Modelling and Synchronisation of Delayed Packet-Coupled Oscillators in Industrial Wireless Sensor Networks [⋆]

Yan Zong, Xuewu Dai, Pep Canyelles-Pericas, Krishna Busawon, Richard Binns, Zhiwei Gao

Department of Mathematics, Physics and Electrical Engineering, Northumbria University, Newcastle upon Tyne, United Kingdom (e-mail: {yan.zong, xuewu.dai, pep.canyelles-pericas, krishna.busawon, richard.binns, zhiwei.gao}@northumbria.ac.uk).

Abstract: In this paper, a Packet-Coupled Oscillators (PkCOs) synchronisation protocol is proposed for time-sensitive Wireless Sensor Networks (WSNs) based on Pulse-Coupled Oscillators (PCO) in mathematical biology. The effects of delays on synchronisation performance are studied through mathematical modelling and analysis of packet exchange and processing delays. The delay compensation strategy (i.e., feedforward control) is utilised to cancel delays effectively. A simple scheduling function is provided with PkCOs to allocate the packet transmission event to a specified time slot, by configuring reference input of the system to a non-zero value, in order to minimise the possibility of packet collision in synchronised wireless networks. The rigorous theoretical proofs are provided to validate the convergence and stability of the proposed synchronisation scheme. Finally, the simulations and experiments examine the effectiveness of PkCOs with delay compensation and scheduling strategies. The experimental results also show that the proposed PkCOs algorithm can achieve synchronisation with the precision of $26.3\mu s$ (1 tick).

Keywords: Time Synchronisation, Packet-Coupled Oscillators, Mathematical Modelling, Delay, Pulse-Coupled Oscillators, Wireless Sensor Networks, Internet of Things.

1. INTRODUCTION

Nowadays, the wide use of Internet of Things (IoT) in various sectors, such as transportation systems, electrical grids, smart cities, and so on, is significantly improving the quality of our life. As a fundamental technology of IoT, Wireless Sensor Networks (WSNs) plays an important role in industrial systems by monitoring and collecting surrounding context and environmental information. To guarantee such time-sensitive and mission-critical industrial applications work properly, Time Synchronisation (TS), which is the key enabling technology of WSNs, is required to provide a common timescale for local clocks of WSN nodes.

Due to the importance of time synchronisation, many synchronisation protocols have been proposed for WSNs in the communication engineering community. They are well represented by TS algorithms such as Precision Time Protocol (PTP) (IEEE 1588 (2008)) and Reference Broadcast Synchronisation (RBS) of Elson et al. (2002). It is well known that in the PTP synchronisation scheme, error in each hop is accumulated, and the precision of such a synchronisation algorithm degrades by increasing hop distance. Compared to the PTP protocol, RBS achieves higher performance at the expense of requiring more packet transmission and reception, while the Radio Frequency (RF) module is the most energy-consuming component of a wireless sensor node, which would increase energy complexity (Sivrikaya & Yener (2004)).

Next, owing to the significance of synchronisation in various disciplines, it has naturally attracted attention from mathematics and physics communities, where it is applied to networked oscillators consisting of a set of oscillators coupled with each other through a certain network topology. Inspired by the synchronous flashing of fireflies observed in certain parts of southeast Asia, a typical networked oscillators model, the Pulse-Coupled Oscillators (PCO), is derived by Mirollo & Steven (1990). As a result of its inherent simplicity and scalability, this biological model guarantees the possibility of synchronisation in a network consisting of N oscillators under simple networking protocols; its discretisation also lets this model apply to wireless sensor networks easily.

However, the PCO's ideal coupling mechanism restricts its employment to any real network. Specifically, in PCO, oscillators' states are instantly known by their coupled oscillators and no delays between connected oscillators, which is impossible in industrial wireless networks. Thus, it needs to be improved for application in WSNs.

As a result, in this paper, inspired by PCO, the Packet-Coupled Oscillators (PkCOs) is proposed to achieve time synchronisation in WSNs. The main focus is particularly

[⋆] Yan Zong gratefully acknowledges financial support from the University of Northumbria at Newcastle *via* a postgraduate research studentship.

laid on the problem of the ideal coupling mechanism (i.e., packet exchange delay and processing delay in WSNs).

In PkCOs, an oscillator works in either free-running or interactive mode. In the free-running mode, the oscillator behaves as an isolated oscillator, whose state P linearly increases from zero to the threshold. Once P reaches the threshold value, the oscillator fires (which means a *Sync* packet is being generated and broadcasted) while state P is reset. This procedure keeps repeating continuously during the free-running mode. When the oscillator receives a *Sync* from another oscillator, it moves to the interactive mode, where generating a local timestamp, and adjusting its state by offset estimated from the timestamp, after which it returns to the free-running mode.

Typically, in embedded systems, the clock module of a WSN node is implemented by a counter register (or a chain of counters) driven by the crystal oscillator. The counter is reset to zero when it matches the pre-defined threshold value. This periodic resetting feature is similar to PkCOs' firing-resetting behaviour, where the oscillator fires and resets its state. Thus, the clock module in embedded systems can be described as an uncoupled PkCOs' oscillator, and periodic packet transmission in WSNs is equivalent to *Sync* firing in PkCOs.

1.1 Related Work

In PCO, the presence of pulse coupling delay results in the occurrence of infinite feedback and instability of networks (Hong & Scaglione (2005)). To prevent WSNs from being unstable, several works (e.g., Hong & Scaglione (2005), Pagliari & Scaglione (2011)) introduce a refractory period where the reception of *Pulse*s has no effect on local oscillator's state, and refractory period must be larger than twice coupling delay (Gentz et al. (2016), Tyrrell et al. (2008)). However, Zong et al. (2018a) shows that, in multi-hop networks, PCO synchronisation error (resulting from coupling delay) accumulates over multiple hops. Therefore, TS error may be larger than the refractory period, which is utilised to guarantee the network's stability. Moreover, due to the impossibility of real-time computing, the effects of processing delay (which is required for calculation) need to be considered in the synchronisation protocol.

In the literature (e.g., see Tyrrell et al. (2010), Proskurnikov & Cao (2017), Wang et al. (2012), Hong & Scaglione (2005), Pagliari & Scaglione (2011) and An et al. (2011)), once a WSN consisting of numerous sensor nodes achieves synchronisation, a large number of *Sync* packets are sent to the wireless channel simultaneously. However, each time only a single wireless packet is permitted to be transmitted in the channel. As a consequence, packet collision occurs, and no packets can be received successfully. To address this issue, Ashkiani & Scaglione (2012) and Gentz et al. (2016) propose a scheduling scheme to guarantee *Sync* packets to be transmitted in a distributed fashion, thereby minimising the possibility of packet collision. However, the dithered quantisation function, which is used for scheduling scheme, is not naturally integrated into WSN nodes. The microprocessor in sensor nodes has to provide extra hardware resources to realise dithered quantisation function, which also increases energy consumption, puts a strain in batteries, and reduces the autonomy of nodes. Furthermore, the use of software floating-point unit in dithered quantisation function may reduce its calculation precision.

1.2 Contribution and Paper Organisation

In this paper, through mathematical modelling of packet exchange and processing delays in both time dimension and state dimension, the delays' effects on PkCOs in these dimensions are analysed. Delay compensation strategy (i.e., feedforward control) is introduced in the packet coupling scheme to cancel the impacts of two delays, rather than simply introducing a refractory period to maintain the network's stability, which cannot improve synchronisation performance effectively. Also, the packet coupling mechanism allows the natural allocation of time slots for *Sync* packet transmission, and the simple scheduling function of packet coupling mechanism reduces the need and consumption of hardware resources, compared to scheduling functions in Ashkiani & Scaglione (2012) and Gentz et al. (2016).

In addition, the convergence and stability of PkCOs are analysed under two different scenarios (i.e., with and without feedforward control). The effectiveness of the proposed delay compensation scheme is also evaluated in simulation and hardware-based experiments. The experimental results show that, by using the proposed synchronisation strategy, the time synchronisation precision of $26.3\mu s$ (1 tick) is achieved under clock resolution of $30.5\mu s$; and packet coupling scheme is also capable of allocating *Sync* packet transmission to a specified time slot.

The rest of this paper is organised as follows: Section 2 presents the PkCOs algorithm and mathematical modelling of delays in the time dimension. Section 3 details the implementation of a delay compensation scheme in the state dimension. The rigorous theoretical proofs for the algorithm's convergence and stability are also provided. Additionally, the scheduling function is briefly introduced in this section. Section 4 and Section 5 present simulation and experimental results. Finally, conclusions are drawn in Section 6.

2. PROBLEM FORMULATION

In embedded systems, the crystal oscillator is widely used as a clock source due to its trade-off between high-quality signal accuracy and cost. In the following, the crystal oscillator-based clock is modelled as a free-running oscillator of PkCOs. An identical clock model is considered since the crystal oscillator provides a reliable clock signal. To understand how clock and delays are modelled, their equations are derived and briefly presented.

2.1 Modelling An Identical Clock

Referring first to the case of a perfect clock, in WSNs, the clock system is constructed from two parts: (i) a crystal oscillator, ticking at the nominal frequency $f_0 = 1/\tau_0$ where τ_0 is crystal oscillator period, and (ii) a counter counts the number of ticks generated by a crystal oscillator. Specifically, through the process of counting, the periodic signal produced by a crystal oscillator is converted into an integer that is increased by one per crystal oscillator period. Once the cumulative value of a counter matches the pre-defined threshold, it is reset and starts counting from zero again, after which this procedure repeats. Let $t[n]$ denote the time reported by such a crystal oscillator-based clock at the n-th clock event, $t[n]$ is calculated as

$$t[n] = n \times \tau_0 = \frac{n}{f_0}. \tag{1}$$

For an ideal crystal oscillator-based clock whose frequency is exactly the same as the nominal frequency f_0, $t[n]$ is accurate and referred to as the *reference time*. Such a clock is also called the *reference clock* or *master clock* thereafter.

In the PkCOs synchronisation method, the time synchronisation cycle T is much greater than the clock period τ_0 (i.e., $T \gg \tau_0$), it is reasonably assumed that the clock is updated m_0 times during a single synchronisation cycle, following $T = m_0 \times \tau_0$. Taking the clock's periodic resetting behaviour into account, the dynamics of the master clock's state $P_0[n]$ are represented as

$$P_0[n] = t[n] - \sum_{j=0}^{k} \varphi_0[j], \tag{2}$$

where φ_0 is the master clock threshold, $k = \lfloor n/m_0 \rfloor$ represents how many clock resetting has occurred from $n = 0$ to the n-th clock event, where floor function $\lfloor n/m_0 \rfloor$ denotes the greatest integer less than or equal to n/m_0. Recalling that synchronising clocks occurs when a clock resets, k also denotes the number of synchronisation cycles so far. In other words, the clock is at the k-th synchronisation cycle.

However, in reality, the state of real crystal oscillator-based clock $P_i[n]$ is different from ideal state $P_0[n]$ due to manufacturing tolerance and environmental conditions (e.g., temperature). The difference between ideal clock's state $P_0[n]$ and $P_i[n]$ reported by the i-th unperfect clock is referred to as the clock offset $\theta_i[n]$:

$$\theta_i[n] = P_i[n] - P_0[n]. \tag{3}$$

Clearly, in Fig. 1, the amount of offset $\theta_i[n]$ of the crystal oscillator-based clock is moderate throughout experimental recordings at room temperature. By using the linear polynomial fitting, it is shown that the clock possesses a smaller clock skew $\gamma_i[n]$ [1], which is only around 1.4 ppm (parts per million). Thus, it is reasonable to assume that the frequency $f_i[n]$ of the crystal oscillator-based clock is identical to nominal clock frequency $f_0[n]$ (i.e., $\gamma_i[n] = 0$). Thus, only the crystal oscillator's phase contributes to the inaccuracy of local clocks. The phase noise of the i-th clock is modelled by a random process $\phi_i[n]$, representing all the instant phase deviation from $t[0]$ to time $t[n]$. The clock state $P_i[n]$ at the n-th clock event is modelled as

$$P_i[n] = t[n] + \phi_i[n] - \sum_{j=0}^{k} \varphi_i[j], \tag{4}$$

where φ_i is the i-th clock threshold. The local unperfect clock is also called the *slave clock*. For the purpose of analysis, the recursive state equation of slave clock, which

[1] clock skew $\gamma_i[n]$ is defined as the normalised difference between the i-th clock frequency $f_i[n]$ and *reference clock* frequency f_0, yielding $\gamma_i[n] = (f_i[n] - f_0)/f_0$.

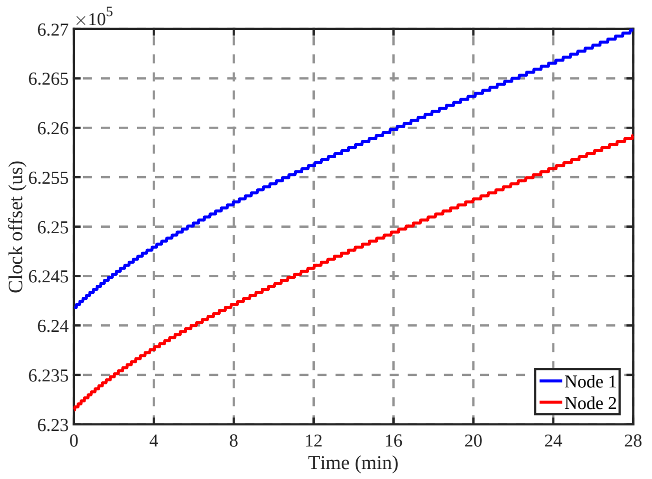

Fig. 1. Offsets of 32.768kHz crystal oscillator-based clocks at wireless nodes (Atmel SMART SAM R21).

describes the behaviour of a clock with identical frequency, is written as

$$\theta_i[n+1] = \theta_i[n] + \omega_{\theta_i}[n], \tag{5}$$

where white Gaussian random process $\omega_{\theta_i}[n] = \phi_i[n+1] - \phi_i[n]$ denotes offset noise.

2.2 Mathematical Modelling of Delays

In the PkCOs' packet coupling scheme, at the k-th synchronisation cycle, when the slave node receives the master's *Sync* after packet exchange delay κ_i, it generates a timestamp $\hat{P}_i[k]$ based on the local clock. While, due to the impossibility of real-time computing, the processing delay η_i is required for a processor to estimate offset $\hat{\theta}_i[k]$. Once the offset estimate $\hat{\theta}_i[k]$ is determined, the correction input $u_i[k]$ is employed to drifting clock (5) instantly.

Obviously, the inaccuracy of clock correction action $u_i[k]$ results from following two factors: (i) packet exchange delay κ_i including send time, access time, propagation time and receive time; and (ii) processing delay η_i which is required for a processor to compute clock correction input from the local timestamp. These factors are taken into accounts as shown in Fig. 2.

Packet Exchange Delay: As shown in Fig. 2, the perfect clock fires and transmits a *Sync* packet at t_k, which indicates the start of the k-th synchronisation cycle. The i-th node receives the *Sync* at $t_k + \kappa_i$, due to the transmission delay, propagation delay, and other factors explained above. The packet exchange delay κ_i is assumed as Gaussian random process with non-zero mean of $\bar{\kappa}_i$ and finite variance of $\sigma^2_{\kappa_i}$. That is $\kappa_i \sim (\bar{\kappa}, \sigma^2_{\kappa_i})$.

When the i-th node receives the *Sync*, a local timestamp $P_i[n]$ is generated by reading node's counter, we get:

$$\hat{P}_i[k] = \hat{P}_i^{t_k+\kappa_i}. \tag{6}$$

Once local timestamp $\hat{P}_i[k]$ is obtained, the offset estimate $\hat{\theta}_i[k] = \theta_i[k] + \kappa_i$ between two clocks can be determined directly via the following expression:

$$\hat{\theta}_i[k] = \begin{cases} \hat{P}_i[k] & if \hat{P}_i[k] < \dfrac{\varphi_i}{2} + \kappa_i \\ \hat{P}_i[k] - \varphi_i & if \hat{P}_i[k] \geq \dfrac{\varphi_i}{2} + \kappa_i \end{cases}. \tag{7}$$

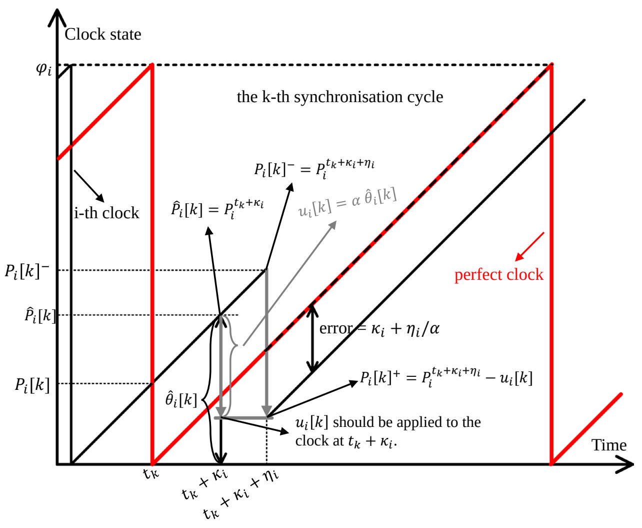
Clock state
φ_i
the k-th synchronisation cycle
i-th clock
$P_i[k]^- = P_i^{t_k+\kappa_i+\eta_i}$
$\hat{P}_i[k] = P_i^{t_k+\kappa_i}$
$u_i[k] = \alpha\,\hat{\theta}_i[k]$
perfect clock
$P_i[k]^-$
$\hat{P}_i[k]$
error $= \kappa_i + \eta_i/\alpha$
$P_i[k]$
$P_i[k]^+ = P_i^{t_k+\kappa_i+\eta_i} - u_i[k]$
$\hat{\theta}_i[k]$
$u_i[k]$ should be applied to the clock at $t_k + \kappa_i$.
Time
t_k
$t_k + \kappa_i$
$t_k + \kappa_i + \eta_i$

Fig. 2. Example of the slave clock correction.

Processing Delay: Next, the offset estimate should be employed to a local clock at $t_k + \kappa_i$:

$$P_i[k]^+ = P_i[k]^- - u_i[k], \tag{8}$$

where $P_i[k]^+/P_i[k]^-$ is the i-th clock's state after/before it is corrected at the k-th synchronisation cycle. However, due to the limitation of hardware resources, the processing delay η_i is required for clock offset estimation. Thus, the local clock actually is corrected at the time $t_k + \kappa_i + \eta_i$, yielding

$$P_i[k]^+ = P_i^{t_k+\kappa_i+\eta_i} - u_i[k], \tag{9}$$

where the processing delay η_i is also considered as Gaussian random process with non-zero mean $\bar{\eta}_i$ and finite variance of $\sigma^2_{\eta_i}$, which is $\eta_i \sim (\bar{\eta}, \sigma^2_{\eta_i})$.

In (3) and (4), it is indicated that the existence of clock offset leads to the clock state's inaccuracy. Therefore, the clock correction action to clock state is essentially equivalent to the application of correction input $u_i[k]$ to clock offset. For the purpose of analysis, the clock correction algorithm (9) can be rewritten as

$$\theta_i[k]^+ = \theta_i^{t_k+\kappa_i+\eta_i} - u_i[k], \tag{10}$$

where $\theta_i[k]^+$ is the clock offset after it is adjusted.

Until now, it is clear that due to the packet exchange delay and processing delay in the time dimension, the direct employment of complete offset estimate to a local clock, yielding $u_i[k] = -\hat{\theta}_i[k]$, results in degradation of PkCOs synchronisation performance.

3. PACKET COUPLING SCHEME WITH FEEDFORWARD CONTROL

To avoid the occurrence of over-correction and synchronisation performance's degradation, this paper applies a proportional controller scheme for better synchronisation performance. Moreover, by effectively compensating the effects of delays in the state dimension, the feedforward control is capable of further improving synchronisation precision.

Specifically, at the k-th synchronisation cycle, once the clock offset $\hat{\theta}_i[k]$ is measured, the local clock is adjusted by a partial amount of offset estimate, following $u_i[k] = -\alpha\hat{\theta}_i[k]$ where α is the coefficient of the proportional controller.

In this paper, since the frequency of clocks is assumed to be the same as nominal frequency f_0 (i.e., $f_i = f_0$), these do not contribute extra value to the offset. Assuming the clock period is sufficiently small, the value of clock state increment within delay duration (which is in the state dimension) is equal to the value of delays (which is in the time dimension). In other words, if clock resolution is sufficiently high, delay in the time dimension is equivalent to delay in the state dimension, and the following expressions are obtained:

$$P_i^{t_k+\kappa_i} - P_i^{t_k} = \kappa_i \tag{11}$$

for the packet exchange delay and

$$P_i^{t_k+\kappa_i+\eta_i} - P_i^{t_k+\kappa_i} = \eta_i \tag{12}$$

for the processing delay.

Therefore, the packet exchange delay and processing delay in the state dimension can be written in following equations

$$\begin{cases} \theta_i[k+1] = \theta_i[k] + u_i[k] + \omega_{\theta_i}[n] \\ \hat{\theta}_i[k] = \theta_i[k] + \kappa_i \\ u_i[k] = \alpha(0 - \hat{\theta}_i[k]) - \eta_i \end{cases} . \tag{13}$$

It is important to note that, ideally, the local clock offset should be corrected to target value of $(\theta^{t_k+\kappa_i} - \alpha\hat{\theta}_i[k])$ at $(t_k + \kappa_i)$. However, in practice, the i-th clock offset can only be adjusted to $(\theta^{t_k+\kappa_i+\eta} - \alpha\hat{\theta}_i[k])$ at the time $(t_k + \kappa_i + \eta_i)$, owing to the processing delay η_i which is required for clock offset estimation. This means that the value of extra η_i is unintentionally used to correct the local clock, this procedure is modelled as $u_i[k] = \alpha\hat{\theta}_i[k] - \eta_i$, as indicated in (13).

Theorem 1. For any $\alpha \in (0, 2)$, the clock offset in algorithm (13) converges to $(-\bar{\kappa}_i - \bar{\eta}_i/\alpha)$ for all initial conditions.

Proof. The characteristic equation of the closed-loop system is

$$\lambda - 1 + \alpha = 0.$$

The eigenvalue of the characteristic equation above is

$$\lambda = 1 - \alpha.$$

As specified in Ogata (2009), if the coefficient α is within the unit circle (i.e., $\alpha \in (0, 2)$), the system achieves steady state. Moreover, using standard techniques (such as z-transformation and final value theorem), it is easy to find that the clock offset $\theta_i[k]$ asymptotically converges to a certain value (i.e., asymptotic error) being satisfied:

$$\begin{aligned} \underset{k\to\infty}{\theta_i[k]} &= \lim_{z\to 1}(z-1)\left(-\frac{z(\alpha\bar{\kappa}_i + \bar{\eta}_i)}{(z-1+\alpha)(z-1)} + \frac{(\sigma^2_{\theta_i} - \sigma^2_{\kappa_i} - \sigma^2_{\eta_i})}{(z-1+\alpha)}\right) \\ &= -\bar{\kappa}_i - \frac{\bar{\eta}_i}{\alpha}. \end{aligned} \tag{14}$$

Clearly, delays in the time dimension lead to the asymptotic error of steady state. However, by introducing feedforward control input $\mu_i = \bar{\eta}_i - \alpha\bar{\kappa}_i$ to the closed-loop control system (13), it is possible to obtain a zero asymptotic error. This also means that, in practice, the synchronised wireless nodes will transmit *Sync* packets simultaneously, and the interference of wireless packet will occur. To avoid the collision of *Sync* packet transmission, the time slot t_{d_i} is allocated to the i-th node to guarantee only the corresponding node can access the wireless channel for *Sync* packet transmission.

Finally, a more advanced closed-loop control system, with features of delay compensation and avoidance of packet collision, is written as

$$\begin{cases} \theta_i[k+1] = \theta_i[k] + u_i[k] + \omega_{\theta_i}[n] \\ \hat{\theta}_i[k] = \theta_i[k] + \kappa_i \\ u_i[k] = \alpha(t_{d_i} - \hat{\theta}_i[k]) - \eta_i + \mu_i \end{cases} . \tag{15}$$

Theorem 2. For any $\alpha \in (0, 2)$, the offset in algorithm (15) tracks t_{d_i} for all initial conditions.

Proof. Using standard techniques (such as z-transformation), it becomes almost natural to find transfer function $G(z)$ of the closed-loop control system (15) from $t_{d_i}(z)$ to $\theta_i(z)$:

$$G(z) = \frac{\theta_i(z)}{t_{d_i}(z)} = \frac{\alpha}{(z - 1 + \alpha)}.$$

Therefore,

$$G(1) = 1. \tag{16}$$

This means that, by using the feedforward control, in the steady state, the clock offset $\theta_i[k]$ always tracks desired reference input t_{d_i}. In other words, in the synchronised state, the i-th node transmits the *Sync* at the allotted time slot t_{d_i}.

4. SIMULATION RESULTS

To validate theoretical results presented in the preceding section, simulations are conducted in Simulink employing parameters obtained from real implementation settings. In the simulations, an identical clock, which corresponds to the crystal oscillator-based clock on Node 1 in Fig. 1, is simulated. The clock offset is subject to a random perturbation with variance of 244.4990×10^{-12}, which is taken from Zong et al. (2020a). The synchronisation cycle is configured to $1s$. In addition, the mean values of packet exchange delay and processing delay are set to $349\mu s$ and $514\mu s$, respectively (see Zong et al. (2020b)). The network consisting of one master and a single slave node is used. The configurations of simulations are summarised in Table 1.

Table 1. Simulation configurations

Symbol	Value	Unit
$\theta_i[0]$	0.6	Second (s)
$\sigma^2_{\theta_i}$	244.4990×10^{-12}	
T	1	Second (s)
$\bar{\kappa}_i$	349	Microsecond (μs)
$\bar{\eta}_i$	514	Microsecond (μs)
α	0.5	
t_{d_i}	$0, 9.15$	Microsecond (μs)

Fig. 3 shows the evolution of clock offset under the closed-loop control system (13). Clearly, it is demonstrated that, without the aid of feedforward control, the clock offset converges to an asymptotic error $-\bar{\kappa}_i - \bar{\eta}_i/\alpha = -1.212ms$, after several synchronisation cycles.

Moreover, Fig. 4 indicates that, by using feedforward control and non-zero reference input, in the steady state,

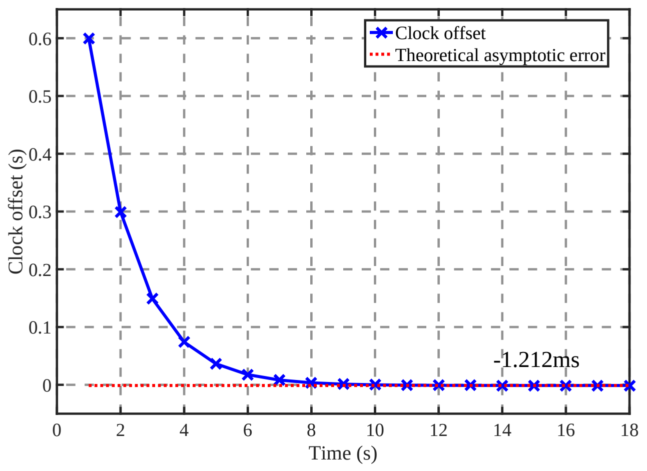

Fig. 3. Evolution of clock offset on the closed-loop control system without delay compensation strategy.

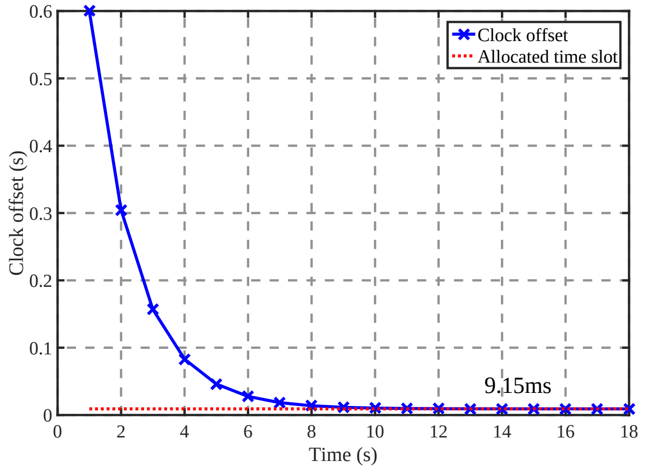

Fig. 4. Evolution of clock offset on the closed-loop control system with delay compensation and avoidance of packet collision strategies.

the clock offset tracks reference input $t_{d_i} = 9.15ms$. This means that, in practice, the i-th slave node transmits the *Sync* at its allotted time slot t_{d_i} within each synchronisation cycle.

5. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed PkCOs synchronisation scheme with features of delay compensation and packet collision avoidance, it is implemented on the Atmel SMART SAM R21 board developed in Zong et al. (2018b). The experiments consider a two-node wireless network consisting of one master node and a single slave node. The master node is connected to a Trimble ThunderBolt E GPS Disciplined Clock for providing the PPS (Pulse Per Second) signal. Once it receives the PPS signal from GPS, the master issues an external interrupt to toggle the GPIO PA19 pin firstly. Meanwhile, it broadcasts a *Sync* packet to the wireless channel directly.

As for the slave node, the PkCOs state is represented by a 32-bit COUNT register of the Real-Time Clock (RTC) module, which uses a 32.768kHz external crystal oscillator as the clock source. When the COUNT register matches

32767 of the compare register COMP, it is reset to zero. At the same time, an RTC interrupt is triggered, and then, a new cycle begins. In the RTC interrupt handler, the GPIO PA19 pin is toggled first; then, a wireless *Sync* packet is transmitted to the channel directly.

In the slave's receive mode, once addressing fields of the received *Sync* packet match local addresses, an AMI (Address Match Interrupt) is issued to generate a timestamp by reading the COUNT register. After the timestamp is generated, the local clock is corrected by using estimated clock offset as described in (7).

The signal outputs from master and slave nodes are attached to a logic analyser that records the time of master's *Sync* packet transmission and slave clock's firing. Also, the difference between *Sync* packet transmission time and signal toggled in the RTC handler of slave node is calculated for analysing synchronisation performance. To guarantee a fair comparison of performance, the precision in the synchronised state, denoted by Δ_i, is defined as the difference among the i-th clock fires time $t_{P_i=\varphi_i}$, allocated time slot t_{d_i} and perfect clock fires time $t_{P_i=\varphi_0}$ of the master node, chosen as the reference. That is

$$\Delta_i := t_{P_i=\varphi_0} - t_{d_i} - t_{P_i=\varphi_i}. \tag{17}$$

Many time synchronisation protocols are proposed and implemented in several hardware platforms that select different types of crystal oscillator as clock sources; thus, the direct comparison of these results is unfair, and obtained conclusions are also questionable (Masood et al (2017)). In addition to Δ_i, we also adopt the clock tick as an evaluation metric, in order to compare the performance of multiple synchronisation algorithms in different hardware environments. This metric relates to maximum achievable accuracy on the underlying platform, leading to reasonable results.

Fig. 5 presents the evolution of synchronisation precision over time. Time is measured in terms of the number of synchronisation cycles. The precision converges to some extent to a steady state with an asymptotic error, which is close to the theoretical asymptotic error of ($\bar{\kappa}_i+\bar{\eta}_i/\alpha = 1.1895ms$). However, due to the limitation of clock resolution (i.e., the clock period $\tau_0 = 30.5\mu s$), the assumption of (11), (12) cannot be met, and the value of delays (i.e., $\bar{\kappa}_i + \bar{\eta}_i/\alpha$) is impossible to be integer multiple of the clock period. Thus, there always exists a bias between the value for compensating delays, saying $26.6\mu s$.

Moreover, in the experiments, only the clock offset is corrected, and the local clock's frequency is slightly different from that of the GPS clock. The existence of clock skew leads to sawtooth behaviour of achieved synchronisation precision. In future work, the clock frequency correction scheme will be applied to local clocks, in order to further improve synchronisation performance.

In Fig. 6, by employing feedforward control μ_i to packet coupling scheme, the achieved synchronisation accuracy improves to $26.3\mu s$ (1 ticks). Even though Degesys et al. (2007) utilises MAC-level timestamping mechanism to generate timestamp, the achieved synchronisation is only around $100\mu s$ (3 ticks). In addition, Elson et al. (2002) reports that RBS is capable of achieving synchronisation with the precision of about $11\mu s$ (11 ticks). By using similar hardware configurations, the proposed synchronisation method has a clear potential to overcome the results in Elson et al. (2002) and Degesys et al. (2007).

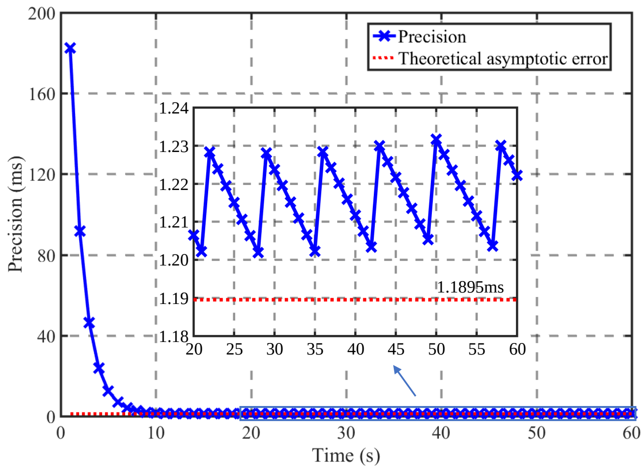

Fig. 5. Evolution of synchronisation precision Δ_i under the PkCOs scheme without delay compensation ($\alpha = 0.5, \bar{\kappa}_i = 518.5\mu s, \bar{\eta}_i = 335.5\mu s$).

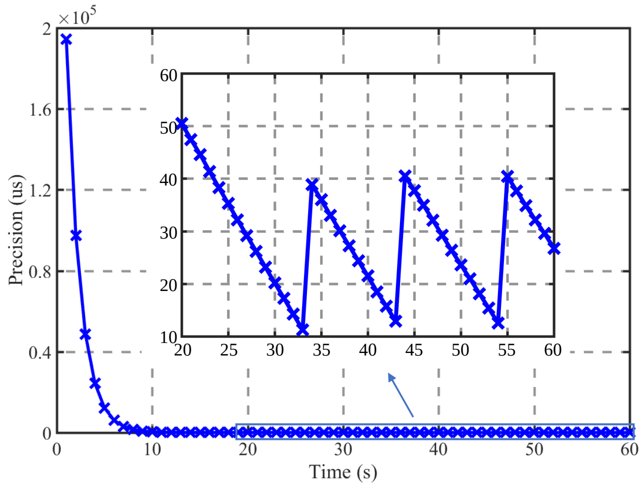

Fig. 6. Evolution of synchronisation precision Δ_i under the PkCOs protocol with delay compensation and avoidance of packet collision ($\alpha = 0.5, \bar{\kappa}_i = 518.5\mu s, \bar{\eta}_i = 335.5\mu s$).

Fig. 7 demonstrates results of the utilisation of feedforward control and non-zero reference input $t_{d_i} = 12.81ms$ in the system. Clearly, in the steady state (i.e., synchronised state), the time of the i-th clock packet transmission event relative to the master node is $12.81ms$. In other words, the i-th node's clock always tracks reference input t_{d_i}. This means that the proposed packet coupling scheme can naturally allocate the time slot for *Sync* packet transmission, thereby minimising the possibility of packet collision.

6. CONCLUSION

In this paper, through mathematical modelling and experimental validation of packet exchange delay and processing delay, the impacts of two delays on PkCOs are studied. The analysis includes packet exchange delay and processing delay in both time and state dimensions. The feedforward

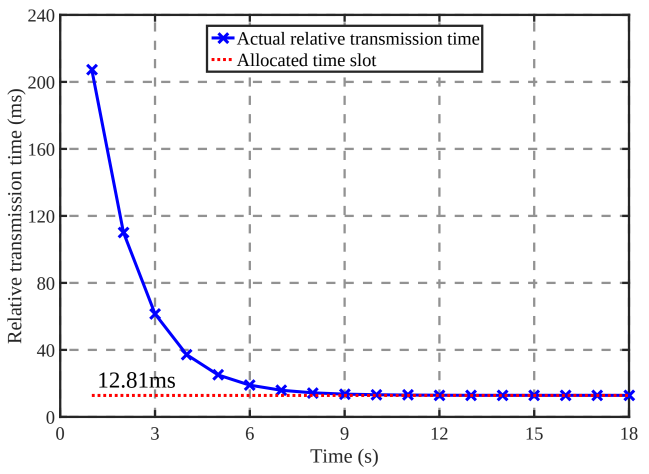

Fig. 7. Transmission time relative to the master node during each synchronisation cycle ($t_{d_i} = 12.81ms$).

control (i.e., delay compensation strategy) is introduced in the packet coupling scheme to cancel delays' effects. Meanwhile, the proposed simple packet coupling scheme allows the natural allocation of a time slot for *Sync* packet transmission. The rigorous theoretical proofs validate the effectiveness of the proposed PkCOs scheme with delay compensation and packet avoidance characteristics. Finally, the experimental results show that, in the PkCOs synchronisation protocol, the utilisation of feedforward control can achieve synchronisation with the precision of $26.3\mu s$ (1 tick). It is also capable of allocating the *Sync* packet to a specified time slot, thereby minimising the possibility of packet collision in wireless sensor networks.

In future work, the hardware resource consumption of PkCOs requires to be analysed in the experiments. Moreover, it is also necessary to study synchronisation precision of the PkCOs protocol in multi-hop large-scale WSNs.

REFERENCES

An, Z., Zhu, H., Li, X., Xu, C., Xu, Y. and Li, X. (2011). 'Nonidentical Linear Pulse-Coupled Oscillators Model with Application to Time Synchronization in Wireless Sensor Networks'. *IEEE Transactions on Industrial Electronics*, volume 58, issue 6, 2205–2215.

Ashkiani, S. and Scaglione, A. (2012). 'Pulse Coupled Discrete Oscillators Dynamics for Network Scheduling'. in *Annual Allerton Conference on Communication, Control and Computing.*

Degesys, J., Rose, I., Patel, A. and Nagpal, R. (2007). 'DESYNC: Self-Organizing Desynchronization and TDMA on Wireless Sensor Networks'. in *International Symposium on Information Processing in Sensor Networks.*

Elson, J., Girod, L. and Estrin, D. (2002). 'Fine-Grained Network Time Synchronization Using Reference Broadcasts'. *ACM SIGOPS Operating Systems Review*, volume 36.

Gentz, R., Scaglione, A., Ferrari, L. and Hong, Y. (2016). 'PulseSS: A Pulse-Coupled Synchronization and Scheduling Protocol for Clustered Wireless Sensor Networks'. *IEEE Internet of Things Journal*, volume 3, issue 6, 1222–1234.

Hong, Y. and Scaglione, A. (2005). 'A Scalable Synchronization Protocol for Large Scale Sensor Networks and Its Applications'. *IEEE Journal on Selected Areas in Communications*, volume 23, issue 5, 1085–1099.

'IEEE Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems'. IEEE Standard.

Mirollo, RE. and Steven, H. (1990). 'Synchronization of Pulse-Coupled Biological Oscillators'. *SIAM Journal on Applied Mathematics*, volume 50, issue 6, 1645–1662.

Ogata, K. (2009). 'Modern Control Engineering'. Pearson.

Pagliari, R. and Scaglione, A. (2011). 'Scalable Network Synchronization with Pulse-Coupled Oscillators'. *IEEE Transactions on Mobile Computing*, volume 10, issue 3, 392–405.

Proskurnikov, AV. and Cao, M. (2017). 'Synchronization of Pulse-Coupled Oscillators and Clocks under Minimal Connectivity Assumptions'. *IEEE Transactions on Automatic Control*, volume 62, issue 11, 5873–5879.

Sivrikaya, F. and Yener, B. (2004). 'Time Synchronization in Sensor Networks: A Survey'. *IEEE Network*, volume 18, issue 4, 45-50.

Tyrrell, A., Auer, G. and Bettstetter, C. (2010). 'Emergent Slot Synchronization in Wireless Networks'. *IEEE Transactions on Mobile Computing*, volume 9, issue 5, 719–732.

Tyrrell, A., Auer, G. and Bettstetter, C. (2008). 'On the Accuracy of Firefly Synchronization with Delays'. in *International Symposium on Applied Sciences on Biomedical and Communication Technologies.*

Wang, Y., Nunez, F. and Doyle, FJ. (2012). 'Energy-Efficient Pulse-Coupled Synchronization Strategy Design for Wireless Sensor Networks through Reduced Idle Listening'. *IEEE Transactions on Signal Processing*, volume 60, issue 10, 5293–5306.

Zong, Y., Dai, X., Gao, Z., Binns, R. and Busawon, K. (2018a). 'Simulation and Evaluation of Pulse-Coupled Oscillators in Wireless Sensor Networks'. *Systems Science & Control Engineering*, volume 6, issue 1, 337–349.

Zong, Y., Dai, X., Gao, Z., NG. W., Busawon, K. and Binns, R. (2020). 'Synchronisation of Non-identical and Time-varying Packet-Coupled Oscillators in Wireless Sensor Networks'. in *IEEE Access* (under review).

Zong, Y., Dai, X. and Gao, Z. (2020). 'Proportional-Integral Synchronisation for Non-identical Wireless Packet-Coupled Oscillators with Delays'. in *IEEE Transactions on Industrial Electronics* (under review).

Zong, Y., Luo, K. and Dai, X. (2018b). 'Implementation of Timestamped Pulse-Coupled Oscillators in IEEE 802.15.4 Networks'. in *International Conference on Automation and Computing.*

Masood, W., Schmidt. J., Brandner. G. and Bettstetter, C. (2017). 'DISTY: Dynamic Stochastic Time Synchronization for Wireless Sensor Networks'. *IEEE Transactions on Industrial Informatics*, volume 13, issue 3, 1421–1429.