Available online at http://scik.org

J. Math. Comput. Sci. 11 (2021), No. 1, 109-124

https://doi.org/10.28919/jmcs/5072

ISSN: 1927-5307

A NOVEL PRECISE AND ACCURATE CLOCK SYNCHRONIZATION ALGORITHM

PHURAILATPAM DEVAKINANDAN SHARMA[1,*], KANGUJAM PRIYOKUMAR SINGH[1,2]

[1]Department of Mathematical Sciences, Bodoland University, Kokrajha-783370, Assam, India

[2]Department of Mathematics, Manipur University, Canchipur, Imphal-795003, India

Copyright © 2021 the author(s). This is an open access article distributed under the Creative Commons Attribution License,which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract: Execution of a task in a synchronized manner in a distributed setting, one needs the common idea of time. To accomplish this task, clock synchronization is conceptualized. Clock drifting is an intrinsic property of a clock, which necessitates the synchronization and resynchronization. A network consists of nodes, which may behave appropriately or behave badly. The mischievous node may pose difficulty in clock synchronization. We require a method to overcome the effect of mischievous nodes. Synchronization can be achieved in two ways namely accuracy and precision. In this paper, we present an accurate weighted average clock synchronization algorithm to execute a coordinated activity in a fault-tolerant manner. We use trusted clock value from GPS/GLONASS/IRNSS as a reference clock to achieve better accuracy. We also use behavior of the node to compute normalized weight in a localized manner, which lies between unit intervals. This weight assignment enables us to contain the effect of misbehavior up to some extent. This algorithm offers better accuracy and precision while tolerating mischievous nodes. The upper bound of tolerance boundary is one-third of the network size.

Keywords: network, accuracy; precision; IEEE 1588; synchronization; tolerant; mischievous; window; weighted average.

2010 AMS Subject Classification: 68M10.

*Corresponding author

E-mail address: devakinandan.bu@gmail.com

Received September 29, 2020

1. INTRODUCTION

A distributed network consists of spatially scattered diverse nodes. These nodes are essential to carry out cooperative functions in close synchronization to one another. This is achievable only when they share some familiar idea of time. This ensures chronological precedence order across the network. In the contemporary world, with proliferation of internet and development of information technology, many facets of our life are integrated over the web. Internet of Things (IoT) is the trend which integrate all our smart devices over the web so as some major aspects of our life [1]. Our smart devices collect various data and share information over the network for better management. All of these systems are able to function only they are synchronized over the network. The clock synchronization process will help in faithful collection of data and its transmission [2]. IoT seldom works in isolation. Many synchronization algorithms use GPS/GLONASS/similar trusted clock signal. However, there are certain applications in which such trusted clock signal cannot be used for certain reasons. Some of these applications may use these clock signal for some time and as per design may not use at other times. We have designed an algorithm in which there is an option to use the trusted clock signal whenever necessary. This will help a network to work in isolation without any contact with outside world, if there should exist such a requirement.

2. RELATED WORK

Time and message format for synchronization are defined in IEEE 1588 standards [3][4]. In the IEEE standards a grand Master node propagates periodically synchronization information to the slave nodes over the boundary clock and the transparent clocks. The synchronization information may also be carried over either by boundary clock or transparent clock only. There are multiple Precision Time Protocol (PTP) ports in a boundary clock. The boundary clock can also act as master clock which can synchronize other clocks, as slave clocks. In such a scheme the grand master will first synchronized its neighbouring slave clocks and then these slaves in turn will

synchronize their neighbouring slave clock. Here the slave clocks performed the role of a master clocks.

Over a period of time, many researchers had suggested various precise clock synchronization algorithm which are IEEE 1588 compliant. Some of them had also demonstrated their algorithm [5] – [9]. These suggested algorithms mainly cater for single hop network. Ability of these algorithms working in multi hop distributed network is suspect since for such multi hop environment, the requirement for high accuracy is mandatory. The clock synchronization error increases as number of hops increases in a distributed network. This is a major limitation in boundary clock synchronization scheme. Peak – to – peak clock synchronization error increases to 0.8, 08, 40 & 300 microsec for 01, 03, 05 & 10 hops respectively for such scheme. Simulation for the same is reported in [10].

Many researchers had proposed various clock synchronization algorithm catering for various requirement including that of IoT [11] – [16]. Maintaining time records of information sent and received is a basic function of synchronization algorithm. Timestamp for each for such bits of information is always enclosed with the information sent/received. The slave clock is able to synchronize by calculating various clock parameters like clock offset, compensate time etc. A basic protocol is under numerated:

a) Master clock forwards a Sync message to the neighbouring slave clocks and it records the transmit time t_1.

b) On receiving the message, this slave clock records the reception time as t_2.

c) The master clock may also immediately forwards another sync message carrying t_1.

d) The slave clock, then, forwards a message requesting to forward the delay in transmission to the master clock.

e) The slave clock, then, will initiate the calculation of the reverse transmit delay and records the transmit time t_3.

f) On receiving this message, master clock records the received time as t_4.

g) Now on receiving the message requesting delay, the master clock forwards back a message carrying t_4 as response to the request.

Now the slave clock has 04 time stamps t_1, t_2, t_3 & t_4. Utilizing the time records we can calculate the total round trip delay between the master and slave clocks as $[(t_2 - t_1) + (t_4 - t_3)]$. As the network is symmetric, the one-way delay between the master and slave clocks is $[(t_2 - t_1) + (t_4 - t_3)]/2$. Therefore, the clock deviation of the slave clock from the master clock is:

$$\text{offset} = (t_2 - t_1) - [(t_2 - t_1) + (t_4 - t_3)]/2 = [(t_2 - t_1) - (t_4 - t_3)]/2$$

Malicious nodes in the network may however use these timestamp to exploit the system. The problem become more acute if the slave nodes require to go through multiple hops for synchronization with the master node. W. Dong [16] has suggested a novel graph theoretical algorithm to counter the effect of malicious behaviour. In the algorithm malicious behaviour is detected at the message level only. This approach is however computing resource exhaustive. Hence, there is a requirement of an approach to counter malicious behaviour while ensuring resource efficient clock synchronization.

3. OUR PROPOSED CLOCK SYNCHRONIZATION SCHEME

Clock synchronization in a distributed system has been exhaustively deliberate in the previous few decades. Many synchronization algorithms used averaging method as convergence function with varying degrees of competence and complexities. Some of the important algorithm are already discussed in the section above.

Here we are giving some essential definitions, which help to build up our system model and synchronization algorithm.

Definition 1. Correct Behavior: Correct Behavior in context of distributed network is the performance of a variety of function by the network component as expected according to the design or algorithm.

Definition 2. Malfunction: A malfunction is a deviation from a correct behavior. Any component or subsystem, which deviates from their correct behavior, is considered to be failed.

Definition 3. Faulty Component: Any component, which fails, is faulty.

Definition 4. Mischievous Behavior: Mischievous behavior is an arbitrary fault that occurs during the execution of an algorithm by a distributed system. It is a serious type of malfunction, which gives inaccurate, untimely and conflicting information. Mischievous behavior includes 'dual-faced' clocks, which give different time to different nodes at the same real time.

3.1. System model and problem statement

In this section we give a thorough explanation of the system model. We also describe the problem statement of the paper subsequently.

3.1.1 System Model

The system model is explained in two parts; at first we describe the network architecture and then the Malfunction model of the network.

3.1.2 Network Architecture

We consider a connected distributed static network which has

three levels of hierarchies. As shown in the Fig. 1, the layers are Reference layer (RL), Pseudo reference layer (PRL) and Non reference layer (NRL). In the reference layer there are one or more than one Master Node (MN) which receives reference clock value for outside trusted sources such as GPS/GLONASS/ IRNSS/Galileo etc. The MN can utilize this clock value for clock synchronization process for rest of the network. However, these MN can also switch off receiving this clock value. The MN in the reference layer are connected to Pseudo Master Nodes (PMN) in the Pseudo reference layer. The number of MN is much fewer than PMN. There are at least two PMN in the Pseudo layer and any MN is connected to the at least two or more PMN. All PMN are fully connected and they are much fewer than the number of nodes. PMN in the pseudo reference are connected to Nodes in the non-reference. Here any node is connected to at least one PMN and the nodes are generally connected to their neighboring nodes but not necessarily

Each of the MN, PMN and Nodes has a local physical clock from where the logical clock of the node is derived. We assumed that the physical clocks run continuously with respect to the real time. This physical clock can drift at rate ρ apart from the real time due to factors like temperature, pressure or aging. We make the following assumptions in our paper:

Drift-Bound: The Physical Clock for any node n_i , $p_{n_i}(t)$ ρ-bounded for all real time t as following:

$$(1+\rho)^{-1} \leq \frac{dP_{n_i}(t)}{dt} \leq (1+\rho)$$

The logical clock is related to the physical clock as L(t) = P(t) + A(t) where A(t) is the called the adjustment. This adjustment can be done discretely or linearly.

Initial synchronization: The logical clock of any two nodes n_i and n_j at t_0 where t_0 is the real time at the start of the synchronization algorithm differ by real time α.

$$\left|L_{n_i}(t) - L_{n_j}(t)\right| \leq \alpha$$

This is basically the initial synchronization assumption. Here we are assuming that all the nodes are initially synchronized at real time t_0.

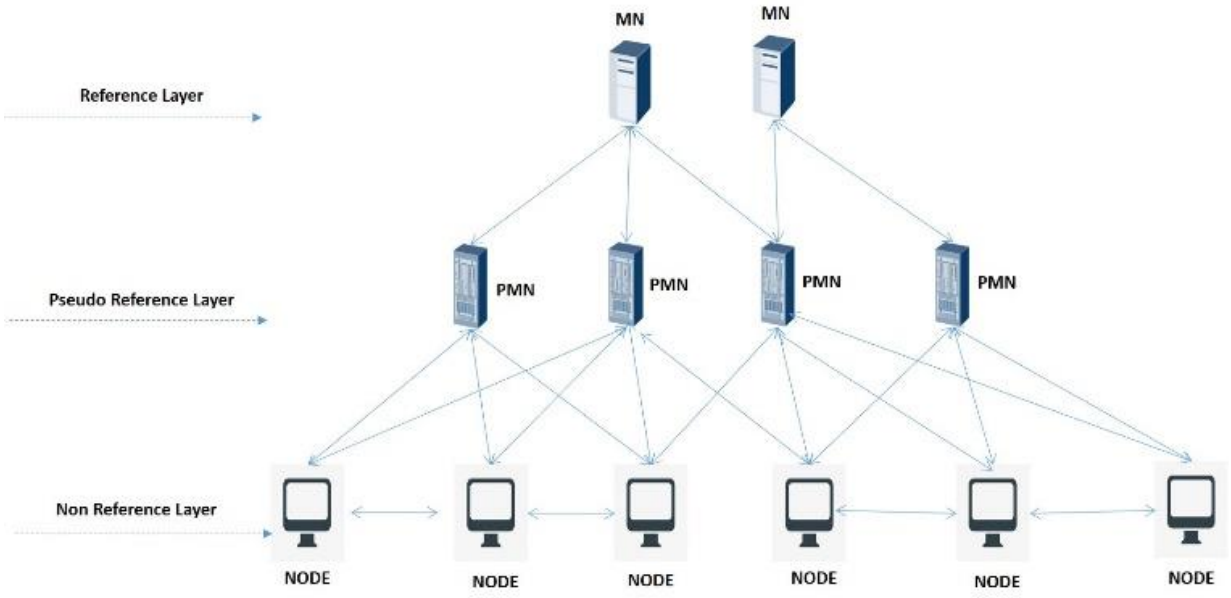

Fig. 1: Network Architecture

3.1.3 Malfunction Model

In our system malfunctions can occur to malfunction of nodes, their clock or due to link malfunction between the nodes or a combination of some or all of these malfunction. A clock is considered failed if the Drift-Bound assumption is violated. We are basically considering two types of clock malfunctions namely timing malfunction and mischievous clock. If a clock gives inaccurate, untimely and conflicting information or behaves like a 'dual-faced' clock then we assumed the clock is a mischievous clock. A clock is a good clock if it is running without any malfunction. We also assumed that the maximum number of bad clock is f.

Maximum Clock Malfunction: There are at maximum f number of faulty clock in the network for 3f+1 nodes.

3.1.4 Problem Statement

Our algorithm uses trusted reference clock value and a weighted averaging and sliding window method for attaining synchronization of the nodes in a distributed network. The synchronization obtained is both accurate and precise. The synchronization algorithm satisfies the following properties:

Agreement Property: For good clocks of nodes n_i and n_j at real time t immediately after re-synchronization, $C_{n_i}(t)$ and $C_{n_j}(t)$ satisfy:

$$\left|C_{n_i}(t) - C_{n_j}(t)\right| \leq \Pi$$

where $\Pi \leq (\delta + \varepsilon)$ is the precision of our synchronization algorithm.

Incremental Step property: The correct clocks change every time by some amount ξ at each synchronization.

Accuracy property: The correct clock of node n_i, for some constants a, b, c and d holds for real time $t_1 < t_2$ the following:

$$\frac{(t_2 - t_1)}{a} - b \leq C_{n_i}(t_2) - C_{n_i}(t_1) \leq (t_2 - t_1)c + d$$

We would like to make Π as small as possible so that clocks are close to one another. The validity of the Agreement property and Incremental Step property implies that our good clocks are running linearly with respect to real time. We would also like to make $(t_2 - t_1)$ as close as possible to $C_{n_i}(t_2)$ - $C_{n_i}(t_1)$ for any node n_i. This ensures the time elapsed by the clock of any node n_i is same to the real time elapsed in the interval. For optimal accuracy a and c should be equal to one and b and d should be zero in expression of Accuracy property.

3.1.5. Accurate Weighted Average Synchronization Algorithm (AWASA)

A synchronization algorithm which achieves clock synchronization by using a weighted averaging as a convergence function is described in [17]. This algorithm is called WASA and it utilizes the concept of sliding window to find the minimum variance data set upon which the convergence function is used. WASA achieves a precise clock synchronization. Here we are designing an algorithm which is both precise and accurate. Since this algorithm is an improvement on WASA, we call this algorithm as AWASA.

In this algorithm the MN receive reference clock value from trusted source. This clock value is then transmitted to the PMN in the PRL. There are much fewer MN as compare to PMN. Once the clock value is received by the PMN, they themselves synchronized with the new clock value by updating themselves. Since the clock value is from trusted source (MN) and on a single hop line of communication, we assume they are free of corruption. We trust the clock value and accept them as authentic value.

We conduct a check on the clock value of the PMN to ascertain whether any of its value is faulty. In the RL all the MN are receiving trusted clock value and the MN send the clock value to all PMN connected to them. In the PRL each PMN exchanges its clock values and stores in an array in an order of their clock values. All the clock values received by the PMN are from a trusted source however since any clock could have been gone bad due some malfunction as described in the malfunction model, we run a check before sending these clock values further for synchronization. Now the clock values distributed in the array can be visualized in the following fashion: n clock

values are distributed in the array i.e. we can visualize this as n slots to be filled with n-f good clock values and f bad clocks. The good and bad clock values can be anywhere in the n slots. There are ${}_nP_f$ distinct possibility.

3.1.5.1 WORKING OF AWASA

In any network, two kind of players are there with conflicting goals. One set of players which have an objective to stabilize the system are called good nodes. The other set of players which aim to destabilize the system are called bad nodes. Our goal is to synchronize the network even in presence of bad nodes. The most serious types of bad nodes are those nodes which give inaccurate, untimely and conflicting information. They may also give different time to different nodes at the same real time. These mischievous behaviors of nodes can be classified as follows: Consistent misbehavior - Nodes exhibit the same mischievous behavior in temporal domain. Inconsistent misbehavior - Nodes display mischievous behavior is inconsistent fashion in temporal domain. A node can misbehave in individual capacity or can collectively collude. These misbehaviors can be categorized as follows: Individual misbehavior: A node is displaying misbehavior in individual manner without consulting other bad nodes. Collaborative mischievous behavior: Bad nodes may consult each other and decide their strategy in a collective fashion.

In this paper our focus is to counter the individual misbehavior which includes consistent and inconsistent behavior both with the aim to achieve an accurate and precise clock synchronization. We have initially taken the trusted clock value from reputed authentic source like GPS/ GLONASS/IRNSS/ Galileo etc to the MN. This clock value is used as a reference to synchronized the rest of the network but after undergoing a check. This is done since the behavior of the clocks at PMN may have, by any probability, gone corrupted. We have devised a sliding window technique to find the set of values, which show minimum deviation among the values. We know that Gaussian distribution is widely present in nature. We capitalized this knowledge to design the weight function. This weighted approach is used to mitigate the misbehavior.

3.1.5.2. Detailed View of AWASA

We give a detailed description of AWASA in this section. The synchronization timeline as show in fig. 2. The synchronization at every regular interval of time, R_N. The periodicity or the time length every is same and decided according to the requirement of the system design like deviation range allow before a resynchronization is necessary. Within this time span the process of resynchronization has to take place, the time required for resynchronization is depicted as T_{rn}, Resync Process Time. The T_{rn} functions in three phases namely resynchronization of Reference Layer (t_{ref}), resynchronization of Pseudo Reference Layer (t_{pref}) and resynchronization of Pseudo Non-Reference Layer (t_{nref}). During each t_{pref} , the PMN executes WASA. WASA can counter consistent mischievous nodes. It also works in presence of inconsistent mischievous behavior provided the mischievous nodes remain consistent during T_{rn}.

In t_{ref} phase the MN receive trusted clock value from authentic source such as GPS/GLONASS/IRNSS etc. The ultimate aim to synchronize all types of nodes in the network to this clock value. In such a scenario, the clock synchronization will be very accurate as all the nodes will have clock value of the trusted/ reference clock.

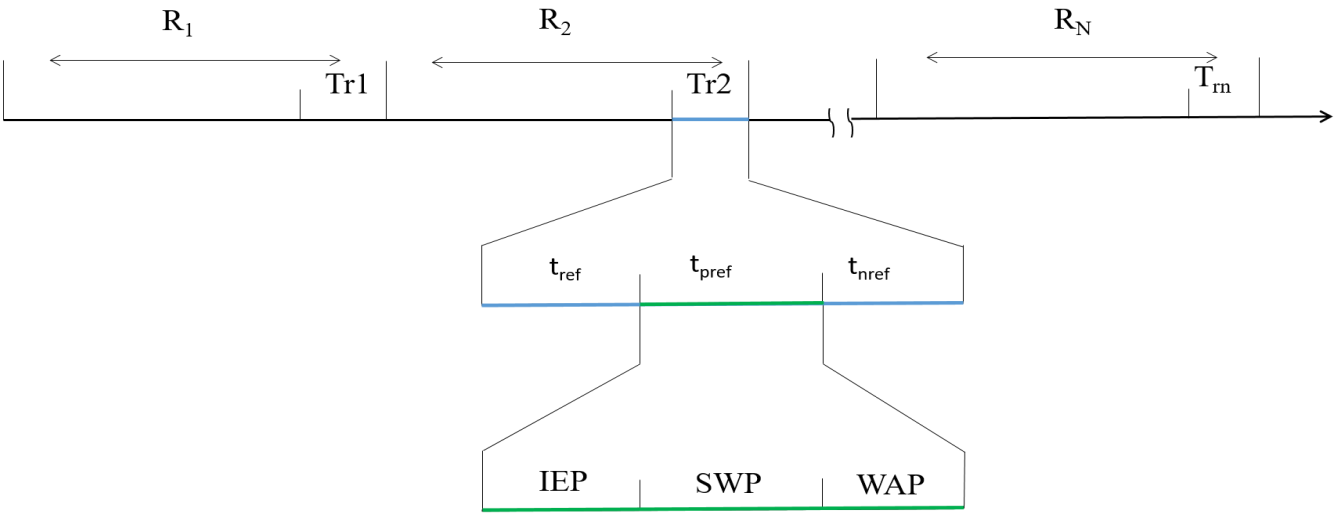

Fig. 2: Synchronization Timeline

However, synchronization of all nodes in the network with GPS/similar trusted clock value will not be possible. Some of the possible reasons for the same is given below. AWASA can overcome this type of scenario easily without effecting the synchronization process. The same will be explained subsequently.

a) Every node may not have a GPS/ similar receiver

b) Some nodes may be in such a location where they are not able to access the GPS signal

c) There arises such a situation in which the system administrator considers it not to utilize such external source of clock input, may be, in order to maintain secrecy

d) The GPS/similar sys is no longer trustworthy, etc.

When the MN is at time $t_{s,}$ it sends a resync request to all PMN. If the clock state of the PMN is between t_{rs} and t_{re}, an ACK is replied after due verification of the request. Timeline for one such t_{ref} is depicted in Fig.3. This indicates the start of the resynchronization process. However, if no ACK is received, the MN will understand that the resynchronization time has not yet started and waits for next resynchronization time and again send out its resync request. On receiving the ACK, the MN sends the trusted clock value received from GPS/similar source to all connected PMN. The PMN then assigns this clock value as their new clock value and hence resynchronize with the MN. For each resynchronization process, we follow the IEEE 1588 standard: as already described in the related work section.

In t_{pref} phase as all the PMN are now in sync with the MN, the PMN now need to further resync with the connected node. However before sending out the clock value to all connected Nodes with IEEE 1588 standard, all the PMN under a WASA [17] check. WASA consists of three phases namely Information Collection Phase (ICP), Sliding Window Phase (SWP) and Weighted Average Phase (WAP). The same is described briefly below.

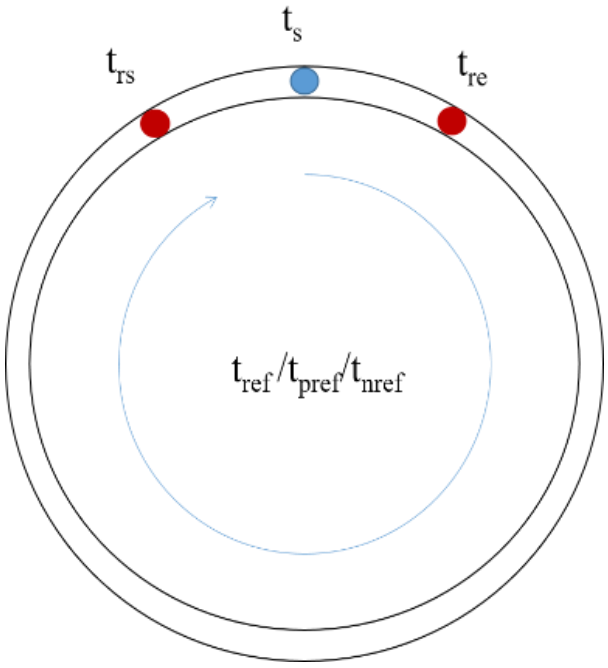

Fig. 3: Timeline for one Sub_sync period

3.1.5.3. Information Collection Phase (ICP)

In this phase every PMN share their clock with one another. PMN stores these clock values in an array. The clock values in the array are then arranged in an ordered fashion as per their values.

3.1.5.4. Sliding Window Phase (SWP)

We introduced the notion of window. The size of window is n-f where n network size and f is allowed bad PMN, if any. Initially we placed the left boundary of the window at the beginning of the array. So first window consists of first $n - f$ values of the array. Thereafter we slide the window one step in the right direction and this consists of next $n - f$ values. We continue this process until the right boundary of the window reaches the end of the array. In this process we find the f data set of size $n - f$. Now out of f data sets we find out the data set with minimum variance.

3.1.5.5. Weighted Average Phase (WAP)

We use the minimum variance data set calculated in the SWP for finding the weighted average with some modifications. Since the size of the minimum variance window is $n - f$, it contains at

least $n-2f$ good clocks and at most f bad clocks. For the worst case the minimum variance window contains $n-2f$ good clocks and f bad clocks. There can be also at maximum f good clocks outside the minimum variance window.

We now push these good clocks outside the minimum variance window inside the window by assigning the value of the good clock nearest to them inside the window. Now data set values inside the minimum variance window are assigned weights as per weight function and the weighted average is calculated. These weighted average value, θ, is the new clock.

For assignment of the weights, clock value within one ± σ distance from μ are assigned weight ω_1 , where μ is the mean and σ is the standard deviation of the window. Clocks within ± σ to ± 2 σ are assigned weight ω_2 and clocks within ± 2 σ to ± 3 σ are assigned weight ω_3. Finally clocks beyond ± 3 σ are assigned zero weight.

3.1.5.6. Best and Worst Case Scenario of WASA

The precision of WASA in an ideal condition where there is no PMN clock is $\Pi = \varepsilon$, which is optimal. The worst possible precision of WASA for large value of bad PMN is within:

$$\Pi \leq \varepsilon + \frac{2(\delta+\varepsilon)}{3}$$

The worst possible precision guaranteed by WASA is given by

$$\Pi \leq \varepsilon + \frac{2(\delta+\varepsilon)}{3}$$

Since $\varepsilon \ll \delta$,

$$\Pi \leq \varepsilon + \frac{2(\delta+\varepsilon)}{3} < (\delta + \varepsilon)$$

Hence WASA guaranteed Agreement Property.

The worst case precision by SWA^{det}_{mean} [18] is given as for n ≥ 4f

$$\Pi \leq \varepsilon + f\frac{(\delta+\varepsilon)}{n-f} + f\frac{(2\delta+\varepsilon)}{n-f+1}$$

If f ≫ 1 and taking δ ≫ ε, WASA is at least 33% tighter.

Now we have a highly accurate and precise clock value at the end of this phase. This clock value is now required to be sent to all Nodes in NRL for further resync with PMN at PRL. We used the IEEE 1588 standard for the same within the t_{nref} on completion of the t_{pref}.

As discussed in section 3.1.5 for synchronization of PMN in PRL, trusted clock value from RL are utilized after running a WASA check. There may be certain scenario where this trusted clock value are no more available or the system administrator decides not to use it as brought out in the section. In absence of such trusted clock value at PRL, the PMN after waiting for time t_{ref} during the resync time T_r, do the WASA check using own clock value. This is a smart strategy since up to now all the PMN are in highly accurate synchronization with the MN. Upon completion of the WASA check, the PMN with then get synchronized as described earlier. The PMN will also update the MN clock with its latest clock value using IEEE 1588 standard. This will ensure both the RL and NRL is precisely in sync with PRL. On availability of the trusted clock value, the resynchronization will revert back to its usual protocol.

4. CONCLUSION

In this paper, we devise and present AWASA algorithm for clock synchronization, which is suitable for fully connected network. AWASA achieve at least 33% tighter accuracy and precision in the worst-case scenario. Even the worst-case scenario occurrence is very less but our study provides meaningful analytically insight. AWASA guarantees clock synchronization in presence of mischievous clocks if the upper bound of mischievous nodes is just under one-third of the network size. Using this algorithm, we are able to achieve highly accurate and precise synchronization when there is presence of GPS/GLONASS/IRNSS etc clock value. The algorithm also works in absence of such trusted clock values as well. However, since a reference clock value is not present, the algorithm will give highly precise synchronization value only. In our current work, we provide the analysis for static network. In future work we will try to achieve synchronization for a dynamic connected network.

CONFLICT OF INTERESTS

The authors declare that there is no conflict of interests.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, IEEE Commun. Surv. Tutorials. 17 (2015), 2347–2376.

[2] E. Xu, Z. Ding, S. Dasgupta, Target Tracking and Mobile Sensor Navigation in Wireless Sensor Networks, IEEE Trans. Mobile Comput. 12 (2013), 177–186.

[3] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std. 1588-2002, 2002.

[4] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std. 1588-2008,2008.

[5] D. Kohler, A Practical Implementation of an IEEE1588 supporting Ethernet Switch, in: 2007 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, IEEE, Vienna, Austria, 2007: pp. 134–137.

[6] R. Holler, T. Sauter, N. Kero, Embedded SynUTC and IEEE 1588 clock synchronization for industrial Ethernet, in: EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.03TH8696), IEEE, Lisbon, Portugal, 2003: pp. 422–426.

[7] T. Neagoe, M. Hamdi, V. Cristea, Frequency Com pensated, H ardw are IEEE-1588 Im plem entation, in: 2006 IEEE International Symposium on Industrial Electronics, IEEE, Montreal, Que., 2006: pp. 240–245.

[8] T. Cooklev, J.C. Eidson, A. Pakdaman, An Implementation of IEEE 1588 Over IEEE 802.11b for Synchronization of Wireless Local Area Network Nodes, IEEE Trans. Instrum. Meas. 56 (2007) 1632–1639.

[9] J. Kannisto, T. Vanhatupa, M. Hännikäinen, T.D. Hämäläinen, Precision Time Protocol Prototype on Wireless LAN, in: J.N. de Souza, P. Dini, P. Lorenz (Eds.), Telecommunications and Networking - ICT 2004, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004: pp. 1236–1245.

[10] J. Jasperneite, K. Shehab, K. Weber, Enhancements to the time synchronization standard IEEE-1588 for a system of cascaded bridges, in: IEEE International Workshop on Factory Communication Systems, 2004. Proceedings., IEEE, Vienna, Austria, 2004: pp. 239–244.

[11] M. Maróti, B. Kusy, G. Simon, Á. Lédeczi, The flooding time synchronization protocol, in: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems - SenSys '04, ACM Press, Baltimore, MD, USA, 2004: p. 39.

[12] M. Akhlaq, T.R. Sheltami, RTSP: An Accurate and Energy-Efficient Protocol for Clock Synchronization in WSNs, IEEE Trans. Instrum. Meas. 62 (2013), 578–589.

[13] J. He, J. Chen, P. Cheng, X. Cao, Secure Time Synchronization in WirelessSensor Networks: A MaximumConsensus-Based Approach, IEEE Trans. Parallel Distrib. Syst. 25 (2014), 1055–1065.

[14] L. Schenato, F. Fiorentin, Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks, Automatica. 47 (2011), 1878–1886.

[15] R. Solis, V.S. Borkar, P.R. Kumar, A New Distributed Time Synchronization Protocol for Multihop Wireless Networks, in: Proceedings of the 45th IEEE Conference on Decision and Control, IEEE, San Diego, CA, 2006: pp. 2734–2739.

[16] W. Dong, X. Liu, Robust and Secure Time-Synchronization Against Sybil Attacks for Sensor Networks, IEEE Trans. Ind. Inf. 11 (2015) 1482–1491.

[17] Phurailatpam Devakinandan Sharma. A Precise Clock Synchronization Algorithm in Network. J. Commun. Eng. Syst. 10 (2020), 12-21.

[18] M.J. Pfluegl, D.M. Blough, A New and Improved Algorithm for Fault-Tolerant Clock Synchronization, J. Parallel Distrib. Comput. 27 (1995), 1–14.