Synchronization Quality of IEEE 802.1AS in Large-Scale Industrial Automation Networks

Marina Gutiérrez, Wilfried Steiner
TTTech Computertechnik AG
{*marina.gutierrez, wilfried.steiner*}*@tttech.com*

Radu Dobrin, Sasikumar Punnekkat
Mälardalen University
{*radu.dobrin, sasikumar.punnekkat*}*@mdh.se*

***Abstract*—Industry 4.0 and Industrial Internet of Things projects work towards adoption of standard IT technologies for real-time control networks in industrial automation. For this the IEEE 802.1 Time-Sensitive Networking (TSN) Task Group has developed and continues to develop a set of standards. One of these standards is the IEEE 802.1AS clock synchronization protocol. IEEE 802.1AS can be used to enable time-triggered communication as well as to coordinate distributed actions in industrial networks.**

In this paper we study the synchronization quality of IEEE 802.1AS and we are interested in whether the clocks can be synchronized with sufficiently low error such that the protocol can be used for demanding industrial automation applications. In particular, we study the protocol behavior in large-scale networks while considering critical implementation details. We report analytical worst-case results as well as probabilistic results based on simulations, that show that implementation details such as the PHY jitter and the clock granularity have a big impact on the time synchronization precision.

I. Introduction

Industrial automation networks often implement a pyramid structure according to the Purdue Reference Model, which implies several levels (typically four levels) of communication. The bottom levels address process control and networks deployed in these levels need to guarantee hard real-time communication behavior. The upper levels in the pyramid have relaxed real-time requirements and typically address production management and business planning. Following the different objectives of these levels an industrial plant implements different communication technologies for the different levels: specific fieldbus technologies and industrial Ethernet on the lower levels and Ethernet and IP networks on the higher levels. Ongoing projects in the context of Industrie 4.0 and Industrial Internet of Things (IIoT) [1] work towards a unified communication solution that is applicable for all levels in the communication pyramid. This unified solution is based on IEEE 802.3 Ethernet and is currently under standardization by the IEEE 802.1 Time-Sensitive Networking (TSN) Task Group. The analysis of the TSN protocols for their suitability to real-time communication is, thus, essential for their adoption in the lower levels of industrial communication. We are performing such an analysis in this paper.

In particular, one core protocol standardized by TSN is the IEEE 802.1AS clock synchronization protocol [2]. IEEE 802.1AS defines a master-slave protocol with leader election, i.e., the protocol automatically elects a master clock in the system, which is then called the "grandmaster". This grandmaster synchronizes the clocks in the network along an automatically established synchronization spanning tree of which the grandmaster is the root. The most important quality parameter of a synchronization protocol like IEEE 802.1AS is the "precision", which is defined as the maximum difference of the local clocks in the network of non-faulty nodes at any point in time during the operation of the system. The synchronization precision is a crucial parameter when used together with a time-triggered communication principle, as for example IEEE 802.1Qbv (another TSN mechanism), that allows maximizing the achievable network utilization or accurately coordinating distributed activities in the system. Typically, in an industrial network we aim to achieve a precision of less than one microsecond. In this paper we study the precision of IEEE 802.1AS.

We consider large industrial automation networks with topologies of up to a hundred switches in between the grandmaster clock and a slave clock. Such large-scale networks may be found in motion control applications in which small three-port switches connect to each other in a daisy-chain topology and each switch also connects to a switch-local processing node. Due to the network size we study the precision of IEEE 802.1AS both analytically and by means of simulation using OMNeT++ [3]. Indeed, we expect that the theoretical results achieved in this paper will accelerate the adoption of TSN in industrial settings.

Clock synchronization algorithms have been extensively studied for almost four decades. Lamport [4] is among the first to formally describe the synchronization of logical clocks as well as the synchronization of real-time clocks. Lundelius and Lynch [5] researched fundamental limits in the achievable precision of clock synchronization algorithms in distributed systems. Various protocols for fault-tolerant clock synchronization have been proposed, for example by Lamport [6] and Kopetz and Ochsenreiter [7]. Cristian and Fetzer [8], [9] have established a theory of probabilistic remote clock reading that is actually applied in today's IEEE 1588 and IEEE 802.1AS standards. More recently, the white rabbit project at CERN demonstrated that time synchronization can be achieved in a network with sub-nanosecond accuracy [10].

IEEE 1588 and IEEE 802.1AS have been studied by means of simulation before, for example in [11], but implementation-

978-1-5090-5269-1/17 $31.00 © 2017 IEEE

specific sources of inaccuracy, such as physical layer communication jitter or the local clock granularity have not been explicitly addressed. These implementation-specific details are, however, relevant factors, especially in large-scale networks, because their effect accumulate as the time synchronization information propagates through long paths. These factors are also not taken into account in IEEE 802.1AS, thus, the main contribution of this paper is the evaluation of the influence of these effects on the time synchronization precision that can be achieved using IEEE 802.1AS in large-scale networks.

We continue in the next section with a more detailed background on IEEE 802.1AS. In Section III we present the assumptions and configuration for our system model. Sections IV and V present the main contributions of this paper with the theoretical worst-case analysis and the results of the simulations respectively. Then, in Section VI we comment the main differences between the work presented in this paper and other performance tests on clock synchronization protocols. We conclude in Section VII with some final remarks.

II. IEEE 802.1AS Overview

IEEE 802.1AS is a standard that specifies the clock synchronization protocol to be used in 802.1 bridged networks. An 802.1 bridged network is composed by two types of elements: bridges and end stations. Because they must satisfy the requirement of being able of transporting time synchronization information they are generically referred as time-aware systems.

The clock synchronization protocol defined in IEEE 802.1AS is based on the master-slave paradigm. The grandmaster of the network is chosen using the best master clock algorithm (BMCA) defined in IEEE 1588. The time-aware systems that are not the grandmaster have all a single slave port through which they receive time synchronization information. Additionally, time-aware systems might have one or more master ports through which they transmit time synchronization information. IEEE 802.1AS ensures that two time-aware systems that are six or less hops apart will be synchronized with a precision of 1 μs.

Given that the capability of timestamping messages is core to the synchronization protocol and that this is a media-dependent feature, IEEE 802.1AS contains different specifications for full-duplex Ethernet, IEEE 802.11 (WiFi) and IEEE 802.3 Ethernet passive optical network (EPON). This paper focuses on the IEEE 802.1AS specification for full-duplex Ethernet. For full-duplex Ethernet networks, IEEE 802.1AS defines a profile of the Precision Time Protocol, PTP, as specified by IEEE 1588. This profile known as the generalized PTP, gPTP, includes also some specifications that go beyond PTP. To achieve time synchronization the gPTP describes two mechanisms: the transport of time synchronization information and the propagation delay measurements. Next those two mechanisms are described in detail. The notation used in this paper can be found in Table I.

IEEE 802.1AS	symbol	description
preciseOriginTimestamp	O	time sync information
correctionField	C	time sync information
rateRatio	$r(i\|GM) = r_i$	time sync information
neighborRateRatio	$r(i\|n) = nr_i$	time sync measurement
Pdelay	D_i	time sync measurement
syncLocked	`syncLocked`	configuration parameter
syncInterval	I	configuration parameter
PdelayInterval	I_D	configuration parameter
$\max\|RTE\|$	p	synchronization precision
-	J	PHY jitter
-	τ	residence time
-	g	clock granularity
-	ρ	clock drift rate

TABLE I
MAPPING OF THE NOTATIONS OF THE IEEE 802.1AS TO THE NOTATIONS USED IN THIS PAPER.

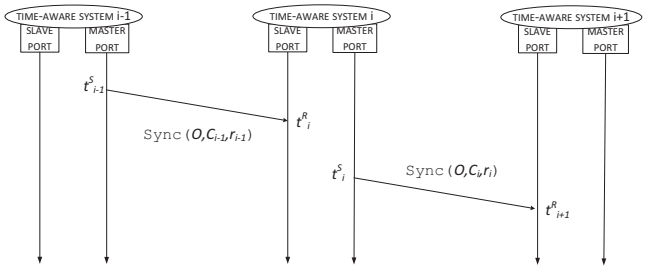

Fig. 1. Transport of time synchronization information between consecutive time-aware systems.

A. Transport of Time Synchronization Information

This mechanism consists on the transmission of a series of values that allow each time-aware system to synchronize to the grandmaster clock. As it has been said, each time-aware system receives time synchronization information in its slave port, and in time, it transmits updated time synchronization through its master port/s. That means that the transport of time synchronization information is made hop by hop (as opposed to end to end).

In Fig. 1 the transport of time synchronization information between consecutive time-aware systems can be seen. Periodically time aware system $i-1$ sends a `Sync` message through its master port/s. Upon the reception of a `Sync` message, time-aware system i records timestamp t_i^R. On a later time, t_i^S, time-aware system i sends a new `Sync` message to the next time-aware system, $i+1$, with updated time synchronization information. The same process is repeated in time-aware system $i+1$ and following time-aware systems successively.

The time synchronization information contained in `Sync` message is the following:

- *Precise Origin Timestamp (O)*: This is the timestamp of the grandmaster clock when the current synchronization information was originated. It is not updated in the time-aware systems.
- *Rate Ratio (r_i)*: This value is the ratio of the frequency of the grandmaster clock to the frequency of the local

clock in the current time-aware system, i. It is calculated in every time-aware system as follows:

$$r_i = r_{i-1} \times nr_i \tag{1}$$

where r_{i-1} is the rate ratio received in the last `Sync` message and nr_i is the neighbor rate ratio, i.e., the ratio of the frequency of the clock of the neighbor time-aware system, $i-1$ to the frequency of the local clock in the current time-aware system, i. IEEE 802.1AS does not describe the algorithm to measure the neighbor rate ratio, however it assumes that the measurement can be done within $\pm$ 0.1 ppm.

- *Correction Field (C_i)*: This value reflects the time that it takes for the synchronization information to arrive from the grandmaster to the i-th time-aware system. It is the sum of the propagation time (in the wire) and the residence time (in the time-aware systems). Every time-aware system updates this value as follows:

$$C_i = C_{i-1} + D_{i-1} + (t_i^S - t_i^R)r_i \tag{2}$$

where C_{i-1} is the correction field received in the last `Sync` message and D_{i-1} is the propagation delay on the wire as measured by the slave port of the current time-aware system, i. The mechanism to measure the propagation delay is described in next section.

The standard defines two variations of the time synchronization information transport protocol: one-step mode (the one depicted in Fig. 1 and described above) and two-step mode. The two-step mode uses a second message called `Follow-up` that is sent after the `Sync` message. The idea is to use the `Sync` message just to generate the timestamps t_{i-1}^S and t_i^R and the `Follow-up` message to transport the time synchronization information: O, C_{i-1} and r_{i-1}. In this paper we assume one-step mode of operation for all the time-aware systems.

With the time synchronization information contained in the `Sync` message it is possible to correct the local clock of the time-aware system so that it is synchronized with the grandmaster clock. Concretely if we define $GM(t)$ as a function that gives the time at the grandmaster for the local time t, then at t_i^R in the time-aware system, the time in the grandmaster is

$$GM(t_i^R) = O + C_{i-1} + D_{i-1} \tag{3}$$

B. Propagation Delay Measurements

The mechanism defined in IEEE 802.1AS to measure propagation delay on the link consists of a series of messages being exchanged and timestamped between the current time-aware system, i and its neighbor time-aware system, $i-1$, from which it receives the synchronization information.

The message exchange (depicted in Fig. 2) goes as follows:

1) At $t = t_1$ time-aware system i sends a delay request message, `Pdelay_Req`, to the time-aware system $i-1$. t_1 is also recorded in the time-aware system i.

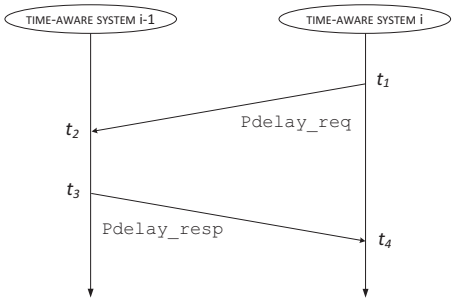

Fig. 2. Propagation Delay Measurements.

2) At $t = t_2$ the time-aware system $i-1$ receives the `Pdelay_Req` message.
3) At $t = t_3$ the time-aware system $i-1$ sends out the delay response messag, `Pdelay_Resp`, containing both t_2 and t_3.
4) At $t = t_4$ the time-aware system i receives the `Pdelay_Resp` message and t_4 is recorded.

At the end of the exchange the time-aware system i has the four timestamps from which the propagation delay, D_i can be calculated (4). The interval $(t_3 - t_2)$ is multiplied by the neighbor rate ratio, nr_i, to convert it to the time scale of the time-aware system i.

$$D_i = \frac{1}{2}((t_4 - t_1) - nr_i \times (t_3 - t_2)) \tag{4}$$

As it can be observed this calculation is based on the idea of having the same propagation delay in both directions, i.e., the paths are symmetric. This, however, does not always hold true as shown in [12]. IEEE 802.1AS provides mechanisms to compensate for an a priori known asymmetry but no specific measures are defined to estimate an existing asymmetry. For this paper the assumption is that, either there is no asymmetry in the paths, or some mechanism is in place to compensate it.

C. Forwarding Behavior

While describing the transport of time synchronization information (Section II-A), it has been said that each time-aware system sends periodically a `Sync` message through its master port (after having received another `Sync` message in its slave port). In this section the exact behavior of these two events is explained.

The flow of the synchronization information starts in the grandmaster, which periodically sends `Sync` messages. This period is referred in the standard as the synchronization interval, `syncInterval`. Upon the reception of a `Sync` message, IEEE 802.1AS describes two possible modes for the time-aware systems that are not the grandmaster (see Fig. 3). The use of one mode or the other is signalized with the flag `syncLocked`. If `syncLocked` is true, then the time-aware system will send a `Sync` message through its master port just after the reception of a `Sync` message in its slave port. Otherwise, if `syncLocked` is false, these two events are

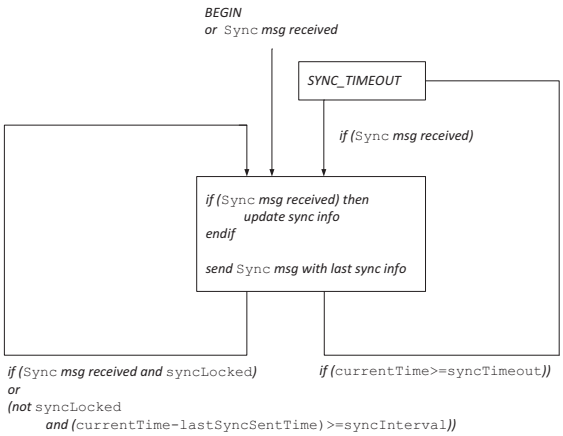

Fig. 3. Simplified state machine that describes the forwarding behavior of time synchronization information in time-aware systems.

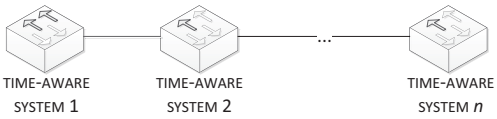

Fig. 4. Network Topology.

decoupled and the time-aware system sends Sync messages periodically regardless of if new synchronization information has been received or not.

Choosing one mode or the other have implications on the performance of the clock synchronization and in the network overall. By setting the flag syncLocked to true the synchronization precision is optimized at the expense of a possible increase in the network workload. On the other hand if the flag is set to false, the behavior of the time-aware system is more deterministic, i.e., it always send Sync messages with the same synchronization interval, regardless of the synchronization interval of the grandmaster. The drawback of this mode is that the synchronization precision gets potentially worse.

III. System Model

As it has been explained in the introduction, the purpose of this work is testing the quality of IEEE 802.1AS for large-scale, realistic networks. In this section the assumptions and behaviors used in our model are explained in detail.

A. Network Topology

In this paper we consider large scale networks of any given topology up to a few thousands elements. In such networks the grandmaster clock is chosen using BMCA and our assumption is that, regardless of the specific topology, the maximum distance between the grandmaster and any time-aware system is of 100 hops. In other words, we assume a subnetwork of n nodes with a linear topology (as in Fig. 4) representing the path from the grandmaster to the farthest time-aware system where n could be 100 in the worst-case.

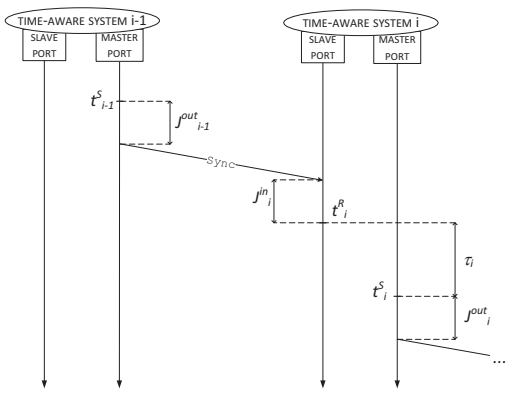

Fig. 5. Time-aware system model, including PHY jitter, J and residence time, τ.

The time-aware systems are connected to each other through IEEE 802.3 full-duplex point-to-point links with a data rate of 100 Mbits/s.

B. Model of Time-aware Systems

Messages sent from and received in a time-aware system suffer a delay while traversing the physical layer. This effect is known as the PHY jitter, J. In Fig. 5 we can see how the PHY jitter affects the overall transmission of a message. Although in that figure we refer to J^{in} and J^{out} to the jitter suffered by a message while being received in and sent from a time-aware system respectively, we consider that both jitters have the same characteristics and therefore $J^{in} = J^{out} = J$. The values and behavior of the PHY jitter assumed in this paper have been taken from the literature. Concretely, in [13] [14] Loschmidt studied clock synchronization enhancements with hardware support and there the measured PHY jitter varies uniformly from 0 to 8 ns.

Additionally to the PHY jitter there is another delay that should be taken into consideration: the residence time (τ in Fig. 5). The residence time is the time that the time-aware system needs to process the Sync message received, update the time synchronization information and forward a new Sync message to the next time-aware system, that is:

$$\tau_i = t_i^S - t_i^R \tag{5}$$

For this paper we assume a maximum residence time of 1 ms. This is a common value in the literature for this parameter as it can seen for example in [11].

C. Clock Model

1) Clock Drift: The fact that two clocks that have been synchronized in a moment in the past will, in a later time, have drifted apart is what generates the need for clock synchronization protocols. If we define t_p as the perfect time-scale of an ideal non-existing clock, then we can express the time in a real clock i, t_i as follows:

$$t_i = t_p + \rho_i t_p = (1 + \rho_i) t_p \tag{6}$$

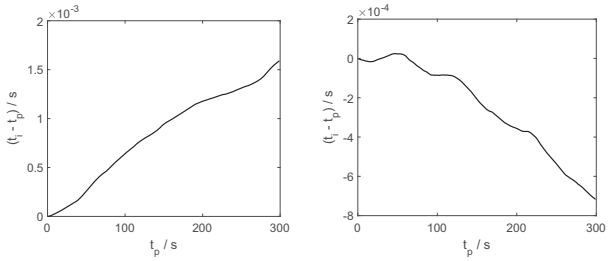

Fig. 6. Two examples of how the time given by a clock t_i that behaves as specified by our model, drifts away from the time given by an ideal clock t_p.

where ρ_i is the drift rate of clock i. Note that ρ_i can have both positive and negative values, i.e., the clock can be a fast clock or a slow clock. The clock drift rate is caused by the non-ideality of the physical oscillators and environmental conditions, for example, heat affecting the frequency of the oscillators. This means that finding a model that describes the exact behavior of a real clock is not a trivial problem.

$$\rho(t) = \rho_0 + \rho'(t) \tag{7}$$

In this paper the model used for the clock drift rate is expressed in (7), where $\rho(t)$ is the clock drift rate, ρ_0 is the initial value of the clock drift rate and $\rho'(t)$ is the change of the drift rate over time, t. We assume that the initial value of the drift rate is in the range, $\rho_0 \in$ [-10,10] ppm for all the clocks and the changes in the clock drift rate, $\rho'(t)$, satisfy these two conditions:

$$\begin{gathered} \rho'(0) = 0 \text{ and} \\ \frac{\Delta\rho'}{\Delta t} \in [0, 1] \text{ ppm/s} \end{gathered} \tag{8}$$

Fig. 6 shows two examples of the behavior of clocks with a drift rate as specified in this system model. The data used to generate those figures has been extracted from the OMNeT++ simulations, results of which are explained in detail in Section V. The initial value of the drift rate, ρ_0 is chosen randomly in the range [10, -10] ppm during the initialization of the simulations.

2) Clock Granularity: To complete the clock model another characteristic has to be taken into account: the granularity. The granularity of a clock is the duration between two consecutive microticks of a digital physical clock [15]. For this paper we consider a clock granularity of 8 ns.

D. IEEE 802.1AS Configuration Parameters

As it can be seen from the overview of IEEE 802.1AS presented in Section II, IEEE 802.1AS has a number of configuration options. The intention in this paper is to configure the network to obtain the best synchronization precision possible, so that the influence of various factors in our model are clearly and realistically represented and taken into account. Thus, the options chosen are the following:

- All time-aware systems transport time synchronization information in one-step mode.
- All time-aware systems operate in `syncLocked` mode.
- The synchronization interval for the grandmaster is set to 31.25 ms. This value is four times lower than the default value of 125 ms defined in the standard.
- The propagation delay measurements interval is set to 1 s. This is the default value used in literature [11].

IV. Expected Theoretical Outcome

The clock synchronization algorithm of IEEE 802.1AS is based on the assumption that the value returned by function $GM(t)$ in equation (3) is really the time in the grandmaster t_{GM} so that:

$$GM(t_i^R) - t_{GM} = 0 \tag{9}$$

In that case we can calculate the deviation between the grandmaster clock and the time-aware system i clock using their clock drift rates and applying (6):

$$t_i - t_{GM} = (\rho_i - \rho_{GM})t_p \tag{10}$$

Now if we assume that in every synchronization interval of length I the clock in the time-aware system i is adjusted to the clock in the grandmaster using (3), then the maximum deviation occur just the moment before the arrival of the new `Sync` message and it would be:

$$p_i = \max(t_i - t_{GM}) = (\rho_i - \rho_{GM})I \tag{11}$$

Equation (11) gives the precision, p_i, of the clock synchronization protocol in time-aware system i. Using $\rho_i = -\rho_{GM} = 10$ ppm as the worst case value for the drift rate and 31.25 ms for the synchronization interval (as assumed in our system model in Section III) give us an estimation of the synchronization precision of 0.625 μs.

The synchronization precision as calculated so far does not depend on the number of hops between the grandmaster and the time-aware system i and that, of course, is not what happens in reality. Next we analyze the factors that affect the synchronization precision in each hop and weight their influence in the precision of the last time-aware system in our topology.

What does not hold in reality from our previous argumentation is equation (9). That means that with the time synchronization information that the time-aware system receives it cannot recreate accurately the time in the grandmaster. In (3) we can see how this value is calculated. Assuming that the precise origin timestamp O is not mistakenly altered, then the factors that cause the error are the correction field C and the propagation delay D, so we can express the error in the GM function, δGM as:

$$\delta GM(t_i^R) = \delta C_{i-1} + \delta D_{i-1} \tag{12}$$

where δC_{i-1} and δD_{i-1} are the errors in the correction field and the propagation delay respectively. Let us now estimate their values.

To calculate further the influence of the error of magnitude on a value that is calculated with that magnitude we use the following general principle of error propagation: if $f(x, y, z)$ is function that depends on the variables x, y and z then the error in f, δf is:

$$\delta f = \frac{\partial f(x,y,z)}{\partial x \partial y \partial z} = \left|\frac{\partial f}{\partial x}\right|\delta x + \left|\frac{\partial f}{\partial y}\right|\delta y + \left|\frac{\partial f}{\partial z}\right|\delta z \quad (13)$$

For the correction field C_{i-1}, that is calculated using (2), all the factors are subject to error, so using (13) we obtain:

$$\delta C_{i-1} = \delta C_{i-2} + \delta D_{i-2} + (t^S_{i-1} - t^R_{i-1})\delta r_{i-1} + r_{i-1} 2\delta t_{i-1} \quad (14)$$

where δr_{i-1} is the error in the rate ratio and $\delta t^S_{i-1} = \delta t^R_{i-1} = \delta t_{i-1}$ are the errors in the timestamps. Analogously we see in (1) the dependencies of r_{i-1} and therefore its error is:

$$\delta r_{i-1} = r_{i-2}\delta nr_{i-1} + nr_{i-1}\delta r_{i-2} \quad (15)$$

being δnr_{i-1} the error in the neighbor rate ratio. And finally, from (4), the error in the propagation delay:

$$\delta D_i = (1 + nr_i)\delta t + \frac{1}{2}(t_3 - t_2)\delta nr_i \quad (16)$$

We see both in the expressions for the error of the correction field (14) and the rate ratio (15), how there is an iterative relationship with the previous values. In order to do an estimation, we assume that some of the worst case values for the following magnitudes are the same in each time-aware system. For example, we consider that the residence time is always the same ($(t^S_1 - t^R_1) = (t^S_i - t^R_i) = ... = (t^S - t^R)$), and so are the propagation delay ($D_1 = D_i = ... = D$) and the neighbor rate ratio ($nr_1 = nr_i = ... = nr$). The same assumption goes for their errors: $(\delta t_1 = \delta t_i = ... = \delta t), (\delta D_1 = \delta D_i = ... = \delta D)$ and $(\delta nr_1 = \delta nr_i = ... = \delta nr)$. Then using these simplifications in (15) we have:

$$\begin{aligned}\delta r_{i-1} =& (r_{i-2} + nr \times r_{i-3} + nr^2 \times r_{i-4} + ...)\delta nr \\ =& \delta nr \sum_{j=2}^{i-1}(r_{i-j} \times nr^{j-2}) = \delta nr \sum_{j=2}^{i-1}(nr^{i-j} \times nr^{j-2}) \\ =& \delta nr \sum_{j=2}^{i-1}(nr^{i-2}) = nr\delta nr \times (i-2)\end{aligned} \quad (17)$$

And for the correction field (14):

$$\delta C_{i-1} = (\delta D + (t^S - t^R)\delta r_{i-1} + 2r_{i-1}\delta t) \times (i-1) \quad (18)$$

Equations (16), (17) and (18) are the expressions for the errors that influence the clock correction (3). With the values defined in the system model we can estimate the new synchronization precision. For the error in the timestamps we consider $\delta t = 16$ ns, that is the sum of the PHY jitter and the clock granularity. For the error in the neighbor rate ratio we use the worst value tolerated by the IEEE 802.1AS $\delta nr = 0.1$ ppm. Besides the errors, we also need to assign values for the magnitudes. For the residence time, that is both $(t^S - t^R)$ in Fig. 1 and $(t_3 - t_2)$ in Fig. 2 we use the maximum value as defined in the system model: 1 ms. To calculate the worst case for the neighbor rate ratio we consider that the two neighbor clocks suffer the maximum drift (as defined in our system model) but in different directions, i.e., they are the slowest clock and fastest clock possible:

$$\max(nr) = \frac{1 + \max(\rho)}{1 - \max(\rho)} = \frac{1 + 10\text{ppm}}{1 - 10\text{ppm}} = 1.00002 \quad (19)$$

This is sufficient to estimate the error in the propagation delay as expressed in (16):

$$\begin{aligned}\delta D =& (1 + 1.00002) \times 16\text{ns} + \frac{1}{2} \times 1\text{ms} \times 0.1\text{ppm} = \\ =& 32\text{ns}\end{aligned} \quad (20)$$

And from (17) the error in r_{i-1} for $i = 100$:

$$\delta r_{99} = 1.00002 \times 0.1\text{ppm} \times 98 = 9.8\text{ppm} \quad (21)$$

For the estimation of the error in the correction field, we need a worst case value for the rate ratio r_{i-1}. The assumption here is that in each hop there is the worst case neighbor rate ratio:

$$\max(r_{i-1}) = \max(nr) \times (i-2) \quad (22)$$

Thus for $i = 100$, $\max(r_{99}) = 1.0002^{98} = 1.002$. And substituting in (18) finally the error in the correction field:

$$\begin{aligned}\delta C_{i-1} =& (32\text{ns} + 1\text{ms} \times 9.8\text{ppm} + 2 \times 1.002 \times 16\text{ns}) \times 99 \\ =& 6.3\mu\text{s}\end{aligned} \quad (23)$$

This value is two orders of magnitude greater than the error in the propagation delay, so we can say that $\delta GM = \delta C$, and therefore the synchronization precision that can be achieved in a more realistic system is:

$$p_i = \max(t_i - t_{GM}) = (\rho_i - \rho_{GM}) + \delta GM = 6.925\mu\text{s} \quad (24)$$

With this calculations we have not just obtained the worst case value for the synchronization precision, which is three orders of magnitude greater than what we would obtain in a system without PHY jitter and clock granularity, we have also shown how the different factors affect the final error. From equations (16), (17) and (18) it gets clear that there are two main sources of error: the error in the timestamping, δt (caused mainly by the PHY jitter and the clock granularity) and the error in the neighbor rate ratio, δnr (that

has been considered monolithic in this paper). However what is probably more interesting is how the error depends on the number of hops between the grandmaster and the time-aware system i. Concretely in (23) we see how the factor $(i-1)$ is what makes the biggest difference as, for example, for $i = 10$ the synchronization precision would be only of 1,2 μs. From all this we can conclude that the influence of the realistic factors that have been considered in this paper, i.e., the PHY jitter and clock granularity, accumulate when transporting time synchronization through long paths, making them crucial factors in time synchronization quality for large-scale networks.

V. Simulation Results

In the previous section we show an estimation of the synchronization precision that can be obtained in a time-aware system that is 100 hops away from the grandmaster. However that is a worst-case estimation. In a real-life scenario the randomness of the errors that affect the synchronization will cancel out resulting in a better synchronization precision. To show this effect we have modeled IEEE 802.1AS in OMNeT++ [3] and we have performed simulations with 100-node long chained networks. We use INET [16] for the Ethernet models and Levesque's [17] implementation of real hardware clocks. OMNeT++ has a int64-based, fixed-point `SimTime` class to represent the time. The accuracy is determined by a scale exponent and falls in the range -18..0.

First we run simulations to test the accuracy of the propagation delay measurements, then we take those results and use them in 100-node long networks simulations.

A. Neighbor Rate Ratio

As it has been explained IEEE 802.1AS does not prescribe a method to measure the neighbor rate ratio. In this paper we are not using any specific method either. Instead we obtain the perfect value and then we introduce the maximum error allowed by IEEE 802.1AS, i.e., 0.1 ppm. What follows is an explanation of how the neighbor rate ratio is calculated inside the simulations.

Assuming that in a given moment we know the exact value of the drift rate of the clock in time-aware system i, ρ_i, then from the model of the drifting clock defined in Section III and using equation (6), we express the relationship between the times given by the clock of two adjacent time-aware systems, i and $i-1$ as follows:

$$t_{i-1} = \frac{(1+\rho_{i-1})}{(1+\rho_i)} t_i \tag{25}$$

And from that is clear that the neighbor rate ratio nr_i, is:

$$nr_i = \frac{(1+\rho_{i-1})}{(1+\rho_i)} \tag{26}$$

The values of the clocks drift rates are known locally in every time-aware system in the simulation, therefore we include this information as part of the time synchronization information that is transported in the `Sync` messages. Thus, the neighbor rate ratio is calculated using (26), with the received neighbor clock drift rate, ρ_{i-1}, and the local clock drift rate ρ_i.

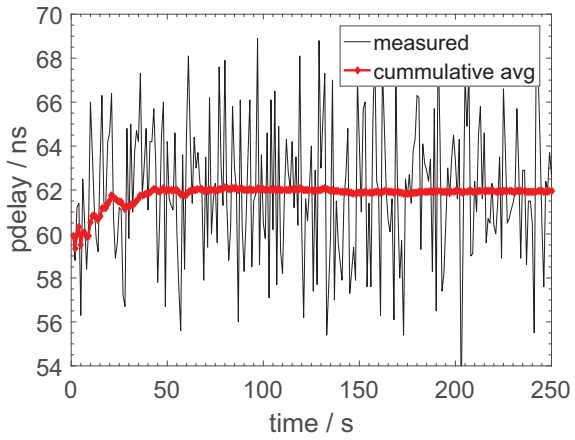

Fig. 7. Propagation delay measurements. The black line is the evolution of the measured propagation delay. The red line is the cumulative average of the propagation delay. Propagation delay interval = 1 s.

B. Propagation Delay Measurements

To test the performance of the propagation delay measurements we set up a simulation with just two time-aware systems separated 10 m (which is equivalent to a 50 ns propagation delay in copper wire). Those time-aware systems have all the characteristics described in the system model, i.e., PHY jitter, residence time, clock granularity and clock drift. The initial value of the drift as defined in equation (7) is chosen randomly at the beginning of each simulation. To cover as many cases as possible, 1000 simulations were performed, each of them for 1000 seconds. The propagation delay measurement interval is set to 1 second.

Fig. 7 shows the results of the propagation delay measurements for a single simulation. The values obtained range from 52 ns to 72 ns and the average is 62 ns. Given that the propagation delay was set to 50 ns it might look like the protocol has failed to measure the right value, however taking into consideration the assumptions made in the system model those results match the expected outcome.

To understand this we must take into consideration all the events that happen between the moment a message is sent from a time-aware system and the moment it is received in the next one. Fig. 5 shows that after leaving the time-aware system $i-1$ at time t_{i-1}^S, the message goes through the physical layer, which introduces a delay that we call the PHY jitter, J. Then, after the propagation delay (time in the wire), the message goes through the physical layer of the time-aware system i. Finally the timestamp t_i^R is also subject to the clock granularity, g (t_i^S is not affected by clock granularity as the event of sending a message from the time-aware system is triggered by the clock in that system). This process is exactly the same as what happens in the propagation delay

measurements (see Fig. 2). That means that the protocol does not obtain the real propagation delay, D_0, but instead:

$$D_i = D_0 + 2J + g \tag{27}$$

The value for the PHY jitter range from 0 to 8 ns and the clock granularity is also 8 ns. Thus in the worst case the deviation from the real propagation delay can be up to 24 ns. However, the variations on the PHY jitter are random and so is the error derived from the clock granularity, which means that on average the mean values are more likely to happen. This accounts exactly for the deviation of 12 ns that we observe for D_0 making a total of 50 ns + 12 ns = 62 ns.

In Fig. 7 we can see how the individual measurements can have variations of ± 10 ns. However if instead of using the last measurement of the propagation delay, the time-aware system uses also previous measurements and averages them, then the value is more stable. Concretely the red line in Fig. 7 shows the values obtained for the propagation delay using cumulative average of the previous values, where the variations are up to a maximum of ± 3 ns.

We have seen how the protocol defined in IEEE 802.1AS does not provide an accurate value for the propagation delay. Instead it gives the sum of the propagation delay and the effect of PHY jitter and clock granularity. Although this might seem like a problem, in reality it helps the synchronization protocol as it accounts for those effects. This result is used in the next section to test the synchronization accuracy. For that we consider that the propagation delay suffer variations of 3 ns.

C. Time Synchronization Precision

Now we are ready to test the time synchronization precision of a time-aware system that is 100 hops away from the grandmaster. The simulations are made with a network topology like the one depicted in Fig. 4. Each time-aware system has the same characteristics of PHY jitter, residence time, clock granularity and drift as defined in the system model in Section III. For the propagation delay we use the value obtained in the previous section, 62 ns. The synchronization interval is set to 31.25 ms and each simulation run for 100 s. A total number of 100 different simulations were made.

Before going to the results, it is important to explain how we evaluate the synchronization precision. In Section III to calculate the synchronization precision we use the maximum deviation between the grandmaster clock and the time-aware clock. However as discussed before, those clocks are changing dynamically due to the clock drift and in the case of the time-aware system, the clock corrections. Ideally we could obtain the synchronization precision by monitoring continuously the clocks, calculate their difference and simply find the maximum. Of course that is not possible in the reality or in the simulations. What we do instead is selecting significant time instants to check the clocks.

To select those time instants we use the approach of the ingress method defined in IEEE 1588 for measuring the time synchronization error. The ingress method relies in the slaves

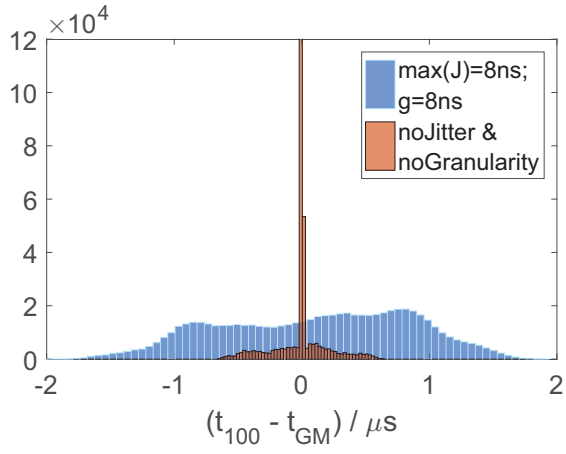

Fig. 8. Distribution of the time deviation measurements of the clock on time-aware system 100 with respect to the grandmaster clock. Red bars: without jitter and clock granularity. Blue bars: max PHY jitter = 8 ns, granularity = 8 ns.

to provide their offset relative to the grandmaster just before the clock correction defined in (3). Analogously we take measurements just before and just after the clock correction in every time-aware system. A known shortcoming of this method is that it is possible to miss the maximum time error between two incoming `Sync` messages. However with the variations of the drift rate as defined in the clock model (8) and the value chosen for the synchronization interval, that possibility is almost non existing.

Fig. 8 shows the distribution of the time deviations between the clock of the time-aware system 100 and the grandmaster clock. In red we see the results for a the perfect case, i.e., without PHY jitter or clock granularity. The distribution presents a prominent peak at $t_{100} - t_{GM} = 0$ and the maximum deviation is around 0.6 μs. This fits with the worst case analysis made in Section IV in which we obtain a value of 0.625 μs. In the same figure, in blue we see the same distribution but now for the case that we want to analyze, i.e., with PHY jitter and clock granularity. The influence of these two factors gets clear as we observe how the distribution has change drastically: now it is almost flat and it ranges from -2 μs to 2 μs. The increase in the time deviation is an expected result after the analysis made in Section IV. However it does not get to the calculated worst case of 6 μs. The reason for that is not only that the worst case, by definition, is not likely to happen, but also that in many cases the error introduced by the jitters and the granularity compensate each other, keeping the time deviation lower.

1) Probability of Synchronization: By looking at Fig. 8 we can say that a synchronization precision of 2 μs can be achieved in a time-aware system that is 100 hops away from the grandmaster. However it is interesting to extend our study a little bit further and see which precision can be achieved in other time-aware systems that are closer to the grandmaster. To do that we define a probability, within the set

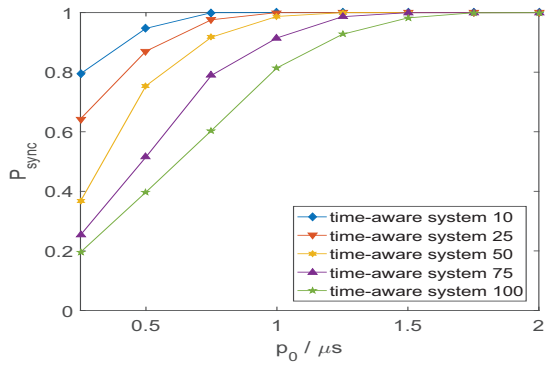

Fig. 9. Probabilty of synchronization within different precision thresholds.

of timestamps gathered by the simulations, for a time-aware system of being in synchronization with the grandmaster with a certain precision. To do that we classify the timestamps as follows:

$$\begin{aligned} |t_i - t_{GM}| < p_0 &\longrightarrow \text{in sync} \\ |t_i - t_{GM}| > p_0 &\longrightarrow \text{out of sync} \end{aligned} \tag{28}$$

Thus the probability of synchronization within a precision p_0 is:

$$P_{sync} = \frac{measurementsInSync}{totalMesurements} \tag{29}$$

In Fig. 9 we can see the probability of synchronization for different time-aware systems in our topology. As it is expected, the more hops apart from the grandmaster the time-aware system is, the worst the synchronization accuracy gets. For time-aware system 100 (that is 100 hops away from the grandmaster), the probability of synchronization gets to 100% for a precision of 2 μs. This is in line with what we observe in Fig. 8. It is also important to notice how for the time-aware system 10, the probability of synchronization within a precision of 1 μs is 100%, which confirms that even with the inclusion of PHY jitter and clock granularity, the IEEE 802.1AS can still guarantee its original requirement of 1μs for up to 6 hops distance.

Fig. 10 gives more information on the probability of synchronization within 1 μs for all time-aware systems. We see that the 1 μs precision is achievable until up to 30 hops with the assumptions made in our system model.

2) Simulations Coverage: As it was explained before, to test the quality of the synchronization that can be obtained with IEEE 802.1AS in our system model, 100 different simulations have been run, each of them of 100 seconds duration. In this section we evaluate the coverage of the simulations made.

Given that the clock drift rates vary from -10ppm to 10 ppm and that there is a total of 100 time-aware systems a rough estimation give us a total of 10^{130} number of cases, which of course make covering all the cases impossible. Because the drift of the clocks change with time, we assume that every

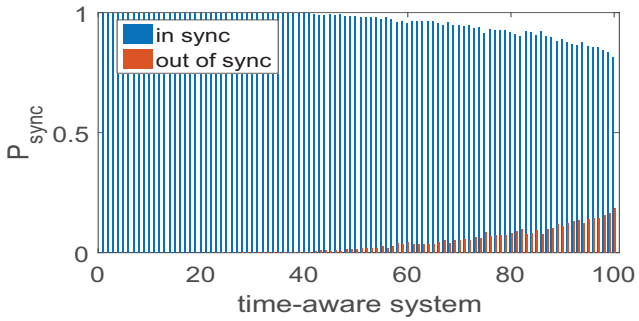

Fig. 10. Probability of synchronization within 1 μs for all time-aware systems on the network.

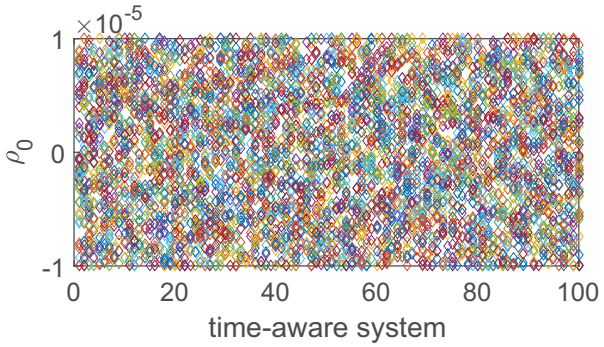

Fig. 11. Initial values of the clock drift rate, ρ_0 of all the time-aware systems in every simulation.

synchronization interval represents a different case. Thus, in a simulation of 100 seconds there are 3200 cases, and in 100 simulations, there are a total of 320.000 cases. This number is way smaller than the total number, however in Fig. 11 can be seen how, at least, those cases are spread homogeneously. That guarantees that our simulations are representative of the general behavior.

VI. Related Work

Being a rather new protocol, there is not so much work done on IEEE 802.1AS. Garner, the editor of the project IEEE 802.1AS in the TSN task group, has several publications considering different aspects of the protocol [11] [18]. He has tested the protocol performance both with simulations and in a real-world set up and many of the assumptions made in this paper are based on his work. However he has always constrained to small networks of up to seven hops and never considered explicitly the influence of PHY jitter.

In [19] IEEE 802.1AS performance is tested in the context of in-car networks. They present results based on simulations made also with OMNeT++ and studied the influence of the network workload. Yet again they do not consider PHY jitter and because of the use-case it is also a small network (7 hops). If we focus on the time-aware systems closer to the grandmaster (time-aware system 10 in Fig. 9) their results are similar to the ones presented in this paper, as they obtain also synchronization precision of 0.7 μs.

If we widen the search, we can find also interesting works on IEEE 1588 that, given its similarities with IEEE 802.1AS are relevant to this paper. Concretely in [20] Wallner presents a simulation framework for IEEE 1588. This framework is built also in OMNeT++ and aims to ease the exploration of the configuration space of PTP. Among the configuration parameters that can be changed one of the most relevant is the synchronization interval. Thus they explore the synchronization precision that can be achieved with different synchronization intervals in a 50-node long network. For a synchronization interval of 31.25 ms (the same as the one used in our simulations), they obtained a synchronization precision of around 0.7 μs. This precision is much better that the one we obtain, 1.25 μs (from Fig. 9). The reason for that difference is no other that the absence of PHY jitter in their model.

Finally another performance test of IEEE 1588, but this time in a real-world situation can be found in [21]. In this work they obtained synchronization precision of 10 μs for a 5 hop network. This result is worse than our worst case analysis for 100 hops, however there are differences between the two situations that explain it. First they are using a synchronization interval of 2 seconds, 64 times longer that our synchronization interval. And second and probably most important they are using ordinary (non time-aware) systems, which means that the clock synchronization is made end to end (as opposed to hop by hop).

VII. Conclusions and Future Work

In this paper we show the synchronization quality that can be achieved with IEEE 802.1 in large-scale realistic networks. To do so we incorporate a series of assumptions (such as PHY jitter, clock granularity or residence time) in our system model with the intention to be as accurate as possible. These level of detail, together with the large-scale topology chosen for the network, make this work an important result for the adoption of TSN standards in industrial communications.

The results obtained highlight the importance of taking into consideration the implementation details (such as the PHY jitter and clock granularity) when calculating the synchronization precision of our network. We show how the effect of these two factors accumulate as the synchronization information moves further away from the grandmaster, making the synchronization precision to increase from 0.6 μs to around 2 μs for the last time-aware system in a chain of 100 hops. This value is the double of the value typically required in industrial networks 1 μs. However the simulations show that 1 μs precision is still achievable for time-aware systems that are up to 30 hops away from the grandmaster.

As for the future work it would be reassuring to validate the results obtained with a real life implementation. However it is important to notice that simulations are a powerful tool that allow us to easily explore the vast design space of the synchronization protocol, test corner cases and fiddle with hardware parameters like the PHY jitter and clock granularity. Things that would be very challenging to experiment with in a real testbed. In fact, the simulation framework developed in OMNeT++ will allow us to study further the influence of implementation details on the time synchronization quality by adding even more details to the current model.

Acknowledgment

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement 607727.

References

[1] Wilfried Steiner, Pablo Gutiérrez Peón, Marina Gutiérrez, Ayhan Mehmed, Guillermo Rodríguez Navas, Elena Lisova, and Francisco Pozo. Next Generation Real-Time Networks Based on IT Technologies. In *21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016.

[2] IEEE Standard for Local and metropolitan area networks– Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks– Corrigendum 2: Technical and Editorial Corrections. *IEEE Std 802.1AS-2011/Cor 2-2015 (Corrigendum to IEEE Std 802.1AS-2011)*, pages 1–13, April 2016.

[3] OMNeT++ Discrete Event Simulator. http://www.omnetpp.org/, 14 October 2016.

[4] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[5] Jennifer Lundelius Welch and Nancy Lynch. A new fault-tolerant algorithm for clock synchronization. *Information and computation*, 77(1):1–36, 1988.

[6] Leslie Lamport and P Melliar Smith. Byzantine clock synchronization. *ACM SIGOPS Operating Systems Review*, 20(3):10–16, 1986.

[7] Hermann Kopetz and Wilhelm Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, 100(8):933–940, 1987.

[8] Flaviu Cristian. Probabilistic clock synchronization. *Distributed computing*, 3(3):146–158, 1989.

[9] Flaviu Cristian and Christof Fetzer. Probabilistic internal clock synchronization. In *Reliable Distributed Systems, 1994. Proceedings., 13th Symposium on*, pages 22–31. IEEE, 1994.

[10] Pedro Moreira, Javier Serrano, Tomasz Wlostowski, Patrick Loschmidt, and Georg Gaderer. White rabbit: Sub-nanosecond timing distribution over ethernet. In *International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2009.

[11] Geoffrey M. Garner and Hyunsurk Ryu. Synchronization of audio/video bridging networks using IEEE 802.1AS. *IEEE Communications Magazine*, 49(2):140–147, February 2011.

[12] R. Exel, T. Bigler, and T. Sauter. Asymmetry Mitigation in IEEE 802.3 Ethernet for High-Accuracy Clock Synchronization. *IEEE Transactions on Instrumentation and Measurement*, 63(3):729–736, March 2014.

[13] Patrick Loschmidt. *On Enhanced Clock Synchronization Performance Through Dedicated Ethernet Hardware Support*. PhD thesis, Vienna University of Technology, Vienna, Austria, December 2010.

[14] Patrick Loschmidt, Reinhard Exel, and Georg Gaderer. Highly Accurate Timestamping for Ethernet-Based Clock Synchronization. *Journal of Computer Networks and Communications*, 2011.

[15] Roman Obermaisser. *Time-Triggered Communication*. Embedded Systems. Taylor & Francis, 2011.

[16] INET Framework. https://inet.omnetpp.org/, 14 October 2016.

[17] Martin Levesque and David Tipper. ptp++: A Precision Time Protocol Simulation Model for OMNeT++ / INET. *CoRR*, abs/1509.03169, 2015.

[18] Geoffrey M. Garner, Aaron Gelter, and Michael Johas 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*, pages 1–7, Oct 2009.

[19] Hyung Taek Lim, Daniel Herrscher, Lars Völker, and Martin Johannes Waltl. IEEE 802.1AS time synchronization in a switched Ethernet based in-car network. In *Vehicular Networking Conference (VNC)*, Nov 2011.

[20] Wolfgang Wallner, Armin Wasicek, and Radu Grosu. A simulation framework for IEEE 1588. In *2016 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pages 1–6, Sept 2016.

[21] Endance Technology Limited. White paper: IEEE 1588 PTP Clock Synchronization over a WAN Backbone. Technical report, 2016.