Extended Synchronization Protocol Based on IEEE802.1AS for Improved Precision in Dynamic and Asymmetric TSN Hybrid Networks

Haytham Baniabdelghany
University of Siegen
Siegen, Germany
haytham.baniabdelghany@uni-siegen.de

Roman Obermaisser
University of Siegen
Siegen, Germany
roman.obermaisser@uni-siegen.de

Ala' Khalifeh
German Jordanian University
Amman, Jordan
ala.Khalifeh@gju.edu.jo

***Abstract*—This paper proposes an extension of the precision time protocol (IEEE 802.1AS) to improve clock synchronization in hybrid networks including both wireless and wire-bound segments. This extension targets systems based on Time Sensitive Networking (TSN) such as industrial applications and vehicular systems. By considering the deterministic delays and the clock drift, the precision of the clock synchronization of the standard 802.1AS scheme can be significantly improved. Furthermore, in order to support dynamic communication environments with mobile nodes, the problem of asymmetric delays in the transmission of the timing packets of the synchronization protocol is also considered. Therefore, a Path Deviation Delay (PDD) filter is introduced to monitor the traffic behavior of timing packets as well as to exclude outlier values which can occur as a result of dynamic and asymmetric scenarios.**

The paper presents a simulation framework for the proposed protocol to evaluate it with TSN-based systems that support the TSN features (e.g. IEEE 802.1Qci and IEEE 802.1Qbv sub-standards). The simulation results show that the proposed protocol improves the synchronization precision compared with the standard 802.1AS protocol. It shows that the proposed protocol enhances the synchronization precision to less than 1 microsecond. In contrast, the synchronization error increases significantly with the standard protocol in the presence of different asymmetric ratios and in mobile node scenarios.

***Index Terms*—Clock synchronization, hybrid systems, TSN, IEEE802.1AS, asymmetric delays.**

I. INTRODUCTION

At the present time, there is a huge success and widespread deployment of hybrid networks including wired (e.g., Ethernet networks) and wireless technologies, especially in automation systems, smart cities and home area networks [1]. On one hand, Ethernet networks enable higher reliability and performance as a result of topology stability and less probability of packet loss. On the other hand, wireless networks give more flexibility and reduce the weight of the wired infrastructure [1], [2]. Recently, many automation and vehicular systems have introduced hybrid networks in applications with real-time and safety requirements. For wirebound networks, the Time Sensitive Networking(TSN) task group [3] introduced a series of IEEE 802.1 sub-protocols to offer a fault tolerant and deterministic communication infrastructure for mission-critical systems over Ethernet-based networks.

Clock synchronization is one of the most important services to coordinate distributed access to shared resources, compare timestamps established at different locations, and measure the performance of distributed real-time systems [3]. The aim of clock synchronization is the establishment of a global notion of time with a predefined precision by ensuring bounded maximum offsets between any two nodes [4].

In order to meet requirements for precision of the clock synchronization, IEEE published the IEEE 1588 standards [5]–[7] in 2002 to 2008, and IEEE 802.1AS [8]. Hybrid networks are gaining importance in safety-critical distributed systems, and the clock synchronization is a critical service to realize safety relevant services that span both wired and wireless subsystems. Moreover, in hybrid wireless networks, there are different delays that need to be considered in clock synchronization, namely deterministic delays (hardware-processing, propagation, transmission and receiving) and random delays [9], [10] (channel access and transmission jitter).

Therefore, this paper aims to build a synchronization protocol on top of the TSN IEEE 802.1qbv substandard [11], which is used to distinguish different message types in a stream (e.g. time-triggered and best effort) and store them in separate egress queues, and on top of the TSN IEEE 802.1Qci substandard [11], which is used for filtering and policing the time-triggered traffic over the hybrid networks. To satisfy the timing requirements of TSN in hybrid networks with dynamic data rates and asymmetric wireless characteristics, this paper proposes an extension of the 802.1AS synchronization protocol to improve the precision within the hybrid networks. The proposed simulation framework provides an opportunity to evaluate the correctness of real-time capability of TSN solutions in realistic dynamic and asymmetric networking scenarios.

The rest of this paper is organized as follows. The next

978-1-7281-6949-1/20/$31.00 ©2020 IEEE

section discusses the principles of the IEEE 802.1AS protocol. Section 3 introduces related work. The explanation of the proposed protocol is provided in Section 4. Section 5 demonstrates the simulation models for TSN clock synchronization. The experiments and evaluation are presented in section 6. Finally, section 7 draws conclusions and describes future work.

II. STANDARD 802.1AS PROTOCOL

In the first phase, IEEE802.1AS [8] uses an algorithm called Best Master Clock Algorithm (BMCA), where all participating time-aware nodes in the IEEE802.1AS domain send announce messages, the grandmaster is selected depending on the highest priority, and the rest are assigned as slave nodes. The grandmaster node sends periodically synchronization messages with its local time to all slave nodes. Moreover, a peer delay mechanism is applied between each time-aware node and its adjacent time-aware nodes, where every node responds to the *delayreq* message that is sent by an adjacent node, by sending a *delayrep* message, as shown in Fig.1. All timing messages in the peer delay mechanism are encapsulated with the local time of the node that sent or received a message. As a result of the peer delay mechanism, the link delay between the adjacent aware nodes can be measured. Once the synchronization message is received to any time-aware relay, the link delay and the processing time (residence time) will be added to the accumulated value in *the correction field* in the synchronization message and forwarded to the other time-aware devices. In case of a series of time-aware relay clocks exists between a grandmaster clock and a slave clock, the link delay and residence time measurement process will be repeated for each of them. At the end, every time-aware end-station clock adjusts its local time relating to the grandmaster clock.

III. RELATED WORK

In the last years, several works studied different time synchronization protocols using various simulation frameworks. The authors of [9] applied the Precision Time Protocol (PTP) to IEEE 802.11 WLAN to solve the problem of asymmetric links. A delay filtering algorithm based on Kalman filters was designed as a pure software-based system and applied to coordinate each local clock with its master clock. The algorithm dealt with asymmetric delays in a single-hop WLAN environment by considering deterministic and random delays. Software time stamping in the application layer has a negative impact since there will be an extra overhead passing through layers. It can also take longer, because of more CPU usage and add more complexity. In contrast, hardware time stamping makes the time synchronization more precise.

In [12] an optimized version of PTP was proposed for wireless networks. By utilizing the delayed transmission feature in the DW1000 transceiver, the authors of [12] inserted the exact sending timestamp into the synchronization message to reduce the timing packet overhead. All experiments were performed in simple wireless networks without considering the asymmetric delay of dynamic wireless networks.

The goal of [13] was computing clock drift between the master and slave clocks more precisely by applying adaptive time compensation. The implementation used only standard Time-Slotted Channel Hopping (TSCH) control messages. This paper dealt with synchronization errors as a result of the drift but they did not consider other factors like asymmetric transmission delays.

The authors in [14] incorporated a clock drift factor; thereby significantly improving the accuracy of the clock offset calculation of a standard PTP scheme. The symbol timing synchronization was also studied, which enabled robust timestamp message decoding during the clock synchronization period. The authors considered just symmetric traffic during the exchange of the timing packets.

In order to enhance the precision of the PTP synchronization, the authors of [15] provided two approaches. The first approach considered the drift in the frequency, the second one reduced the asymmetric delays caused by the imbalanced hardware processing delays on download and upload links. The work did not deal with variable surrounding conditions such as traffic congestion and link loss as a result of dynamic topologies.

In contrast to the related work, our work aims to improve the time synchronization for scalable hybrid network with considering all aforementioned factors that effect on the time synchronization accuracy in an extended synchronization protocol. The time synchronization process is implemented on MAC layer rather than other higher layers to avoid extra overhead with faster response for exchange time synchronization messages. This paper focuses accurately on measuring the clock drift and path delay. Whilst the path delay is a sensitive value, the deterministic delay in the peer delay mechanism is considered to reduce the asymmetric delays caused by unpredictable surrounding behaviors (e.g., mobility [16], data rate and direction) for these nodes. As an additional step, a Path Deviation Delay (PDD) filter is applied on the value of the path delay to exclude outlier values as a result of traffic congestion or lossy links in a high dynamic hybrid networks. To the best of our knowledge, this work is the only one that addresses a global and unified clock for hybrid network with TSN based systems. While there is a trend to apply hybrid (wired and wireless) TSN networks in various industrial fields [17].

IV. THE PROPOSED 802.1AS PROTOCOL

Asymmetric delays in time-aware hybrid networks are one of the main reasons of the inaccuracy of the synchronization process. The synchronization precision is affected by the asymmetric delays in such a way that the degree of precision depends on the accuracy of the calculated path delay.

In time-aware systems based on the 802.1AS synchronization protocol, all slave nodes in the same domain share a global time base which is realized using the robust synchronization mechanism. The standard 802.1AS synchronization protocol is designed mainly for Ethernet networks and assumes that the communication links between nodes are symmetric. Therefore,

determining the path delay is simple by calculating the mean delay value for *delayreq* and *delayrep* messages.

In wireless/hybrid networks, this is not an optimal solution especially in asymmetric wireless networks. Moreover, additional conditions such as variable data rate and the mobility of the nodes would affect negatively the synchronization precision. To make it worse, asymmetric delays may vary according to the surrounding environment. This variation can change frequently for the same node depending on the traffic during the transmission of timing messages. The delay variation actually depends on many factors. In this paper, the variable data rate and the mobility of the time-aware system are considered.

To get more accurate synchronization time at a slave clock, the frequency drift between the grandmaster clock (as a reference clock) and the slave clock must be considered. To achieve that, a frequency ratio of the neighbor time-aware devices, known as Neighbor Rate Ratio (NRR), is computed [18]. The NRR value is computed in each time-aware relay and it is equal to the ratio of the neighbor clock frequency and the own clock frequency. On the other side, Cumulative Scaled Rate Offset (CSRO) is an aggregated value by adding the system's NRR and is carried by the synchronization message [18]. Therefore, CSRO is used for adjusting the frequency of an oscillator in the time-ware device (syntonization) with the grandmaster clock through travelling the synchronization message. It is considered that the frequency of the oscillator of each node may drift from the ideal case as a result of the temperature, or the clock age [19], [20]. The purpose of this work is to measure more accurate delay values for the *delayrep* messages that were sent from each adjacent time-aware node. Because the delay of the *delayrep* message represents the actual link delay of the synchronization message as it passes to reach the time-aware slave nodes. Therefore, it is required to be accurately measured as shown in this section.

A. Modeling of delayrep_message delay (Delay_reply)

As mentioned earlier, the slave node adjusts its local time with the grandmaster node. The offset value with the grandmaster should be minimized as much as possible. Therefore, if the peer delay mechanism is incorrect, it leads to a bias or offset value in the time between the local clock and its grandmaster clock. When asymmetric traffic is used, the difference of the propagation and transmission delays must be known, and this difference should not be changed during the messaging traffic. But some procedures for identifying the difference in the peer delay traffic (upload and download) are not mentioned in the standard protocol. This leads to synchronization precision problems for dynamic wireless environments, where the data rate is frequently changed as a result of channel condition, mobility or the characteristics of the service. If we consider that the grandmaster node is located in the wired network, the synchronization traffic will be transmitted through the wired, wireless bridge until reaching the wireless time-aware end-stations. These wireless links would be WLAN, 2nd, 3rd or 4th generation broadband technologies (such as mobile WiMAX and LTE). The changes in the data rate for the wireless links relates to the following reasons: firstly, the speed of the wireless time-aware node (i.e. sensor). If the speed is very low then the speed cannot be considered as a factor to make anaccurate synchronization process. However, if the mobile node increases its speed, this leads to fluctuation in its data rate and makes it unstable due to changes in the quality of the wireless link. Secondly, a wireless link is a shared link for simultaneous users who want to send data at the link. Thus, the allocated time varies for each scheduled interval. Some conditions in the link such as channel fading and errors will affect the date rate. Thirdly, the wireless technologies use asymmetric duplexing mechanisms. For example, asymmetric traffic for specific services (i.e. webpage) is used with higher bandwidth for downloading than uploading. LTE and WiMAX use Time Division Duplexing (TDD) for the web services. These asymmetric characteristics cause time offsets between the grandmaster clock and its slave clocks.

Thus, because of integrating the synchronization process in time sensitive networks, which contains different node models, random mobility, dynamic data rates and asymmetric characteristics, it is essential to propose a model to deal with the mentioned variables.

1) Symmetric Degree Ratio (SDR) expression: In order to measure the ratio of delays for *delayreq* and *delayrep* messages during the peer delay mechanism, the Symmetric Degree Ratio (SDR) ratio is introduced as shown in (1). If its value is unity or very close to unity, this means symmetric delays for *delayreq* and *delayrep* messages. By considering and identifying all the differences in the peer delay traffic, the SDR value becomes approximately one. Therefore, SDR ratio gives an indication of the symmetric peer delay mechanism. If the SDR value deviates from unity, it means we have to consider asymmetric environmental characteristics in the wireless dynamic network.

The SDR ratio of a time-aware system can be calculated as:

$$SDR = \frac{Delay\_request}{Delay\_reply} \tag{1}$$

Where, *Delay_request* and *Delay_reply* are the times needed to transmit *delayreq* and *delayrep* messages, respectively, as shown in Fig.1.

$$Delay\_reply = t4–t3 \tag{2}$$

$$Delay\_request = t2–t1 \tag{3}$$

Where, *t1* and *t2* are the timestamps of the sender and receiver clocks when *delayreq* message is sent and received, respectively. Similarly, *t3* and *t4* are the timestamps of the sender and receiver clocks when *delayrep* message is sent and received. The difference in the transmission time equals the absolute value of subtracting (4) and (5).

$$Delay\_request\_trans\_time = \frac{Delay\_request\_size}{data\_rate} \tag{4}$$

$$Delay\_reply\_trans\_time = \frac{Delay\_reply\_size}{data\_rate} \quad (5)$$

Data-rate in (4) and (5) refers to the sending node of *delayreq* and *delayrep* messages, respectively.

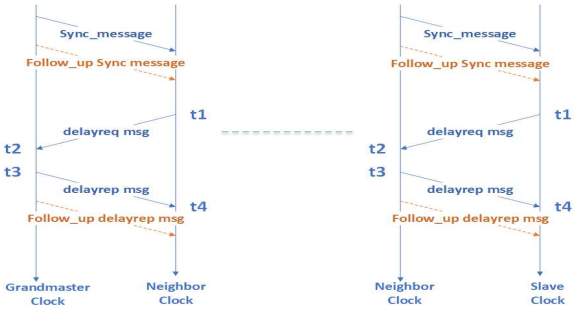


Fig. 1: Peer Delay Mechanism.

According to the continuous movement of the time-aware wireless nodes, the distance between them will change. Therefore, the propagation delay for *delayreq* and *delayrep* messages is calculated using (6) and (7), respectively.

$$Delay\_request\_propagation\_time = \frac{Dist\_req}{speed\_of\_light} \quad (6)$$

$$Delay\_reply\_propagation\_time = \frac{Dist\_rep}{speed\_of\_light} \quad (7)$$

Where, *Dist_req* and *Dist_rep* are computed based on the distances between the message sender and the receiver for *delayreq* and *delayrep* messages, respectively. In contrast, the coordinates of the adjacent node that receives *delayreq* and sends *delayrep* messages are user configurable.

The difference in the propagation time equals the absolute value of subtracting (6) and (7).

The fixed (deterministic) time difference equals the sum of the difference in the transmission time and the difference in the propagation time. Eq. (8) shows the consideration of the fixed time difference value in the SDR ratio.

$$SDR = \frac{Delay\_reply}{Delay\_request + Fixed\_time\_diff} \quad (8)$$

The fixed-time difference deals with delays required for the transmission on the physical layer and the traffic propagation in the medium. Even if the fixed-time difference is computed and considered in the SDR ratio, in some simulation scenarios, the SDR ratio deviates from unity as a result of highly dynamic changes in the size and the topology of the network, besides the random access delays. Later, this issue can be dealt with by conducting further studies.

Equation (9) shows how to compensate the fixed-time difference for *delayreq* and *delayrep* messages.

$$t4\text{–}t3 = (t2\text{–}t1) + Fixed\_time\_diff \quad (9)$$

Dividing (9) by (t4 - t3), and by using (1), the SDR ratio can be expressed using (10).

$$\mathbf{SDR = 1 - \frac{Fixed\_time\_diff}{t4\text{–}t3}} \quad (10)$$

2) Neighbor Rate Ratio: As mentioned earlier, the NRR value is used to compensate the drift time of a local clock. Therefore, to simulate time-aware systems with different drift rates, (11) and (12) show the expressions of the timing offset and the normalized frequency offset (the clock drift factor) between each current node clock and its neighbor clock.

$$t2 - t1 = Delay\_request\text{–}Offset\text{–}Drift \quad (11)$$

$$t4 - t3 = Delay\_reply + Offset + Drift \quad (12)$$

$$Where, Drift = NRR * (current\_time\text{–}last\_sync\_time) \quad (13)$$

The current_time represents the local time of the node once it receives the synchronization message. In contrast, *last_sync_time* is the local time of the node since the last time it received the synchronization message. NRR represents the clock drift factor between the current node, which receives the synchronization message with its neighbor node that has sent the synchronization message.

Equations (11) and (12) are reformulated as shown in the following equations.

$$t2 = t1 + Delay\_request\text{–}Offset\text{–}NRR * (t1\text{–}last\_sync\_time) \quad (14)$$

$$t4 = t3 + Delay\_reply + Offset + NRR * (t4\text{–}last\_sync\_time) \quad (15)$$

The value of NRR can be computed via the last two consequent *delayrep* messages, as shown in Fig. 2.

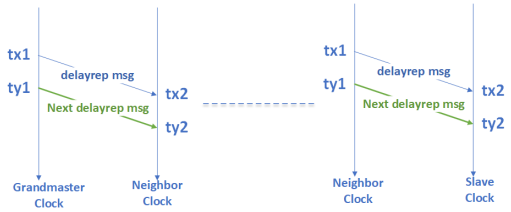


Fig. 2: Two consequent *delayrep* messages.

Similarly to (15) the expressions of *tx2* and *ty2* can be shown as follows:

$$tx2 = tx1 + Delay\_reply + Offset + NRR * (tx2\text{–}last\_sync\_time) \quad (16)$$

$$ty2 = ty1 + Delay\_reply + Offset + NRR * (ty2\text{–}last\_sync\_time) \quad (17)$$

Where, *tx2* and *tx1* are previous timestamps of the *delayrep* message in the receiving node clock and the sending neighboring time-aware node, respectively. Similarly, *ty2* and *ty1* are the current timestamp values of the same *delayrep* message. In contrast, the neighboring node is where the synchronization message was sent or forwarded.

By using (16) and (17), the expression of NRR can be calculated using (18):

$$\mathbf{NRR = 1 - \frac{ty1 - tx1}{ty2 - tx2}} \quad (18)$$

It is worth to note that if the neighboring time-aware systems have the same drift rate, then the measured NRR value is unity, this means there is no difference in the frequency between them, which leads to inaccuracy in the synchronization process at the slave node. Because of the actual frequency difference between the neighboring systems, the CSRO is used to accumulate the values of NRR during the synchronization message traffic between the grandmaster and the slave clocks. If there are multiple time-aware relays between the grandmaster and the slave node, each intermediate time-aware relay will compute its NRR and add it to the accumulative CSRO value, and then the CSRO value will be loaded in *the CSRO field* of the synchronization message. Even with NRR unity values between neighboring systems, the respective frequency ratio will be obtained by the aggregated CSRO values.

3) Formulating Delay_reply value: After formulating the SDR and NRR equations from the previous sections, it is the time to express the Delay_reply formula as will be shown in this section.

By using (14) and (15), (19) is formulated as follows:

$$(t2+t4)-(t1+t3)=Delay\_request+Delay\_reply+NRR*(t4-t1) \quad (19)$$

Dividing (19) by the Delay_reply value, the Delay_reply value is obtained using (20):

$$\mathbf{Delay\_reply} = \frac{(\mathbf{t2}-\mathbf{t1})+(\mathbf{t4}-\mathbf{t3})-\mathbf{NRR}*(\mathbf{t4}-\mathbf{t1})}{\mathbf{SDR}+\mathbf{1}} \quad (20)$$

Substituting the outputs of (10) and (18) in (20), an accurate Delay_reply value (the delay of the *delayrep* message) is obtained. Thereafter, it will be used as a path link delay instead of the mean path delay in the time-aware hybrid TSN systems. The processing time of the synchronization message in the time-aware relay system will be also computed, it will be added besides to the Delay_reply value into *the correction field* of the synchronization message.

In the symmetric scenario (ideal scenario), the value of NRR and SDR should be 0 and 1, respectively. According to (20), the Delay_reply value in the symmetric scenario is calculated as follows:

$$Delay\_reply = \frac{(t2-t1)+(t4-t3)}{2} \quad (21)$$

Equation (21) is used normally in the symmetric frameworks such as the wired networks, but it is not feasible in the dynamic wireless environments because of NRR and SDR values mostly are not 0 and 1, respectively.

Regarding to the previous analysis, the synchronization precision in the asymmetric wireless networking depends on computing accurate values for time and frequency offsets, this ensures that the slave time is nearly equal to the grandmaster time.

As a general review, the peer delay mechanism is used to compute NRR and the Delay_reply values of the time-aware neighboring systems. Once the time-aware relay receives the synchronization message from the grandmaster, it will add its NRR to the aggregated CSRO value, then the added value will be inserted in the *the CSRO field* of the synchronization message. At the same time, it will also add the summation of its computed Delay_reply value and the processing time into *the correction field* of the synchronization message. Thereafter, the synchronization message will be forwarded to the time-aware end-station system, when it receives the synchronization message, the compensated time is calculated as a result of adding the values in the CSRO and the correction fields to the original grandmaster time that was already loaded in the synchronization message in *the original grandmaster time field*.

B. Path Deviation Delay Filter Test

In the previous section, the Delay_reply value is calculated according to the accurately measured SDR and NRR values. With more attention, it is noted that the value of SDR seems to be deceptive, because the SDR ratio might give us unacceptable values for Delay_request (in the numerator) and for Delay_reply (in the denominator) but an acceptable SDR ratio, especially when dealing with a time sensitive environment, where the deadline conditions play a critical role in the time sensitive applications. Therefore, to deal with acceptable values for Delay_reply and to ensure that it is not an outlier value; a filter called Path Deviation Delay (PDD) filter is used to compare it with other values stored in the time-aware relay.

After receiving the synchronization message and exchanging peer delay messages at the time-aware relay system, the Delay_reply value is computed according to the measured NRR and SDR values. In this filter, the previously stored Delay_reply values are utilized in a table called Path Delay Table (PDT). The idea is to observe the standard deviation of the Delay_reply value with previously stored values. If the value of the standard deviation is an outlier value (away from the mean), the Delay_reply value is excluded because it does not satisfy the TSN application requirements [11] and it might be an indication of traffic congestion or out of coverage. Therefore, we just keep the previously updated synchronized time until another synchronization message with an acceptable and successful filter test is coming from the time-aware neighbor system. If the standard deviation is close to the previous values, at this moment, it can be used to update the time of the time-aware relay clock and upload it in *the correction field* to the next node until reaching the time-aware end station, as shown in Fig. 3. This filter contributes to removing the effects of random delays. It is worth to note that discarding several consecutive synchronization messages will reduce the synchronization precision, thus the utilized simulation framework described in the simulation section, may not work perfectly in very high congested dynamic environments.

V. HYBRID NETWORK SIMULATION MODELS FOR TSN CLOCK SYNCHRONIZATION

In this section, the simulation model and environment used to evaluate the effectiveness of the proposed synchronization

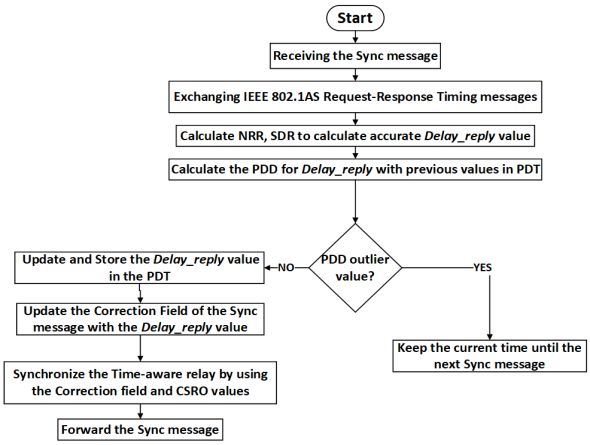


Fig. 3: PDD Filter Test.

protocol is described. The TSN synchronization functionalities of the wireless nodes and bridges are implemented and integrated with the existing TSN models in [21]. These models were tested and implemented in the OPNET, which is a discrete event simulator that allows to modify the existing models and uses them in different platforms [22]. The models are an extension to the work in [18], where the standard wired nodes and Ethernet relays were modeled to be time-aware devices in OPNET network environment. The focus of this paper is to model the wireless devices (wireless nodes and bridge models) as shown below.

- Time-Aware Wireless End-station: All packets belonging to the IEEE 802.1AS protocol are formulated into WLAN frames. Therefore, the existing standard wireless node model is used for modeling the time-aware wireless end-station.
- Time-aware Wireless Bridge: The functionalities of the boundary clock as proposed in the IEEE 802.1AS protocol were applied in the time-aware wireless bridge. Thus, the existing bridge model in OPNET is used for modeling the time-aware bridges.

All models (including Ethernet and wireless models) incorporate the synchronization process and the BMCA algorithm by using the proposed IEEE 802.1AS protocol, where, these models are re-defined to support IEEE 802.1Qbv, IEEE 802.1Qci and IEEE 802.1CB protocols [21]. These protocols control all time-triggered streams by using Time-Aware Shaper (TAS) and protecting them against faults.

VI. EXPERIMENTS AND EVALUATIONS

In the hybrid TSN simulation framework, different time-aware nodes will be used including wired and mobile wireless end-stations, Ethernet relays and wireless bridges. The OPNET simulation runs on a PC (Intel i5) with 24 GB of memory.

A. Experimental Setup

To evaluate the behavior of the proposed protocol, an example layout of a hybrid embedded system is used. As illustrated in Fig. 4, two wireless time-aware bridges connect a group of time-aware smart sensors. On the other side of the bridges, time-aware wired relays are connected with a time-aware Human-Machine Interface (HMI) and a monitoring application. Data traffic of every sensor is sent to the corresponding Central Computing Unit (CCU). Then, the CCU forwards the traffic to the HMI and the monitoring application. After that, the HMI sends the processed data back to the CCUs.

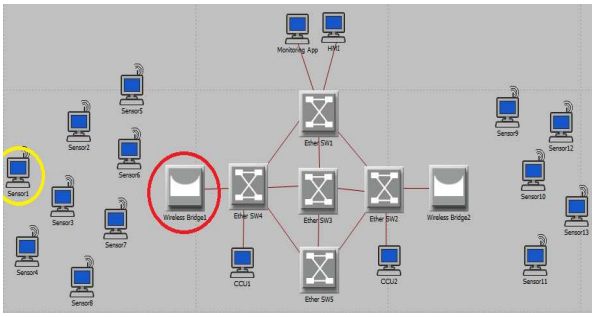


Fig. 4: Time-aware Hybrid Network.

B. Time-Aware Hybrid Network Simulation

The BMCA algorithm is executed in our experimental network for all the time-aware systems. In [18], the authors show the configuration parameters for clock synchronization and how the BMCA algorithm works to select the grandmaster. In the emulated work, the HMI is selected to be the grandmaster according to the highest priority. It is worth to note that all parameters are user configurable and can be changed depending on the network specifications.

The proposed protocol is evaluated with different scenarios, including different mobile speeds, different oscillation drifts between the grandmaster and the slave clocks, increasing number of hops between them, and different asymmetry ratios. The simulation results of both the proposed and the standard protocols are compared and a detailed analysis is presented. The oscillation drifts are measured in the parts per million (ppm) unit which it indicates how much the crystal's frequency may deviate from the nominal value [19], [20]. The asymmetric ratio is a ratio in the data-rate between the wireless uplinks and downlinks.

Fig. 5 shows the computed synchronization error in (ns) according to the variation in ppm between the grandmaster (HMI) and sensor1. The synchronization error is calculated by referring to OPNET simulation time (i.e. *op_sim_time ()*). Fig. 5 shows how the variation in ppm between the grandmaster and the slave clocks affects the synchronization precision. It shows linear increment in the synchronization error with increasing ppm variation, thus it is necessary in this paper to consider the variation in the frequency by calculating the accumulated value of CSRO at the slave clock. The CSRO value gives the actual deviation in the frequency, which should be considered in the synchronization process. It is obvious from the results of the clock drift synchronization error that it equals 0 in case of no clock-drift or when both node clocks

are operating with the same frequency drifts as a result of the same frequencies.

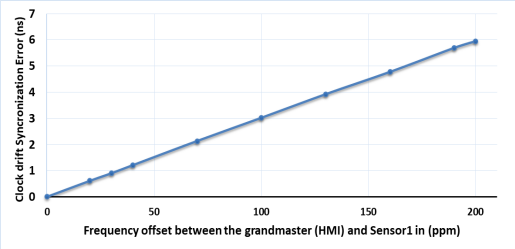


Fig. 5: Clock drift synchronization error between grandmaster (HMI) and sensor1 (fixed speed = 20 m/s) with variable frequency offsets in (ppm), uplink and downlink = 24 Mbps, asymmetric ratio = 1.

In the following scenarios, ppms of wireless/wired end-stations, wireless bridge and wired relay are fixed with 1000, 500 and 200 respectively. Fig. 6 shows the synchronization error that is produced between the grandmaster (HMI) and sensor1 at different mobile speeds in random directions. The speed of sensor1 affects the synchronization error because the transmission time for the synchronization messages varies depending on the speed, topology and direction of the node that receives the synchronization messages.

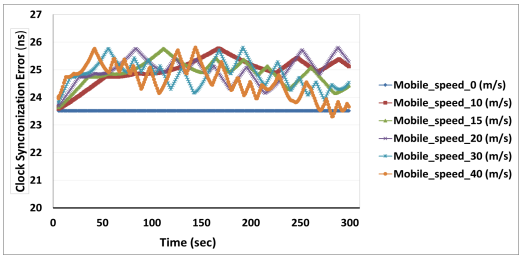


Fig. 6: Clock synchronization error between grandmaster (HMI) and sensor1 in different mobile speeds, uplink and downlink = 24 Mbps, asymmetric ratio = 1.

Fig. 7 shows the measured synchronization error in sensor1 according to the asymmetric data ratio between sensor1 and wireless bridge1. As shown, the synchronization error for the standard algorithm increased as the asymmetric ratio between the uplink and the downlink increased. In contrast, the proposed IEEE 802.1AS provides accurate synchronization for dynamically changing asymmetric wireless systems compared to the standard protocol. The proposed correction model shows that the average synchronization error is around 0.645 microseconds. This is due to the wireless data-rate awareness in the proposed correction model, which enables accurate path delay calculation for dynamic changes in uplinks and downlinks and dynamic mobile network topologies.

Fig. 8 shows the calculated synchronization error according to the number of intermediate hops between HMI and a selected time-aware station. The synchronization error for the standard and the proposed protocols increases as the number of hops increases. This is as expected, but the proposed correction model shows a smaller synchronization error. The proposed correction model could be utilized to minimize the upward increase in the synchronization error by selecting the best path through different paths between the grandmaster and the slave end-stations.

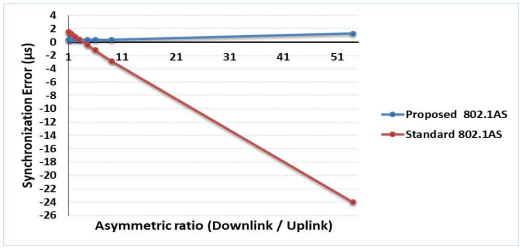


Fig. 7: The synchronization error for the standard and the proposed IEEE 802.1AS protocols with increasing the asymmetric ratios (downlink = 54 Mbps, uplink 1 - 54 Mbps) between sensor1 (speed = 20 m/s) and Wireless Bridge1.

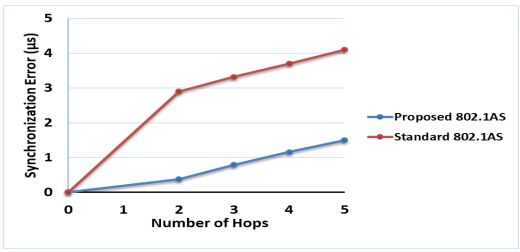
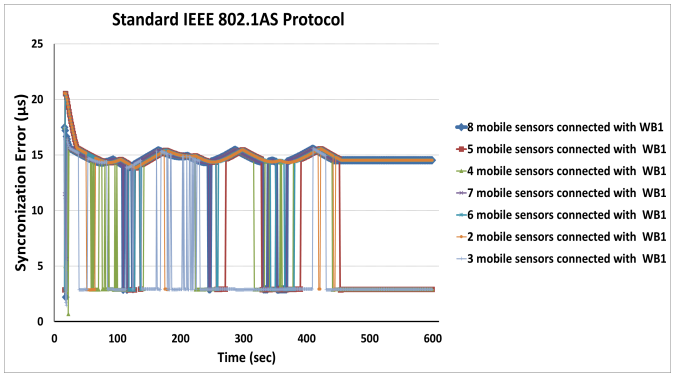
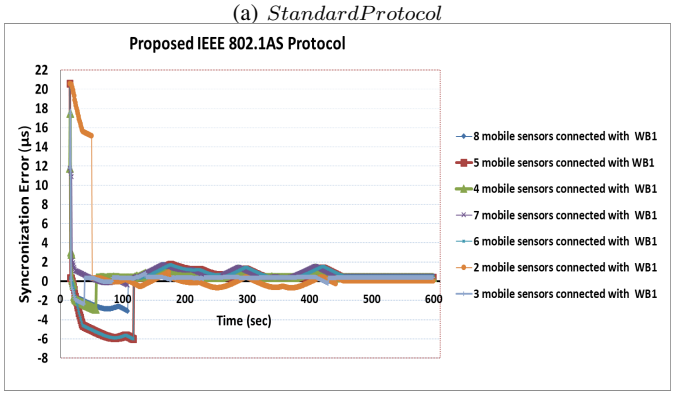


Fig. 8: The synchronization accuracy for the standard and the proposed IEEE 802.1AS protocols with increasing the number of hops between HMI and a selected Time-aware station, uplink and downlink = 24 Mbps, asymmetric ratio = 1.

Another scenario to evaluate the proposed protocol and comparing it with the standard protocol is illustrated in Fig. 9. It shows the synchronization error for sensor1 in case of increasing the number of mobile sensors that are connected with it to Wireless Bridge1. The actual synchronization precision in sensor1 is not correctly computed by the standard IEEE 802.1AS protocol. A significantly more accurate time synchronization is achieved by the proposed IEEE 802.1AS protocol by incorporating precisely calculated deterministic delays and considering the oscillation factor with the grandmaster, besides eliminating any case of asymmetric or outlier values in the path delay values. Fig. 9 (a) shows that in the standard protocol, the synchronization error stabilizes on the range (2.899 - 14.522) microseconds because of increasing the number of mobile sensors that are located in the same network, as a result of asymmetric behavior of the standard protocol. On the other hand, at the beginning of simulation, Fig. 9 (b) shows positive and negative synchronization errors, the reason is that the compensated time upon receiving the synchronization message at the end station is less or more than *op_sim_time* respectively. But over time, every end station receives more synchronization messages. Therefore, the compensated time is computed accurately and the synchronization errors stabilizes to be on the range (0.0558 - 0.5251) microseconds although the number of the mobile sensors increases.



(a) *StandardProtocol*



(b) *ProposedProtocol*

Fig. 9: The synchronization error for sensor1 by using the standard (a) and the proposed (b) protocols with variable number of sensors (range speed 0 - 30 m/s) connected with it to Wireless Bridge1 (WB1), uplink and downlink = 24 Mbps, asymmetric ratio = 1.

VII. CONCLUSIONS AND FUTURE WORK

An enhanced version of the standard IEEE 802.1AS time protocol has been presented. By computing the deterministic delays in dynamic TSN hybrid networks and considering the frequency drift between the slave and the grandmaster clocks, a formula to calculate accurate values for the link delay is derived. This synchronized time is used for scheduling and filtering in our simulation framework. In addition, the proposed protocol deals with the dynamic and asymmetric behavior of mobile hybrid networks by applying PDD filter, which is used to exclude the outlier behavior in the wireless networks as a result of the mobility and traffic congestion. As shown by the simulation results, the proposed protocol outperforms the standard IEEE 802.1AS and provides a significant gain in terms of synchronization precision. In the future work, further studies are planned in some research topics such as analyzing the random delays (i.e. the transmission jitter delays and the symbol timing synchronization error) and the channel access delay. In ad hoc networks, selecting the best path through different paths between the grandmaster and the slave clocks, or using multiple grandmaster clocks, can also be considered.

REFERENCES

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE industrial electronics magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[2] U. Okpeki, J. Egwaile, and F. Edeko, "Performance and comparative analysis of wired and wireless communication systems using local area network based on ieee 802.3 and ieee 802.11," *Journal of Applied Sciences and Environmental Management*, vol. 22, no. 11, pp. 1727–1731, 2018.

[3] J. H. Cho, H. Kim, S. Wang, J. Lee, H. Lee, S. Hwang, S. Cho, Y. Oh, and T.-J. Lee, "A novel method for providing precise time synchronization in a distributed control system using boundary clock," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 8, pp. 2824–2829, 2009.

[4] M.-g. LI and H.-n. SONG, "Research on computer clock synchronization technology [j]," *Acta Simulata Systematica Sinica*, vol. 4, 2002.

[5] J. Eidson and K. Lee, "Ieee 1588 standard for a precision clock synchronization protocol for networked measurement and control systems," in *Sensors for Industry Conference, 2002. 2nd ISA/IEEE*. Ieee, 2002, pp. 98–105.

[6] J. Jasperneite, K. Shehab, and K. Weber, "Enhancements to the time synchronization standard ieee-1588 for a system of cascaded bridges," in *IEEE International Workshop on Factory Communication Systems*. Citeseer, 2004, pp. 239–244.

[7] S. Lee, "An enhanced ieee 1588 time synchronization algorithm for asymmetric communication link using block burst transmission," *IEEE communications letters*, vol. 12, no. 9, pp. 687–689, 2008.

[8] Institute of Electrical and Electronics Engineers, Inc., "802.1asrev - timing and synchronization for time-sensitive application in time-sensitive networking task group," IEEE, 2017, http://www.ieee802.org/1/pages/802.1AS-rev.html, Last accessed on 2020-03-16.

[9] W. Chen, J. Sun, L. Zhang, X. Liu, and L. Hong, "An implementation of ieee 1588 protocol for ieee 802.11 wlan," *Wireless networks*, vol. 21, no. 6, pp. 2069–2085, 2015.

[10] F. Cristian, "Probabilistic clock synchronization," *Distributed computing*, vol. 3, no. 3, pp. 146–158, 1989.

[11] Institute of electrical and electronics engineers, time sensitive networking, "Time-sensitive networking task group," IEEE, 2017, http://www.ieee802.org/1/pages/tsn.html, Last accessed on 2020-03-16.

[12] G. von Zengen, K. Garlichs, Y. Schrcöder, and L. C. Wolf, "A submicrosecond clock synchronization protocol for wireless industrial monitoring and control networks," in *2017 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2017, pp. 1266–1270.

[13] A. Elsts, S. Duquennoy, X. Fafoutis, G. Oikonomou, R. Piechocki, and I. Craddock, "Microsecond-accuracy time synchronization using the ieee 802.15. 4 tsch protocol," in *2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops)*. IEEE, 2016, pp. 156–164.

[14] D. Shrestha, Z. Pang, and D. Dzung, "Precise clock synchronization in high performance wireless communication for time sensitive networking," *IEEE Access*, vol. 6, pp. 8944–8953, 2018.

[15] D. K. Lam, K. Yamaguchi, Y. Nagao, M. Kurosaki, and H. Ochi, "An improved precision time protocol for industrial wlan communication systems," in *2016 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2016, pp. 824–829.

[16] A. Swidan, H. B. Abdelghany, R. Saifan, and Z. Zilic, "Mobility and direction aware ad-hoc on demand distance vector routing protocol," *Procedia Computer Science*, vol. 94, pp. 49–56, 2016.

[17] A. Mildner, "Time sensitive networking for wireless networks-a state of the art analysis," *Network*, vol. 33, 2019.

[18] M. Pahlevan, B. Balakrishna, and R. Obermaisser, "Simulation framework for clock synchronization in time sensitive networking," in *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 2019, pp. 213–220.

[19] Y. Liu, K. Xu, and M. Wei, "A study on mems oscillators 'frequency drift of temperature," in *MATEC Web of Conferences*, vol. 189. EDP Sciences, 2018, p. 11002.

[20] C. W. Nicholls and P. Wu, "Method and system for correcting oscillator frequency drift," Mar. 18 2014, uS Patent 8,674,778.

[21] 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)*. IEEE, 2018, pp. 1–7.

[22] A. S. Sethi and V. Y. Hnatyshin, *The practical OPNET user guide for computer network simulation*. Chapman and Hall/CRC, 2012.