Modeling Time Synchronization in WLANs in OMNeT++

Anas Bin Muslim, Carolin Christoph, Ralf Tönjes
Faculty of Engineering and Computer Sciences
University of Applied Sciences Osnabrueck
Osnabrueck, Germany
{a.bin-muslim, carolin.christoph, r.toenjes}@hs-osnabrueck.de

***Abstract*—This paper presents a framework for OMNeT++ which includes time synchronization model for WLANs. Synchronization is based on the Generalized Precision Time Protocol (gPTP) standard, which aims to achieve an accuracy of less than 100 nanoseconds. The presented model is developed and implemented in OMNeT++, a discrete event network simulator, using its INET library. A new type of WLAN node is modeled which supports time synchronization at the Link layer. A clock module for WLAN nodes is also modeled which implements variable clock drift to simulate noise interference in clock frequency oscillators. Simulations with our WLAN nodes are done and the results show that using gPTP based time synchronization in wireless networks, accuracy of $\pm$3ns can be achieved.**

***Index Terms*—time synchronization, wlan, gPTP, tsn, OMNeT++**

I. Introduction

With the IEEE 802.11ax [4] standard approved in 2021, work on the 802.11be amendment is already ongoing. Although an initial draft of 802.11be was planned to be presented in March 2021, the work will continue until its approval, which is planned for 2024. A major part of 802.11be is to have Time-Sensitive Networking (TSN) integrated which must provide inter-working capabilities with Ethernet-based TSN and 5G-TSN.

There are two types of networks, one that can tolerate higher latency and one that can't. The first type of network is called Delay Tolerant Network (DTN) while the latter can be called a real-time network. Real-time networks consists of real-time systems which are required to complete their tasks within a specific time for the whole network to function properly. Depending on the system, tasks can be a-periodic or periodic such as moving a robotic arm to a specified position when a command is received or sending a command repetitively after a specific amount of time.

In addition to types of tasks, each system can have two types of deadlines, i.e. hard and soft deadlines. A soft deadline, if missed, only compromises the performance of that specific system module, while a hard deadline, if missed, can be catastrophic for the whole system. To avoid such issues and system failures, the clock time of real-time systems must be synchronized. In the case of wireless nodes, such time synchronization becomes difficult due to mobility, packet loss, and low Signal to Interference and Noise Ratio (SINR).

In this paper, we introduce a time synchronization model based on IEEE 802.1AS [2] [3] Generalized Precision Time Protocol (gPTP) for WLANs with updated clock model using variable drift, developed in OMNeT++[1], a discrete event simulator. The IEEE 802.1AS standard defines methods for time synchronization not only in Ethernet-based networks but also in wireless networks. The developed model considers the propagation delay and synchronizes the clocks of slave nodes with their masters.

The rest of this paper is divided into 5 sections. Section II provides an overview of generalized Precision Time Protocol and discusses the requirements of the time synchronization process in the WLAN networks. Section III discusses related work. Section IV derives the model for time synchronization and section V presents the simulation scenario along with its results. Finally, section VI mentions the future works.

II. Generalized Precision Time Protocol

Generalized Precision Time Protocol is a profile of the Precision Time Protocol (PTP) defined in IEEE 1588-2008 [1], also referred to as PTP version 2.0. gPTP adds timing features and improves the timing accuracy in the base Precision Time Protocol profile. It works on the Master-Slave principle, which means all slaves under the same master synchronize their clocks with the master's clock. In a multi-hop network, a primary master is selected as Grand Master (GM) and its slave nodes further synchronize their slave nodes and thus, are referred to as Bridge nodes. The hierarchy of such network is shown in figure 1. Thus, three types of nodes exist in the gPTP domain:

- Grand Master (GM) - There exist only one GM per gPTP domain. The GM sends the time synchronization messages in the network.
- Slave - Multiple slaves can exist in a gPTP domain at the same time.
- Bridge - Multiple bridges can exist in a gPTP domain. A bridge is a slave to solely one master and can be a master to multiple slaves.

European Regional Development Funds (EFRE) Project AgraNet (project number: ZW-85018457)

[1] www.omnetpp.org

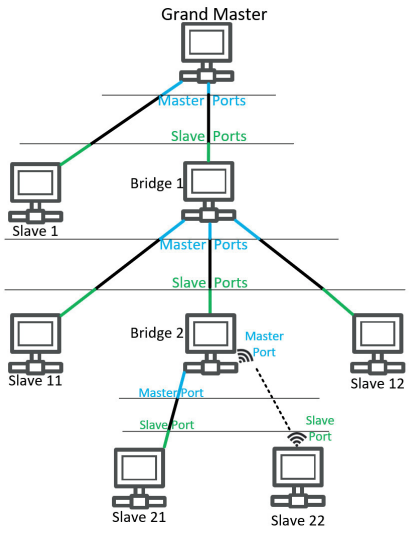

Fig. 1. gPTP network topology

Selection of the Grand Master is done by the Best Master Clock Algorithm (BMCA) [2]. Multiple clock properties are considered for the Grand Master selection. But since the focus of this paper is the time synchronization aspect of gPTP in WLANs, the BMCA is not discussed here. In our implementation of Time synchronization in WLANs, nodes are pre-assigned the roles of (grand-)master or slave, thus eliminating the need of BMCA.

Each of the master, slave, and bridge nodes in a time-aware system has a specific type of port that performs the time synchronization functionality. The GM has a master port while slaves have slave ports. A slave can have multiple slave ports but at any given time, it can only synchronize to a single master. Bridge nodes on the other hand have both slave and master ports. These ports are always in one of the following states:

- Master port - the port that sends time synchronization information to slaves and bridge nodes.
- Slave port - a port that receives time synchronization information from the master node and synchronizes the node clock with the master's clock.
- Passive port - a port not capable of time synchronization via gPTP.

Once gPTP based synchronization starts, slave nodes request their master via sending timing measurement request, to which the master replies with the acknowledgment. Acknowledgment sent by the master contains an origin timestamp on which slaves must synchronize. Further information is needed to complete the synchronization, which is discussed in the following section.

A. Requirements for Time Synchronization in Wireless Networks

For the clock time correction, various types of information are needed. This includes the propagation delay information, the rate ratio (r) information, and the Correction Field (CF) information. The propagation delay is the time that a packet takes to transfer from master to its slave after it is transmitted over air. It takes two round trip times to compute the propagation delay for time synchronization in WLAN. The difference in clock oscillator frequencies of master and slaves also needs to be considered. This is taken care of via the rate ratio which is the ratio of the oscillator frequencies of the synchronization requester and the responder. Finally, the CF is required if the slave computing the time difference is synchronizing to a bridge. The CF is the corrected information that every bridge node computes and shares with its slaves. It represents the time differences of the bridge node to its master so that the slave nodes can also consider this difference while adjusting their clock.

In addition to the propagation delay, CF, and rate ratio, mobility speed, SINR, Bit Error Rate (BER), etc. can also affect the time synchronization process. They directly affect the synchronization process when sync or ack packets are lost. In such cases, slaves have to wait for the next successful consecutive synchronization packets to synchronize with the master.

1) Rate Ratio: Time-aware systems use hardware clocks whose properties may vary, e.g. precision, error rate, and deviation of the oscillator, which results in a variable frequency, within a specific range for that oscillator. To have a consistent time base between the nodes being synchronized, a rate ratio is used. Rate ratio is the ratio of the frequency of the master clock to the frequency of the slave clock, as per [3]. Its equation is mentioned in eq. 1. The rate ratio brings both, the slave and the master, on a common frequency base for propagation delay computation.

$$r = \frac{f_{requester}}{f_{responder}} \tag{1}$$

With the rate ratio, it can be pointed out whether the slave clock is lagging behind the master clock or precedes the master clock, but it does not tell which clock runs correctly. That decision is taken by Best Master Clock Algorithm (BMCA) when the master node is chosen.

From equation 1, it is seen that

- if $r > 1$, slave clock precedes the master clock.
- if $r < 1$, slave clock lags behind the master clock.
- if $r = 1$, master and slave clocks are at same frequency.

In our clock model, we have not yet fully implemented the oscillator frequency model, therefore we use an alternative method for rate ratio computation. Since the interval between two consecutive measurement request frames (at slave) is not always equal to the interval between their reception times at the master, thus this difference in the intervals can be used to compute the rate ratio between slave and master as shown in eq. 2. Similarly, the rate ratio can also be computed via acknowledgments sent by the master.

$$r = \frac{t_1^{'} - t_1}{t_2^{'} - t_2} = \frac{t_3^{'} - t_3}{t_4^{'} - t_4} \tag{2}$$

ISBN 978-3-8007-5873-9

2) Propagation Delay: Propagation delay, as mentioned before, is the propagation time of a packet from master to slave or vice versa. In the wireless networks with mobile nodes, propagation delay can vary and become important to factor in, for clock corrections. The propagation delay measurement in IEEE 802.11 links is accomplished via round trip frame exchange. A timing measurement request is initiated by the requester (slave) and is sent to the responder (master) time-aware system, which then responds with an acknowledgment. In this exchange, both time-aware systems capture 4 timestamps.

(a) The timing measurement requesting station generates a measurement frame and sends it to the requester on a wireless link at time t_1 and records t_1.
(b) The responding station receives the timing measurement frame and records the frame reception time t_2.
(c) The responding station generates an acknowledgment frame and sends it to the requester at time t_3 and records t_3.
(d) The requesting station receives the acknowledgment frame and records the frame reception timestamp t_4.

In each measurement frame exchange, these 4 timestamp are captured and the responder (master) shares its captured timestamp with the requester (slave) in the next timing frame exchange. Thus, after the second timing measurement exchange, the requester has all four of the timestamps from the last timing measurement exchange and is capable of measuring the precise propagation delay using equation 3. This can be seen in figure 2. After the first Round Trip (RTT), only 3 timestamps are available at the slave hence P_{delay} cannot be calculated. After another RTT t_3 also becomes available to the slave node thus P_{delay} is calculated after 2*RTTs for the first time. Afterward, at each received ack from the master node, a new P_{delay} can be calculated.

$$P_{delay} = \frac{r \cdot (t_4 - t_1) - (t_3 - t_2)}{2} \tag{3}$$

Figure 2 shows the timing measurement and acknowledgment frames exchange between a master and slave. It also shows what information is available at the slave at any given time and how it is used for the propagation delay calculations.

3) Correction Field: In a multi-hop network, the nodes that are not directly connected to the GM, must be able to synchronize with the GM via their local masters, i.e. Bridges. For such nodes to synchronize, their bridges compute correction fields and share them with their slaves. The CF is the adjusted time of the bridge node. Once the slave of the bridge knows the CF, it can synchronize with the master. CF is calculated via equation 4.

$$CF(i) = CF(i-1) + P_{delay} + T_{residence} + T_{transmission} \tag{4}$$

4) Residence Time: The time that a frame spends waiting in queues or being processed, after being received by node and before it is processed in synchronization method is called Residence Time. Queuing delay is also considered in the residence time. Once the frame is received at physical layer, it goes through various processes before it is processed by time synchronization functions. It is necessary to include this time in the synchronization process.

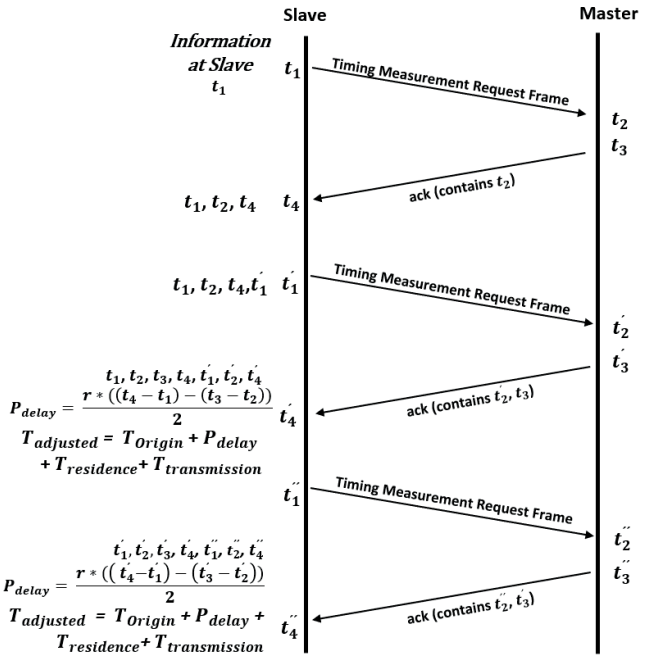

Fig. 2. Propagation delay measurement sequence

5) Transmission Delay: Transmission delay is the time that a measurement request needs to be transmitted on the wireless medium. It is directly related to the available bitrate. In our model, transmission delay is calculated by diving the number of bits in the frame by the bitrate at which the frame is sent.

B. Time Synchronization

In addition to rate ratio, propagation delay, and correction field, residence time and transmission time of the timing measurement packets are also used in synchronization. The equation that is used in our model for time synchronization is

$$T_{synced} = T_{origin} + CF + P_{delay} + T_{residence} + T_{transmission} \tag{5}$$

For slaves and bridges directly connected to Grand Master, CF in equation 5 becomes zero but for all other nodes, CF is computed and sent by their master and is then used in the time synchronization process.

These various delays are shown in figure 3.

III. Related Work

One of the main starting points for investigating time synchronization for any type of time-aware system is the selection of synchronization procedures. [7] takes a look at various synchronization procedures and then presents a PTP variation for time synchronization. They mention

ISBN 978-3-8007-5873-9

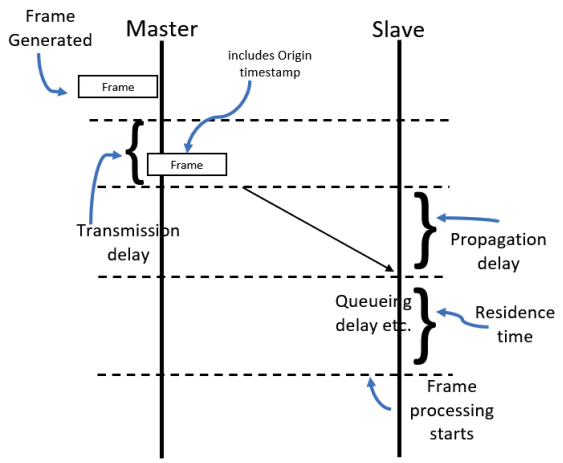

Fig. 3. Time differentiation in Sync process

- Global Positioning System (GPS) (which uses signals from four different satellites and uses trilateration to calculate the exact position and time of the device).
- Network Time Protocol (NTP) (which uses public internet to synchronize devices to UTC with the accuracy of less than one millisecond within LANs and accuracy of tens of milliseconds over public internet).
- Reference Broadcast Time Synchronization (RBS) and Reference Broadcast Infrastructure Synchronization (RBIS) (where both uses synchronization packets and acknowledgments for WLANs . RBIS works for AP based WLANs while RBS works in Ad-Hoc networks, in similar fashion to gPTP).
- Precision Time Protocol (PTP) (which is related with IEEE 1588 standard [1], IEEE 802.1AS standard [2], and its revision in [3]).

In [6], a Wireless Precision Time Protocol is proposed, which is an extension to PTP. It is implemented and tested in Cooja Simulator on Contiki Operating System[2] nodes. Authors in [5] present an IEEE 1588 styled time synchronization for Wireless Links but the research is focused on the network embedded systems and using Kalman filter along with PI controller for clock frequency stability. Their implementation is hardware-based and shows promising results. In [9] authors present a PTP model for OMNeT++ simulator which is based only on Ethernet and no model for WLAN time synchronization is provided. Another PTP implementation is presented in [13], but the authors focus on modeling clock and estimating clock noise using Power Law Noise (PLN). While their clock noise model is of interest to us and we integrate their PLN model in our implementation, their framework is still missing time synchronization in WLANs. The Open-source framework CORE4INET, which implements real-time Ethernet and time synchronization, is presented in [12], but it does not provide synchronization functionalities for WLANs. Another IEEE 802.1AS clock synchronization implementation is presented in [11], which is gPTP based but also implements only Ethernet-based synchronization. The clock model in their framework is minimal and implements constant clock drift. In our implementation, we take this clock model and update it with PLN based variable clock noise. In our future developments, we will integrate this Ethernet-based time synchronization model with our WLAN time synchronization framework. Authors of [8] present a real-time Ethernet synchronization but they focus on time-aware shaping. None of these papers present a comprehensive time synchronization framework for WLANs and network simulators like ns-3 or OMNeT++ and also do not list any of such frameworks either. Thus in this paper, we present a time synchronization model and a framework for WLANs in OMNeT++ simulator.

[2]www.contiki-os.org

IV. Modeling Time Synchronization for WLANs

Time synchronization modeling requires modeling a gPTP capable MAC layer, synchronization capable nodes, and a clock with clock drift functionality. Furthermore, the requester node must be able to send the timing measurement frames and process the acknowledgments. The responder node has to process the timing measurement frame and send timestamps along with the origin timestamp to the requester via ack frames.

OMNeT++ is a discrete event network simulator that is used to implement, simulate and test our time synchronization model for wireless time-aware systems. The gPTP standard explained in the previous sections is implemented using the INET framework[3] of OMNeT++. Some of the requirements and assumptions that are made in this modeling are as follows:

- BMCA for choosing the grand-master in the network is not implemented, as it is out of the scope of this paper. Instead, nodes are pre-configured as master or slaves
- The base clock model that we use is taken from [11]. It is then updated with variable clock noise (using PLN) to have realistic clock behavior.
- INET framework for OMNeT++ is used. Node models, OSI layers, and all other models e.g. propagation are used from INET. Our implementation extends the MLME layer for wireless nodes to add gPTP based time synchronization.

gPTP synchronization is performed at the Link layer. All timestamps are captured as soon as timing measurement frames are received at the MAC layer or are transmitted to the physical layer. The timing measurement frames and their acknowledgments are generated and consumed by the MLME or management entity at the MAC layer of the IEEE 802.11 node. Since the timing synchronization and ack frames are generated by the management module at the Link layer, therefore these frames get the highest priority due to being management frames. This results in data rate for application data being dropped and management frames being prioritized in case the network is throttling, during a simulation.

[3]inet.omnetpp.org

ISBN 978-3-8007-5873-9

Various models are created to implement the timing synchronization functionality. These models are:

- The Clock model - This model keeps track of the node's local time. It implements variable drift and responds with drift time when queried. When a timing measurement frame is received, the local clock's time is adjusted according to the master's clock.
- The MLME model with gPTP - This MAC layer management module implements gPTP functionality. It computes the rate ratio, propagation delay, and sends updates to the clock module when clock time must be adjusted.
- A model that interfaces slave and master ports in the network - This module is needed for the bridge nodes. When a bridge node synchronizes its local clock to its master, it must transmit this information to its slave nodes too. Information to be shared with slaves is stored in this module.
- Time aware network node called *GPTP Wireless Host* - gPTP wireless host contains the above-mentioned modules and can take the role of master or slave depending on the user-defined configuration.
- Wireless access point from INET framework is also updated and gPTP capable management module is added at its Link layer.

gPTP capable wireless nodes consist of several simple and compound modules and their structure is shown in figure 4. Modules highlighted in red are added in INET based wireless hosts. Link layer of INET wireless host is present inside WLAN compound module, therefore it is updated there.

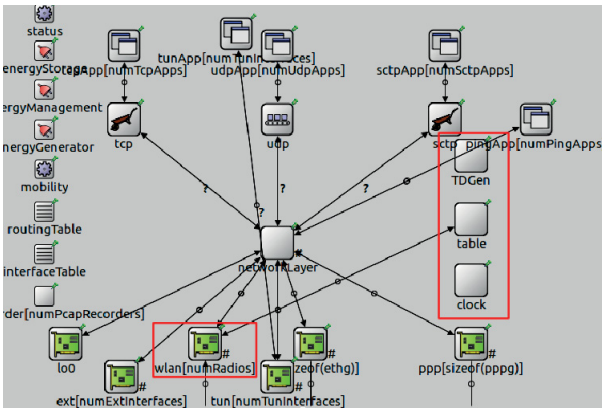

Fig. 4. gPTP node structure

V. Simulation and Results

A simulation scenario is created using gPTP wireless hosts. This scenario is simulated with 2, 5, 10, and 15 gPTP slave nodes. The number of nodes does not affect simulation primarily because OMNeT++ is a discrete event simulator, therefore, whenever a frame or packet is being processed, the simulation time halts until the processing is done.

One of the nodes is configured as a master and the rest of the nodes as slaves. The simulation area is set to be 100 x 100m. Simulation modules necessary for wireless communication are added. These consist of a configurator and a radio medium module. A random way-point mobility model is used in the simulation with wait times of 0 to 5 sec. The mobility speed of the nodes is set via uniform distribution between 1 and 3 meters per second. Simulation time resolution is set to 1 ns. IEEE 802.11 NIST error model is used as error model and, for propagation model constant speed propagation is used; both from INET framework. An Isotropic antenna is specified for all nodes.

Each timing measurement request is sent after 125ms. IEEE 802.11g standard is used. Variable clock drift is configured for the slaves. The Master's clock is assumed to be perfect therefore no drift is added to it. Even if we would introduce drift in the master's clock, all the slaves try to synchronize with the master, therefore it doesn't make a difference if drift is added to the master or not. Thus the master's clock runs at simulation time while slave clocks have the continuous variable drift.

With these parameters, the simulations are run and slaves synchronize with the master for the first time after two RTTs. After that, slaves can synchronize with their master at each acknowledgment frame until the timing measurement request or ack frame is lost. In that case, two-timing measurement requests and acks are needed to synchronize again.

Simulations with 2, 5, 10 and 15 nodes are done. The clock drift that our clock model generates in simulation with 2 nodes is shown in figure 5. Since the drift is negative this means that both slave clocks lag behind the master clock and try to synchronize with the master at every synchronization interval. Furthermore, as the drift is variable, it varies between 0ns-2ns. From the same figure, it can be seen that just after 10 seconds of simulation, if clock correction is not done via synchronization, node 1 will be about 80ns and node 2 will be about 77ns behind the master's clock respectively.

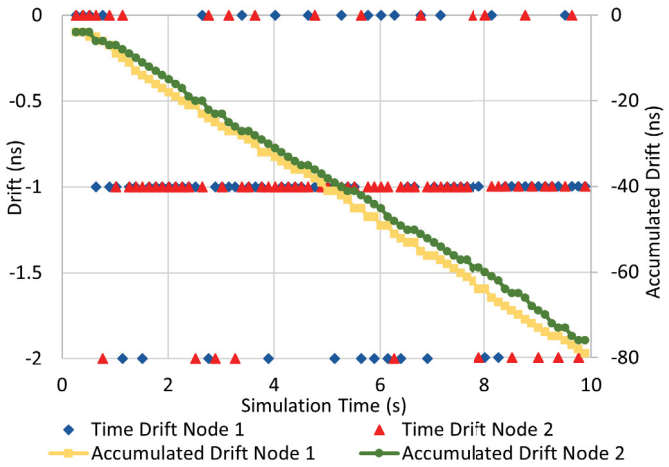

Fig. 5. Slave clock drifts vs. Simulation time

In figure 6, the slave's clock difference to the master's clock is shown just before and immediately after the synchronization

takes place. An important point to note here is that the time difference after synchronization is in nanoseconds, while before synchronization it is in milliseconds.

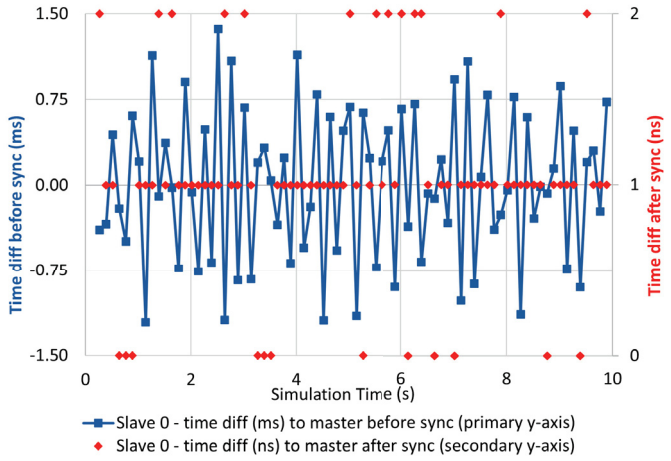

Fig. 6. Slave's clock diff to master's clock before and after the synchronization

Lastly, in figure 7, it is shown that for all simulations the time difference between slave and master's clock remains between ± 2 ns after synchronization with only a few outliers at +3 ns and -3 ns.

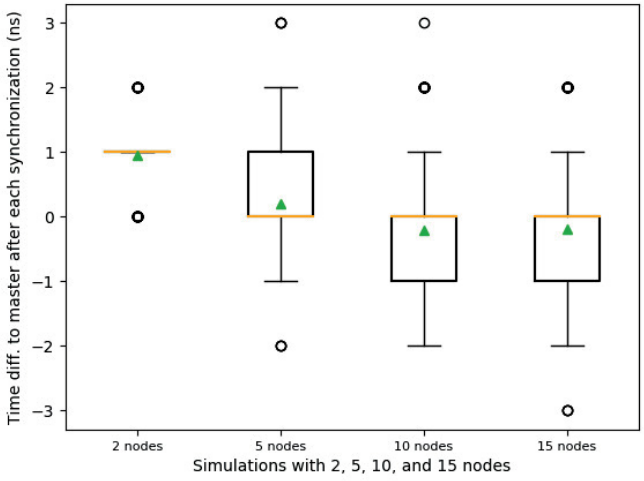

Fig. 7. All slaves time difference to their masters

VI. Conclusion and Future Works

Clock with variable drift using PLN noise and time synchronization in gPTP capable WLAN nodes is presented in this paper. Each gPTP capable node is equipped with clock module which has variable drift. This results in the node's clock drifting ahead or lagging behind the Master's clock even after synchronization takes place. Thus, we use modified gPTP algorithm to synchronize all the nodes with Master after every 125 milliseconds. Modifications are made to gPTP algorithm for it to function properly with WLAN nodes. The original concept of gPTP algorithm remains the same. With gPTP capable WLAN nodes, simulations are done using 2, 5, 10, and 15 nodes. Results of each simulation shows that synchronization between Master and Slave nodes can be achieved successfully using our WLAN gPTP capable Framework. This framework is first of the kind which combines gPTP and WLAN, and it can be used to develop and simulate TSN protocols over wireless networks.

We plan to add further parameters, e.g. temperature, movement, etc., and clock noise variation dependency on those parameters to model realistic clock behavior. Integrating Ethernet based gPTP nodes in this framework hybrid network simulations is also planned as future work. The final goal is to have an OMNeT++ based framework that is capable of simulating both Ethernet and WLAN based TSN protocol stack.

References

[1] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, in IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), vol., no., pp.1-269, 24 July 2008, doi: 10.1109/IEEE STD.2008.4579760.

[2] IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks, in IEEE Std 802.1AS-2011, vol., no., pp.1-292, 30 March 2011, doi: 10.1109/IEEE STD.2011.5741898.

[3] IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications, in IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011), vol., no., pp.1-421, 19 June 2020, doi: 10.1109/IEEE STD.2020.9121845.

[4] IEEE Approved Draft Standard for Information technology– Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN, in IEEE P802.11ax/D8.0, October 2020 (approved draft), vol., no., pp.1-820, 10 Feb. 2021.

[5] H. Abubakari and S. Sastry, "IEEE 1588 style synchronization over wireless link," 2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, 2008, pp. 127-130, doi: 10.1109/ISPCS.2008.4659226.

[6] A. Garg, A. Yadav, A. Sikora, and A. S. Sairam, "Wireless Precision Time Protocol," in IEEE Communications Letters, vol. 22, no. 4, pp. 812-815, April 2018, doi: 10.1109/LCOMM.2017.2781706.

[7] D. Krummacker et al., "Intra-Network Clock Synchronization for Wireless Networks: From State of the Art Systems to an Improved Solution," 2020 2nd International Conference on Computer Communication and the Internet (ICCCI), 2020, pp. 36-44, doi: 10.1109/ICCCI49374.2020.9145977.

[8] C. Le and D. Qiao, "Evaluation of Real-Time Ethernet with Time Synchronization and Time-Aware Shaper Using OMNeT++," 2019 IEEE 2nd International Conference on Electronics Technology (ICET), 2019, pp. 70-73, doi: 10.1109/ELTECH.2019.8839517.

[9] Martin Lévesque and David Tipper. 2015. "ptp++: A Precision Time Protocol Simulation Model for OMNeT++ / INET".

[10] Christoph Mayer and Thomas Gamer. 2008. "Integrating Real World Applications into OMNeT++".

[11] Henning Puttnies, Peter Danielis, Enkhtuvshin Janchivnyambuu, and Dirk Timmermann. 2018. "A Simulation Model of IEEE 802.1AS gPTP for Clock Synchronization in OMNeT++." In Proceedings of the 5th International OMNeT++ Community Summit (EPiC Series in Computing, Vol. 56).

[12] Till Steinbach, Hermand Kenfack, Franz Korf, and Thomas Schmidt. 2011. An "Extension of the OMNeT++ INET Framework for Simulating Real-time Ethernet with High Accuracy".

[13] Wolfgang Wallner. 2016. "Simulation of the IEEE 1588 Precision Time Protocol in OMNeT++".