Wireless Precision Time Protocol

Ayush Garg, Akash Yadav, Axel Sikora, and Ashok Singh Sairam, *Senior Member, IEEE*

***Abstract*— The IEEE 1588 precision time protocol (PTP) is a time synchronization protocol with sub-microsecond precision primarily designed for wired networks. In this letter, we propose wireless precision time protocol (WPTP) as an extension to PTP for multi-hop wireless networks. WPTP significantly reduces the convergence time and the number of packets required for synchronization without compromising on the synchronization accuracy.**

***Index Terms*— IEEE 1588, time synchronization, wireless network.**

I. Introduction

MANY cyber physical systems (CPS) are based on wireless sensor networks, where time synchronization is crucial to realize the coordinated duty-cycling of nodes as well as global timestamping of sensor data. Precision Time Protocol (PTP), which is specified as IEEE 1588 ([1]), is a generic approach to provide such a timestamping. In addition, there have been many efforts to increase the synchronization accuracy of nodes in wireless sensor networks since many years ([2], [3]).

In this paper, we propose a novel approach to provide sub-microsecond synchronization accuracy without compromising the traffic load on the channel and in the energy consumption of the nodes. This approach is based on PTP, but proposes an extension for wireless multi-hop networks. Although there also have been already some attempts in realizing PTP for wireless networks ([4], [5]), they mainly examined the limitations posed by the physical layer of wireless networks. In Figure 1, we show the straight forward application of PTP in a wireless environment. As can be seen from the figure, the protocol iterates in levels. In the first iteration, the level 1 nodes are synchronized, these nodes then act as masters for the second level nodes and so on the protocol proceeds. The equation to compute the offset ($\triangle$) is given in Equation 1. Delay can be computed by adding the two terms instead of subtracting.

$$\triangle = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{1}$$

Manuscript received September 11, 2017; revised November 6, 2017; accepted December 4, 2017. Date of publication December 11, 2017; date of current version April 7, 2018. This work was partially supported by Department of Science and Technology, India and German Academic Exchange Service (DAAD), Grant INT/FRG/DAAD/P-250/2015. The associate editor coordinating the review of this paper and approving it for publication was B. Rong. *(Corresponding author: Ashok Singh Sairam.)*

A. Garg and A. Yadav are with the Department of Computer Science and Engineering, IIT Patna, Patna 801103, India (e-mail: ayushgrg04@gmail.com; akash.pcs13@iitp.ac.in).

A. Sikora is with the Hochschule fur Technik Wirtschaft und Medien Offenburg, 77652 Offenburg, Germany (e-mail: axel.sikora@hs-offenburg.de).

A. S. Sairam is with the Department of Mathematics, IIT Guwahati, Guwahati 781039, India (e-mail: ashok@iitg.ernet.in).

Digital Object Identifier 10.1109/LCOMM.2017.2781706

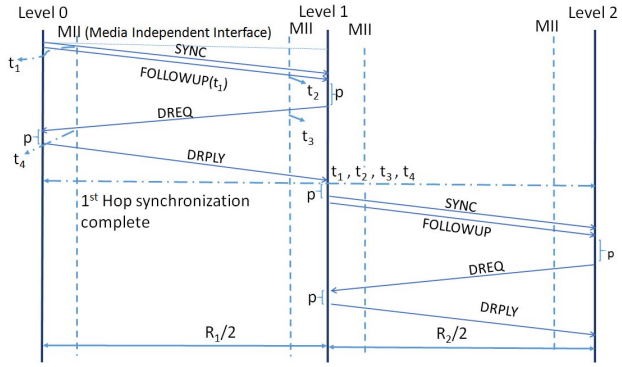

Fig. 1. IEEE 1588 PTP working procedure.

In PTP, the synchronization process of the next level nodes starts only after completion of synchronization of the nodes in the current level. In wireless networks, all communications are inherently broadcast. Our proposed protocol exploits the broadcast nature of wireless communication to overlap the synchronization process between adjacent levels. The proposed protocol thus saves on the number of packets exchanged as well as reduces the convergence time.

The main contributions of our work is (a) proposed a sub-microsecond precision, time synchronization framework for multi-hop wireless networks (b) provide an in depth performance analysis of the system, and (c) validated the analytical results with experiments.

II. System Model

We consider a multi-hop wireless network with nodes distributed over a finite geographical area. The root or master node starts the synchronization process. The proposed protocol follows the master-slave synchronization architecture, where the slave nodes synchronize themselves with the master node's clock. The slave nodes can be connected in one of the two ways with the master node: (i) Directly connected: nodes which lie within the transmission range of the master node (i.e. nodes in level 1). (ii) Indirectly connected: nodes which do not lie within the transmission range of master node. These nodes use the intermediate nodes as hops to connect with master (i.e. nodes in level n where $n > 1$). The hop or level at which a node lies can be ascertained using the level discovery phase in [6] with the root assumed to being at level 0.

III. Working of WPTP

In WPTP, the synchronization of directly connected and indirectly connected nodes are handled differently. The synchronization process of WPTP is shown in Figure 2.

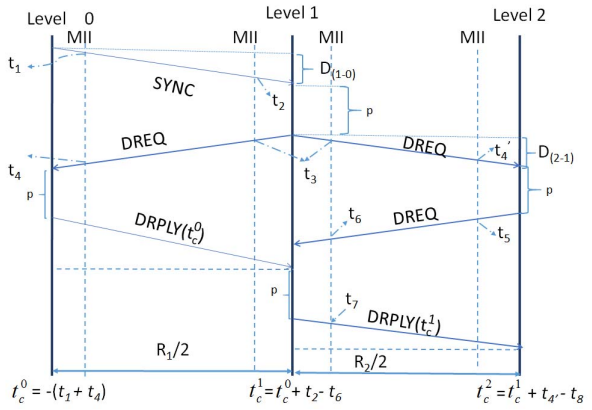

Fig. 2. Working of Proposed WPTP.

A. Synchronization of Directly Connected Nodes

The equation to compute offset as given in Equation 1 can be re-written as,

$$\triangle = \frac{(t_2 + t_3) - \overbrace{(t_4 + t_1)}^{level\ 0}}{2}$$

Here the time stamp t_1, t_4 are from level 0, hence they can be calculated at that level itself and can be sent as a single consolidated time stamp. Let t_c^0 denote the consolidated timestamp from level 0, then $t_c^0 = -(t_4 + t_1)$. Thus at level 1, the offset can be calculated as

$$\triangle_{(1,0)} = \frac{(t_2 + t_3) + t_c^0}{2} \tag{2}$$

So in WPTP only three packets will be required to calculate and correct the clock offset. To account for the message transmission delay, a second set of *SYNC* and *FOLLOWUP* messages are used with the corrected clock. The process to compute delay will be same as that of computing offset, except that the node at lower level will send the difference of the timestamps. In the remaining part of the paper, we concentrate only on computation of the offset.

1) Synchronization of Indirectly Connected Nodes: In wireless networks, since every transmission is a broadcast, the same *DREQ* message send by nodes in level 1 to those nodes in level 0, can act as a *SYNC* message for the nodes in level 2. Figure 2 show how the nodes at level 2 can represent their clock time w.r.t. node at level 0. The offset can be calculated as:

$$\begin{aligned} t_4' &= D_{(2,1)} + p + D_{(1,0)} + \triangle_{(2,0)} + t_1 \\ \Rightarrow \triangle_{(2,0)} &= t_4' - D_{(2,1)} - p - D_{(1,0)} - t_1 \end{aligned} \tag{3}$$

In Equation 3, $D_{(2,1)}$ represents transmission delay between level 2 and level 1, p represents the processing time which is assumed to be same for all the nodes. $\triangle_{(2,0)}$ is the clock offset between the local cocks at level 2 and level 0. By using the timestamps shown in Figure 2 and the equation for delay ([6]), the Equation 3 can be expressed as:

$$\begin{aligned} \triangle_{(2,0)} &= t_4' - \left(\frac{(t_4' - t_3) + (t_6 - t_5)}{2}\right) - (t_3 - t_2) \\ &\quad - \left(\frac{(t_4 - t_3) + (t_2 - t_1)}{2}\right) - t_1 \\ &= \frac{1}{2}\Bigg(\underbrace{(t_4' + t_5)}_{\text{by } level\ 2} + \underbrace{(t_2 - t_6)}_{\text{by } level\ 1} + \overbrace{\underbrace{(-t_4 - t_1)}_{\text{by } level\ 0}}^{t_c^0}\Bigg) \\ &= \frac{1}{2}\Bigg(\overbrace{(t_4' + t_5) + (t_2 - t_6) + (-t_4 - t_1)}^{t_c^1}\Bigg) \end{aligned} \tag{4}$$

Equation 4 shows the timestamp contributed by each level. As level 1 has already calculated the value of t_c^0 from level 0, it can send a single consolidated timestamp, $t_c^1 = (t_2 - t_6) + t_c^0$, to level 2. Thus the resulting formula for calculating offset at level 2 will be reduced as follows:

$$\triangle_{(2,0)} = \frac{t_4' + t_5 + t_c^1}{2} \tag{5}$$

The first two term t_4' and t_5, in Equation 5 denote the timestamp when the node at level 2 receives and sends a *DREQ* packet. In order to make the equation generic, let $t^n_{r(n-1)}$ denote the time when a node at level n receives a *DREQ* packet from a node at level $n-1$. Similarly let $t^n_{s(n-1)}$ denote the time when a node at level n sends a *DREQ* packet to a node at level $n-1$. Thus the generic equation to compute offset for a node at the n^{th} level w.r.t. the node at level 0 is given as:

$$\triangle_{(n,0)} = \frac{1}{2}\left(t^n_{r(n-1)} + t^n_{s(n-1)} + t_c^{n-1}\right) \tag{6}$$

where t_c^{n-1} can be calculated as:

$$t_c^n = \begin{cases} t_c^{n-1} + t_r^n - t^n_{r(n+1)}, & \text{if } n \geq 1 \\ -(t_4 + t_1), & n = 0 \end{cases} \tag{7}$$

In this protocol, a node sends the consolidated timestamp value to the next hop nodes using the *DRPLY* message, while the *SYNC* and *DREQ* packet do not contain any timestamps and are only used to initiate the protocol. Thus this mechanism will greatly reduce the overall convergence time and the number of packets without compromising on the synchronization accuracy. The steps of WPTP can be summarized as follows:

- Master node(s) initiate the synchronization process.
- Nodes at level n receive either *SYNC* message (if $n = 1$) or *DREQ* message (if $n > 1$) from nodes at level $(n-1)$ and record its receiving time, i.e. $t^n_{r(n-1)}$.
- These nodes at level n, then broadcast the *DREQ* message and record the sending time i.e. $t^n_{s(n-1)}$.
- The nodes at level $(n-1)$ receive the *DREQ* message and record the receiving time i.e. $t^{(n-1)}_{r(n)}$.
- If the node at level $(n-1)$ is already synchronized then it sends a *DRPLY* message containing the consolidated timestamp t_c^{n-1}. Otherwise it waits for a *DRPLY* message and then replies.

At the end of these steps, nodes at level n will have all the timestamps to calculate the offset. The nodes at level $(n+1)$ receive *DREQ* message from nodes at level n and the process repeats for each level.

IV. Performance Analysis

This section gives the quantitative analysis of *WPTP* over *PTP* in terms of total convergence time, total number of packets required and storage overhead.

A. Convergence Time Analysis

We consider a general N-hoped network and let R_l is the round trip time between nodes of level $(l-1)$ and l. The average propagation delay is $R_l/2$. The processing time p is same for all the nodes.

1) Convergence Time for PTP: In Figure 1, we can see that for the nodes in level 1 to get synchronized, four packets are exchanged. However, the *SYNC* and *FOLLOWUP* packets are send almost simultaneously and thus we consider a single propagation delay for these two packets. The convergence time T_{PTP} is given as:

$$\begin{aligned} T_{PTP} &= (\frac{3}{2}R_1 + 2p) + (\frac{3}{2}R_2 + 3p) + \cdots (\frac{3}{2}R_N + 3p) \\ &= 3/2(R_1 + R_2 + \cdots R_N) + (3N-1)p \end{aligned} \tag{8}$$

The first term in the equation refer to the convergence time for nodes in level 1, the second term for nodes in level 2 and so on.

2) Convergence Time for WPTP: The convergence time for WPTP will depend on the order of arrival of the synchronization message. In this work, we consider the case when the *DREQ* message from the next hop arrives before the *DRPLY* message from the previous hop, as can be seen for the level 1 nodes in Figure 2. Then as per the algorithm, the node has to wait for the *DRPLY* message to get the value of t_c^{i-1} before replying to the next level node, where i is the current level of the node. The convergence time T_{WPTP} can be calculated as:

$$\begin{aligned} T_{WPTP} &= (3/2R_1 + 2p) + (p + R_2/2) + \cdots (p + R_N/2) \\ &= R_1 + 1/2(R_1 + R_2 + \cdots R_N) + (N+1)p \end{aligned} \tag{9}$$

The other extreme case can occur when a node first receives the $DRPLY$ message from the lower level node, gets synchronized and then receives $DREQ$ message from the next higher level node. In such a case, the node can immediately send the calculated value of t_c^i. It can be shown that the convergence time for such case will also be equivalent to Equation 9.

Further, for ease of calculation, we assume R to be the average of all the round-trip times $(R_1, R_2, \cdots R_N)$ in the network. Thus, the convergence time of PTP and WPTP given in Equation 8 and 9 respectively can be expressed as:

$$T_{WPTP} \approx R + \frac{N}{2}R + (N+1)p \tag{10}$$

$$T_{PTP} \approx \frac{3}{2}NR + (3N-1)p \tag{11}$$

The gain in convergence time of WPTP with respect to PTP is calculated as:

$$\begin{aligned} &\frac{(\frac{3}{2}NR + (3N-1)P) - \frac{N+2}{2}R - (N+1)P}{\frac{3}{2}NR + (3N-1)P} \\ &\Rightarrow \frac{(N-1)R + 2(N-1)P}{\frac{3}{2}NR + (3N-1)P} \\ &\leq \frac{(N-1)R + 2(N-1)P}{\frac{3}{2}(N-1)R + 3(N-1)P} \\ &\Rightarrow \frac{R+2P}{\frac{3}{2}R + 3P} \approx 0.67 \end{aligned} \tag{12}$$

Thus, by using WPTP, the upper bound of gain in convergence time as compared to PTP is about 67%.

B. Number of Packets Used for Synchronization

The network used for analysis is assumed to contain K number of nodes distributed over the n hops and it is assumed to be collision free. Here level n contains k_n number of nodes, such that $\sum_{n=0}^{N} k_n = K$. The level 0 contains k_0 number of master nodes, where k_0 usually equals one.

1) Synchronization Packets Required for PTP: In PTP, all the k_0 master nodes will broadcast the *SYNC* and *FOLLOWUP* message and then for each node of level 1, one *DREQ* and one *DRPLY* message will be required. So the total number of packets required to synchronize all the k_1 nodes will be $2k_0 + 2k_1$. In general the number of packets required to synchronize the nodes of level l will be: $2k_{l-1} + 2k_l$. So total number of packets required to synchronize the whole network is:

$$\begin{aligned} Packets_{PTP} &= (2k_0 + 2k_1) + \cdots + (2k_{N-1} + 2k_N) \\ &= 2(k_0 + k_N) + 4(k_1 + k_2 + \cdots k_{N-1}) \end{aligned} \tag{13}$$

2) Synchronization Packets Required for WPTP: In *WPTP*, the master node(s) will only broadcast *SYNC* message to initiate the process and then for each slave node, one *DREQ* and one *DRPLY* message will be required. Thus the total number of packets required to synchronize all the k_1 nodes will be $k_0 + 2k_1$. Now since the same broadcasted *DREQ* message will act as a *SYNC* message for the nodes at level 2, so for each node only one *DREQ* and one *DRPLY* packet will be required. The same process will follow for all the further hops. Hence the total number of packets required in *WPTP* will be:

$$\begin{aligned} Packets_{WPTP} &= (k_0 + 2k_1) + (2k_2) + \cdots (2k_N) \\ &= k_0 + 2(k_1 + k_2 \cdots k_N) \end{aligned} \tag{14}$$

Comparing Eq. (13) and (14), it is observed that WPTP synchronizes the whole network by using almost half the number of packets as compared to PTP.

C. Storage Overhead

The main overhead of WPTP is in terms of additional storage at the motes. A node at level i typically need to store the DREQ packets received from those nodes of level $(i+1)$, which it is synchronizing. The size of a DREQ packet is 6 bytes. Thus the average storage overhead at each node of level i is $\frac{k_{i+1}}{k_i} * 6$ bytes.

V. Implementation Details

In this work, we implemented both PTP and WPTP using the *Contiki Operating System* ([7]) and *Cooja Simulator*. The network is assumed to be homogeneous and static with the number of hops varying from 1 to 6. The master node is assumed to be at at hop 0 or level 0. The protocol stack used is *Rime*. It is an alternative network stack, that is used when the overhead of the IPv4 or IPv6 stack is prohibitive.

There are three possible states of a node. A node remains in the INACTIVE state before the start of the synchronization

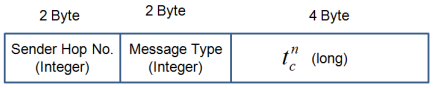

Fig. 3. Packet Format of DRPLY.

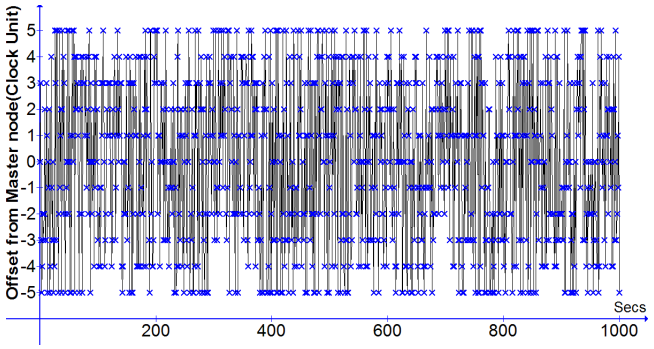

Fig. 4. Offset error from Master Node.

TABLE I
AVERAGE OFFSET ERROR OF PTP AND WPTP

Level	WPTP	PTP
1	2.6	2.6
2	3.4	3.8
3	4.4	4.2
4	4.2	3.8
5	4.8	4.8
6	5	5.4

process. When a node receives the first synchronization packet, its state changes from INACTIVE to ACTIVE. The state of a node changes from ACTIVE to SYNCED once it completes the synchronization process.

The structure of *DRPLY* packet used in our implementation is shown in Figure 3. The other packets have similar structure as that of PTP. Protothreads is a mix of event-driven and multi-threaded programming mechanisms used in Contiki. The three main protothreads used in our implementation are *BroadcastMsg()*, *UnicastMsg()* and *MainPTP()*.

Experiment 1: Figure 4 shows the values of clock offset of a level 1 slave node from the master node. The timestamps have been recorded using the *Rtimer* library of Contiki, whose one clock unit equals 32μs. The results show that the clock offset error ranges between 5 and -5 clock units with an average value of 0.057 clock units and standard deviation of 3.18 clock units. This result (in term of clock units) is comparable to the results of implementation of PTP in wireless sensor networks [4]. However, the absolute offset error in our implementation is high since we are recording the packet's sending and receiving timestamps in the radio layer rather than using the hardware triggered set-up.

Experiment 2: Table I show the comparison of average offset error of WPTP and PTP for a network with 6 levels of nodes. Here the average offset error is computed by considering the absolute value of the offset error of all the nodes in the same level, hence the values are always positive. The average offset

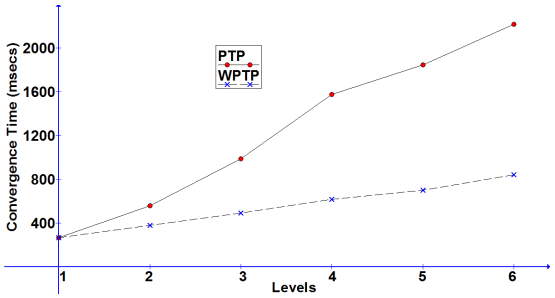

Fig. 5. Comparison of convergence time.

error for WPTP and PTP is 4.175 clock units and 4.25 clock units respectively.The results show that offset error is almost equal in PTP and WPTP.

Experiment 3: Figure 5 show the comparison of convergence time of WPTP and PTP for a network with 6 levels of nodes. Convergence time is defined as the time duration taken to synchronize all the nodes. The gain in convergence time of WPTP as compared to PTP is exponential for the initial levels and from the fourth level the gain is almost constant about 62% which concur with the analytical result given in Equation 12. The overall performance improvement of WPTP is about 50%.

VI. CONCLUSION

In this letter, we proposed wireless precision time protocol (WPTP), a time synchronization protocol for multi-hop wireless networks. WPTP works on the principles of the precision time protocol (PTP), a high precision time synchronization protocol for wired networks. WPTP exploits the broadcast communication property to significantly reduce the number of control packets required. The performance improvement is 50% as compared to that of PTP. The simulation results prove the proposed analytical model that the performance improvement in convergence time of WPTP does not compromise its synchronization accuracy.

REFERENCES

[1] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2008, 2008.
[2] A. Mahmood, R. Exel, H. Trsek, and T. Sauter, "Clock synchronization over IEEE 802.11—A survey of methodologies and protocols," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 907–922, Apr. 2017.
[3] R. Sharma, A. S. Sairam, A. Yadav, and A. Sikora, "Tunable synchronization in duty-cycled wireless sensor networks," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Nov. 2016, pp. 1–6.
[4] H. Cho, J. Jung, B. Cho, Y. Jin, S.-W. Lee, and Y. Baek, "Precision time synchronization using IEEE 1588 for wireless sensor networks," in *Proc. Int. Conf. Comput. Sci. Eng.*, vol. 2. Aug. 2009, pp. 579–586.
[5] W. Wallner, A. Wasicek, and R. Grosu, "A simulation framework for IEEE 1588," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Sep. 2016, pp. 1–6.
[6] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst.*, 2003, pp. 138–149.
[7] *Contiki, the Open OS for the Internet of Things*. Accessed: Mar. 2017. [Online]. Available: http://www.contiki-os.org/