國 立 交 通 大 學

電信工程研究所

碩 士 論 文

在小型無線感測網路下，時間同步演算法的設計與實作

Time Synchronization Protocol for Small-Scale Wireless Sensor Networks

研究生：黃愈翔

指導教授：伍紹勳

中 華 民 國 98 年 10 月

在小型無線感測網路下，時間同步演算法的設計與實作

學生：黃愈翔　　指導教授：伍紹勳　博士

國立交通大學電信工程學系（研究所）碩士班

摘　　要

在此論文中我們在此論文中我們提出了循環式訊息交換同步機制（RRTE），並與傳統的 TPSN 與 PBS 演算法做比較，在功率消耗方面，RRTE 比 TPSN 減少相當多，若與 PBS 相比，RRTE 將功率消耗分散在各個感測節點中，更有利於整個網路的運作。在同步準確度方面，RRTE 介於 TPSN 與 PBS 之間，並且可藉由調整感測網路上的節點數目來做變化，這對於小型無線感測網路來說是相當有用的特色。再者，為了更近一步改進同步的精確度，我們引入了遞迴式非線性迴歸來抵抗每次同步所有延遲的變異量對同步誤差造成的影響，並且這將不需要太多繁複的運算。

Time Synchronization Protocol
for
Small-Scale Wireless Sensor Networks

Student：Yu-Hsiang Huang　　Advisor：Dr. Sau-Hsuan Wu

Department of Communication Engineering
National Chiao Tung University

ABSTRACT

A Round-Robin Timing Exchange (RRTE) protocol is proposed for the distributed synchronization of wireless sensor network. Compared to the Timing-sync Protocol for Sensor Network (TPSN) and the Pairwise Broadcasting Synchronization (PBS) scheme, the power consumption for each sensor node (SN) of RRTE is much smaller than that of TPSN and is evenly distributed among each SN in contrast to the PBS method. In addition, the synchronization accuracy of RRTE falls within that of TPSN and PBS, and can be adjusted by controlling the number of SNs in one cycle of synchronization. This makes the synchronization protocol particularly useful for small-scale wireless sensor networks. Furthermore, to improve the accuracy of synchronization, a recursive second-order regression method is also proposed to smooth the timing adjustment of each synchronization step without requiring complicated computations.

誌　　謝

在研究所的這兩年，經歷了許多的風風雨雨，也讓我從懵懂無知的大學生，學到了做研究的精神，這一切的一切首先要感謝便是指導教授伍紹勳老師，老師對於研究的熱忱令我佩服不已，在研究上常常給予我們諸多的建議以及幫助，在生活上老師也是給我們無微不至的照顧，並且也很設身處地為大家著想，老師投注全部的心力在大家身上，讓大家在研究的路上能夠無憂無慮的前進，在此，必須要再慎重的說一句：老師謝謝您。

再來要感謝的是實驗室的戰友們，大家在研究上常常能夠互相討論，並且不吝給予協助以及指導，大家的感情非常好，在處於巨大壓力的研究生涯中，擁有這們好的同伴們真的是一件很幸福的事，再次感謝 Lab711 的各位夥伴們。

最後要特別感謝的是我的父母及女友，在研究所的這兩年，父母無微不至的關懷，讓遠在異鄉的我內心倍感溫暖，不管是在物質或是心靈上，父母都給予全心全力的協助，女友的陪伴及鼓勵，讓我研究或生活遇到挫折時有重新站起來的力量，謝謝你們！

誌於 2009.10 新竹 交通大學

愈翔

Contents

List of Figures

Abstract

A Round-Robin Timing Exchange (RRTE) protocol is proposed for the distributed synchronization of wireless sensor network. Compared to the Timing-sync Protocol for Sensor Network (TPSN) and the Pairwise Broadcasting Synchronization (PBS) scheme, the power consumption for each sensor node (SN) of RRTE is much smaller than that of TPSN and is evenly distributed among each SN in contrast to the PBS method. In addition, the synchronization accuracy of RRTE falls within that of TPSN and PBS, and can be adjusted by controlling the number of SNs in one cycle of synchronization. This makes the synchronization protocol particularly useful for small-scale wireless sensor networks. Furthermore, to improve the accuracy of synchronization, a recursive second-order regression method is also proposed to smooth the timing adjustment of each synchronization step without requiring complicated computations.

Chapter 1

Introduction

Wireless sensor networks (WSN) [1] has found many applications in home surveillance, environment monitoring and industrial process controls [2], *etc.* More recently, WSN has also evolved into wireless body area network (WBAN) to meet the specific requirements on human health monitoring and physical rehabilitation. [3]. To acquire physical signals from distributed locations, a network coordination mechanism is necessary to synchronize the actions of distributed wireless sensor nodes (SNs). This synchronous distributed sensing would not be made possible without a reliable distributed timing synchronization protocol.

In the past decade, there has been rich research results on distributed synchronization for WSN, (see [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]). Among which, a *reference broadcasting synchronization, RBS,* protocol is first proposed in [16]. To improve the accuracy of synchronization and to reduce the numbers of message exchanges in RBS, a peer-to-peer synchronization scheme, known as the *Timing-sync Protocol for Sensor Network, TPSN*, is proposed in [17] which requires only two message exchanges for each nodes in the WSN. While, the number of transmissions of the reference node in TPSN increases proportionally with the number of SNs in the network, making the power consumption of the reference node unproportionately large in comparison with

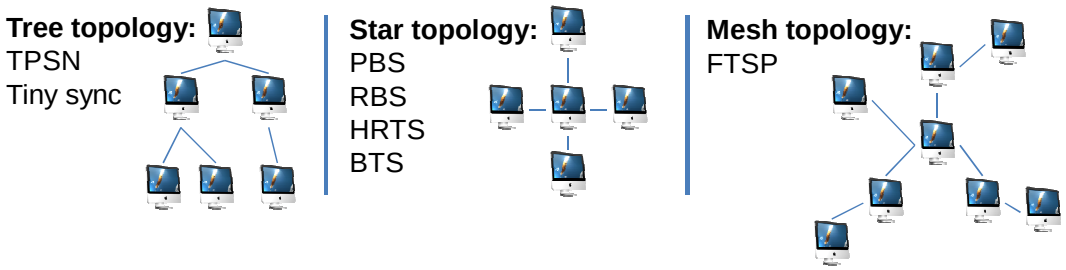


Figure 1.1: Suitable network topology for each protocol.

other SNs.

To reduce the number of message exchanges in RBS and TPSN, a *Pairwise Broadcast Synchronization, PBS,* is proposed in [18], in which every SN only needs to listen to the timing messages broadcasted by the two reference nodes (RNs) in the network. This makes the power consumption of each SN smaller than that of TPSN, however, the synchronization error is still equivalent to that of RBS and the power of the RNs will be drained out far more quickly than that of the SNs. Below is the summary of some famous protocols. Tiny-sync [12] is similar with TPSN, and the difference of them is the way they process the time-stamp after message exchange. Tiny-sync uses the manner like linear regression to get the linear equation of the two clock to reduce the synchronization error. FTSP [5] uses the flooding mechanism to synchronize all the network. Reference node periodical broadcasts its time and so do the synchronized nodes. It also uses linear regression scheme to increase error performance. HRTS [11] and BTS [13] are both similar with PBS, but the synchronization of them is started through reference node instead of WSNs. They also need one more message passing than PBS to get the adjustment. The differences of BTS and HRTS are that BTS employs piggy back to decrease the number of packet transmitted times in one synchronization cycle and the linear regression scheme is also performed in it. In addition, the suitable network topology for each protocol is different, and we illustrate it in Fig. 1.1.

Inspired by the above results, we propose in this work a *Round-Robin timing ex-*

change, RRTE, protocol for the distributed synchronization of SNs. Compared to the TPSN scheme, the power consumption of each SN is much smaller, and is evenly distributed among each SN in the network in contrast to the PBS method. Besides, it requires only one transmit and one receive in each synchronization cycle. Analysis shows that the synchronization accuracy falls within that of TPSN and PBS, and can be adjusted by controlling the number of SNs in one cycle of synchronization. This makes the synchronization protocol particularly useful for small-scale WSN. Furthermore, to improve the accuracy of synchronization, we also propose a recursive second-order regression method, which can smooth the timing adjustment in each synchronization step, while does not involve complicated mathematical calculations.

Chapter 2

Distributed synchronization for WSN

We consider a small scale WSN which consists of a reference node (RN) and a number of wireless SNs. Each node is operating at a certain clock frequency plus a certain amount of frequency offset. The frequency offset depends on the temperature, air pressure and humidity, etc, and drifts in between a limited range. As the local time of a SN is maintained based on its clock frequency, a distributed synchronization (DS) protocol is thus required to control the local times of all nodes within an acceptable range.

The accuracy of a DS scheme depends on the its protocol as well as the processing delays involved in the execution of the protocol. The delay factors for sending data packets from node i to node j can be classified into several categories as illustrated in Fig. 2.1 and are summarized below:

- ***Send time***: The time required for passing data packets from application layer to medium access control (MAC) layer
- ***Access time*** The waiting time for packets to get access to the channel at MAC layer.

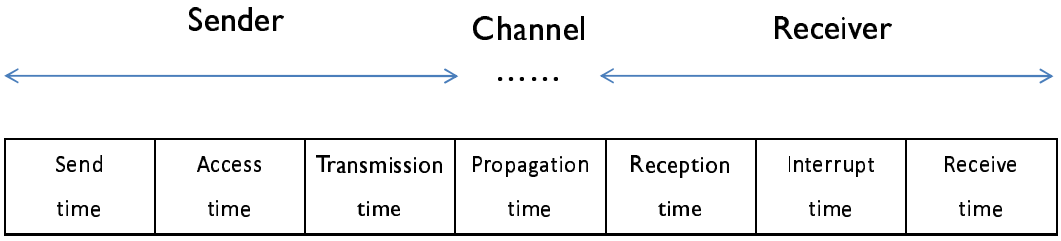


Figure 2.1: The delay factors involved in transmit and receive.

- ***Transmission time:*** The time for transmitting a packet at physical layer. It also depends on data bandwidth and packet size.
- ***Propagation time***: The time for radio signal to propagate from one node to another.
- ***Reception time:*** The time for processing a received packet at physical layer. It is similar to the transmission time.
- ***Interrupt time:*** The waiting time for received packets to be processed by MAC layer.
- ***Receive time:*** The time for passing data packets from MAC to application layers.

For convenience of expression, the send time, access time and transmission time at a node are lumped together and denoted by T_S^i where i is the node index. The propagation time from node i to j is denoted by $T_P^{i\to j}$. And the sum of the reception, interrupt and receive time at node j is denoted by T_R^j.

Taking into account the processing delays, clock drifting and the relative local time offset between two nodes, the local time of node j when receiving a data packet sent at the local time T_1^i of node i is given by

$$T_2^j = T_1^i + T_S^i + T_P^{i\to j} + T_R^j + O_1^{j\to i} + D_{1\to 2}^{j\to i} \tag{2.1}$$

where $O_1^{j\to i}$ stands for the relative time offset of node j to node i at time index 1

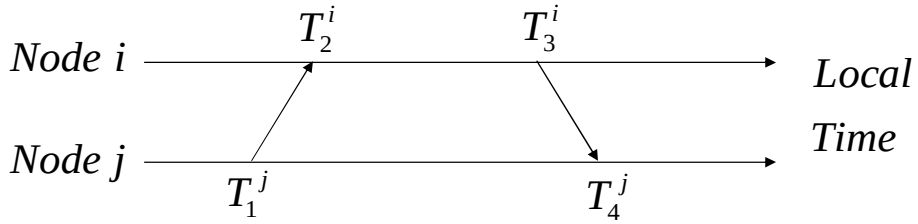


Figure 2.2: TPSN synchronization: Node j can adjust its time by adding $(T_2^i - T_1^j + T_3^i - T_4^j)/2$.

when the time stamp T_1^i is marked at node i. And $D_{1\to 2}^{j\to i}$ is the relative local time drift of nodes j to i during the time indices 1 and 2 when the time stamp T_2^j is marked at node j. Now suppose that $|O_1^{j\to i}| \gg T_S^i + T_P^{i\to j} + T_R^j + D_{1\to 2}^{j\to i}$. To have node j's local time synchronize with node i's local time, a straightforward method is to subtract $\triangle_j \triangleq T_2^j - T_1^i$ from T_2^j, forcing the local time T_2^j to T_1^i when the local time at node i is $T_2^i = T_1^i + T_S^i + T_P^{i\to j} + T_R^j + D_{1\to 2}^{i\to i}$.

To reduce the synchronization error $T_2^i - T_1^i$, TPSN [17] employs a two-way timing message exchange method as shown in Fig. 2.2. Following the notations of (2.1), we have

$$T_2^i = T_1^j + T_S^j + T_P^{j\to i} + T_R^i + O_1^{i\to j} + D_{1\to 2}^{i\to j}. \tag{2.2}$$

$$T_4^j = T_3^i + T_S^i + T_P^{i\to j} + T_R^j + O_3^{j\to i} + D_{3\to 4}^{j\to i}. \tag{2.3}$$

Having node i send back T_2^i and T_3^i to node j gives

$$\begin{aligned} \triangle_j &\triangleq (T_2^i - T_1^j + T_3^i - T_4^j)/2 \\ &= [(T_S^j - T_S^i) + (T_P^{j\to i} - T_P^{i\to j}) + (T_R^i - T_R^j) + \\ &\quad (O_1^{i\to j} - O_3^{j\to i}) + (D_{1\to 2}^{i\to j} - D_{3\to 4}^{j\to i})]/2. \end{aligned} \tag{2.4}$$

Assuming that node i and j have similar processing delays and the time offsets remain the same between the time indices 1 and 4, then $\triangle_j \simeq (O_2^{i\to j} - O_4^{j\to i})/2 \simeq O_4^{i\to j}$. As a

result,

$$T_4^j + \triangle_j \simeq T_4^i + O_4^{j\to i} + O_4^{i\to j} = T_4^i. \tag{2.5}$$

Though accurate when $T_S^j \cong T_S^i$, TPSN requires one transmission and one reception for each SN to synchronize with the reference node. In addition, for random access network where the variation of T_S becomes large, the accuracy of TPSN will degrade. To improve the power efficiency, a pairwise broadcast synchronization(PBS) method is introduced in [18] for DS, based on the RBS scheme [16]. Different from the RBS, two super reference nodes as shown in Fig. 2.3 are used to broadcast the timing messages collaboratively, with the synchronization between the super reference nodes maintained with TPSN.

At the beginning of a synchronization cycle, the first RN, r_1, broadcasts the time stamp $T_1^{r_1}$ to the second RN, r_2, and all SNs in the network. Therefore, we have

$$T_2^{r_2} = T_1^{r_1} + T_S^{r_1} + T_P^{r_1\to r_2} + T_R^{r_2} + O_1^{r_2\to r_1} + D_{1\to 2}^{r_2\to r_1}. \tag{2.6}$$

$$T_3^{j} = T_1^{r_1} + T_S^{r_1} + T_P^{r_1\to j} + T_R^{j} + O_1^{j\to r_1} + D_{1\to 3}^{j\to r_1}. \tag{2.7}$$

By broadcasting $T_2^{r_2}$ of r_2, a SN j now has

$$\begin{aligned} T_3^j - T_2^{r_2} &= (T_P^{r_1\to j} - T_P^{r_1\to r_2}) + (T_R^j - T_R^{r_2}) + \\ &\quad (O_1^{j\to r_1} + D_{1\to 3}^{j\to r_1}) - (O_1^{r_2\to r_1} + D_{1\to 2}^{r_2\to r_1}) \\ &\simeq O_1^{j\to r_2} + D_{1\to 3}^{j\to r_2} = O_3^{j\to r_2} \end{aligned} \tag{2.8}$$

assuming that $T_P^{r_1\to j} \simeq T_P^{r_1\to r_2}$ and $T_R^j \simeq T_R^{r_2}$. Thus, substracting $O_3^{j\to r_2}$ from $T_6^j = T_6^{r_2} + O_6^{j\to r_2}$ adjusts the local time of node j approximately equal to $T_6^{r_2}$ if $O_3^{j\to r_2} \approx O_6^{j\to r_2}$. We note that the synchronization accuracy of PBS is equal to that of RBS, and it largely depends on the time drift $D_{3\to 6}^{j\to r_2}$.

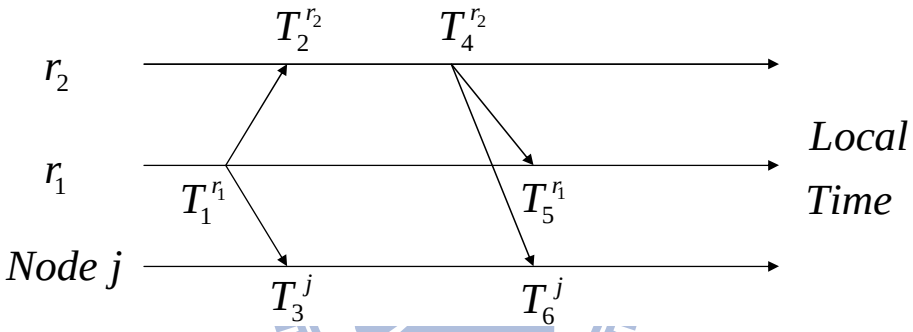


Figure 2.3: PBS synchronization: Node j can synchronize to Node i by adding $\triangle_j \triangleq T_2^{r_2} - T_3^j$.

Chapter 3

Round-Robin Timing Exchange Synchronization (RRTE)

As shown in Fig. 2.3, PBS uses two receptions for each SN, making it more power efficient than TPSN. Nevertheless, the RNs in PBS still employ one transmission and one reception each. Consequently, the power of the RNs in PBS or TPSN are drained out far more quickly than other SNs. For applications, such as health monitoring, where the lifetime of a WSN is determined by the shortest lifetime of a node in the network, then the above synchronization protocols will make the lifetime of the WSN dominated by the RNs. Inspired by the RBS/PBS and TPSN DS protocols, we propose a modified synchronization scheme to better compromise between the power consumption and the synchronization errors.

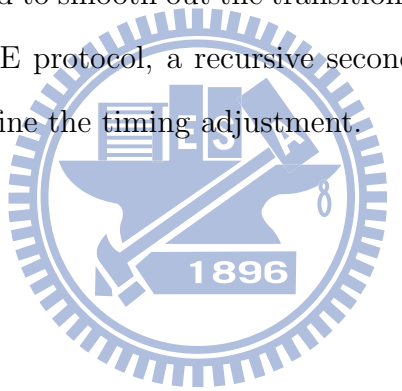
Suppose that there are m SNs in the network to synchronize with the reference node (RN). In each synchronization cycle, one SN is chosen out of the network in a Round-Robin fashion to do timing message exchanges with the RN like TPSN. Other SNs in the network perform synchronization similar to PBS by overhearing the message exchanges between the SN and the RN as shown in Fig. 2.3. As every node equally obtains one "TPSN" for every $m-1$ "PBS", the average synchronization error of the SNs is likely

to be

$$\delta_{RRTE} = \frac{1}{m}\delta_{TPSN} + \frac{m-1}{m}\delta_{PBS} \tag{3.1}$$

where δ_{TPSN} is the synchronization error of using TPSN only, and δ_{PBS} is the synchronization error of using PBS only. A rigorous asymptotic analysis will be provided in Chapter 5.

In addition, since each SN takes turn to do timing exchanges with the RN, the overall power consumption for DS is much lower than that of TPSN and is evenly distributed among the SNs in the network. This prevents from draining unproportionally large power from a single SN and is thus crucial to some applications such as WBAN in which a failure of a SN may lead to the end of the entire network. Nevertheless, to compensate the synchronization errors and to smooth out the transition between the TPSN and PBS phases in the proposed RRTE protocol, a recursive second-order regression method is employed in Chapter 4 to refine the timing adjustment.



Chapter 4

Recursive Second-Order Regression

Based on the discussions in the previous chapter, when a SN j as shown in Fig. 2.2 performs synchronization with the RN using TPSN, the adjustment for SN j is $\triangle^j \triangleq (T_2^r - T_1^j + T_3^r - T_4^j)/2$, [*c.f.* (2.4)]. While for a SN k doing synchronization by overhearing the message exchanges between the SN j and the RN, the adjustment is given by $\triangle^k \triangleq T_3^k - T_2^r$, [*c.f.* (2.8)]. As the accuracies of $\triangle^j$ and $\triangle^k$ are usually different and both vary with time, a second order regression method is provided herein to improve the accuracies.

Let p stand for the synchronization cycle of the network and y_p be the adjustment for the p-th cycle. Given the rough estimates $\triangle_p,\ \forall p \in [1, n]$, which are obtained either by TPSN or PBS, we want to find $y_p,\ \forall p \in [1, n]$ that minimizes

$$J_n = \sum_{p=1}^{n} \lambda^{n-p}(y_p - \triangle_p)^2 \tag{4.1}$$

where n is the current synchronization cycle and $\lambda \in [0, 1]$ is used to control the effective size of the processing data.

Suppose that y_p can be modeled by a second-order equation for the processing data

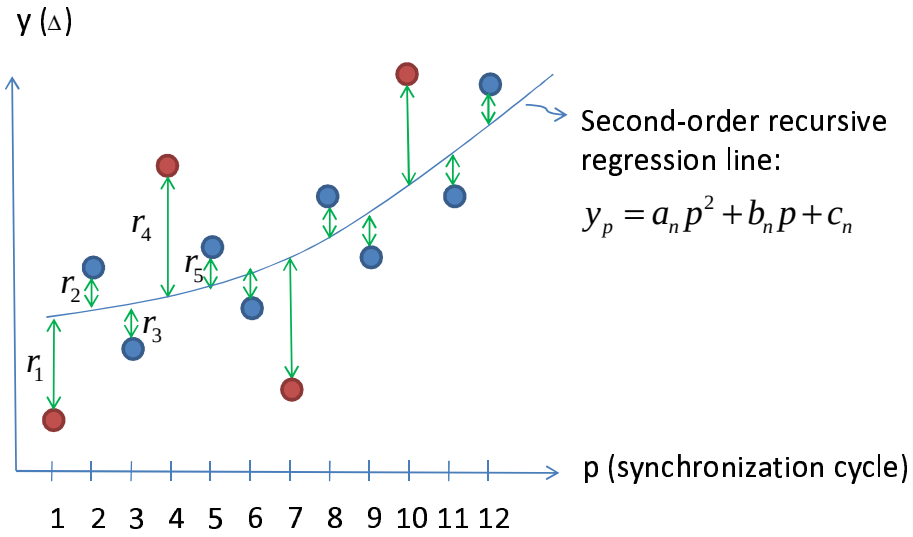


Figure 4.1: Second-order regression to refine the adjustment △.

up to n as shown in Fig. 4.1. Then

$$y_p = a_n p^2 + b_n p + c_n \tag{4.2}$$

where a_n, b_n, and c_n are the coefficients of the regression line shown in Fig. 4.1. Now, define $h_n \triangleq [a_n\ b_n\ c_n]$. Having $\frac{\partial J_n}{\partial h_n} = 0$ yields $\hat{h}_n$ for the n-th synchronization cycle. To lower down the computational complexity, the derivative is further expanded with the Taylor series expansion into [19]

$$\frac{\partial J_n}{\partial h_n} = \frac{\partial J_n}{\partial h_n}\bigg|_{h_n=\hat{h}_{n-1}} + \frac{\partial^2 J_n}{\partial {h_n}^2}\bigg|_{h_n=\hat{h}_{n-1}} (h_n - \hat{h}_{n-1}) = 0. \tag{4.3}$$

This leads to the Gauss-Seidel recursive form of h_n, given by

$$h_n = \hat{h}_{n-1} - \left(\frac{\partial^2 J_n}{\partial {h_n}^2}\bigg|_{h_n=\hat{h}_{n-1}}\right)^{-1} \left(\frac{\partial J_n}{\partial h_n}\bigg|_{h_n=\hat{h}_{n-1}}\right). \tag{4.4}$$

Define

$$P_n \triangleq \begin{pmatrix} S_{p^4,n} & S_{p^3,n} & S_{p^2,n} \\ S_{p^3,n} & S_{p^2,n} & S_{p^1,n} \\ S_{p^2,n} & S_{p^1,n} & S_{p^0,n} \end{pmatrix}^{-1} \text{ and } Q_n \triangleq \begin{pmatrix} S_{\triangle p^2,n} \\ S_{\triangle p^1,n} \\ S_{\triangle p^0,n} \end{pmatrix}$$

with $S_{p^q,n} \triangleq \sum_{p=1}^{n} \lambda^{n-p} p^q$ and $S_{p^q\triangle,n} \triangleq \sum_{p=1}^{n} \lambda^{n-p} p^q \triangle_p$. By some mathematical manipulations, it can be shown that

$$\hat{h}_n = P_n Q_n \tag{4.5}$$

and $Q_n = \lambda Q_{n-1} + \triangle_n V_n$, with $V_n \triangleq [n^2,\ n,\ 1]^T$. Furthermore, with the Matrix Inversion Lemma, we obtain

$$P_n = \frac{1}{\lambda} P_{n-1} - \frac{1}{\lambda} P_{n-1} V_n \left(\lambda + V_n^T P_{n-1} V_n\right)^{-1} V_n^T P_{n-1}. \tag{4.6}$$

This gives a recursive second-order regression scheme to adjust the timing offset at each synchronization cycle. Besides, it only requires to store P_n and Q_n. And no matrix inversion is needed in the computation.

Chapter 5

The Synchronization Error of RRTE

We investigate herein the asymptotic synchronization error of the proposed RRTE protocol and compare the error with that of the TPSN and PBS protocols. For convenience of analysis, the forgetting factor λ in (4.1) is set to one. For cases with $\lambda < 1$, the analysis is still valid if we have n equal to the effective window size $1/(1-\lambda)$ of the recursive scheme. Therefore, having $\frac{\partial J_n}{\partial a_n} = \frac{\partial J_n}{\partial b_n} = \frac{\partial J_n}{\partial c_n} = 0$ with $\lambda = 1$ in (4.1) leads to

$$y_n = C_n \sum_{p=1}^{n} \triangle_p F_n(p) \tag{5.1}$$

where $C_n = \frac{3}{n(n+1)(n+2)}$ and

$$F_n(p) = 10p^2 - (8n+6)p + (n^2+3n+2). \tag{5.2}$$

Furthermore, as the synchronization is performed periodically, we can expect an asymptotic error of synchronization, denoted by Δ_a, between the RN and SNs. Define the transient error in the synchronization cycle p as $\bigtriangledown_p \triangleq \triangle_p - \Delta_a$. Then the transient response of synchronization can be characterized by

$$\delta_n \;=\; y_n - \Delta_a = C_n \sum_{p=1}^{n} \bigtriangledown_p F_n(p) \tag{5.3}$$

as

$$C_n \sum_{p=1}^{n} F_n(p) = 1. \tag{5.4}$$

Now, define $\triangle_a^{SR}$ and $\triangle_a^{RR}$ to be the average synchronization errors corresponding, respectively, to the sender-receiver (SR) synchronization scheme of TPSN and the receiver-receiver (RR) protocol of RBS/PBS. Since, RRTE uses 1 SR synchronization of TPSN for every (m-1) RR synchronization of PBS, therefore we assume $\triangle_a = [\triangle_a^{SR} + (m-1)\triangle_a^{RR}]/m$. As a result, (5.3) can also be expressed as

$$\begin{aligned} \delta_n &= \delta_n^{RR} + C_n \sum_{p=1,1+m,1+2m,\ldots}^{\lceil n/m \rceil} \left(\Delta_p^{SR} - \Delta_p^{RR}\right) F_n(p) \\ &\quad - \frac{1}{m}(\triangle_a^{SR} - \triangle_a^{RR}). \end{aligned} \tag{5.5}$$

Suppose $n = mk$ and k is an integer. It can be shown that

$$\begin{aligned} C_n \sum_{p=1,1+m,1+2m,\ldots}^{n-m+1} F_n(p) \\ = \frac{n^2 + 6n - 3nm + 5m^2 - 21m + 18}{(n^2 + 3n + 1)m}. \end{aligned} \tag{5.6}$$

Therefore,

$$C_n \sum_{p=1,1+m,1+2m,\ldots}^{n-m+1} F_n(p) = \frac{1}{m}, \quad n \to \infty \tag{5.7}$$

Substituting this back into (5.5) yields

$$\delta_\infty = \delta_\infty^{RR} + C_\infty \sum_{p=1,1+m,1+2m,\ldots} (\bigtriangledown_p^{SR} - \bigtriangledown_p^{RR}) F_\infty(p) \tag{5.8}$$

where $\bigtriangledown_p^{SR} \triangleq \Delta_p^{SR} - \triangle_a^{SR}$ and $\bigtriangledown_p^{RR} \triangleq \Delta_p^{RR} - \triangle_a^{RR}$.

Since $\bigtriangledown_p^{SR}$ remains almost unchanged for $p = mk+1, \ldots, (m+1)k$, if m is not too large. Then, we have

$$\begin{aligned} \delta_\infty &\simeq \delta_\infty^{RR} + \frac{C_\infty}{m} \sum_{p=1}^{\infty} (\bigtriangledown_p^{SR} - \bigtriangledown_p^{RR}) F_\infty(p) \\ &= \delta_\infty^{RR} + \frac{1}{m}(\delta_\infty^{SR} - \delta_\infty^{RR}) \\ &= \frac{1}{m}\delta_\infty^{SR} + \frac{m-1}{m}\delta_\infty^{RR} \end{aligned} \tag{5.9}$$

where $\delta_n^{RR} \triangleq y_{n,RR} - \bigtriangleup_a^{RR}$ and $\delta_n^{SR} \triangleq y_{n,SR} - \bigtriangleup_a^{SR}$. This shows that the synchronization error of the proposed RRTE protocol will asymptotically converge to a value between that of the TPSN and PBS. The value depends on the number of SNs in the network.

Chapter 6

The Power Efficiency of RRTE

Our goals In this chapter we compare the the power efficiency of the RRTE protocols with that of TPSN and PBS/RBS schemes.

As the transmit power dominates the power consumption of a wireless SN, we in particular compare the number of transmissions and receptions for each SN of RRTE to that of TPSN and PBS protocols.

Suppose there are m SNs in the network, (the RN itself is not included). TABLE 6.1 summarizes the numbers of transmit and receive times for PBS, TPSN and RRTE, respectively. We can see that all have the same total number of transmit and receive times. However, the number of transmit times for TPSN is much higher than that of PBS and RRTE.

On the other hand, Table 6.2 lists the time required for one cycle of synchronization over the network. For TPSN, the time required for synchronization is proportional to m

Table 6.1: Power consumption comparison

Protocol	Receive Times	Transmit Times	Total Times
PBS	$2m$	0	$2m$
TPSN	m	m	$2m$
RRTE	$2m-1$	1	$2m$

Table 6.2: Needed time for synchronization

Protocol	Required Time units to Achieve Total Time Synchronization
PBS	1
TPSN	m
RRTE	1

as all SNs in the network must sequentially perform timing message exchanges with the RN in one synchronization cycle. While for PBS and RRTE protocols, the SNs perform synchronization all together by overhearing the messages exchanges between the RNs or between the SN and the RN. The time required for one cycle of synchronization is thus only $1/m$ of that for TPSN, making them more friendly to congestion control and have higher spectral efficiency.

Chapter 7

Experiment Result

In this chapter, we present the implementation and measurement results for the proposed RRTE synchronization methods and compare them with that of the TPSN and PBS protocols. The entire WSN and the synchronization protocols are developed using a Zigbee development kit, U-NET01, manufactured by Uniband Electronic Corp (UBEC) (see Fig. 7.1) [20]. The embedded processor of U-NET01 is a 8051 micro-controller, and the wireless transceiver module of it uses UBEC's UZ2400, an IEEE 802.15.4 compliant transceiver which has passed the Zigbee Compliant Platform (ZCP) certification in 2006.

There are 4 U-NET01 boards in our environment, and Table 7.1 shows the clock frequency of all these boards. We can see that there are some clock drifts and offsets between them, therefore the difference of the local time maintained by each of them will become larger without synchronization as time passes by.

To make sure our experimental results match our analysis, Fig. 7.2 shows all the delay factors we had measured. Note that the Receive time can be ignored because we make time stamp in the beginning of the outer interrupt service routine(ISR) (See Fig 7.3). Once the RF detects the packet, it will trigger the outer interrupt and run into it to process the packet. So we can make time stamp here instead of making time stamp in the application layer. The ***Send time*** can be calculated by checking how many clock

Figure 7.1: The U-NET01 Zigbee development kit.

Table 7.1: Clock frequency of each UNET-01 boards in MHz

Node	Mean	Min	Max
1	24.94690	24.89668	24.99122
2	24.82279	24.77488	24.86820
3	24.83406	24.77810	24.88306
4	24.91826	24.87578	24.96026

passed from the application layer to the mac layer. The ***Transmission Time*** and ***Reception time*** are all the same because the data rate of the Zigbee is 250k bps and our synchronization packet is fixed 35 bytes. The mean value of the ***Interrupt Time*** due to the variable announcement time difference between CPN and WSN. Because it may encounter the situation that the CPU is serving another ISR when the RF trigger the outer interrupt. Note that there is another ISR for maintaining its local time, and the serving time for CPN and WSN are about 8.04us and 14.92us, respectively. It has to wait for the CPU finishing serving another ISR and able to serve it, so the variance set to 5 is make sense.

In the experimental setting, we assign one of the U-NET01 boards as a CPN, and use the others as wireless SNs. Packets of synchronization messages are periodically exchanged between the CPN and the SNs in a Round-Robin fashion. Figure 7.4 and 7.5 represent the code flow chart of CPN and WSN, respectively.

Delay Type	Magnitude	
	CPN	WSN
Send Time	N(394.49,85.85) μs	N(431.107,8.077) μs
Access Time	320 μs x (Uniform random between $[0, 2^{BE}-1]$)	
Transmission Time	1120 μs	
Propagation Time	Can be ignored	
Reception Time	1120 μs	
Interrupt Time	N(16.04,5) μs	N(17.88,5) μs
Receive Time	Can be ignored	

Figure 7.2: The delay factors for sending data packets.

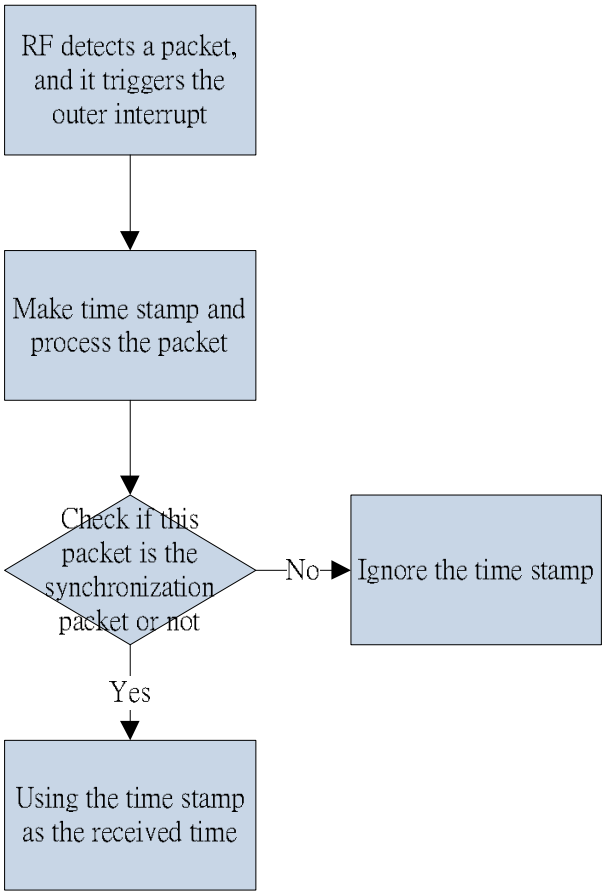


Figure 7.3: The flow chart of the received time stamp.

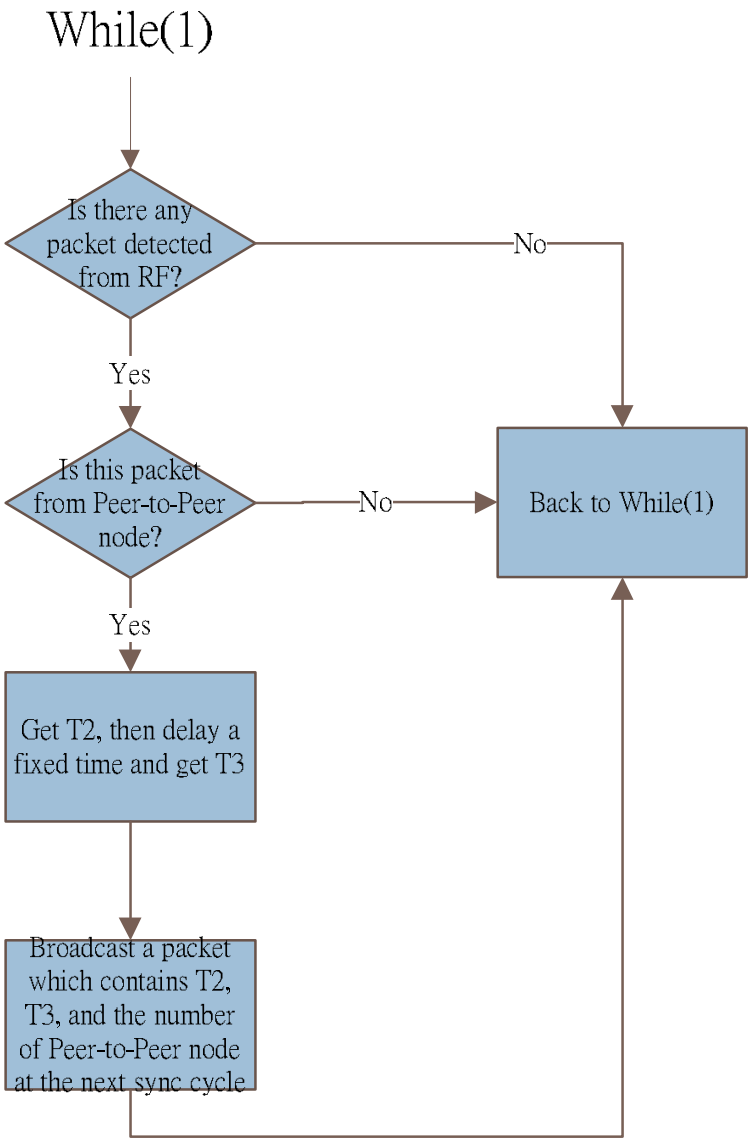


Figure 7.4: Code flow chart of CPN.

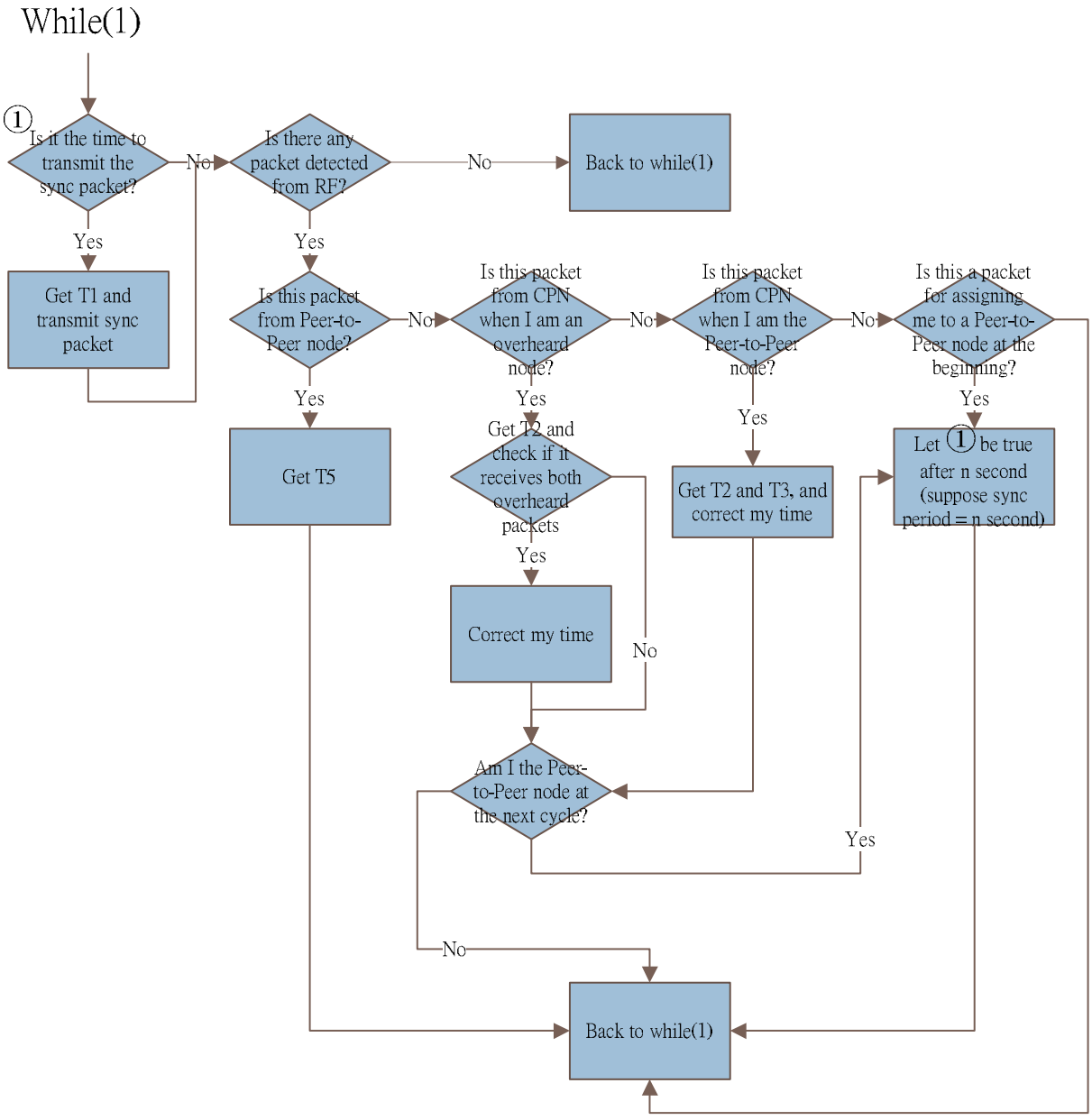


Figure 7.5: Code flow chart of WSN.

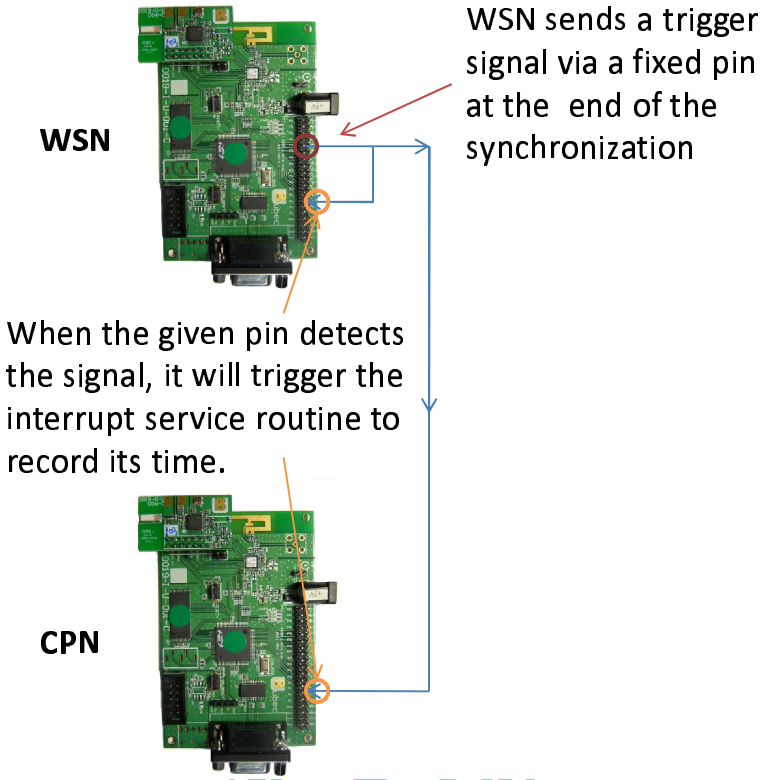


Figure 7.6: WSN triggers the interrupt to record the local time.

To measure the synchronization errors, a fixed WSN is used to trigger the interrupts of the CPN and itself simultaneously after the end of the synchronization. Once the interrupts are triggered, the ISR of the CPN and the WSN will record their current local times. We use the difference between the recorded time of the CPN and that of the WSN as the synchronization error. Fig 7.6 shows how the WSN triggers both of them to record the time.

Table 7.2 shows the average absolute of synchronization errors for the three protocols under different maximum random access delays without using the regression scheme. The maximum access delay of CSMA-CA can be set to uniform random between [0 , $0.32(2^{BE}-1)]ms$ in Zigbee with $BE = 0, \cdots, 3$. The synchronization period is $0.5s$, and the total number of experiments is 250.

We can see that the average synchronization errors of TPSN degrade greatly as the

Table 7.2: Synchronization error in μs without regression

Protocol	BE=0	BE=1	BE=2	BE=3
TPSN	33.5	104.48	220.2	417.93
RRTE (1CPN and 3WSNs)	14.8	36.48	77.79	139.6
PBS	5.552	5.4	5.28	6.32

Table 7.3: Synchronization error after adding received delay in μs without regression

Protocol	BE=0	BE=1	BE=2	BE=3
TPSN	73.52	122.26	231.64	430.6
RRTE (1CPN and 3WSNs)	115.26	130.8	170.1	238.8
PBS	137.58	136.8	136.12	135.14

variation of T_S increases, and the RRTE protocol is a good compromise between the TPSN and the PBS. Besides, the synchronization errors of RRTE closely follow the rule: $\delta_{RRTE} = \frac{1}{3}\delta_{TPSN} + \frac{2}{3}\delta_{PBS}$ of (3.1) and (5.9). We can see that the performance of PBS protocol for different BE are all good. It is because our experiment environment is simple. To simulate the more practical environment, we add a uniform random delay between $(0, 320us)$ at the receiver before it making time stamp. Table 7.3 shows the synchronization error after adding the random received delay of the three protocols. We can notice that TPSN is better than PBS at BE=0 and 1. But when BE is more higher, PBS is still better than TPSN because the error performance of PBS is just relative to the received delay but not transmitted one.

To verify the proposed synchronization algorithm, we simulate the synchronization protocols of PBS, TPSN, and RRTE on the "Simsync" simulator developed by Xu et al. [21]. The all delay factor are modeled by our measurement (See Fig. 7.2). Besides, to model the propagation delays between the SNs in a small BAN, the SNs are randomly allocated within a circle with a radius of 5 meters.

Table 7.4 summarizes the synchronization error between TPSN, RRTE and PBS

Table 7.4: Synchronization error in μs without regression via simsync

Protocol	BE=0	BE=1	BE=2	BE=3
TPSN	37.658	103.5	222.148	421.491
RRTE (1CPN and 3WSNs)	15.595	38.813	75.59	146.611
PBS	5.805	5.35	5.42	5.65

Table 7.5: Synchronization error after adding received delay in μs without regression via simsync

Protocol	BE=0	BE=1	BE=2	BE=3
TPSN	73.839	124.707	234.258	437.658
RRTE (1CPN and 3WSNs)	109.34	130.35	166.975	232.273
PBS	134.683	133.6	136.51	135.844

without regression, and Table 7.5 are that after adding the received delay. It can be seen that the error performance are all closed to our experiment result.

On the other hand, regression scheme is helpful for the huge delay variation environment. Table 7.6 represents the absolute synchronization error of using the recursive second order regression method proposed in Chapter 4 in the case of BE=3 and with received delay.

Furthermore, we use Simsync to simulate the relation between number of WSNs and synchronization error in our proposed RRTE protocol for BE=2 and with/without received delay (See Fig. 7.7, and Fig. 7.8). We can see that if the number of node become larger, the error performance will approach to that of PBS. On the other hand, the average synchronization error of regression case will approach to that without regression when the number of nodes are larger enough in Fig. 7.8. So we can know that the regression scheme is helpful when 1) the number of nodes are less than 4 in the case of without received delay and 2) the performance of PBS is bad due to received delay variation.

Fig. 7.10 presents three realizations of synchronization errors for SNs performing

Table 7.6: Synchronization error after adding received delay in μs with regression

Protocol	Absolute synchronization error
TPSN	269.42
RRTE (1CPN and 3WSNs)	177.48
PBS	118.5

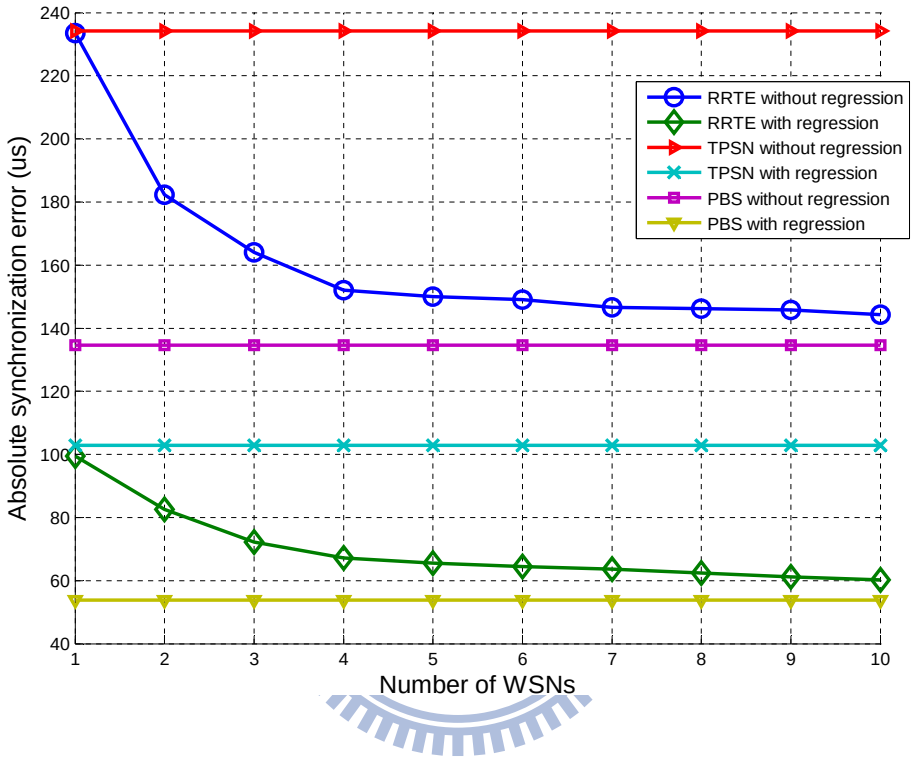


Figure 7.7: Number of WSNs in RRTE v.s. absolute synchronization error with received delay.

synchronization with RRTE and regression as oppose to the realizations without using regression in Fig. 7.9. We can see that the synchronization errors jump rapidly in Fig. 7.9, while they are much more smooth in Fig. 7.10.

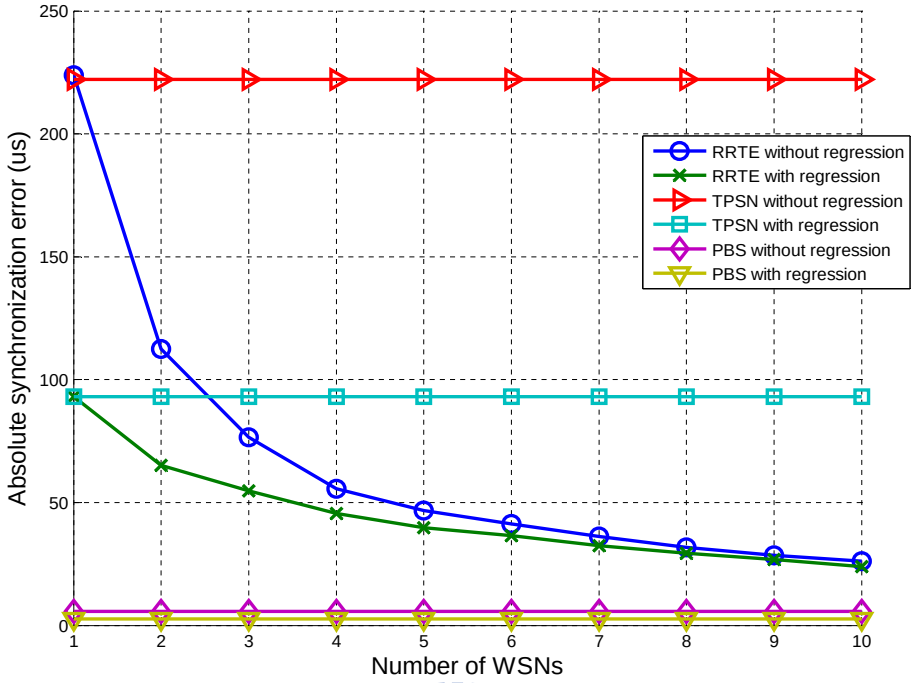


Figure 7.8: Number of WSNs in RRTE v.s. absolute synchronization error without received delay.

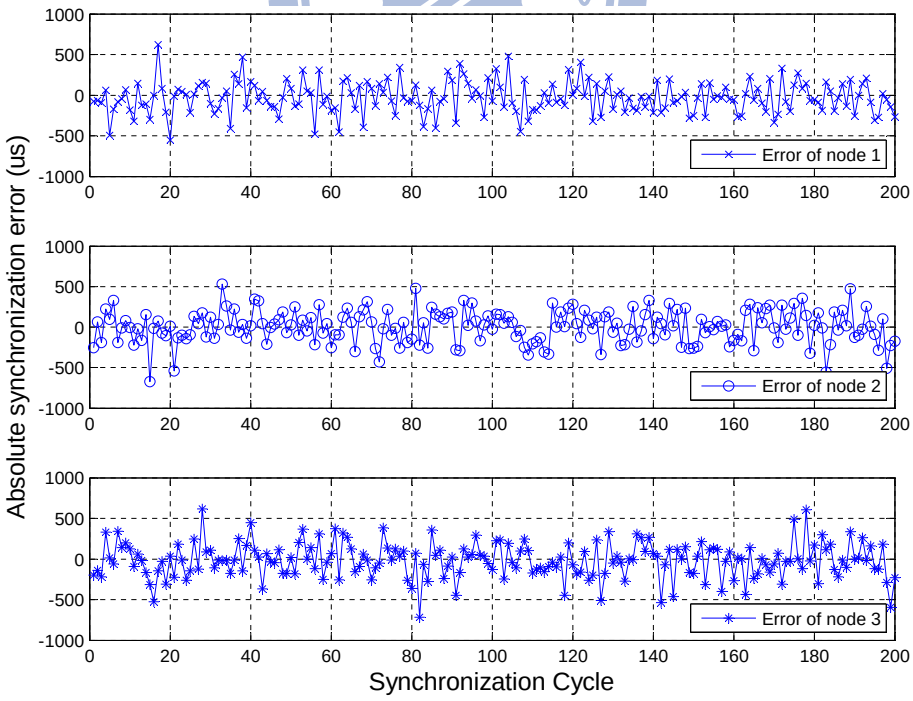


Figure 7.9: Error distribution of RRTE without regression.

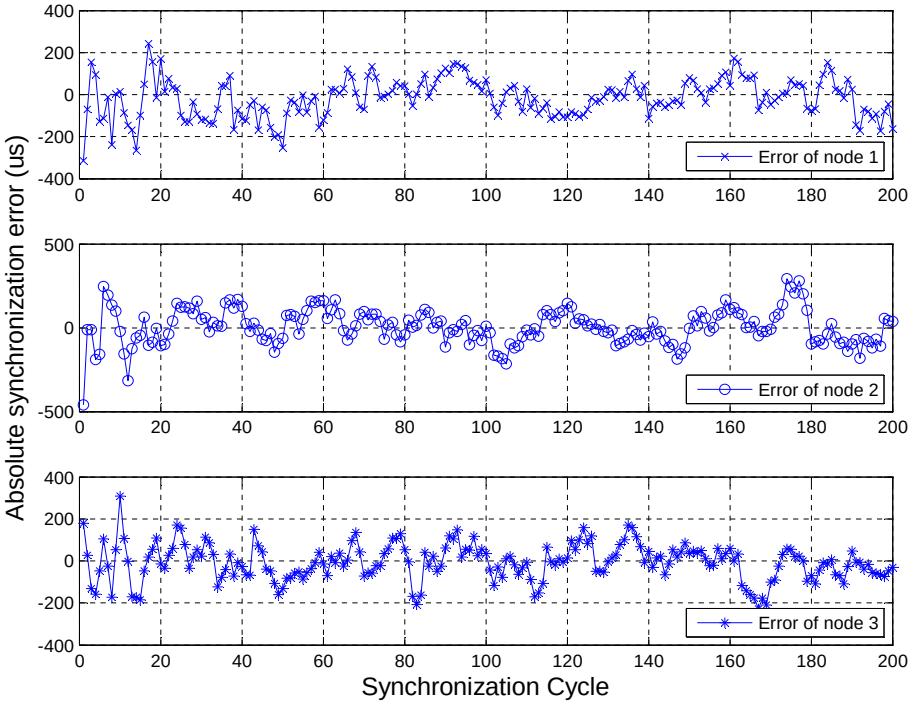


Figure 7.10: Error distribution of RRTE with regression.

Chapter 8

Conclusions

A Round-Robin Timing Exchange (RRTE) protocol was proposed for the distributed synchronization of wireless sensor network. Compared to the TPSN and the PBS protocols, the power consumption for each sensor node of RRTE is much smaller than that of TPSN and is comparable to that of PBS. Furthermore, to improve the accuracy of synchronization, a recursive second-order regression method was also introduced to smooth the timing adjustment of each step. Implementation results also verified that the synchronization accuracy of RRTE falls within that of TPSN and PBS, and can be adjusted by controlling the number of SNs in one cycle of synchronization. This makes the synchronization protocol particularly useful for small scale wireless sensor networks.

Bibliography

[1] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proc. ACM Mobicom Conference*. Seattle, WA, Aug. 1999.

[2] C. S. Raghavendra, K. M. Sivalingam, and T. F. Znati, *Wireless Sensor Networks*, Springer, 2nd edition, 2004.

[3] E. Jovanov, A. Milenkovic1, C. Otto1, and P. C de Groen, "A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 2, no. 1, pp. 6–16, Mar. 2005.

[4] B. Sundararaman, U. Buy, and A. D. Kshemklyani, "Clock synchronization in wireless sensor networks: a survey," *Ad hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.

[5] M. Marotia, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. 2nd International Conference on Embedded Networked Sensor Systems*. Baltimore, U.S.A., Nov. 2004.

[6] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Proc. IEEE Wireless Communications and Networking Conference*. New Orleans, LA, Mar. 2003.

[7] J. Elson and D. Estrin, "Time synchronization for wireless sensor networks," in *Proc. International Parallel and Distributed Processing Symposium, Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*. San Francisco, CA, Apr. 2001.

[8] D. Cox, E. Jovanov, and A. Milenkovic, "Time synchronization for zigbee networks," in *Proc. 37th SSST*. Tuskegee, AL, Mar. 2005.

[9] J. V. Greunen and J. Rabaey, "Lightweight time synchronization for sensor networks," in *Proc. 2nd ACM International Workshop on Wireless Sensor Networks and Applications*. San Diego, CA, Sep. 2003.

[10] S. Yoon and M. L. Sichitiu, "Analysis and performance evaluation of a time synchronization protocol for wireless sensor networks," in *Proc. International Conference on Telecommunication Systems, Modeling and Analysis*. Dallas, TX, Nov. 2005.

[11] H. Dai and R. Han, "Tsync: A lightweight bidirectional time synchronization service for wireless sensor networks," *Newsletters of ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 1, pp. 125–139, Jan. 2004.

[12] S. Yoon, C. Veerarittiphan, and M. L. Sichitiu, "Tiny-sync: Tight time synchronization for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 3, no. 8, Jun. 2007.

[13] C. Xu, Lei. Zhao, Yongjun Xu, and Xiaowei Li, "Broadcast time synchronization algorithm for wireless sensor networks," in *Proc. International Conference on Sensing, Computing and Automation*. ChongQing, China, May. 2006.

[14] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications Magazine*, vol. 7, no. 5, pp. 16–27, Oct. 2000.

[15] C. Otto, A. Milenkovic, C. Sanders, and E. Jovanov, "System architecture of a wireless body area sensor network for ubiquitous health monitoring," *Journal of Mobile Multimedia*, vol. 1, no. 4, pp. 307–326.

[16] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proc. 5th Symposium on Operation System Design and Implementation*. Boston, MA, Dec. 2002.

[17] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 1st ACM Conference on Embedded Networked Sensor Systems*. Los Angeles, CA, Nov. 2003.

[18] K. L. Noh, E. Serpedin, and K. Qaraqe, "A new approach for time synchronization in wireless sensor networks: Pairwise broadcast synchronization," *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3318–3322, Sep. 2008.

[19] F. Albu and C. Palelologu, "A recursive least square algorithm for active noise control based on the gauss-seidel method," in *Proc. IEEE International Conference on Electronics, Circuits and Systems (ICECS 2008)*. St. Julien's, Malta, Aug. 2008.

[20] "Uniband electronic corp.," http://www.ubec.com.tw.

[21] C. Xu, L. Zhao, Y. Xu, and X. Li, "Simsync: An effective time synchronization simulator for sensor networks," in *Proc. First International Workshop on Sensor Networks and Applications*. Beijing, China, Oct. 2005.