Proceedings of the 23rd International Conference on Automation & Computing, University of Huddersfield, Huddersfield, UK, 7-8 September 2017

A Software Simulator of Discrete Pulse-Coupled Oscillators (PCO) Time Synchronization in Wireless Sensor Networks

Yan Zong, Xuewu Dai, Zhiwei Gao
Department of Mathematics, Physics and Electrical Engineering
Northumbria University
Newcastle upon Tyne, United Kingdom
yan.zong@northumbria.ac.uk, xuewu.dai@northumbria.ac.uk, zhiwei.gao@northumbria.ac.uk

Abstract—**Time synchronization, aiming to provide a common timescale among distributed sensor nodes, is a key enabling technology for many applications, such as collaborative condition monitoring and localization detection. Due to the complexity of time synchronization in wireless sensor networks, the Discrete Event Simulator is recommended to adopt resulting from the feature that the behavior of a complex is represented as an order sequence of well-defined event in time. In this paper, the PCO clock is firstly implemented into the open-source software simulator OMNeT++ by using *desynchronization* mechanism under a realistic scenario, and it can also be simulated at an adjustable and higher resolution. The developed relay node, can either be a Full Function Device or a Reduced Function Device, enables the high scalability for multi-node and multi-hop simulation. In addition, the shared code of this project on the GitHub directly benefits the researchers and engineers in communication.**

Keywords-time synchronization; pulse-coupled oscillators; desynchronization; OMNeT++; wireless sensor networks

I. INTRODUCTION

Time Synchronization (TS) in Wireless Sensor Networks (WSNs), aiming to provide a common sense of timing among distributed sensor nodes, is one of key enabling technologies for many wireless applications, such as data fusion and Time Division Multiple Access (i.e., TDMA) technique. In such applications, a network of distributed sensors are dedicated to cooperatively monitor physical or environmental conditions such as temperature, sound, pressure and motion at different locations [13],[16].

Some existing TS algorithms, such as PTP (Precise Time Protocol), RBS (Reference Broadcast Synchronization), TPSN (Time-sync Protocol for Sensor Networks) and FTSP (Flooding Time Synchronization Protocol), achieve the high accuracy TS by improving the timestamp. And the increase of hop of WSNs does weaken the performance of these TS algorithms. In addition, these complicated algorithms demand the high resource of WSNs [21]. Inspired by one of the most famous periodic synchronized activities in autonomous system in biological system, synchronized flashing of fireflies observed in certain parts of southeast Asia [10], a bio-inspired mathematical model, namely, Pulse-Coupled Oscillators (PCO) has been developed, and applied to sensor networks to enable time synchronization in WSNs since its simplicity of PCO.

In the classic PCO, one significant assumption is that the fired *Pulse* is broadcasted and receipted immediately and simultaneously, namely, there is no delay (e.g., propagation delay) in complex system. In the WSNs, one technological limitation of wireless transceiver is the half-duplex constraint (i.e., nodes cannot transmit and receive at the same time) [15]. Another limitation is time needed for packet transmission (i.e., transmission delay). Therefore, PCO cannot be implemented directly to enable the common timescale of WSNs, and the *desynchronization* (DESYNC), logical opposite of synchronization, is applied to sensor networks to transmit the *Pulse* as far away as possible from all other nodes, rather than *Pulse* broadcasted at the same time [4].

Due to the complexity of TS in WSNs, it is common to use the Discrete Event Simulator (DES) to simulate the network realistically, and feature the behavior of TS as an order sequence of well-defined event in time [11]. There are some famous DESs in WSNs simulation, e.g., the commercial simulator OPNET and open-source simulator NS2 and OMNeT++.

[20] theoretically and experimentally proves that the OMNeT++ is an excellent WSNs simulation platform from some factors, such as simulation library, debugging and tracing, delivery ratio and memory usage. Two simulation models of OMNeT++ are for simulating time synchronization in WSNs, (i.e., Castalia with clock model [7] and realistic PTP TS simulator [11]). Both [7] and [11] provide the realistic drifting clock model, and [11] also simulates the timestamp uncertainties, delays, jitters and node movements with good reality. The developed simulator of [11] achieving realistic delays and jitters simulation of IEEE 802.15.4 is beneficial to develop the PCO model. However, the master and slave nodes of [11] can only realize the functions of Reduced Function Device (RFD) of ZigBee. And the PCO clock model has not been implemented into the realistic software simulator.

Therefore, the main contribution of this paper is to implement the PCO clock into realistic software simulator by using DESYNC mechanism. Then, the TS superframe is proposed based on the DESYNC mechanism and IEEE 802.15.4 beacon-enabled superframe. Thirdly, the relay node, developed to either be the Full Function Device

(FFD) or RFD of ZigBee, enables the high scalability for multi-node and multi-hop simulation. In addition, the shared code of this project on the GitHub directly benefits the researchers and engineers in communication [22]. The rest of this paper is as follows: Section 2 presents the simulation framework of developed simulator. Section 3 introduces the developed drifting PCO clock model, proposed superframe and measurement offset calculation, which are implemented into software simulator OMNeT++ in Section 4. Finally, the simulation results are given in Section 5, and Section 6 details the conclusion.

II. Simulation Framework

The general structure of proposed simulation framework is shown in Fig. 1, and four components of simulation framework are as follows [11]:

- The *World Manager* module defines the environment model representing the geographical environment of WSN, namely, the size of the WSN's deployment area and obstacles blocking/attenuating wireless signals.
- *Connection Manager* module manages the mobility and connectivity, such as the varies of wireless channel and the change of network topology resulting from the moving nodes.
- A wireless channel denotes the features of the wireless channels and its impacts on packet exchange, namely, propagation delay, collision and attenuation.
- WSN nodes: there are three types of nodes, namely, master, slave and relay nodes. The relay node used in the multi-hop network is to realize the functions of FFD.

In Fig. 1, four layers of the Internet Protocol (IP) are implemented into the simulator by C++ classes (referred as modules), namely, the application layer (*app*), the network layer (*netw*), the MAC layer (*mac*) and the physical layer (*phy*). And two additional modules are implemented into the simulator, the *TS* module which is for timestamp, and the *Core* module is for functions of WSN nodes.

In addition, three other supporting modules used in the simulator are as follows [11]:

- *Clock* module, providing local time to other modules of node, simulates the node's clock.
- *Mobility* module stores and updates the location and speeds of nodes, reporting them to the *World Manager* and *Connection Manager*.
- *World Manager* and *Connection Manager* update the location of nodes, and configure the wireless links in the network topology.

III. Modelling Dynamics of Oscillator Clock

A. Model of Drifting Clock

Clock time in embedded systems is discrete and usually provided by a crystal oscillator and a series of *counter* [13]. A periodic sine-wave signal or square-wave signal is generated by the crystal oscillator [11],[19]. The *counter*, usually called *timer* in embedded systems, is used to count the number of the generated periodic signal. When the value of the *counter* register reaches the pre-defined threshold value, an interrupt, called a clock *tick*, is generated and the register returns to default value. The interrupt increments the clock value stored in the memory at each clock *tick*. This clock value can be used to timestamp an event [19].

The output of an ideal sinusoidal oscillator with nominal frequency can be defined as

$$V(t) = V\sin(2\pi f_0 t) \tag{1}$$

where V is a constant amplitude, and the period of the sinusoidal wave is $\tau_0 = 1/f_0$.

As the sinusoidal wave is converted into a pulse, the *counter* counts the pulse to generate the clock *tick*, meanwhile, the clock value can be obtained. This process is modelled by comparing the instant phase $(2\pi f_0 t)$ against 2π [11]

$$k = \left\lfloor \frac{2\pi f_0 t}{2\pi} \right\rfloor = \lfloor f_0 t \rfloor = \left\lfloor \frac{t}{\tau_0} \right\rfloor \tag{2}$$

where k represents an integer indicating how many cycles have occurred, the *k-th* counting event can be

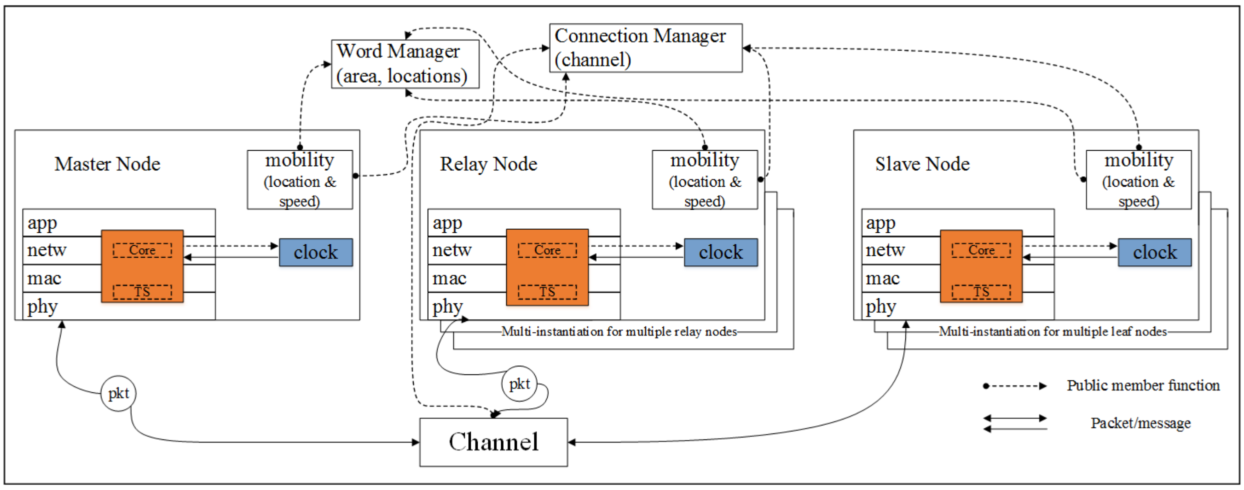

Fig. 1. Simulation Framework

referred as the event of the counter reaching the value k. And floor function operator $\lfloor x \rfloor$ means the largest integer not greater than x. The time instant $t[k]$ can be used to represent the global reference time at the *k-th* event, in addition, it is common to use $C[k]$ to represent the clock time of *k-th* event. The clock time of an ideal clock is given by

$$C[k] = t[k] = k \times \tau_0 \tag{3}$$

Due to the crystal manufacturing tolerance, crystal capacitive loading mismatch and oscillator temperature drift [2], the frequency of each oscillator of WSNs is non-identical and time-varying.

In order to model the non-identical and time-varying oscillators in the distributed system, a random process $\phi(t)$, representing all the instant phase deviation, can be used to model the phase noise of the oscillator. And $\alpha(t)$ is adopted to represent the oscillator frequency change at time t, the drifting frequency $f(t)$ at instant time t is $f_0 + \alpha(t)$, and the drifting clocks is modelled as [11]

$$V(t) = V \sin\left(2\pi\left(f_0 t + \int_0^t \alpha(\tau)\, d\tau\right) + \phi(t)\right) \tag{4}$$

And the k of (2) is modified to

$$k = \left\lfloor f_0 t + \int_0^t \alpha(\tau)\, d\tau + \frac{\phi(t)}{2\pi} \right\rfloor \tag{5}$$

Therefore, the time of a drifting clock at *k-th* event is

$$\begin{aligned} C[k] &= \left(f_0 t + \int_0^{t[k]} \alpha(\tau)\, d\tau + \frac{\phi(t[k])}{2\pi} \right) \times \frac{1}{f_0} \\ &= t[k] + \frac{\int_0^{t[k]} \alpha(\tau)\, d\tau}{f_0} + \frac{\phi(t[k])}{2\pi f_0} = t[k] + \theta[k] \end{aligned} \tag{6}$$

It is obvious that the clock $C[k]$ is inaccurate and different from the reference $t[k]$ due to the second and third terms of (6) (i.e., the phase noise $\phi(t)$ and the accumulated frequency variations). And the difference of clock time between the ideal clock and drifting clock is clock offset, $\theta[k]$.

Even though the clock frequency is affected by the oscillator temperature drift, the frequency of drifting clock can also be considered as a constant value due to the high clock update frequency. Therefore, the term $\int_0^{t[k]} \alpha(\tau)\, d\tau$ in (6) can be piecewise discretized as $\sum_{i=0}^{k-1} \alpha(i)\tau_0$. In addition, the term $1/(2\pi f_0)$ can be viewed as a scaling factor. The $\phi(t[k])/(2\pi f_0)$ can be rewritten as a discrete form $\phi[k]$, because the possibility destiny function (PDF) of $\phi[k]$ is similar to that of $\phi(t[k])/(2\pi f_0)$.

Therefore, in the discretized clock model, the clock offset $\theta[k]$ of *k-th* event can be modelled as

$$\theta[k] = \frac{\sum_{i=0}^{k-1} \alpha(i)\tau_0}{f_0} + \phi[k] \tag{7}$$

In addition, the skew $\gamma[k] = (f(t[k]) - f_0)/f_0$ is used to denote the deviation from the nominal frequency. For discretized clock model, the skew $\gamma[k]$ is equal to $\gamma[k] = \alpha(t[k])/f_0$ for one clock update period of $[t[k], t[k+1]]$. Thus, by introducing $\omega_\theta[k] = \phi[k+1] - \phi[k]$, the offset of *(k+1)-th* is rewritten as

$$\theta[k+1] = \theta[k] + \gamma[k] \times \tau_0 + \omega_\theta[k] \tag{8}$$

The skew $\gamma[k]$ is generally affected by environmental factors (e.g., temperature), and it can be assumed as a constant value within diminutive clock update period due to the slow change of skew. A better model, auto-regressive (AR) model, can be used to model the skew as a time-varying process in an auto-regressive manner with a small perturbation [11]. And the skew of *(k+1)-th* event is given by

$$\gamma[k+1] = p \times \gamma[k] + \omega_\gamma[k] \tag{9}$$

where ω_γ is the noise with zero mean, and p, the parameter of the first-order AR model, is close to 1.

In addition, both offset noise ω_θ and skew noise ω_γ, two uncorrelated random process, are subjected to zero-mean Gaussian distribution with standard deviation σ_θ and σ_γ respectively [11].

B. *Pulse-Coupled Oscillators*

The Pulse-Coupled Oscillators, also called MS (Mirollo and Strogatz) model, is developed by [14] based on the model proposed by Peskin. The PCO consists of two states, namely, the free running state and interacting state [17].

In the free running state of classic PCO, the state variable of node increases to threshold value. When the state variable reaches the threshold, it is reset to zero, meanwhile a *Pulse* is generated and broadcasted, and increases to threshold, and so on [18].

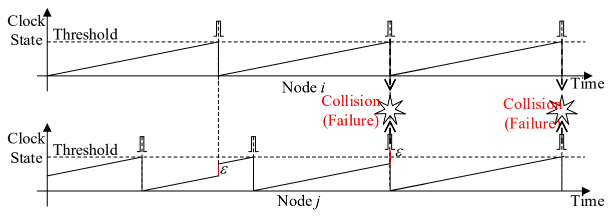

Fig. 2. Interacting state of PCO

In the interacting state of classic PCO, the state variable evolves as mentioned above, in addition, the state X_j is adjusted by ε upon reception of a *Pulse* from node i due to the nodes coupled with each other. A *Pulse* is generated and broadcasted immediately if the adjusted state variable exceeds the threshold, and the interaction between two non-identical oscillators are shown in Fig. 2. All the oscillators will broadcast a *Pulse* simultaneously when the synchronization is achieved. The interacting behavior of PCO can be described by

$$X_i(t) = \phi_{th} \rightarrow$$
$$X_j(t^+) = \begin{cases} X_j(t^-) + \varepsilon, if\left(X_j(t^-) + \varepsilon\right) < \phi_{th} \\ 0, if\left(X_j(t^-) + \varepsilon\right) \geq \phi_{th} \end{cases} \qquad (10)$$

where t^+ represents an infinitesimal time instant after t, similarly, the t^- represents an infinitesimal time instant before t. The ϕ_{th} is the pre-defined threshold state value, the increment function ε is referred to as phase response curve (PRC) [15],[17],[18].

The classic PCO is approved that TS can be achieved under the two following assumptions: The oscillators are identical. And there is no time delay during the pulse exchange among the oscillators. In addition, the *Pulse* of oscillators will be broadcasted when all oscillators achieve the TS.

However, in the realistic WSNs, the clock frequency is time-varying and non-identical due to the environment factors (e.g., temperature) and manufacturing tolerance. And the transmission time is needed to enable RF (Radio Frequency) module of sensor nodes transmit or receipt a packet. In addition, the RF module of sensor nodes either only works in the transmission mode or reception mode, and the pulse-exchange collision will occur when synchronization of classic PCO is achieved. As a result, the classic PCO cannot be applied to the realistic WSNs directly.

C. Desynchronization and Superframe Structure

Different from synchronization that nodes attempt to transmit the *Pulse* at the same time, *desynchronization* enables nodes broadcast the periodic *Pulse* as far as possible from all other nodes, namely, in a uniformly distributed fashion (i.e., TDMA fashion) [4].

The superframe structure of Fig. 3 , used in beacon-enabled operation of IEEE 802.15.4 MAC layer, is bounded by two neighboring beacons. There are three types of periods in superframe structure, namely, contention access period (CAP), contention-free period

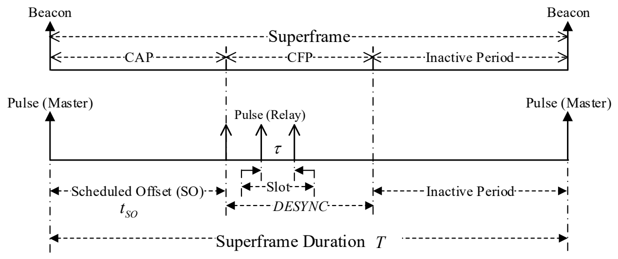

Fig. 3. IEEE 802.15.4 superframe (upper), proposed superframe (lower)

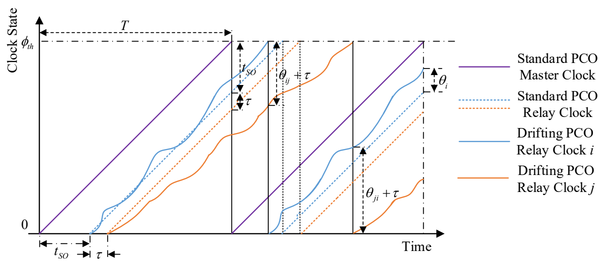

Fig. 4. The clock offset calculation among one standard PCO clock and two drifting PCO clocks by implementing DESYNC mechanism

(CFP) and inactive period. The CSMA (Carrier Sense Multiple Access) mechanism needs to be used in CAP to access a frequency channel, and there is no guarantee for each node to access the channel when it is needed. However, during the CFP, the specified slot is guaranteed to specified node, and TDMA mechanism is used in CFP, rather than CSMA. In addition, the inactive period enables the node enter power-saving mode (i.e., sleep mode) [6].

Similar to the superframe of IEEE 802.15.4 MAC layer, the proposed superframe (i.e., time synchronization cycle) in Fig. 3, bounded by neighboring *Pulse* broadcasted by master node, consists of three types of periods, namely, Scheduled Offset (SO), *DESYNC* and Inactive Period. The SO is used to transmit the data. The *DESYNC* is to enable WSN nodes broadcast the *Pulse* to achieve the *desynchronization* of WSNs. And the inactive period is used to enable the sensor nodes sleep to reduce the power consumption. In addition, at *n-th* time synchronization cycle, the time t_p of *Pulse* generated by *i-th* relay node is defined as

$$t_p = n \times T + t_{SO} + i \times \tau \qquad (11)$$

where t_{SO} is duration of SO, and τ means the slot duration representing the time between neighboring *Pulse* by relay nodes, and T is the superframe duration (i.e., time synchronization interval). Moreover, the slot duration is needed to enable the sensor nodes transmit or receipt the *Pulse* to avoid the pulse-exchange collision.

By introducing the DESYNC mechanism, the *i-th* relay clock of (6) is remodified to

$$C'[k] = t[k] + \theta[k] - (t_{SO} + i \times \tau) \qquad (12)$$

It is notable that the clock time of (12) will be the global reference clock (i.e., standard clock), if the clock is the standard clock, due to the zero skew and offset of

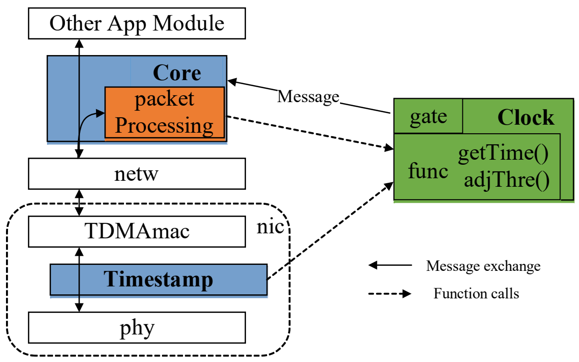

Fig. 5. General structure of WSN node

standard clock. Otherwise, the offset $\theta[k]$ and skew $\gamma[k]$ is updated based on the (8) and (9) to produce the drifting clock.

At *n-th* time synchronization cycle, the PCO clock time can be generated by comparing the instant clock time $C'[k]$ against multiple of threshold ($n\times\phi_{th}$), and this progress is modelled as

$$P[k]=C'[k]-n\times\phi_{th} \tag{13}$$

The timestamp will be generated, when broadcasted *Pulse* is receipted. And timestamp is defined as

$$P^{*}(t)=P(t)+\omega_{\eta}(t) \tag{14}$$

where $P^{*}(t)$ is real reception PCO clock time, and $\omega_{\eta}(t)$ means the measurement noise which is subjected to zero-mean Gaussian distribution with standard deviation σ_{η} .

As shown in [3],[12], the higher layer timestamp, the higher measurement noise, due to the interrupt and data processing. And the timestamp between the physical layer and MAC layer can realize the relatively balanced trade-off between the time synchronization accuracy and the expandability and cost of system.

The network topology, two-hop WSN of Fig. 4 consisting of one master (i.e., standard clock) and two relay nodes (i.e., drifting PCO Clock A is more stable than Clock B [9],[11]), is analyzed and simulated. Two kinds of clock correction mechanism are used, one mechanism is RBS-like clock correction mechanism, the drifting clock of relay node is corrected based on the measurement offset relative to the standard clock of master. Another algorithm is PCO-like clock correction mechanism, the drifting clock of relay is adjusted based on measurement offset relative to another relay node. And the measurement clock offset of *i-th* relay node relative to the master node, θ_i , is given by

$$\theta_i=P_i^{*}(t)-\phi_{th} \tag{15}$$

By introducing the DESYNC and presence of delay (i.e., transmission delay), the measurement offset of (15) is modified to

$$\theta_i=\left(\left(P_i^{*}(t)-\kappa\right)+\left(t_{SO}+n\times\tau\right)\right)-\phi_{th} \tag{16}$$

where κ is the transmission delay. Similarly, measurement clock offset of *j-th* (or *i-th*) relay about the *i-th* (or *j-th*) relay, θ_{ij} (or θ_{ji}), is defined respectively

$$\begin{aligned}\theta_{ij}&=\left(P_j^{*}(t)-\kappa\right)-\phi_{th_j}+\tau\\ \theta_{ji}&=\left(P_i^{*}(t)-\kappa\right)-0-\tau\end{aligned} \tag{17}$$

IV. SIMULATOR IMPLEMENTATION NODE STRUCTURE

The proposed simulator for PCO time synchronization is developed on previous work in [11] and simple simulator in [8]. In the simulator, three additional modules (i.e., PCO *Clock*, TDMA *mac* and relay *Core*) are developed to implement PCO clock and *desynchronization* mechanism. The *Pulse* is modelled by the 74-byte *SYNC* packet, based on the ZigBee and IEEE 1588 standards [1],[5],[6].

A. Node Structure

Besides the master and slave nodes of [11], the relay node, named *rnode* in simulator, has been developed to realize the functions of FFD. All three kinds of nodes are in the deployment area defined by the *World Manager* module, and *Connection Manager* is used to connect these nodes via a wireless channel [11]. In addition, several relay nodes can be simulated simultaneously in one network (i.e., *rnode*[0], *rnode*[1], *rnode*[2], …).

According to the IP structure, a WSN node of Fig. 5 is constructed with three additional modules, namely, PCO *Clock*, *Core* and *Timestamp*. And the *nic* (network interface card) of [11] is also modified to realize the TDMA function, rather than the CSMA function.

The PCO clock and the clock correction mechanism are simulated realistically by *Clock* module, and *Clock* also provides two interfaces to other modules [11]. One interface is an OMNeT++ gate, which is used to simulate the *Pulse*. Each time the clock reaches the threshold, a message will be sent to the *Core* module so that the *Core* module is able to transmit a *SYNC* packet based on the received message of the *Clock* module. The packet will be sent to *phy* layer to broadcast immediately via *Timestamp* module when a *SYNC* is received by *mac* layer from upper layer.

Another interface of *Clock* module is public member function [11]. By calling these interface functions, e.g., *getTimestamp()* and *adjustClock()*, the time of *Clock* module can be accessed, and clock can be adjusted more

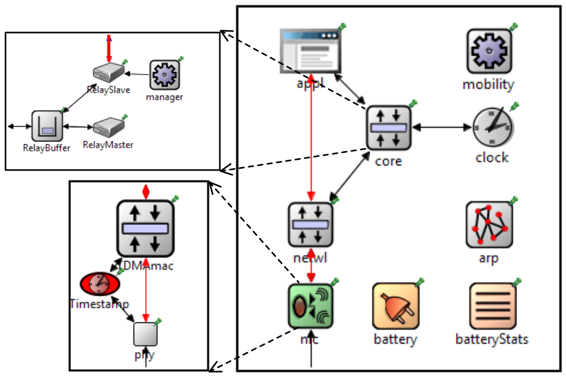

Fig.6. Implementation of relay node

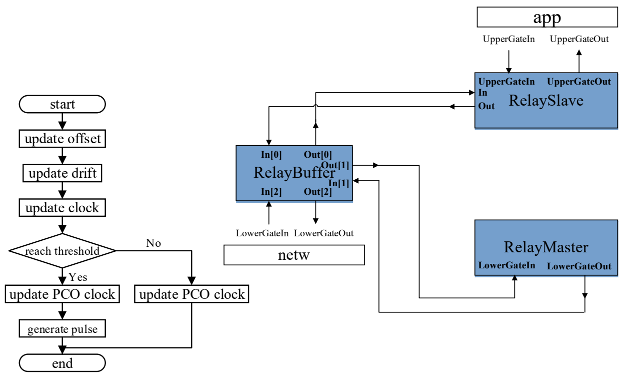

Fig. 7. Flowchart of PCO clock (right), structure of *Core* Module of relay node (left)

easily.

Due to the character of PCO, it is necessary to use the TDMA, rather than CSMA, to realize the behavior of PCO. In simple simulator by [8], a simple TDMA *mac* layer is developed to realize the function of the TDMA.

In addition, the *Timestamp* module of [11], modelling (14), has be implemented into the *nic* consisting of MAC (*TDMAmac*) layer and physical (*phy*) layer to model the timestamp. In the *Timestamp* module between the *mac* layer and *phy* layer, the packet will be sent to the *phy* layer immediately when a packet is received from the *mac* layer. Besides the packet will be transmitted to the upper layer when *Timestamp* receives a packet from the lower layer, a timestamp will be generated (i.e., *getTimestamp()* is called) by the *Timestamp* module to inform the *Clock* module the time instant of received *Pulse* time.

B. Implementation of PCO Clock

In the flowchart of Fig. 7, the clock time is updated based on the (12). And the PCO clock time is obtained by comparing the instant clock time against multiple of threshold (i.e., (13)).

Different from the master clock with zero offset and skew, the offset and skew of the drifting clock are updated with the specified clock frequency.

C. Implementation of Relay Node

The implementation of relay node is indicated in Fig. 6, similar to the master and slave nodes of [11], the *rnode* module is composed of several modules, namely, a compound module *nic* consisting of *phy* layer, *mac* layer and *Timestamp* module, a basic network layer *netwl* and general application layer *appl*.

Different from *Core* module of master and slave nodes in simulator by [11], the *Core* of relay node, compound module consisting of *RelayBuffer*, *RelaySlave* and *RelayMaster* modules, has been developed to realize the functions of both master and slave nodes. The structure of *Core* module is shown in Fig. 7. In *RelaySlave* module, the *handleMessage* function will be called to process the message from upper layer or *RelayBuffer* Module. If the received message is for itself, *RelaySlave* will process it, otherwise the received message will be sent out. The responsibility of *RelayBuffer* module is to transmit the message to respective modules, namely, *RelaySlave*, *RelayMaster* modules and network layer.

There are two kinds of working mechanisms in *RelayMaster* module of simulator. One mechanism is that all *RelayMaster* module works at the same time when network is built by using *scheduleAt()* of OMNeT++ API in the *initialization* function of *RelayMaster*. Another one is that once the time synchronization of previous hop is completed, *RelayMaster* module starts to work to active the synchronization of next hop. In addition, the *Clock* module of relay node is the drifting clock, rather than the standard clock.

TABLE I. SIMULATION CONFIGURATION

Symbol	Value	Unit
θ_0	0	Second
γ_0	2×10^{-5}	
σ_θ	$10^{-7}\left(ClockA\right),10^{-6}\left(ClockB\right)$	Second
σ_γ	$10^{-9}\left(ClockA\right),10^{-8}\left(ClockB\right)$	Second
σ_η	$10^{-8},10^{-2}$	Second
T	1	Second
τ	2.368	Millisecond
t_{SO}	100	Millisecond
κ	2.176	Millisecond

V. SIMULATION RESULTS

In this section, the impacts of parameter variations (i.e., initial drift, offset noise, drift noise and offset noise) on the PCO clock model have been evaluated on network topology (i.e., two-hop WSN consisting of one master and two relay nodes). The clock update frequency is configured to *32.768KHz* to simulate the typical real-time clock (RTC) frequency in WSNs, and the superframe duration, T , is set to *1* to model the PPS (Pulse Per Second). The configurations of simulations are summarized in Table 1.

The measurement offset and clock offset of Clock A (i.e., *rnode*[0]) are plotted in Fig. 8 with two kinds of

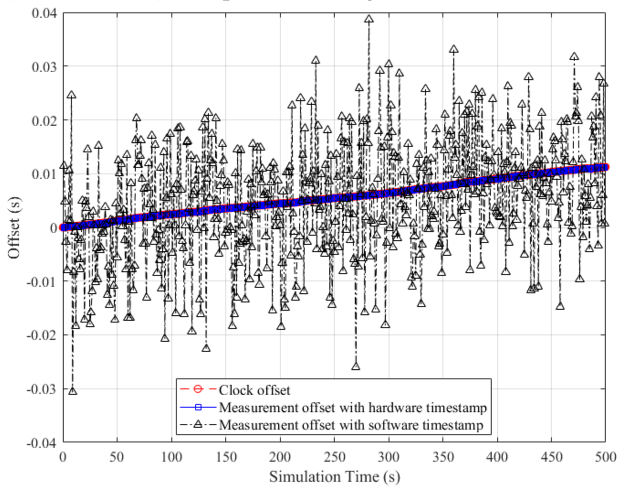

Fig. 8. Measurement offset of drifting PCO clock A relative to standard PCO clock with two kinds of timestamp mechanisms

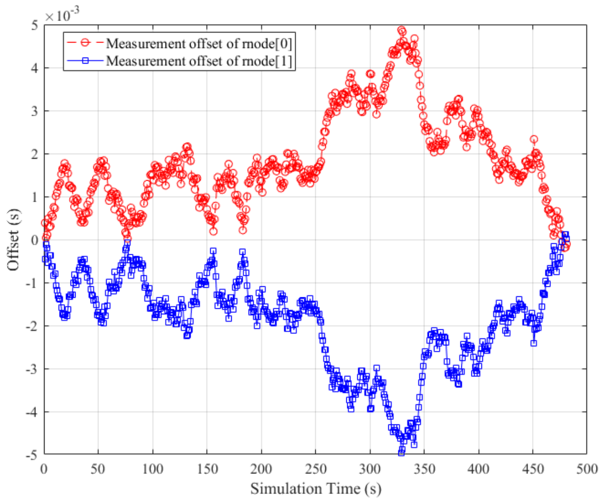

Fig. 9. Measurement offset of drifting PCO clock A (i.e., *rnode*[0]) and B (i.e., *rnode*[1]) by hardware timestamp

timestamp mechanisms, namely, hardware timestamp (i.e., $\sigma_\eta = 10^{-8}$) and software timestamp (i.e., $\sigma_\eta = 10^{-2}$). Both the measurement offset and clock offset rise with the increase of simulation time, and the measurement offset is equal to the clock offset approximately when the hardware timestamp is adopted. The accuracy of timestamp has a significant effect on the accuracy of measurement offset, and affects the performance of TS furtherly when the measurement offset is adopted to correct the drifting clock.

Fig. 9 plots the measurement offset of *rnode*[0] (or *rnode*[1]) about *rnode*[1] (or *rnode*[0]). The measurement offsets of *rnode*[0] and *rnode*[1] are symmetric approximately, since the measurement offsets of *rnode*[0] and *rnode*[1] are same theoretically. And these two measurement offsets are used to realize the PCO-like correction mechanism.

Therefore, the PCO clock is implemented into software simulator by DESYNC successfully, and the measurement clock offset calculation results coincide with the expectation.

VI. Conclusion

In this paper, the PCO clock is implemented into the software simulator by DESYNC algorithm successfully, and relay node is developed to realize the functions of FFD to enable the simulator more generalized and useful. Then, the clock offset calculation results coincide with the expectation. In addition, the code of this project is shared on the GitHub (i.e., [22]) to benefit the researchers and engineers in communication.

In the future, the clock correction mechanism adjusting the drifting PCO clock will be proposed and evaluated on the software simulator.

References

[1] IEEE standard for a precision clock synchronization protocol for networked measurement and control systems.

[2] Atmel. Real-time-clock calibration and compensation. Technical report, Atmel, 2014.

[3] Kendall Correll, Nick Barendt, and Michael Branicky. Design considerations for software only implementations of the ieee 1588 precision time protocol. 2005.

[4] Julius Degesys, Ian Rose, Ankit Patel, and Radhika Nagpal. DESYNC: Self-organizing desynchronization and TDMA on wireless sensor networks. Institute of Electrical and Electronics Engineers (IEEE), 2007.

[5] Fred Eady. Hands-On ZigBee. Elsevier Science, 2010.

[6] Shahin Farahani. ZigBee Wireless Networks and Transceivers. Newnes, 2011.

[7] Federico Ferrari, Andreas Meier, and Lothar Thiele. Accurate clock models for simulating wireless sensor networks. 2010.

[8] Antonio Franco. Lab 1 : Tdma. http://omikron.eit.lth.se/ETSN01/ETSN01/labs/. Accessed: 2017-04-21.

[9] Giada Giorgi and Claudio Narduzzi. Performance analysis of kalman-filter-based clock synchronization in IEEE 1588 networks. IEEE Transactions on Instrumentation and Measurement, 2011.

[10] Yao-Win Hong and A. Scaglione. A scalable synchronization protocol for large scale sensor networks and its applications. IEEE Journal on Selected Areas in Communications, 2005.

[11] Y. Huang, T. Li, X. Dai, H. Wang, and Y. Yang. TS2: a realistic IEEE1588 time-synchronization simulator for mobile wireless sensor networks. SIMULATION, 2015.

[12] Yiwen Huang. Ieee 1588 time synchronization optimization and omnet simulation for wsns. Master's thesis, Southwest University, 2014.

[13] Sami M. Lasassmeh and James M. Conrad. Time synchronization in wireless sensor networks: A survey. mar 2010.

[14] Renato E. Mirollo and Steven H. Strogatz. Synchronization of pulse-coupled biological oscillators. SIAM Journal on Applied Mathematics, 1990.

[15] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. Strogatz. Distributed synchronization in wireless networks. IEEE Signal Processing Magazine, 2008.

[16] Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. Ad Hoc Networks, 2005.

[17] A. Tyrrell, G. Auer, and C. Bettstetter. Emergent slot synchronization in wireless networks. IEEE Transactions on Mobile Computing, 2010.

[18] Alexander Tyrrell, Gunther Auer, and Christian Bettstetter. A synchronization metric for meshed networks of pulse-coupled oscillators. 2008.

[19] Nicola Varanese. Distributed Synchronization Algorithms for Wireless Sensor Networks. PhD thesis, 2011.

[20] Xiaodong Xian, Weiren Shi, and He Huang. Comparison of omnet++ and other simulator for wsn simulation. June 2008.

[21] Chaonong Xu and Zhulin An. The m&s model and its application in time synchronization in wireless multi-hop networks. Computer Applications and Software, 2010.

[22] Yan Zong. Time synchronization for wireless sensor networks. https://github.com/yan-zong/ts2. Accessed: 2017-04-21.