2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)

Simulation Framework for Clock Synchronization in Time Sensitive Networking

Maryam Pahlevan, Balakrishna Balakrishna and Roman Obermaisser
University of Siegen, Germany
Emails: {name.surname@uni-siegen.de}

***Abstract*— Hard real-time systems like industrial control applications have strict temporal requirements. Many hard real-time systems depend on a global time base for coordinating access to shared resources and for time-stamping events. Hence, the Time Sensitive Networking (TSN) task group introduces a fault-tolerant and robust clock synchronization mechanism (i.e. IEEE 802.1AsRev) that results in a synchronized network.**

This paper presents a simulation framework for IEEE 802.1AsRev to evaluate the reliability of the global time base in TSN-based systems. The simulation models are developed on top of our existing TSN models that support the time-based features of TSN (e.g. IEEE 802.1Qbv and IEEE 802.1Qci standards). Moreover, the evaluation of different TSN synchronization modules such as Best Master Clock Algorithm (BMCA), synchronization and peer delay measurement are carried out in our simulation framework. We also study the behavior of IEEE 802.1AsRev in the presence of either a node failure or a link failure using an example scenario of a train communication network. The experimental results validate the correctness and applicability of TSN clock synchronization for modern cyber physical systems with demanding timing requirements.

I. INTRODUCTION

Ethernet technologies are popular in home and office environments due to their high bandwidth and seamless connectivity [1]. However, the standard Ethernet technology by itself only offers limited Quality of Service (QoS) mechanisms and fails to provide deterministic behaviour which is required by real-time applications [2]. Therefore, the Time Sensitive Networking (TSN) group [3] introduces several extensions for standard Ethernet to offer real-time capabilities. In the conventional Ethernet-based networks, the time attribute was only used for performance measurements, not for time synchronization. Therefore, to transport timing information over Ethernet-based networks a wide range of protocols such as Network Time Protocol (NTP) [4], IEEE 1588 Precision Time Protocol (PTP) [5], IEEE 802.1AS [6] and IEEE 802.1AS-Rev [7] were developed. Conversely, the NTP is not suitable for many industrial applications due to its synchronization accuracy in the order of milliseconds.

The IEEE 1588 standard (Precision Time Protocol) was designed to realize high synchronization accuracy over a data network in the field of industrial control and measurement applications. IEEE 1588 permits non-PTP devices to relay the PTP messages within a specific PTP domain. These devices slow down the timing convergence and introduce extra jitter. Thus, PTP is not ideal for mission-critical systems that demand low jitter and guaranteed latency. To resolve this issue, the IEEE 802.1AS standard [6] only allows the time-aware systems to participate in the clock synchronization process. The IEEE 802.1AS standard specifies functionalities that are required by time-sensitive applications across bridged and virtual bridged local area networks to ensure that temporal requirements are met. The IEEE 802.1AS standard is also called generalized Precision Time Protocol (gPTP) since this standard provides similar functionalities as IEEE 1588 with additional modifications, such as using only IEEE 802.1AS-capable systems in the synchronization process and defining new interfaces for obtaining timing information. However, IEEE 802.1AS does not support multiple timescales in a single time-aware network which is essential for industrial networks. Hence, to address this issue, the TSN task group introduce the IEEE 802.1AS-Rev standard.

IEEE 802.1AS-Rev defines procedures which are used for transporting timing information over Local Area Networks (LAN). A time-aware system implementing this standard supports multiple gPTP domains and timescales. Consequently, the number of clock sources that are required in time-aware networks is reduced considerably [7].

Software simulation is a very important step in the verification of any concept or algorithm. Before implementing a method in a real system, simulation is often a prior step, because it is very flexible, avoids physical constraints, saves cost and time. Simulation also plays a key role in the development of new networking protocols, because these protocols are constantly changing during their design phase. Since the IEEE 802.1AS-Rev protocol is not finalized yet, we develop a simulation model for the TSN clock synchronization to verify the correctness and applicability of the IEEE 802.1AS-Rev procedures. This work aims to study the Best Master Clock Algorithm (BMCA), the delay measurement and the synchronization process which are introduced in the IEEE 802.1AS-Rev protocol. To achieve this goal, we extend our TSN simulation models [8] that implemented the time-based features of TSN (i.e. IEEE 802.1Qbv and IEEE 802.1Qci), to support different TSN time synchronization functionalities. A time-aware system requires a local clock with high accuracy in order to guarantee the deterministic delivery of time-triggered traffic. Therefore, we model the local clock of each time-aware system with different drift rates (e.g. linear and non-linear drift rates) and also develop varying time synchronization functions to validate that TSN synchronization

978-1-7281-0151-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ISORC.2019.00046

results in a synchronized network. In addition, we investigate the behavior of the time synchronization modules in the presence of crash failures using the fault injector model developed in [9]. We emulate re-execution of BMCA by failing either the primary grandmaster node or the attached link and evaluate the accuracy of a time-aware system's local clock during the grandmaster failure and the handover period. To the best of our knowledge, our simulation framework is the first work that uses TSN synchronization mechanisms instead of the static reference time to enforce time-based features of TSN including time-aware filtering and shaping. As a result, this work provides an opportunity to evaluate the correctness of real-time capability of TSN solutions in more realistic networking scenarios.

This paper is organized as follows: Section II discusses the related work. In section III, we briefly describe the IEEE 802.1AS-Rev standards along with other time synchronization protocols. The simulation model of gPTP capable devices is explained thoroughly in section IV. Section V is dedicated to the evaluation of simulation results which are carried out in varying use cases. The last section concludes our work.

II. Related Work

In the last years, several works studied different clock synchronization protocols using various simulation frameworks. Hao Guo [10] modeled the PTP protocol in the OPNET simulator. The author in [10] developed only two boundary clocks, namely one for the master device and one for the Slave device and one end-to-end transparent clock. In addition, this framework only evaluates synchronization and delay measurement mechanisms since grandmaster and slave clocks are selected statically. The author also carried out multiple simulation scenarios where a conventional switch is employed between grandmaster and slave clocks instead of an end-to-end transparent clock. This design decision results in a notable growth of the synchronization error as the load of background traffic increases. Thus, the study concludes that a conventional switch is not able to measure accurately the residence time of PTP messages in the presence of background traffic. Furthermore, it recommended to use end-to-end transparent clock if sub-microsecond accuracy is necessary.

Hyung-Taek Lim [11] built a simulation framework for IEEE 802.1AS in OMNeT++. The main goal of the framework was to simulate vehicular networks which support the IEEE 802.1AS standard. To simplify the clock synchronization procedure, the author made some assumptions such as 1) assigning pre-defined clock types instead of implementing BMCA, 2) not using the external timing source and 3) not measuring the phase and frequency discontinuity. The evaluation of the framework is divided into two phases: peer delay measurement and a synchronization process. Once the peer delay messages are exchanged between the two time-aware end stations, the computed propagation delays are converged into the real propagation delay of 40ns within a range of ±10ns. Moreover, the computation of the propagation delay involves two types of filters in order to average the experimental samples. In the synchronization process, all time-aware systems are connected in a daisy-chain based topology with additional traffic generators and receivers (non-time-aware systems). Two synchronization intervals (62.5ms and 125ms) are employed to analyze the synchronization accuracy. Although a maximum synchronization accuracy 1µs is achieved with a periodicity of 125ms in the synchronization process with a 100% network load, a synchronization interval of 62.5ms further improves the accuracy of local clock by approximately 50%.

In this paper, we develop TSN synchronization modules on top of our existing simulation models that support IEEE 802.1Qbv and IEEE 802.1Qci. Unlike the described studies, this work is not only limited to the modeling of delay measurement and synchronization, but it also assigns the role of each device's port dynamically using BMCA. In contrast to existing clock synchronization simulation frameworks, our simulator considers faults and dynamic changes in real-time systems. For this purpose, we simulate different faulty behaviors like crash and link failure within the synchronized network and then assess the impact of each failure on the time synchronization process. To perform an accurate evaluation, this work conducts experiments using a real-life use case based on train communication network.

III. Background

State-of-art protocols for clock synchronization (e.g. NTP) were not suitable for distributed measurement and control applications. Therefore, the IEEE 1588 standard was introduced to realize high time accuracy over a data network that is essential for industrial automation. The first version of IEEE 1588-2002 (PTPv1) was published in 2002. Due to considerable interest from different industries, in the year 2008, the second version of IEEE 1588-2008 (PTPv2) with improved functionalities was published. One of the major advantages of IEEE 1588 for industrial networks is that PTP can be easily deployed over Ethernet, and it does not require any extra time distribution network [5].

IEEE 802.1AS is a clock synchronization standard for time-sensitive applications across bridged local area network which was published by the AVB task group in 2011. The IEEE 802.1AS standard synchronizes all time-aware systems across bridged local area network and thus enables the time-sensitive applications running on the time-aware systems to meet their temporal requirements (e.g. low jitter and bounded end-to-end latency). To achieve synchronized networks, the standard follows similar principles as IEEE 1588 with some modifications. For this reason, IEEE 802.1AS is also known as generalized Precision Time Protocol (gPTP). The major differences in gPTP compared to PTP are 1) reduction in the types of clocks, 2) enabling a single time domain in the steady state, 3) participation of only IEEE 802.1AS-capable systems in the time-aware network, 4) no pre-qualification for either master or foreign-masters, and 5) a new interface definition for obtaining timing information [6].

A. IEEE 802.1AS-Rev Overview

In 2012, the AVB task group was renamed to TSN task group and started introducing new mechanisms for the transmission of time-sensitive traffic over conventional Ethernet-based networks. The IEEE 802.1AS-Rev standard was introduced by the TSN task group and specifies enhanced procedures for the exchange of timing information and clock drift measurements. A significant modification has been identified in the IEEE 802.1AS-Rev draft standard compared to IEEE 802.1AS is that the standard supports multiple gPTP domains and timescales within a single time-aware network. This feature is beneficial for industrial automation since it requires several different timescales for various measurement and control applications.

The time-aware network is a packet switched network where time-aware systems are interconnected by LANs. The time-aware systems are classified into two types: 1) time-aware end station and 2) time-aware relay. The time synchronization process of IEEE 802.1AS-Rev in similar way to IEEE 1588v2 is divided into three subsections: A) grandmaster selection and network establishment, B) delay measurement and C) syntonization. In general, once a grandmaster is selected by BMCA, the grandmaster sends synchronization message by including the current synchronized time to all directly connected time-aware systems. Each of these time-aware systems corrects the received synchronization message and adjust its local clock accordingly. The local clock correction involves adding the received precise grandmaster time in a synchronization message with the propagation time. The propagation time is a path delay between own slave port (received synchronization message time-stamp) and the neighbouring master port (sent synchronization message time-stamp). If the time-aware system is an end station, then the station corrects its local time using the synchronization information received from grandmaster and does not forward the message to other nodes. On the other hand, if the time-aware system is a bridge, then the relay corrects its local time using the synchronization information from the grandmaster and also forwards the message to other time-aware systems after considering the path delay in the synchronization message.

1) Grandmaster Selection and Network Establishment: In a gPTP domain, all devices are time-aware systems and participate in the BMCA to elect the best clock source as a grandmaster within the domain. Therefore, IEEE 802.1AS-Rev determines the synchronization spanning tree as shown in Figure 1. The established spanning tree by IEEE 802.1AS-Rev and the forwarding spanning tree obtained by IEEE 802.1D and IEEE 802.1Q Rapid Spanning Tree Protocol (RSTP) are mainly different. To be more specific, a forwarding spanning tree by RSTP might be even inappropriate for synchronization.

2) Delay Measurement: Time-sensitive networks support four types of link technologies including full-duplex point-to-point links, Ethernet Passive Optical Network (EPON) links, wireless links and generic Coordinated Shared Networks

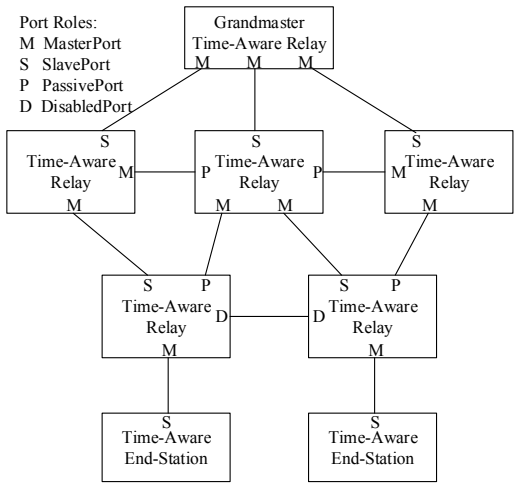

Fig. 1: Master-slave hierarchy after BMCA [7]

(CSN) to interconnect the time-aware systems. All these links have different methods to compute the propagation delay, although they work on the same principle: recording the instant of time that a delay request message is sent from one device to the responder node and then retrieving the time instant when the message is received by the responder device. The same procedure is repeated for a delay response message that is sent by the respondor to the initiator of the delay measurement process as shown in Figure 2.

The delay calculation methods used by different link technologies are explained as follows:

- Full-duplex Ethernet networks use the two-step Peer-to-Peer (P2P) path delay algorithm as defined in the IEEE 1588v2 standard, where *PdelayReq*, *PdelayResp* and *PdelayRespFollowUp* messages are used.
- IEEE 802.11 wireless LANs use the timing measurement procedure defined in IEEE P802.11v, where the messages are either *TimingMeasurementAction* frames or the corresponding acknowledgements.
- EPON link technologies use the discovery process, where the types of messages are either *GATE* or *REGISTERREQ*.
- For CSNs, there are two different ways to measure the path delay: 1) employ the same approach as full-duplex Ethernet 2) implement a procedure which is designed for that specific CSN.

3) Syntonization: In a gPTP domain, the inaccuracy of the local clock of any time-aware system may lead to clock synchronization error. The measurement of the propagation delay and residence time interval play an import role in the clock synchronization process. Therefore, if all the time-aware systems syntonize their clock to grandmaster's time, the time reference for the calculation of the path delay and the residence time will be identical. Adjusting the frequency of an oscillator within a time-ware system according to the Phase Locked Loop (PLL) which was introduced in IEEE 1588, is a slow process and susceptible to gain peaking ef-

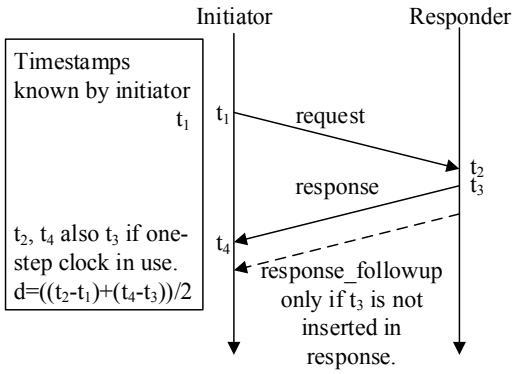

Fig. 2: Simplified delay measurement procedure [7]

fects. Consequently, the gPTP-capable systems diminish the impact of their clock drifts using the grandmaster frequency ratio.

The frequency ratio of adjacent time-aware systems is known as Neighbour Rate Ratio (NRR). The measurement of an NRR takes place at each port of a time-aware system and it is equal to the ratio of the neighbor system's clock frequency to the own clock frequency. On the other hand, Cumulative Scaled Rate Offset (CSRO) is a frequency ratio of the grandmaster clock frequency to the local clock frequency of a specific time-aware system. The CSRO is carried by the *FollowUp* message. The NRR is used for correcting the delay time measurement, while CSRO is used for computing the synchronized time. The CSRO at every time-aware system is calculated by adding the system's NRR as illustrated in Figure 3.

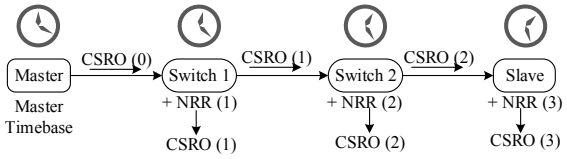

Fig. 3: Relation between CSRO and NRR [7]

There are two main reasons why the CSRO is derived from the accumulation of NRRs: Firstly, NRR is continuously measured using the *Pdelay* messages. For instance in case of a network reconfiguration, a new grandmaster is selected, but the NRR need not be measured again because the ratio was already measured by the *Pdelay* messages. This helps to reduce the duration of any transient state in the reference time during a network reconfiguration process. In addition, the clock's drift rate of a slave port is compensated by NRR even if no synchronization messages are received for a certain time interval. Secondly, there are no gain peaking effects since an error in a CSRO at one time-aware system does not affect the frequency offset at downstream systems. Although the time-aware system with an inaccurate CSRO computes the residential time erroneously [7].

IV. Simulation Models for TSN Clock Synchronization

We implement different TSN synchronization functionalities and integrate them with our existing TSN simulation models [8], [9] which were developed in the Riverbed simulator. Riverbed (also known as OPNET) is a discrete event simulation tool that allows to reuse the existing models, and also share the newly developed models between different platforms [12].

A. Time-aware End Station

All messages belonging to gPTP protocol are encapsulated into Ethernet frames. Therefore, the existing OPNET node model of a conventional Ethernet end-system is chosen for time-aware end system modeling.

B. Time-aware Relay

The time-aware relay has the role of an IEEE 1588 boundary clock with the certain functionalities. Thus, the existing advance Ethernet switch model in OPNET is used to model a time-aware relay.

The time-aware end station and relay models incorporate BMCA, synchronization and peer delay measurement mechanisms. We already modified the pre-defined OPNET models to support IEEE 802.1Qbv, IEEE 802.1Qci and IEEE 802.1CB protocols [8], [9]. Based on these models we integrated the TSN synchronization procedure.

At the initialization phase, the time-aware system generates the announce messages and forwards them to other systems within a gPTP domain. Once the master-slave hierarchy is established, the time-aware device acts either as a grandmaster or a Slave gPTP system. If the device becomes a grandmaster, it periodically generates and sends synchronization messages with its own local time to all slave devices in the gPTP domain. Namely, for a synchronization message, the grandmaster obtains the local time when the synchronization message is scheduled to be sent and inserts the corresponding time-stamp into the message. In addition to synchronization messages, the time-aware system also sends announce messages periodically. It is worth to note that, both synchronization and announce messages are scheduled at different cycles.

Moreover, the grandmaster responds to the *PdelayReq* message that is issued by an adjacent time-aware system, by sending a *PdelayResp* message. For the *PdelayResp* message, the time-aware system encapsulates the sending time of the *PdelayResp* message and also the reception time of the *PdealyReq* message.

C. Local Time Modeling

The main aim of the TSN protocol is that all time-aware systems in a gPTP domain follow a single time-base provided by the grandmaster. However, in the OPNET simulator, every network module operates by referring to the OPNET simulation time (i.e. *op_sim_time*()). Therefore, to simulate time-aware systems with different drift rates, designing a

local clock for each system is essential. The local clock of a time-aware system can be modeled as:

$$\begin{aligned}\text{LocalTime} = \text{SyncTime} + \text{MeanPathDelay} \\ + \text{TimeDrift} + \text{ClockTick}\end{aligned} \tag{1}$$

In Equation 1, SyncTime refers to the OPNET simulation time at the beginning of the simulation run and later it is replaced with the precise Grandmaster time that is carried in a synchronization message. It is noteworthy that the OPNET simulator does not provide any time reference apart from the simulation time. Therefore, the local clock of a time-aware systems is derived from either the simulation time or from a received synchronization message. In this work, the local time of grandmaster is set to the simulation time (i.e. *op_sim_time*()). According to Equation 1, the local time of a time-aware systems is obtained by adding time drift, synchronization time, mean path delay and time duration of a clock tick.

The clock tick is measured according to the Equation 2.

$$\text{ClockTick} = \text{CurrentAbsoluteTime} - \text{PreviousAbsoluteTime} \tag{2}$$

Where the PreviousAbsoluteTime is measured when the local time is synchronized to the grandmaster clock. The mean path delay is computed based on the principle that is shown in Figure 2. To calculate the time drift of a time-aware system, we multiply the constant drift rate of the node with the clock tick. Hence, the local time that its drift is computed based on Equation 3, simulates the linear clock and it is used for time-stamping of gPTP messages. In our simulation framework, the drift rate for each time-aware system is user-configurable.

$$\text{TimeDrift} = \text{DriftRate} * \text{ClockTick} \tag{3}$$

Equation 4 shows the compensated time of a slave time-aware system, where the drift rate of the linear clock is compensated by the NRR and the CSRO. In our models, the grandmaster has zero drift rate and it appends its own time (i.e. *op_sim_time*()) to a synchronization message.

$$\begin{aligned}\text{CompensatedTime} = \text{LocalTime} - \text{CompensatedTimeOffset} \\ \text{CompensatedTimeOffset} = (\text{NRR} + \text{CSRO}) * \text{ClockTick}\end{aligned} \tag{4}$$

As shown in Equation 3, the drift increases linearly over time and results in an inaccurate local time. Consequently, NRR plays an important role in identifying the frequency ratio between neighbour devices. NRR along CSRO is used to compensate the drift as stated in Equation 4.

$$\text{NRR} = \frac{\text{PresOriginTimestamp} - \text{MasterPreviousAbsoluteTime}}{\text{PresReceiptTimestamp} - \text{SlavePreviousAbsoluteTime}} \tag{5}$$

The NRR is calculated using Equation 5. This equation has a time duration of a neighbour time-aware system (i.e. synchronization message sent or forwarded) in the numerator and its own node time duration in denominator. The time duration is calculated by subtracting two timestamps where the first timestamp (i.e. PresOriginTimestamp or PresReceiptTimestamp) has the highest drift from the grandmaster time, although the second timestamp (i.e. previous absolute time of slave or master) has zero drift. If neighbouring time-aware systems are running with same drift rate, then the computed NRR results in unity, meaning that no frequency difference is identified. On the other hand, if the neighbour devices are running with different drift rates, then the respective frequency ratio is identified by NRR. The computation of NRR can also be considered as a linearization since it aims to compensate the drift time of a local clock. It is quite straightforward to identify the frequency ratio between two clocks when the drift rate of a clock is linear. However, in a clock with non-linear drift rate, the approximated frequency ratio can be computed but the synchronization interval needs to be adjusted based on the expected synchronization accuracy.

As mentioned earlier, if two neighbour devices are operating with the same drift rate, then the computed NRR results in unity and it leads to an inaccurately compensated time of a slave device, because there is an actual frequency offset between the local clock of a slave device and the grandmaster clock. Thus, CSRO is utilized to carry the frequency offset between the local node (slave device) and grandmaster device.

The CSRO is an aggregation of NRR. If there are multiple time-aware relays between grandmaster and slave end systems, each time-aware relay will compute the NRR and add it to the CSRO field in a synchronization message before forwarding. Moreover, slave time-aware devices operating in same drift rate will receive the CSRO from a synchronization message. Although the slave devices have unity NRR, the respective frequency ratio is obtained from the CSRO.

With the help of CSRO and NRR, the compensated time can be calculated as shown in Equation 4. The CSRO and NRR compensate the drift of a clock and ensure that the compensated time is nearly equal to grandmaster time. This compensated time is used for scheduling and filtering in our simulation framework.

V. Experiments and Evaluation

In the TSN simulation framework, different time-aware systems including end-stations and relays are interconnected using full-duplex 100 Gbps Ethernet links. We run our simulation on a computer with an Intel i5 CPU and 32GB of memory.

A. Experimental Setup

We evaluate the behavior of different TSN synchronization modules using an example layout of a train network. This train layout same as other Ethernet-based train networks encompasses an Ethernet Consist Network (ECN) and an Ethernet Train Backbone (ETB) [13]. As illustrated in Figure 4, time-aware systems of every ECN are connected to the ETB via the redundant ETB lines. Moreover, in our

experimental setup, every smart sensor within a simulated ECN has a sampling rate of 100 milliseconds. The collected samples are sent to the Central Computing Unit (CCU) of a corresponding ECN. After that, the CCU forwards the sensor samples to the monitoring application and the HMI at a rate of 200 milliseconds. At the end, the HMI sends the processed data back to the CCUs every 200 milliseconds.

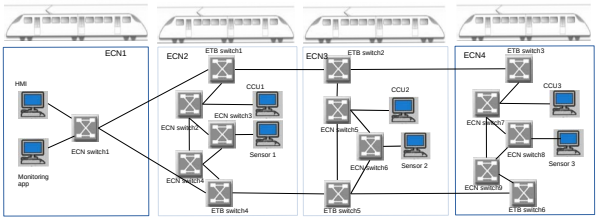

Fig. 4: Time-aware network

In our different simulation scenarios, each time-aware system uses the following configuration parameters for clock synchronization:

- Announce message interval is set to 3 seconds
- Announce message time-out is configured to 5 seconds
- Synchronization message interval is set to 100ms for the first five synchronization messages, later the interval is set to 1 second
- The *Pdelay_Req* message that is used for the mean path delay calculation is generated at the reception of a synchronization message
- The first announce message starts after 5.2 seconds due to the RSTP configuration period
- The first synchronization message is sent after 5.3 seconds since a grandmaster is needed to be selected first.
- Each time-aware system in the simulated network is configured with two priority values for BMCA and a drift rate for the local clock computation.
- The propagation delay of the communication link is set to 8 μs.

It is noteworthy that the all aforementioned parameters are user-configurable and can be changed according to the network requirements (e.g. synchronization accuracy).

B. Time-Aware Network Simulation

Once the simulation begins to run, all time-aware systems will communicate their pre-configured clock priorities with each other via announce messages. In our case, we have 23 time-aware systems and each time-aware system receives 22 announce messages from its neighbours. As soon as the time-aware system receives an announce message, the BMCA will be executed and the time-aware system identifies itself as a grandmaster or as a slave gPTP device based on the BMCA outcome.

In our experimental network, all time-aware systems are grandmaster-capable devices, thus every system participates in the BMCA. We set the priority attribute of the HMI to the highest value in our emulated network. Consequently, HMI is elected as the grandmaster gPTP device and it starts sending synchronization and announce messages periodically. Other time-aware systems in Figure 4 are slave gPTP devices and schedule an announce message time-out event to examine the state of the grandmaster device at a specific interval.

NRR is an important parameter to calculate the frequency difference between grandmaster clock and slave clock. The NRR is calculated each time the mean path delay is computed. After NRR computation, CSRO is calculated based on the principle depicted in Figure 3.

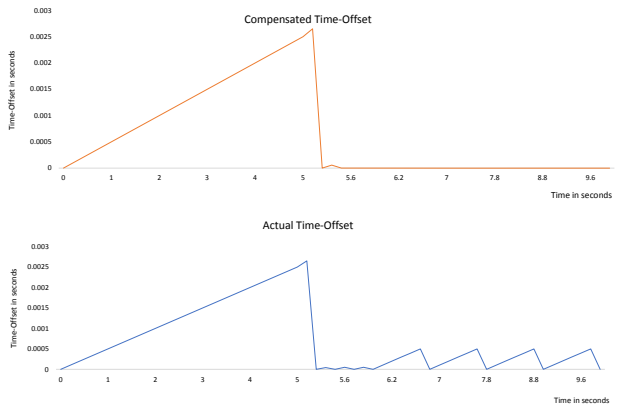

Fig. 5: Time offset of sensor 3 with 500ppm drift rate

Figure 5 depicts the synchronization process of sensor 3 (i.e. slave device). Before receiving the first synchronization message, the actual time offset and compensated time offset are the same since NRR, CSRO and the mean path delay are not computed yet. The compensated time offset is settled approximately to zero after sensor 3 receives five synchronization messages. This means that the linear drift error of sensor 3 clock is compensated with the help of NRR and CSRO. Therefore, it is necessary to reduce the synchronization message interval during initialization phase to achieve zero compensated time offset quickly. However, the path from HMI to sensor 3 comprises six intermediate time-aware relays. Zero compensated time-offset is achieved by exchanging five synchronization messages (at 5.6 seconds). Moreover, the actual time-offset is set to zero upon reception of a synchronization message and then it increases linearly until the device receives the next synchronization message.

Furthermore, a non-linear clock has been modeled based on a reference paper [14] to reflect the difference with respect to the compensated time-offset compared to a linear clock. The drift time of a non-linear clock is computed as follows:

$$c(t) = \rho_0 t + \sigma t^2$$

Where ρ_0 is a skew rate and σ is clock drift rate. A simulation is executed with a skew rate of 0.01, and a drift rate of 1000 ppm. The linear clock results in a maximum of 50 nanoseconds time-offset while the non-linear clock has a maximum 200 nanoseconds time-offset over one-second

synchronization interval. Since NRR calculation involves linearization, there is a difference between the compensated time-offset of a clock with linear and non-linear drift time. However, the non-linear behaviour of a clock within a short duration can be linearized, thus the reduction of the synchronization interval will also reduce the compensated time-offset. Selecting the non-optimal periodicity of synchronization messages leads to either an inefficient network usage or an erroneous NRR. Therefore the synchronization interval needs to be chosen carefully based on the clock behaviour and the expected synchronization accuracy.

The experimental results shows that the same accuracy of the compensated time offset is achieved with both linear and non-linear clocks since in OPNET time can only be represented in microseconds. Thus, the difference between the compensated time-offset of linear and non-linear clocks in nanoseconds does not affect the accuracy of local times.

C. Time-aware Network Simulation with Fault Injection

In the previous scenario, the evaluation of BMCA and synchronization mechanisms is carried out, but the announce message time-out functionality in slave devices needs to be examined. For this purpose, we inject a failure of the primary grandmaster (i.e. HMI) during the simulation run, and investigate how the compensated time-offset in a slave device changes.

In the fault-free network, when a slave device receives an announce message from its primary grandmaster, the announce message time-out attribute is updated. Nevertheless, if there is no announce message within the announce message time-out period, all slave devices store the information about the primary grandmaster for rollback and communicate their clock priorities through an announce message in order to identify a new grandmaster device. The newly elected grandmaster performs the same tasks as the primary grandmaster until the failed primary grandmaster recovers.

At the beginning of this simulation run in a similar way to the previous use case, HMI is elected as grandmaster. The rest of devices act as slave gPTP nodes and schedule an announce time-out event at 10 seconds. First five synchronization messages are transmitted by HMI every 100 milliseconds for the mean path delay, NRR and CSRO computation. Once the correct mean path delay, NRR and CSRO are calculated, the compensated time-offset settles to zero as shown in Figure 6.

After 6 seconds, the primary grandmaster goes to the failure state until 13 seconds. During this period, the compensated time-offset of a slave device (e.g. sensor 3) is zero because its local clock frequency has been syntonized with the grandmaster clock frequency. At 10 seconds, the announce time-out event is triggered in all slave gPTP devices. Upon reception of announce messages, BCMA re-executed in all time-aware systems and the monitoring app is selected as a new grandmaster.

The new grandmaster starts sending periodic announce and synchronization messages to all slave gPTP devices. Nevertheless according to the experimental results shown in

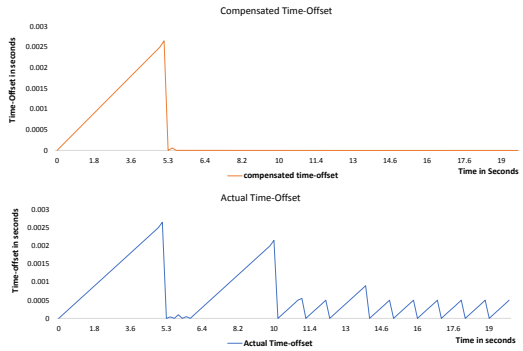

Fig. 6: Time offset of sensor 3 with 500ppm drift rate when the primary grandmaster fails and recovers

Figure 6, irrespective of changes in the grandmaster clock or the synchronization message interval the compensated time-offset of sensor 3 remains unchanged (i.e. zero) because of the constant value of NRR and CSRO. This feature is considered as one of the major advantages of the gPTP standard over the state-of-the-art synchronization mechanisms.

At 13 seconds HMI (i.e. the primary grandmaster) recovers from its failed state. At 14 seconds it sends an announce message to all other time-aware systems to trigger all devices to restore the primary grandmaster clock information and also to schedule an announce message time-out event. The monitoring app (i.e. the current grandmaster) will also move to the slave state, cancel its scheduled announce and synchronization messages and finally create an announce message time-out event like other slave devices.

It is good to mention that another simulation run has been evaluated by failing the link between HMI and its neighbour time-aware relay (i.e. ECN switch1). The resulted behaviour of the time-aware network is similar to the previous node failure test (i.e. HMI crash failure).

D. Time-aware Network Simulation with IEEE 802.1Qbv and IEEE 802.1Qci Integration

In the previous simulation scenarios, all gPTP messages are declared as a best-effort traffic. Moreover, including the best-effort traffic, the Time-Triggered (TT) traffic is scheduled and filtered using the OPNET simulation time ($op\_sim\_time()$). As a result, there were no scheduling and filtering issues related to local clocks because all devices are referring to $op\_sim\_time()$.

In the design of gPTP framework, each time-aware system has its own clock with specific drift drift rate. If all time-aware systems (slaves) are not synchronized to the grandmaster time, the ingress time-based filtering in the time-aware system discards the incoming TT packets due to multiple time-domains in the network topology. Nevertheless, the filtering process in time-aware systems is not applicable for BE traffic and this helps to achieve the fully synchronized

network even though the local clocks are not synchronized yet.

To integrate the IEEE 802.1Qbv and 802.1Qci standards with the IEEE802.1AS-Rev standard, it is necessary not to start sending TT traffic before 6 seconds. From the previous simulation case, it has been identified that all gPTP messages are sent after 5.2 seconds due to the RSTP configuration, and the synchronization accuracy will reach sub microseconds (compensated time-offset) after exchanging five synchronization messages.

Integration of IEEE802.1AS-Rev with existing TSN models is performed by replacing the *op_sim_time*() with the gPTP compensated time in ingress time-based filtering and the egress time-aware shaper.

A simulation run is executed with the same simulation parameters as in the above use cases, and the same synchronization accuracy is achieved although the BE traffic reference time is changed from *op_sim_time*() to the gPTP local clock. The result shows that the integration of IEEE 802.1Qbv and Qci protocols with gPTP is successful since there is no packet drop due to untimely TT traffic. For example, Figure 7 shows the compensated time offset of CCU1 and also the TT traffic received by CCU1 from sensor 1 without any message losses.

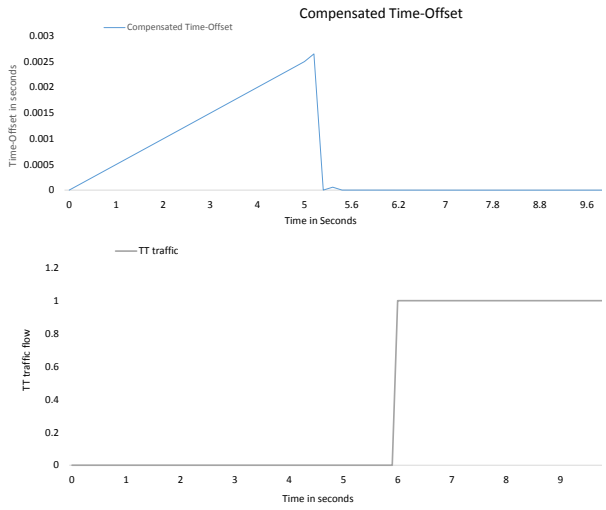

Fig. 7: IEEE 802.1Qbv and Qci-capable device (i.e. CCU1) using gPTP local time for filtering and shaping traffic

VI. CONCLUSION

This paper is focused on developing a simulation framework for the IEEE 802.1AS-Rev standard, and integration with IEEE 802.1Qbv and 802.1Qci standards. The synchronization process based on IEEE 802.1AS-Rev is not affected by other traffic such as TT traffic. Therefore, the synchronization error and its accuracy remain below a fraction of microseconds over seven hops. However, the grandmaster synchronization interval selection, the wrong computation of NRR, the announce/synchronization slave-timeout configuration and the measurement of time-stamps of gPTP event messages have a great impact on the synchronization accuracy. Therefore, based on the type of clock (linear or non-linear) and the expected synchronization accuracy, these parameters should be adjusted. Interoperability between IEEE 802.1Qbv and 802.1Qci with IEEE 802.1AS-Rev is evaluated in this work and the result shows that TSN synchronization offers deterministic behaviour that is the main concern of modern industrial control networks.

ACKNOWLEDGMENT

This work was supported by the DFG research grants DAKODIS OB384/5-1 and ADISTES OB384/6-1.

REFERENCES

[1] T. Nolte, H. Hansson, and L. L. Bello, "Automotive communications-past, current and future," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, vol. 1, pp. 8–pp, IEEE, 2005.

[2] P. Heise, F. Geyer, and R. Obermaisser, "Tsimnet: An industrial time sensitive networking simulation framework based on omnet++," in *8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2016*, pp. 1–5, IEEE, 2016.

[3] "Institute of Electrical and Electronics Engineers, Time-Sensitive Networking," in *Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html*, IEEE, 2017.

[4] "Institute of Electrical and Electronics Engineers, Network Time Protocol," in *[Online]. Available: https://tools.ietf.org/html/rfc5905*, IEEE, 2010.

[5] "Institute of Electrical and Electronics Engineers, IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," in *[Online]. Available: https://standards.ieee.org/findstds/interps/1588-2008.html*, IEEE, 2008.

[6] "Institute of Electrical and Electronics Engineers, Inc. 802.1AS - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," in *Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/files/private/as-drafts/d7/802-1AS-d7-6.pdf*, IEEE, 2010.

[7] "Institute of Electrical and Electronics Engineers, Inc. 802.1AS-Rev - Timing and Synchronization for Time-Sensitive Applications," in *Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/802.1AS-rev.html*, IEEE, 2017.

[8] M. Pahlevan and R. Obermaisser, "Evaluation of time-triggered traffic in time-sensitive networks using the opnet simulation framework," in *26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2018*, pp. 283–287, IEEE, 2018.

[9] M. Pahlevan and R. Obermaisser, "Redundancy management for safety-critical applications with time sensitive networking," in *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 1–7, IEEE, 2018.

[10] H. Guo, *Time Synchronization and Communication Network Redundancy for Power Network Automation*. PhD thesis, The University of Manchester (United Kingdom), 2017.

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

[12] "Introduction to Riverbed Modeler Academic Edition," in *https:splash.riverbed.com/docs/DOC-4833*, 2018.

[13] "IEC 61375-1:2012. Train communication network (TCN) - part 1: TCN general architecture,"

[14] Y. Quan and G. Liu, "Drifting clock model for network simulation in time synchronization," in *2008 3rd International Conference on Innovative Computing Information and Control*, pp. 385–385, IEEE, 2008.