The naive example in Figure 3 pictures the delay between the reception time t and transmission time t' of a message. Without granularity, this duration is 48ns. With a granularity of 10ns, the clock reading is of 10 at reception time and of 50 at transmission time, leading to a duration of 40ns.

B. Communication model

Implementation-specific features of the physical layer technology impact the accuracy of transmission delay measurement, as highlighted by [6] [13]. Although the propagation delay on the link is constant, the delay between the message timestamping and its actual transmission (or between the reception and its timestamping) varies due to hardware implementation and transmission technology. It triggers two kinds of inaccuracies:

- A physical jitter J that varies over time according to a distribution. On a given link, Loschmidt's measurements show that the distribution of this jitter may depend on the direction of the communication. For instance, for 1000Base-T, the delay follows a uniform distribution on one direction and a normal one on the other direction, with different widths. For 100Base-T, the jitter follows the same normal distribution in both directions.
- A constant link asymmetry latency A that induces a larger delay in one direction. It is related to technological choices. For instance, a link layer using an optical fiber where a different wavelength is used per direction induces an asymmetric propagation delay.

A refined characterization of the communication delay is captured by the communication model in Fig 4. It is characterized by a directional communication delay and jitter, link asymmetry latency and residence time. Numerical values have to be set according to the physical layer characteristics.

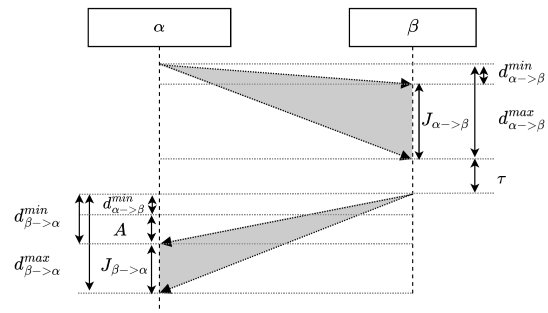

Fig. 4. Illustration of the communication model.

Directional communication delay and jitter: We assume an asymmetric communication delay. For two time-aware systems α and β, the delay $d_{\alpha\to\beta}$ from α to β belongs to an interval $d_{\alpha\to\beta} \in \left[d^{\min}_{\alpha\to\beta}, d^{\max}_{\alpha\to\beta}\right]$. The size of this interval is defined as the jitter $J_{\alpha\to\beta}$:

$$J_{\alpha\to\beta} = d^{\max}_{\alpha\to\beta} - d^{\min}_{\alpha\to\beta} \tag{6}$$

This jitter can be set according to a characterization of its width distribution for a target PHY layer from extensive measurements similar to the ones done in [13]. Similarly, delay $d_{\beta\to\alpha}$ and corresponding jitter $J_{\beta\to\alpha}$ are defined for direction $\beta \to \alpha$. Directional communication delays and jitters are illustrated in Figure 4.

Link asymmetry latency A: In the case of an asymmetrical propagation channel, a constant latency A is added to the delay of one direction. In Figure 4, a link asymmetry latency A is added for direction $\beta \to \alpha$. Thus:

$$d^{\min}_{\beta\to\alpha} = d^{\min}_{\alpha\to\beta} + A \text{ and } d^{\max}_{\beta\to\alpha} = d^{\min}_{\alpha\to\beta} + A + J_{\beta\to\alpha} \tag{7}$$

Residence time τ: Any back-to-back request-response synchronization mechanism, such as the one used for the `Pdelay` computation, necessitates some processing time on the responder side before transmission. This processing time is typically called the residence time and denoted τ.

IV. Bounding the worst-case precision

This section gives the main developments leading to the computation of upper and lower bounds. Starting from the original model in [11], we derive a less pessimistic model of duration measurement error and residence time error, we introduce errors caused by link asymmetry, `neighborRateRatio` measurement inaccuracies, the two-step mode of IEEE802.1AS and variations of the periodic synchronisation interval induced by other flows.

A. Upper and lower bounds on synchronization precision

The instantaneous precision $P_i(t)$ of a time-aware system i is the difference between its estimation of the Grandmaster clock $t_i(t)$ and the Grandmaster clock $t_{GM}(t)$:

$$P_i(t) = t_i(t) - t_{GM}(t) \tag{8}$$

Let P_i^U (resp. P_i^L) be the upper (resp. lower) bound on precision of system i. These bounds are constructed to meet the following constraints:

$$P_i^U \geq \max_t P_i(t) \quad \text{and} \quad P_i^L \leq \min_t P_i(t) \tag{9}$$

This precision depends on two quantities: first, the relative clock drift between the Grandmaster and time-aware system i since last synchronisation point, second, the wrong estimation of the Grandmaster time by the time-aware system i at the previous synchronisation point, which is due to the implementation-specific sources of inaccuracy modeled before.

Let's note $E_{drift_i}(t)$ the clock drift at time t between the Grandmaster and the time-aware system i since the last synchronisation point. This drift lies between $-E^U_{drift_i}$ and $E^U_{drift_i}$. Let's also denote δGM_i the error in the estimation of the Grandmaster clock at the last synchronisation point. It lies between δGM_i^L and δGM_i^U. Consequently, we have:

$$P_i^U = E^U_{drift_i} + \delta GM_i^U \tag{10}$$

$$P_i^L = -E^U_{drift_i} + \delta GM_i^L \tag{11}$$

B. Derivation of $E^U_{drift_i}$

The worst drift occurs when the clocks of the Grandmaster and the time-aware system i drift in opposite directions. This drift is corrected at each synchronisation point, and it increases until the next synchronisation point. Therefore, the largest possible drift is observed right before a synchronisation point. Let's assume that previous synchronization occurred I time units ago. The bound $E^U_{drift_i}$ is given by:

$$E^U_{drift_i} = I \times (|\rho_i| + |\rho_{GM}|) \tag{12}$$

In an ideal situation, I would be equal to *syncInterval* I_s. However, in practice, `Sync` and `Follow_Up` can be delayed by other messages in switch queues. The worst situation is when the first `Follow_Up` message undergoes the smallest possible network traversal delay, while the second one undergoes the largest possible one. In this case, the delay I is the sum of the *syncInterval* I_s and the largest network jitter J_{fup} that the `Follow_Up` message can experience : $I = I_s + J_{fup}$.

If the topology of the network, the port queuing disciplines (TAS, CBS, etc.) and flows are known, J_{fup} can be upper bounded using a worst-case latency analysis, e.g. [15].

C. Derivation of δGM_i^U

δGM_i^U (resp. δGM_i^L) represents an upper bound (resp. lower bound) on the Grandmaster's time estimation error on the time-aware system i made at the last synchronization point. We develop the construction of δGM_i^U and provide final equations for its lower bound counterpart in Table I.

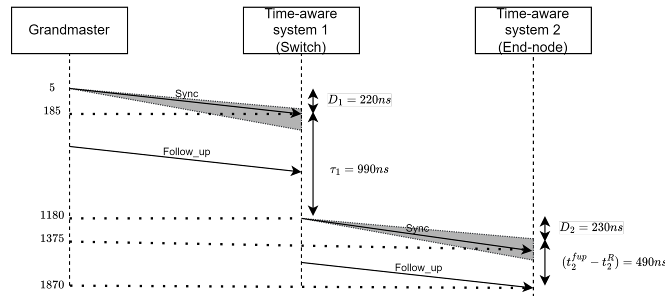

Fig. 5. Illustration of errors that impact GM_i. $G = 10$ns for all systems.

Let's consider the time when the time-aware system i receives a `Follow_Up` message. Let's denote this time t^{fup}_{GM} if expressed in the Grandmaster reference clock and t_i^{fup} if expressed in the clock of time-aware system i. Upon `Follow_Up` message reception, time-aware system i calculates its estimation of the Grandmaster current time using (4):

$$GM_i = O + C_{i-1} + D_i + (t_i^{fup} - t_i^R) \tag{13}$$

By definition, δGM_i^U is upper-bounding the error between its estimation of the Grandmaster clock and t^{fup}_{GM}:

$$\delta GM_i^U \geq \delta GM_i \quad \text{with} \quad \delta GM_i = GM_i - t^{fup}_{GM} \tag{14}$$

This error is illustrated in Figure 5 where in the time base of the Grandmaster, the `Sync` is transmitted at time 5ns and the last `Follow_Up` message is received at time 1870ns because the communication delay is of 180ns for the first hop and 195ns for the second hop, the residence time in the switch is of 995ns and the delay $t_2^{fup} - t_2^R$ is of 495ns.

Conversely, the end node gets $GM_i = 1930$ns, leading to an error of $\delta GM_i = 60ns$ because the *Pdelay* mechanism estimates a link delay of 220ns instead of 180ns and 230ns instead of 195ns, the clock granularity of the switch induces an under-estimation of τ_1 and the clock granularity of the end node an under-estimation of $t_2^{fup} - t_2^R$. Moreover, the initial `Sync` is transmitted at 5ns but the `Follow_Up` carries a value $O = 0$ns because of the clock granularity of the Grandmaster.

The worst synchronization error is observed when the synchronization protocol triggers an estimate of the `Sync` traversal time that is as large as possible compared to the smallest possible `Sync` network traversal delay. The bound on the synchronization error is the sum of bounds on the errors induced by the different components of GM_i:

$$\delta GM_i^U = \delta O^U + \delta C_{i-1}^U + \delta D_i^U + \delta(t_i^{fup} - t_i^R)^U \tag{15}$$

with δD_i^U the upper bound on the *Pdelay* error and δC_{i-1}^U the upper bound on the *correctionField* error. Both types of errors originate from the *Pdelay* mechanism. The *correctionField* error originates as well from errors on the *rateRatio* and on the *residence time* estimation. Bounds on δO^U and $\delta(t - t_i^R)^U$ are a consequence of the granularity on timestamps readings.

In the model of [11], δO is neglected and $\delta(t - t_i^R)$ is not accounted for. In our version, $\delta(t - t_i^R)$ captures the more precise two-step mode of IEEE802.1AS. Our derivation of

δD_i^U differs from [11] since it captures the communication channel asymmetries, the *neighborRateRatio* error and finer residence time error. The derivation of δC_{i-1}^U follows the one of [11] but its numerical values change since it relies on δD_i^U.

1) Bounding Pdelay error with δD_i^U: δD_i^U bounds δD_i, the error made by system i when it estimates the link delay with its parent system j. We have $\delta D_i^U = D_{worst} - D_{best}$, with D_{worst} the highest estimation of *Pdelay* and D_{best} the smallest error-free one, i.e. $d_{j\to i}^{\min}$. Computing δD_i^U comes to maximize D_{worst}. Since D_{worst} follows Eq. (1), it comes to maximize nr and $(t_4 - t_1)$ and minimize $(t_3 - t_2)$.

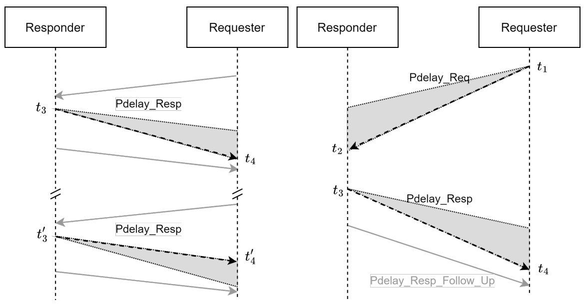

Fig. 6. Left: Illustration of the `Pdelay_resp` propagation delay variation with jitter J (solid gray) and the worst-case propagation delay scenario for the *neighborRateRatio* computation (black dashed arrow)
Right: Illustration of the `Pdelay_req` and `Pdelay_resp` propagation delay variation due to jitter J (solid gray) and the worst-case propagation delay scenario for the *Pdelay* computation (black dashed arrow)

Maximizing nr over-estimation: We define $\delta nr_i^U = nr_{i_{worst}} - nr_i$, with $nr_{i_{worst}}$ the largest possible value of the *neighborRateRatio* and nr_i the error-free one. From Eq. 2, computing $nr_{i_{worst}}$ comes to maximize the numerator $t'_3 - t_3$ and minimize the denominator $t'_4 - t_4$, compared to the real delay that led to these timestamps.

$t'_3 - t_3$ being the difference between two timestamps internal to a time-aware system, the maximum over-estimation of this difference is a granularity unit G (see section III-A2).

For the same reason, the maximum under-estimation of $t'_4 - t_4$ includes a granularity unit G. Additionally, it is impacted by the physical jitter. Indeed, as illustrated in Fig, 6 - Left, the variation of the propagation delay due to physical jitter can lead to a underestimation of $-J_{j\to i}$. In practice, the worst $t'_4 - t_4$ delay happens if the first `Pdelay_resp` message experiences a propagation delay of $d_{j\to i}^{\max}$ and the timestanp t_4 is taken exactly on a clock tick, while the second one experiences the smallest propagation delay $d_{j\to i}^{\min}$ and t'_4 is taken an arbitrary small instant before a clock tick.

Finally, clock drift also has an impact on the computation. A maximum positive drift ρ_j in the responder increases $t'_3 - t_3$ because the clock is faster than reality. Conversely, a maximum negative drift $-\rho_i$ decreases $t'_4 - t_4$. Overall δnr is :

$$\begin{aligned}\delta nr_i^U &= nr_{i_{worst}} - nr_i \\ &= \frac{t'_3 - t_3 + G}{t'_4 - t_4 - G + (d_{j\to i}^{\min} - d_{j\to i}^{\max})} - \frac{t'_3 - t_3}{t'_4 - t_4} \qquad (16) \\ &= \frac{2G + G \times (\rho_j - \rho_i) + J_{j\to i} \times (1 + \rho_j)}{I_p \times (1 - 2\rho_i + \rho_i^2) + (\rho_i - 1) \times (G + J_{j\to i})}\end{aligned}$$

Maximizing $t_4 - t_1$: $t_4 - t_1$ is the duration between the transmission of a request and the reception of a response message. Similarly to the $t'_4 - t_4$ case in nr computation, $t_4 - t_1$ is impacted by the granularity G, the variable propagation delay and the physical asymmetry A. Adding the granularity G maximizes $t_4 - t_1$ (like for $t'_3 - t_3$). The maximum propagation delay $d^{\max}$ occurs twice: once for the `Pdelay_req`, once for the `Pdelay_resp`, as illustrated in Figure 6 - Right. The impact of the physical asymmetry A is experienced in one direction.

Minimizing $t_3 - t_2$: Since $t_3 - t_2$ is a duration between two internal events of a system, granularity G has to be removed.

Calculus of δD_i^U: To summarize, the bound δD_i^U is:

$$\begin{aligned}\delta D_i^U &= \frac{\max(t_4 - t_1)(nr_i + \delta nr_i^U) - \min(t_3 - t_2)}{2} - d_{j\to i}^{\min} \\ &= [(\tau_i + 2d_{j\to i}^{\min} + J_{j\to i} + J_{i\to j} + A)(\rho_i + 1) + G] \\ &\quad (nr_i + \delta nr_i^U) - [\tau_i(1 - \rho_j) - G]/2 - d_{j\to i}^{\min}\end{aligned}$$

2) Bounding the correctionField error : The *correctionField* C_{i-1} in a time-aware system $i-1$ is computed by Eq. (3). It depends on the *correctionField* and the *rateRatio* in the previous system $i-2$, the *neighborRateRatio* and the *Pdelay* between systems $i-2$ and $i-1$ and the *residence time* in $i-1$. Worst-case values of *neighborRateRatio*, *Pdelay* and *residence time* $(t_3 - t_2)$ have been set in part IV-C1.

The *rateRatio* is computed by $r_i = r_{i-1} \times nr_i$. For the rest of the paper, we assume (as done in [11]) that all time-aware systems are identical: same clock with the same drift rate bounds, granularity and physical interface with the same physical jitter and asymmetries. Thus $nr_1 = ... = nr_i = nr$ and $\delta nr_1{}^U = ... = \delta nr_i{}^U = \delta nr^U$. Therefore we have: $r_i = nr^i$. To calculate the bound on *rateRatio* overestimation $\delta r_i{}^U$, we apply the following derivation:

$$\delta r_i^U = (nr + \delta nr^U)^i - nr^i = i \times nr^{i-1}\delta nr^U + ... + (\delta nr^U)^i$$

In order to simplify this equation, the powers of δnr are neglected since they are very small compared to the main term $i \times nr^{i-1}$ as done in [11] and thus:

$$\delta r_i^U \approx i \times nr^{i-1} \times \delta nr^U \qquad (17)$$

We can now compute the upper bound on the error δC in a time-aware system $i-1$. From Eq. (3), we have:

$$\begin{aligned}&\delta C_{i-1}^U = C_{i-1_{worst}} - C_{i-1} \\ &= [C_{i-2} + \delta C_{i-2}^U + (d_{(i-2)\to(i-1)}^{\min} + \delta D_{i-1}^U)(r_{i-2} + \delta r_{i-2}^U) \\ &\quad + (\tau_{i-1} + G)(r_{i-1} + \delta r_{i-1}^U)] \\ &\quad - [C_{i-2} + d_{(i-2)\to(i-1)}^{\min} r_{i-2} + \tau_{i-1} r_{i-1}]\end{aligned}$$

As for the *neighborRateRatio*, we assume that our time-aware systems use the same hardware. Thus we have $d_{GM\to 1}^{\min} = ... = d_{(i-2)\to(i-1)}^{\min}$, $\delta D_0^U = ... = \delta D_{i-1}^U$ and $\tau_0 = ... = \tau_{i-1}$. Thus, previous equation simplifies to:

TABLE I
LOWER BOUND FORMULAS ON SYNCHRONIZATION PRECISION.

P_i^L	$P_i^L = -(\lvert\rho_i\rvert + \lvert\rho_{GM}\rvert)(I_s + J_{fup}) + \delta GM_i^L$
δGM_i^L	$\delta C_{i-1}^L + \delta D_i^L - 2G$
δC_{i-1}^L	$\delta D_{i-1}^L(\frac{nr^{i-2}-1}{nr-1}) - G(\frac{nr^{i-1}-1}{nr-1} - 1)$ $+\delta nr((d^{\max}_{(i-2)\to(i-1)} + \delta D_{i-1}^L)\sum_{j=0}^{i-2} j \times nr^{j-1}$ $+(\tau_{i-1} - G)\sum_{j=1}^{i-1} j \times nr^{j-1})$
δD_i^L	$\frac{[(\tau_i+2d^{min}_{i\to j}+A)(1-\rho_i)-G](nr+\delta nr^L)-(\tau_i(1+\rho_j)+G)}{2}$ $-(d^{\min}_{i\to j} + J_{j\to i} + A)$
δnr^L	$\frac{-[2G+J_{j\to i}(1-\rho_j)+G(\rho_i-\rho_j)]}{I_p(1+2\rho_i+\rho_i^2)+(\rho_i+1)(G+J_{j\to i})}$

$$\delta C_{i-1}^U = \delta D_{i-1}^U \left(\frac{nr^{i-2}-1}{nr-1}\right) + G\left(\frac{nr^{i-1}-1}{nr-1} - 1\right) + \delta nr^U (d^{\min}_{(i-2)\to(i-1)} + \delta D_{i-1}^U)\sum_{j=0}^{i-2} j \times nr^{j-1} + \delta nr^U(\tau_{i-1} + G)\sum_{j=1}^{i-1} j \times nr^{j-1} \quad (18)$$

3) Bounding δO and $\delta(t_i^{fup} - t_i^R)$: The *preciseOriginTimestamp O* gets the value of the clock at the last tick no later than the current instant. Thus O can be under-approximated by up to the granularity G. This erroneous timestamps is carried in the `Sync` messages but can only reduce the value of GM_i. Therefore O cannot be over-approximate and $\delta O^U = 0$.

$(t_i^{fup} - t_i^R)$ is the duration between the reception of the `Sync` and the reception of the `Follow_Up` (when the correction occurs). Being a duration between two events in the time-aware system, it can be over-approximated by the granularity G.

D. Upper bound on precision P_i^U

Finally, we can express the upper precision bound P_i^U as the sum of the drift between the Grandmaster's clock and the time-aware system's clock since the last synchronisation and errors that occurred by estimating the Grandmaster's time on the time-aware system i:

$$P_i^U = (\lvert\rho_i\rvert + \lvert\rho_{GM}\rvert)(I_s + J_{fup}) + \delta C_{i-1}^U + \delta D_i^U + G \quad (19)$$

V. RESULTS

First we compare our model to the state of the art one for 100Base-T technology. Second we instantiate the model with 1000Base-T links and compare the results with the bound obtained with 100Base-T links.

A. Bound tightness validation

Simulations, exhaustive search and measurements are leveraged to address two questions: how close our bound on worst-case precision is and how it compares to the previous model of Gutiérrez et al. [11].

TABLE II
100BASE-T AND 1000BASE-T PARAMETERS.

	G	ρ_{GM}	ρ_{Slave}	d^{min}	$J_{j\to i}$	$J_{i\to j}$
100Base-T	10ns	0.02ppm	10ppm	200ns	75ns	75ns
1000Base-T	10ns	0.02ppm	10ppm	200ns	29.7ns	8ns

	A	τ	J_{fup}
100Base-T	32ns	1ms	2ms
1000Base-T	6.85ns	1ms	2ms

In order to provide a fair comparison, we instantiate our model for 100Base-T links. Unless mentioned, we use the values of granularity, clock drift, propagation delay, jitter and asymmetry from our previous work [13]. These values are specific to the switches that we use as well and thus allows a very fine comparison between the experimental measurements and the bound. Since 100Base-T jitter does not depend on the direction, we denote $J = J_{i\to j} = J_{j\to i}$. For the residence time τ, we use a common value of the literature [11]. J_{fup} is set to 2ms based on network calculus analysis using a commercially available tool[1] on our embedded use-case. Numerical values are given in Table II. For the protocol configuration, we use the default parameters of AS: $I_s = 0.125s$ and $I_p = 1s$.

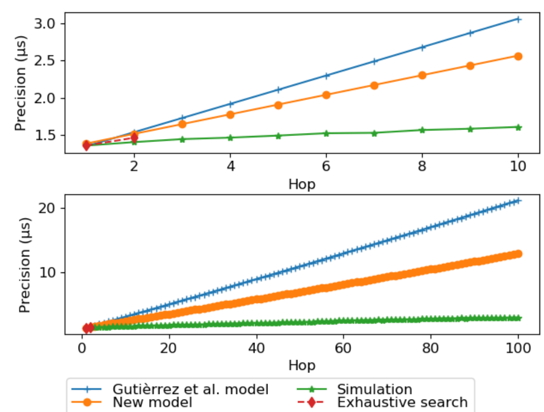

Fig. 7. Comparison of simulated, exhaustive search and precision bound for 100Base-T physical layer according to the number of hops.

1) Simulation and exhaustive search study: Figure 7 compares, as a function of the number of hops, our upper bound and the one of Gutiérrez et al. [11] to simulations calculated with our open-source simulation library [13] and to an exhaustive search as detailed later. Results are produced with the parameters of Table II, except for the Grandmaster drift which is set to 0ppm (perfect clock assumption) and for J_{fup}, set to 0 as well, since we don't simulate data traffic.

For the sake of fairness, we integrate δD_i^U in δGM_i^U for the derivation of [11]. Indeed, authors neglect δD_i^U because they evaluate their bound on a 100-hop network. With a 10-hop network, δD_i^U is not negligible anymore.

Simulation results are obtained from a set of 400 1-hour and 10 12-hour simulations for the upper part and a set of 10 1-hour and 1 12-hour simulations for the lower part of Fig. 7. We

[1] https://www.realtimeatwork.com/rtaw-pegase/

have randomized initial settings (initial clock desynchronization, AS mechanism start time, physical asymmetry) except for the slave clock drift which is set to the worst value (i.e. 10ppm) for fair comparison with the bounds. We have kept the worst precision recorded at each hop among all runs.

The exhaustive search is carried out by testing all the possible combinations of the parameter values in order to determine the time of transmission or reception of synchronization messages, and deduce the timestamps and synchronization calculations. The worst offset between the Grandmaster's clock and the clock of a time-aware system is recorded across all executions. Parameters range and sampling interval are chosen as follows. For the propagation delay, the start time of synchronization, the delay between the reception of a `Pdelay_req` and the transmission of the `Pdelay_resp` or the delay between the transmission of a `Sync` and its corresponding `Follow_Up`, an interval of one granularity is set since it is enough to capture the worst error. The physical jitter is evaluated over its entire interval $[0, J]$. The sampling size has been chosen to get a tractable resolution and meaningful results for a 2-hop network topology. A sampling step of 1.5ns (resp. 0.05ns) for the 2-hop (resp. 1-hop) network triggers 15 billion (resp. 4.1billion) combinations. We limit the search to 2 hop due to combinatorial explosion and as is enough to cover all AS mechanisms.

From Figure 7 - Top, we observe that our bound is two times closer to the worst precision observed during the simulation when compared to the one of Gutiérrez et al. Their larger pessimism is due to an overestimation of the error impacting some delays: the error related to the physical jitter is accounted for any duration while this error never happens for the `Sync` residence time duration or for the duration between the reception of `Pdelay_Req` and the transmission of `Pdelay_Resp`. Moreover, the errors caused by the granularity on a duration measurement are also overestimated in the model of [11] compared to our model. This pessimism is even more obvious with a 100-hop network, as shown in Fig. 7 - Bottom. After 100 hops, the state-of-the-art model reaches 21.174 µs while our bound is 12.843 µs. The simulation reaches 2.952 µs, which is far from the bound because the sequence of events which leads to the worst-case is less likely as the number of hops increases. The evolution of the bounds according to the number of hops being linear, in the following we focus on networks more representative of embedded networks i.e. up to 10-hop. From a complexity point of view, both models are implemented in $\mathcal{O}(N)$.

Figure 8 focuses on the first two hops. For each hop, it shows the precision distribution obtained by simulations, the results of the exhaustive search and the bounds. We observe that our bounds are very close to the exhaustive search worst observation for the 2 first hops. Indeed, for the upper bound (resp. lower), we observe a difference with the exhaustive search of 5.4% (resp. 9.4%) for the first hop and 5.4% (resp. 9.9%) the second hop. We also see that the model of Gutiérrez et al. fails at the first hop as it produces an upper bound which is smaller than the worst observed precision with the simulator and the exhaustive search because [11] doesn't consider the 100Base-T asymmetries.

2) Experimental validation: A 3-hop chain of four Fraunhofer IPMS switches has been deployed where the first switch of the chain acts as the Grandmaster. A netTimeLogic PPS analyzer captures clock progress. The switches use two-step mode, a *pdelayInterval* of 1s and a *syncInterval* of 0.125ms.

20 experiments of 1-hour of precision measurements have been made. Each experiment records the worst and the best precision observed over time. Between each experiment, the interfaces are reset to allow measurements with different random combinations of asymmetry. Since no data traffic was transmitted during the experiments we set $J_{fup} = 0$. To compute our upper and lower mathematical bounds we use the free-running clock drift measured during 10 minutes before each run, relatively to the Grandmaster clock.

Figure 9 compares the upper and lower bounds obtained with our worst-case model to the smallest and largest precision records made over the 20 experiments at each hop. These results show that the bounds nicely frame the actual measurements with little pessimism despite the small amount of measurements made. Moreover, as for the simulation, we observe that when the number of hops increases, it gets harder to measure worst-case events.

We did another campaign of 200 1-hour experiments for a 1-hop network. Since the drift was negative, we focus on lower bound. The smallest precision value recorded is -276ns while the lower bound is -332ns, leading to a difference of 56ns, which exhibits the reduced pessimism of our model.

B. Comparing 1000Base-T to 100Base-T

Parameters for the 1000Base-T model instance are derived from our switches by applying the method of [13] (Table II).

Obtained results are similar to 100Base-T ones. For the upper bound, the difference is 20.1% (resp. 20.5%) between the bound and the exhaustive search at 1 hop (resp. 2 hops). For the lower bound it is 17.3% (resp. 19.8%) at 1 hop (resp. 2

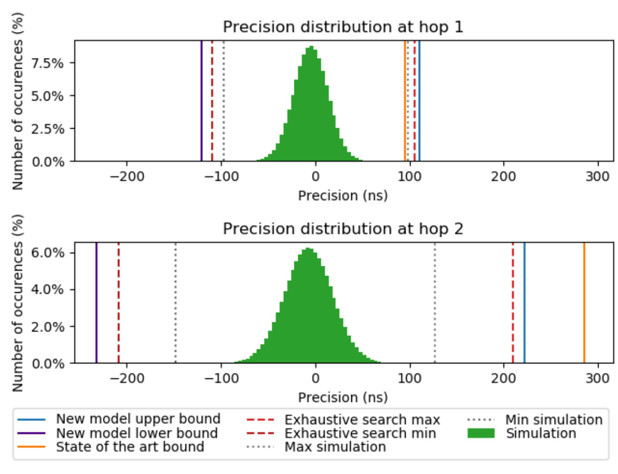

Fig. 8. Comparison of simulated, exhaustive search and precision bound for 100Base-T physical layer on hop 1 and 2

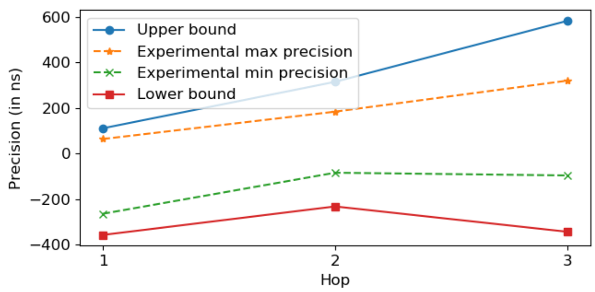

Fig. 9. 100Base-T upper and lower bounds compared to measurements.

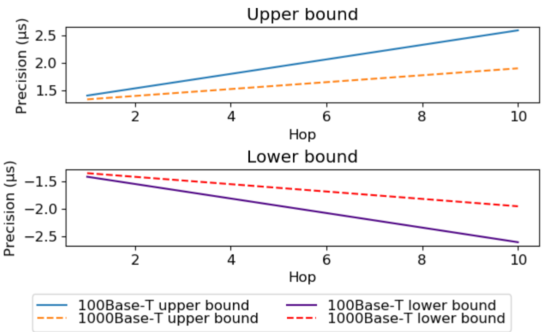

Fig. 10. Comparison of 100Base-T and 1000Base-T precision bounds

hops). This greater difference is explained by the fact that the combination of jitter and granularity obtained for our switches with 1000Base-T does not allow us to meet the conditions described in Section IV and reach the worst case. For example, for the *neighborRateRatio*, the exhaustive search can't observe the condition that leads to the worst $t'_4 - t_4$ delay.

Figure 10 compares the upper and lower bounds obtained for 100Base-T and 1000Base-T instances for our TSN switches. We observe that 1000Base-T gives a more precise bound than 100Base-T: the 1000Base-T upper bound (resp. lower bound) is 36% (resp. 33%) lower after 10 hops. This is due to the smaller physical jitter and asymmetries in the physical layer, thus reducing the worst-case error in the *Pdelay* mechanism (for the upper bound : $\delta D^U = 52.31ns$ for 1000Base-T compared to $\delta D^U = 121.06ns$ for 100Base-T). It does not guarantee that 1000Base-T always offers better precision than 100Base-T with any time-aware system. It is only valid for the hardware used in this study.

VI. Conclusion

In this paper, we propose a refined analytical model to upper or lower bound the precision of an IEEE802.1AS. These bounds rely on a generic communication model which captures link jitter and asymmetries. This communication model can be implemented for different Ethernet physical layer technologies using appropriate parameters. For 100Base-T links, we have shown that the upper bound on synchronization offers reduced pessimism with respect to the state-of-the-art. The quality of our bounds comes as well from a refined characterization of the clock inaccuracies and protocol operations.

In terms of future works, we will leverage these bounds to guarantee a deployment of IEEE802.1AS where a given worst-case precision is ensured, even if some network failures occur. This new model could also be extended to support the wireless physical layer, such as WiFi, to meet the needs of industrial automation networks.

Acknowledgment

This work is supported by the French Research Agency (ANR) and the partners of IRT Saint-Exupéry Scientific Cooperation Foundation (FCS) : 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] M. Ashjaei, L. Lo Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-Sensitive Networking in automotive embedded systems: State of the art and research opportunities," *Journal of Systems Architecture*, vol. 117, p. 102137, 2021.

[2] P.-J. Chaine, M. Boyer, C. Pagetti, and F. Wartel, "TSN support for quality of service in space," in *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, 2020.

[3] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, 2016.

[4] "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.

[5] "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.

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

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

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

[9] M. D. Johas Teener and G. M. Garner, "Overview and timing performance of IEEE 802.1AS," in *2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2008, pp. 49–53.

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

[11] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "Synchronization Quality of IEEE 802.1AS in Large-Scale Industrial Automation Networks," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2017, pp. 273–282.

[12] 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.

[13] Q. Bailleul, K. Jaffrès-Runser, J.-L. Scharbarg, and P. Cuenot, "Assessing a precise gPTP simulator with IEEE 802.1AS hardware measurements," in *ERTS*, 2022.

[14] O. A. Hotescu, K. Jaffrès-Runser, J.-L. Scharbarg, and C. Fraboul, "Towards Quality of Service Provision with Avionics Full Duplex Switching," in *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, Dubrovnik, Croatia, Jun. 2017, pp. 1–3. [Online]. Available: https://hal.science/hal-01919072

[15] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, and M. Boyer, "Latency analysis of multiple classes of avb traffic in tsn with standard credit behavior using network calculus," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 10, pp. 10 291–10 302, 2020.