Time Synchronization and Its Applications in Wireless Sensor Networks

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Xintao Huan

Department of Electrical Engineering and Electronics
School of Electrical Engineering, Electronics and Computer Science

May 2021

Abstract

Time synchronization is an essential component of wireless sensor networks (WSNs) that play a key role in the thriving Internet of Things (IoT), supporting IoT applications from large-scale monitoring & event detection to collaborative interactions. The large-scale applications based on resource-constrained sensor nodes promote the development of WSN time synchronization towards the three major aspects of lower energy consumption, lower computational complexity, and higher multi-hop time synchronization accuracy. It is these three aspects that we focus on in our contributions to the development of WSN time synchronization, which are presented in this thesis together with their applications to optimal bundling and node identification.

First, we concentrate on the *computational complexity* of WSN time synchronization. Through the practical implementation of the state-of-the-art WSN time synchronization scheme based on the asynchronous source clock frequency recovery and the reverse two-way message exchange compensating for propagation delay, we demonstrate that the actual performance of the implemented scheme on a real WSN testbed consisting of low-cost, battery-powered sensor nodes could not reach its simulated performance; the major cause is the limited precision of floating-point arithmetic of those sensor nodes, which suggests that the design of WSN time synchronization schemes should take into account the computational complexity, including the issue of precision loss. To address this issue, we propose an asymmetric high-precision time synchronization scheme (AHTS), where synchronization-related computations are all asymmetrically

done at the head node with abundant computing and power resources and, thereby, the computational complexity on the resource-constrained sensor nodes is significantly reduced while achieving microsecond-level time synchronization accuracy.

Second, we focus on the *energy-efficiency* of WSN time synchronization. Noting that the message exchanges via radio transmissions occupy a large portion of the total energy consumption of the resource-constrained sensor nodes, we propose a beaconless asymmetric time synchronization scheme (BATS) based on the reverse one-way message dissemination, which could significantly reduce the synchronization-related energy consumption. With a real WSN testbed, we experimentally demonstrate that BATS could significantly reduce energy consumption compared to the benchmark flooding time synchronization protocol and still provide microsecond-level time synchronization accuracy. We also present the generalized reverse asymmetric time synchronization framework providing time synchronization with lower energy consumption and computational complexity.

Third, we shift our focus to the improvement of the *multi-hop performance* of WSN time synchronization. Because the cumulative synchronization error caused by the processing delays at intermediate gateway nodes heavily affects the multi-hop time synchronization accuracy, we propose a per-hop delay compensation scheme (PHDC) based on packet-relaying gateways to compensate for the processing delays. We not only compare the performance of the proposed PHDC with that of the conventional multi-hop extension method based on time-translating gateways through systematic mathematical analyses but also experimentally evaluate its performance in comparison to that of three representative time synchronization schemes based on both one-way and two-way message exchange.

Fourth, we also apply our high-precision asymmetric WSN time synchronization scheme to *optimal data bundling* and propose a data bundling scheme optimizing the energy efficiency under the constraints of application-specified end-to-end delay and

time synchronization accuracy, where the optimization problem is formulated as integer linear programming (ILP). A set of optimal bundling numbers are calculated for sensor nodes in a WSN to maximize energy efficiency while fulfilling the required end-to-end delay and time synchronization accuracy. Extensive experimental results on a real WSN testbed demonstrate that, while satisfying the requirements of delay and synchronization accuracy, the proposed optimal message bundling scheme could significantly reduce the number of total message transmissions.

Finally, we study *node identification* based on our high-precision asymmetric WSN time synchronization scheme, which is considered an important security protection means. We propose a new robust clock-skew-based node identification scheme called node identification against Spoofing attack (NISA). NISA can achieve simultaneous node identification and attack detection based on the high-precision estimation of clock skew for each sensor node enabled by BATS and the spatially correlated radio link information unilaterally collected at the head or gateway nodes. We also provide both centralized and distributed implementations of NISA for covering both single- and multi-hop scenarios. Evaluations using a group of WSN nodes demonstrate the effectiveness of both centralized and distributed NISA. We discuss the research challenges and future directions of WSN time synchronization at the end.

Keywords— Wireless sensor network, multi-hop, time synchronization, energy efficiency, computational complexity, synchronization accuracy, data bundling, node identification, spoofing attack.

Acknowledgements

I am deeply indebted to my dear mentor, Assoc/Prof. Kyeong Soo (Joseph) Kim; without his immense help and continuous encouragement for both my study and life, I couldn't be able to publish any paper, not to mention finishing this PhD thesis. I would also thank my co-supervisors, Assoc/Prof. Sanghyuk Lee, Prof. Eng Gee Lim, and Prof. Alan Marshall, for all their supports in my study at both Xi'an Jiaotong-Liverpool University and the University of Liverpool.

I would like to express my deepest gratitude toward my family members, especially the ones who left me. They supported my academic dreams with their great patience and love; they deserve my deepest apology for my absence in family life during my PhD study—which was, in fact, in day and night mode. For them, a thousand times over.

List of Publications

A number of our publications are listed as follows in support of the work we presented in this thesis:

Journal Papers

1. **X. Huan**, K. S. Kim and J. Zhang, "NISA: Node identification and spoofing attack detection based on clock features and radio information for wireless sensor networks", ***IEEE Transactions on Communications***, pp. 1–1, 2021, Early Access. [SCIE]

2. **X. Huan**, K. S. Kim, "Per-hop delay compensation in time synchronization for multi-hop wireless sensor networks based on packet-relaying gateways", ***IEEE Communications Letters***, vol. 24, no. 10, pp. 2300–2304, Oct. 2020. [SCIE]

3. **X. Huan**, K. S. Kim, S. Lee, E. G. Lim and A. Marshall, "A beaconless asymmetric energy-efficient time synchronization scheme for resource-constrained multi-hop wireless sensor networks", ***IEEE Transactions on Communications***, vol. 68, no. 3, pp. 1716–1730, Mar. 2020. [SCIE]

4. **X. Huan**, K. S. Kim, "On the practical implementation of propagation delay and clock skew compensated high-precision time synchronization schemes with resource-constrained sensor nodes in multi-hop wireless sensor networks", ***Com-***

puter Networks, vol. 166, no.106959, Jan. 2020. [SCIE]

5. **X. Huan**, K. S. Kim, S. Lee, and M. K. Kim, "Optimal message bundling with delay and synchronization constraints in wireless sensor networks", ***Sensors***, vol. 19, no. 18:4027, Sep. 2019. [SCIE]

6. **X. Huan**, K. S. Kim, S. Lee, E. G. Lim and A. Marshall, "Improving multi-Hop time synchronization performance in wireless sensor networks based on packet-relaying gateways with per-hop delay compensation", under review in ***IEEE Transactions on Communications***, (2nd round review after major revision). [SCIE]

Conference Papers

1. **X. Huan** and K. S. Kim, "High-precision time synchronization for wireless sensor networks: Lessons from the experiments on a real testbed," in *Proc. 2019 International Conference on Smart Grid Technology and Data Processing (SGTDP)*, Suzhou, China, Mar. 2019.

2. S. Fu, M. Ceriotti, Y. Jiang, C. Shih, **X. Huan**, P. J. Marrón, "An approach to detect anomalous degradation in signal strength of IEEE 802.15.4 links," in *Proc. 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Hong Kong, Jun. 2018.

3. Z. Zhong, Z. Tang, X. Li, T. Yuan, Y. Yang, W. Meng, Y. Zhang, R. Sheng, N. Grant, C. Ling, **X. Huan**, K. S. Kim, and S. Lee, "XJTLUIndoorLoc: a new fingerprinting database for indoor localization and trajectory estimation based on Wi-Fi RSS and geomagnetic field," in *Proc. 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, Takayama, Japan, Nov. 2018, pp. 228-234.

Contents

List of Figures

List of Tables

List of Acronyms

ACK	Acknowledgment
AHTS	Asymmetric high-precision time synchronization
AI	Artificial intelligence
BATS	Beaconless asymmetric time synchronization
CAN	Controller area network
CDF	Cumulative distribution function
CESP	Coefficient exchange synchronization protocol
CNN	Convolutional neural network
CO	Crystal oscillator
CR	Cumulative ratio
CSNI	Clock-skew-based node identification
CTP	Collection tree protocol
DCO	Digitally-controlled oscillator
DF	Device fingerprinting
DoS	Denial of service

DSO Digital storage oscilloscope

E2E End-to-end

EE-ASCFR Energy-efficient time Synchronization based on asynchronous source clock frequency recovery

EGSync External gradient time synchronization protocol

FCSA Flooding with clock speed agreement

FTSP Flooding time synchronization protocol

FTSP-LPL Flooding time synchronization protocol - low power listening

ID Identifier

ILP Integer linear programming

IoT Internet of things

GRU Gated recurrent unit

KNN K-nearest neighbors

LoS Line-of-sight

LQI Link quality indicator

LSTM Long short-term memory

MAC Media access control

MAE Mean absolute error

MCU Microcontroller unit

ML Machine learning

MSE Mean squared error

NISA Node identification against Spoofing attack

PC Personal computer

PHDC Per-hop delay compensation

PRR Racket receive ratio

RBS Reference-broadcast synchronization

ReLu Rectified linear unit

RF Radio frequency

RNN Recurrent neural network

RSP Ratio-based time synchronization protocol

RSS Received signal strength

RSSD RSS Distribution

RSSI Received signal strength indicator

RTSP Recursive time synchronization protocol

SA Synchronization accuracy

SCFR Source clock frequency recovery

SFD Start frame delimiter

SI Synchronization interval

SIMO Single-input and multiple-output

TPSN Timing-sync protocol for sensor networks

TSC Time-series classification

TT Time translation

USRP Universal software radio peripheral

WLAN Wireless local area network

WSN Wireless sensor network

List of Symbols

Notations

$(\cdot)^{-1}$	Matrix inverse
$(\cdot)^{\top}$	Vector transpose
$\mathrm{E}[\cdot]$	Expectation
$\mathrm{frac}(\cdot)$	Fractional part
$\lfloor \cdot \rfloor$	Floor function

Parameters

$\bar{b}$	Binary fraction
Δ	Delay
ϵ	Clock skew
Γ	Number of bundled messages
$\hat{\epsilon}$	Estimated clock skew
$\hat{\theta}$	Estimated clock offset
$\mathbf{\Phi}$	Vector containing estimated clock skew and offset

$\mathbf{E}$	Total energy consumption
$\mathbf{I}$	Measurement interval
$\mathbf{SA}$	Synchronization accuracy requirement
$\mathcal{O}$	Computational complexity
$\mathcal{T}$	Temperature
$\mathcal{V}$	Voltage
$\mathscr{R}$	Relationship between SA and SI
$\mathscr{T}$	Logical clock time of sensor node
Ψ	Computational error
σ	Sign
θ	Clock offset
ε	Precision loss
$\widehat{T}$	Compensated time
$\widetilde{T}$	Translated time
C	Number of variables
E	Energy consumed
e	Integer exponent
G	Number of training samples
I	Current
L	Number of links

M Number of measurements

N Number of hops

P Power

R Clock frequency ratio

S Size of timestamp in bits

T Hardware clock time

t Reference clock time

x Non-zero floating-point number

X, Y, Z Random variables

D Link delay

Chapter 1

Introduction

Wireless sensor networks (WSNs) have been a key for generations of smart applications from conventional monitoring [8, 9] to thriving smart cities [10, 11] to novel device-free wireless sensing [12]. Time synchronization as the essential service enabling data ordering and collaborative actions for WSN applications has been under the spotlight for decades in the research community. WSN time synchronization schemes could be classified into three major categories according to the way of exchanging timing messages, i.e., *two-way message exchange*, *one-way message dissemination*, and *receiver–receiver synchronization* [13, 14]. The schemes belonging to the last category could reduce the time-critical-path [15], while those belonging to the first and the second categories have the advantage of simplicity especially considering the multi-hop extension. Specifically, the two-way schemes can compensate for the propagation delay and, therefore, cover larger areas [16, 17]. The one-way schemes, on the other hand, require fewer message transmissions and are simpler to implement, which make them more popular than the two-way schemes [18, 19]. Note that there are variations of the two-way and one-way schemes based on the reverse asymmetric time synchronization framework [1], which could significantly reduce the energy consumption and the computational complexity of sensor nodes.

Traditionally, the focus of research in WSN time synchronization is on time synchronization accuracy [18], but recent works take into account not only synchronization

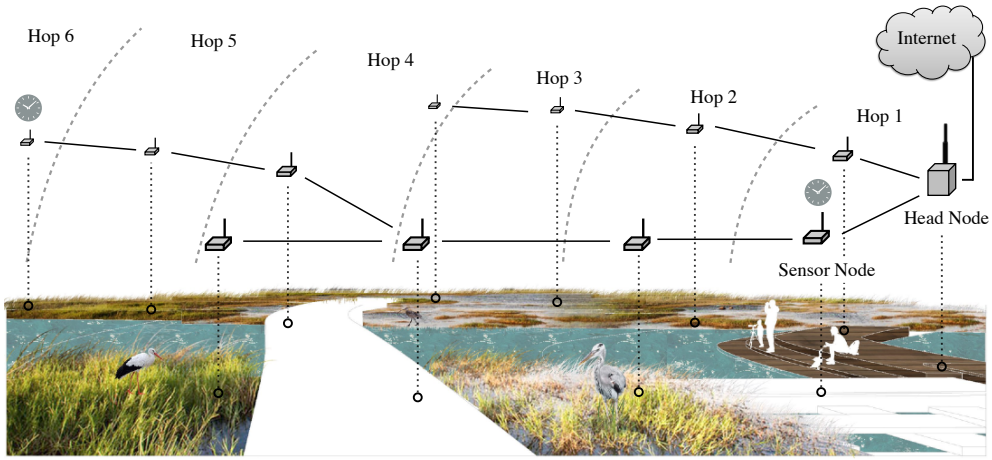


Figure 1.1: A multi-hop WSN with one head node and multiple sensor nodes monitoring an ecological park.

accuracy but also energy-efficiency [2, 20], computational complexity [3, 21], and multi-hop synchronization accuracy [4, 22]; specifically, the latest developments try to improve multi-hop time synchronization accuracy while maintaining the energy consumption and computational complexity as low as possible [1]. Motivated by the aforementioned development focuses, this thesis aims to investigate those three aspects.

In this chapter, we explain the basic concepts of WSN and its time synchronization, including clock models, timing message exchanges, and multi-hop extensions. We also summarize our contributions to the development of WSN time synchronization. We conclude this chapter with the organization of the rest of the thesis.

1.1 WSN and Its Time Synchronization

A typical WSN consists of a large number of resource-constrained, battery-powered sensor nodes and a rather resource-abundant head[1] connected to Internet as shown in Fig. 1.1. Together with Internet of Things (IoT), WSN penetrates into smart cities, smart factories, and smart healthcare as one of the enabling technologies. Through exchanging messages with one another or directly with the head, the sensor nodes not

[1]We consider the head node and a monitoring station—e.g., a PC—connected to it collectively as the head in this thesis.

only pass their measurement data to the head but also achieve network-level collaborative actions. These operations are based on many services provided by WSN, and time synchronization is one of the critical services. The research on WSN time synchronization has been attracting a great deal of attention for recent decades due to its significance to a myriad of smart applications. One up-to-date representative example is the smart healthcare [23], where the collaborative monitoring of vital health signs is of importance, especially during pandemics such as COVID-19 [24].

Note that *time synchronization* is a process of establishing a common time frame with which the nodes in a network can operate one another, whether their clocks are synchronized or not, while *clock synchronization* is a process of synchronizing each node clock to that of a common reference node [2]; clock synchronization, therefore, is a special case of time synchronization. Establishing time synchronization in a WSN relies on three foundations, namely mathematical modeling of hardware and software clocks, ways of exchanging messages for the spread of time information, and multi-hop extension. In the following, we discuss those three foundations of WSN time synchronization as the preliminaries for the work presented in this thesis.

1.1.1 Clock Models

As temperature, battery voltage level, and other environmental factors have a significant impact on the behavior of sensor nodes' hardware clocks, different mathematical models of hardware clocks could be employed under different conditions. Due to the repetitive nature of time synchronization operations, however, static & linear models could be used for sensor nodes' hardware clocks for a relatively short period of time even in the presence of aforementioned environmental factors [1]. For this reason, the first-order affine clock model is widely used in modeling hardware clocks in WSN time synchronization [2].

We consider a typical WSN where its reference clock is the hardware clock of the head and the operations of all sensor nodes are to be based on the reference clock. Because the crystal oscillator of each sensor node runs independently of one another,

the time according to its hardware clock could differ from that of the reference clock. WSN time synchronization, therefore, is to estimate the difference between the reference clock and the sensor nodes' hardware clocks in terms of the parameters of a clock model. With respect to the head's reference clock t, the sensor node i's hardware clock $T_i(t)$ can be described as follows [2]: For $i \in [0, 1, \ldots, n-1]$:

$$T_i(t) = (1 + \epsilon_i)\, t + \theta_i, \tag{1.1}$$

where n is the number of sensor nodes, and $(1+\epsilon_i) \in \mathbb{R}_+$ and $\theta_i \in \mathbb{R}$ denote the frequency ratio and the clock offset, respectively. Note that the clock skew $\epsilon_i \mathbb{R}$ is the difference between two clocks at a given point in time, whose typical value for clocks based on crystal oscillators is in order of tens of ppm, i.e., $\epsilon_i \ll 1$ [2].

Based on the hardware clock model of (1.1), the logical clock $\mathscr{T}_i$ of sensor node i, which is updated during each synchronization round and represents the current estimation of the reference clock as a function of the hardware clock $T_i(t)$, can be described as follows [2]: For $t_k < t \leq t_{k+1}$ $(k=0, 1, \ldots)$,

$$\mathscr{T}_i\Big(T_i(t)\Big) = \mathscr{T}_i\Big(T_i(t_k)\Big) + \frac{T_i(t) - T_i(t_k)}{1 + \hat{\epsilon}_{i,k}} - \hat{\theta}_{i,k}, \tag{1.2}$$

where t_k represents the reference time for the kth synchronization, and $\hat{\epsilon}_{i,k}$ and $\hat{\theta}_{i,k}$ are the estimated clock skew and offset from the kth synchronization. Note that the way and the location of clock parameter estimation depend on the underlying time synchronization scheme: In [2], for instance, a sensor node synchronizes only its clock frequency (i.e., $1+\hat{\epsilon}_{i,k}$) to that of the reference clock, but its clock skew (i.e., $\hat{\theta}_{i,k}$) is estimated & compensated for at the head; in contrast, those estimations are all done at the head in [3].

1.1.2 Timing Message Exchange

Fig. 1.2 (a) illustrates the one-way message dissemination where the head sends a beacon message containing the reference time $T1$ to the sensor node, which records the time $T2$

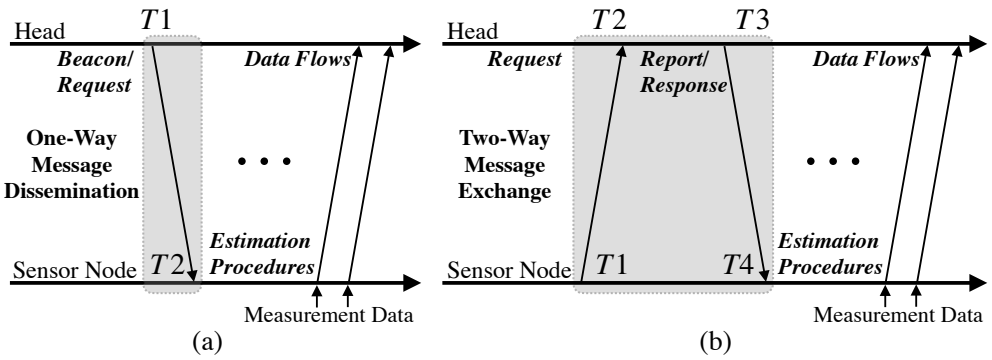


Figure 1.2: Time synchronization based on (a) one-way message dissemination and (b) two-way message exchange [1].

at the moment of receiving the beacon message. Based on the pairs of those timestamps, the sensor node estimates the clock parameters for achieving time synchronization. Note that the propagation delay—i.e., the time the beacon message spends in the air between the nodes—cannot be calculated and, therefore, compensated for, which is the major drawback of the one-way schemes. Thanks to its round-trip nature, by the way, the two-way message exchange could properly compensate for the propagation delay and, as a result, provide more accurate time synchronization. Fig. 1.2 (b) shows the details of two-way message exchange: The sensor node initiates a request for synchronization and records the sending time $T1$. The head receives the request message at time $T2$ and replies a response message containing the reception time $T2$ and the transmission time $T3$ to the sensor node. Finally, the sensor node records the reception time $T4$ and establishes the time synchronization using those four timestamps, where the said propagation delay is also estimated and compensated for.

Although the compensation of the propagation delay is critical for a long distance, the two-way message exchange requires two times the number of message transmissions of the one-way message dissemination. This is its major disadvantage for WSNs consisting of resource-constrained sensor nodes, because the energy for message transmissions occupies a major portion of the total energy consumption at sensor nodes. The one-way message dissemination, in contrast, provides a simpler implementation despite the

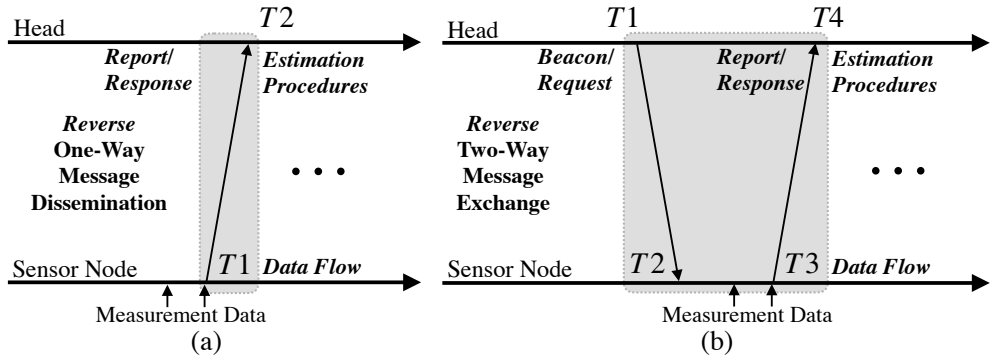


Figure 1.3: Time synchronization based on reverse (a) one-way message dissemination and (b) two-way message exchange with optional bundling of measurement data [1].

uncompensated propagation delay, which could be ignored when the distance between two nodes is short [1].

Fig. 1.3 (b) shows the reverse two-way message exchange employed in [2, 3]. The reversing of the message exchange turns the request and response messages into the pair of beacon and report messages; the head thus initiates the synchronization procedure and establishes the time synchronization. Specifically, the timestamps $T2$ and $T3$ are recorded at the sensor node and transmitted to the head, which establishes the time synchronization using the received timestamps from the sensor node and the timestamps $T1$ and $T4$ it recorded. During the time synchronization, synchronization data could be bundled together with measurement data and carried in a report message to further reduce the number of message transmissions. Note that, unlike the reverse two-way message exchange still relying on beacon messages, the reverse one-way message dissemination is completely free from beacon messages as illustrated in Fig. 1.3 (a). The only synchronization-related message transmission is the report message containing synchronization data—i.e., timestamp $T1$—that could be further bundled together with the measurement data.

Overall, the reverse schemes reduce the number of message exchanges required for time synchronization and relocate the clock parameter estimation procedure from sensor nodes to the head, which could lower both the energy consumption and the computa-

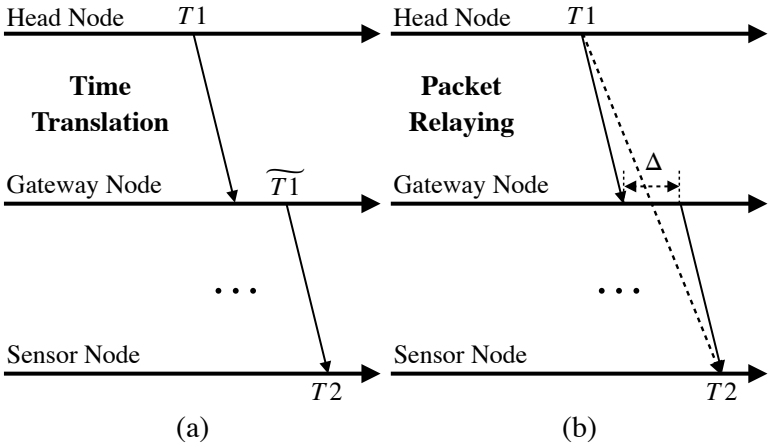


Figure 1.4: Multi-hop extension methods for WSN time synchronization: (a) Time translation and (b) packet relaying.

tional complexity of sensor nodes.

1.1.3 Multi-Hop Extension

A WSN is often deployed in large-scale, where the single-hop communication from the head could not cover the whole deployment area; the multi-hop extension, therefore, becomes indispensable, and so does the multi-hop time synchronization. Note that the multi-hop extension is also employed to either save the energy of single-hop transmission by reducing the hop distance or safeguard line-of-sight wireless communication [4].

In the literature, two major multi-hop extension methods are advocated, namely the time translation and packet relaying [2]. Fig. 1.4 (a) and (b) demonstrate these methods on the conventional one-way message dissemination where the head initiates the synchronization process. In the case of time translation, the head broadcasts a synchronization message containing reference time $T1$ to the gateway node who, in turn, synchronizes its time to that of the head; afterwards, the gateway node translates the reference time to $\widetilde{T1}$ based on the gateway node's forwarding time and then forwards the synchronization message to the sensor node. From the sensor node's perspective, its time is still synchronized to the head's time due to the time translation.

In the packet relaying method, on the contrary, the gateway node does not translate the reference time from the head; it rather directly relays a synchronization message with the original $T1$ to the sensor node. The computational complexity of doing so is much lower than the aforementioned time translation. However, the gateway node confronts a delay—i.e., Δ in Fig. 1.4 (b)—during the relaying process, which deteriorates the synchronization accuracy, especially through multiple hops. It is this processing delay that prevents the use of simple packet relaying in practice. Most WSN time synchronization schemes, therefore, employ the time translation method for multi-hop extension despite its rather complex procedure [1, 18].

1.2 Contributions

The contribution of this thesis centers around the development of WSN time synchronization schemes targeting the three major aspects of energy efficiency, computational complexity, and multi-hop synchronization accuracy. We not only propose high-precision asymmetric WSN time synchronization schemes but also apply them to the problems of optimal bundling with delay & synchronization constraints and node identification against Spoofing attack. We also highlight the challenges and future directions of research in WSN time synchronization. Below is a summary of our contribution:

- Through the implementation of the state-of-the-art WSN time synchronization schemes based on asynchronous source clock recovery and reverse two-way message exchange [2]—called *EE-ASCFR* throughout the thesis—on a real testbed consisting of TelosB sensor nodes [25], we investigate the issue of precision loss resulting from the limited precision in floating-point arithmetic at sensor nodes; the results of the investigation reveals that the design of WSN time synchronization schemes targeting resource-constrained sensor nodes should take into account *computational complexity*. Building on EE-ASCFR, therefore, we propose the asymmetric high-precision time synchronization (AHTS) [3] where most of the computations related with time synchronization are done at the head, and, as a

result, the computational complexity at sensor nodes is significantly reduced.

- Focusing on *energy-efficiency*, we propose a beaconless asymmetric time synchronization (BATS) [1] scheme based on the reverse one-way message dissemination. BATS no longer relies on the beacon message flooding through the network for synchronization purpose; instead, its synchronization data could be embedded into measurement messages through data bundling. The ensemble of the above two measures enables BATS to be energy-efficient. The performance evaluation on a real WSN testbed demonstrates that BATS could conserve significant energy compared to flooding time synchronization protocol (FTSP) [18], the benchmark one-way scheme. The multi-hop extension of BATS is also presented, which employs the per-hop synchronization strategy, i.e., a variation of the time translation method.

- To improve the *multi-hop synchronization accuracy* of WSN time synchronization, we investigate the major cause of the cumulative synchronization errors with a focus on multi-hop extension schemes. To alleviate the cumulative synchronization errors, we propose a per-hop delay compensation (PHDC) scheme [4] based on the multi-hop extension method of packet-relaying [2]. We then comprehensively analyze the errors in the conventional time translation (TT) and PHDC schemes and carry out a comparative analysis of their performance through the experiments on a real WSN testbed, showing the improvement of PHDC over TT on schemes based on both one-way and two-way message exchange.

- Data bundling, which is one of the effective measures for energy conservation [2, 1], not only affects energy consumption but also end-to-end (E2E) delay and, when synchronization data is carried by measurement messages, synchronization accuracy. Tuning the bundling number for energy-efficiency, therefore, should simultaneously consider E2E delay and synchronization accuracy, which turns to be an optimization problem maximizing the number of bundled data in a single message under the constraints of application-specified E2E delay and synchro-

nization accuracy requirements. We apply our proposed high-precision WSN time synchronization scheme and propose an optimal data bundling method that uses integer linear programming to optimize the number of bundled data at each sensor node in a WSN while satisfying the required E2E delay and synchronization accuracy. Extensive practical experiments demonstrate the effectiveness of the proposed approach.

- We also apply the proposed high-precision asymmetric WSN time synchronization scheme to node identification, which is considered an important security measure. The high-precision estimation of clock skews enabled by BATS could greatly enhance the node identifiability. We also investigate the identifiability of the clock skews under environmental variations, i.e., the temperature and the voltage levels of batteries powering sensor nodes. Combining the spatially correlated radio link information, we propose a node identification scheme called node identification against Spoofing attack (NISA) where the node identification and attack detection could be achieved simultaneously in an integrated way. We further implement NISA in both centralized and distributed manner for covering both single- and multi-hop scenarios. The effectiveness of both centralized and distributed NISA is demonstrated through the practical experiments on a real WSN testbed. We finally discuss the research challenges and future directions of WSN time synchronization.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

- **Chapter 2** reviews the representative and up-to-date WSN time synchronization schemes based on the taxonomy and the development focus.

- **Chapter 3** investigates the computational complexity of WSN time synchronization with a major focus on the precision loss in floating-point arithmetic at sensor

nodes and discusses how to address it through the asymmetric high-precision time synchronization scheme.

- **Chapter 4** presents the results of our investigation of the energy efficiency in WSN time synchronization and discusses the resulting energy-efficient beaconless asymmetric time synchronization scheme.

- **Chapter 5** considers the multi-hop synchronization accuracy of WSN time synchronization with a focus on the processing errors at intermediate gateway nodes and provides a solution based on per-hop delay compensation.

- **Chapter 6** applies our proposed WSN time synchronization scheme to optimal data bundling with end-to-end delay and synchronization accuracy constraints.

- **Chapter 7** demonstrates the application of our proposed WSN time synchronization scheme to WSN node identification against Spoofing attack.

- **Chapter 8** concludes our work in this thesis and envisions future research directions of WSN time synchronization and its applications.

Chapter 2

Review of WSN Time Synchronization

In this chapter, we review the representative WSN time synchronization schemes based on the taxonomy discussed in Chapter 1. We also review the up-to-date schemes based on the development focus of energy-efficiency, computational complexity, and multi-hop synchronization accuracy.

2.1 Taxonomy of WSN Time Synchronization

Based on the way of transferring timing messages to sensor nodes, WSN time synchronization schemes could be classified into three major categories: *two-way message exchange*, *one-way message dissemination*, and *receiver–receiver synchronization* [13]. The former two categories could cover most WSN time synchronization schemes, which provide absolute timescales among sensor nodes with respect to the clock of a reference node (often called a head or a root node). Schemes based on two-way message exchange—e.g., timing-sync protocol for sensor networks (TPSN) [16] and recursive time synchronization protocol (RTSP) [17]—can compensate for propagation delay and, therefore, provide more accurate time synchronization. Though not being able to compensate for propagation delay, by the way, schemes based on one-way message dissemination

can save the number of message exchanges and simplify the implementation at the expense of synchronization accuracy, which makes them popular for resource-constrained WSNs with moderate coverage (e.g., 1 µs propagation delay for 300 meters); the flooding time synchronization protocol (FTSP) [18]—a representative of the one-way WSN time synchronization schemes—was the first to synchronize multi-hop WSNs through flooding synchronization messages, and many schemes leveraging the flooding time synchronization framework like flooding with clock speed agreement (FCSA) protocol [26], PulseSync [19] and external gradient time synchronization protocol (EGSync) [27] have been proposed to enhance the accuracy and coverage time of synchronization. Note that the introduction of media access control (MAC)-layer timestamping [18, 17] reduces the effect of random delays in timestamping and therefore greatly improves the synchronization accuracy of the schemes based on either two-way message exchange or one-way message dissemination.

The receiver-receiver synchronization, on the other hand, has been studied due to its distributed nature and reduction of the time-critical-path [15]: The reference broadcast synchronization (RBS) algorithm [28] exploits the broadcast channel through which messages from a sender are delivered to multiple receivers approximately at the same time. The receiver-to-receiver referenceless synchronization (termed in R4Syn) protocol proposed in [29] further distributes the role of the reference to all sensor nodes, which makes it completely decentralized. The reference broadcast infrastructure synchronization (RBIS) [30] investigates the applicability to industrial and home automation networks, while coefficient exchange synchronization protocol (CESP) [31] enhances the energy-efficiency.

2.2 Development Focus of WSN Time Synchronization

From initially concentrating on the multi-hop synchronization accuracy alone [18] to recently taking energy efficiency, computational complexity, and multi-hop synchronization accuracy into simultaneous consideration [1], WSN time synchronization has

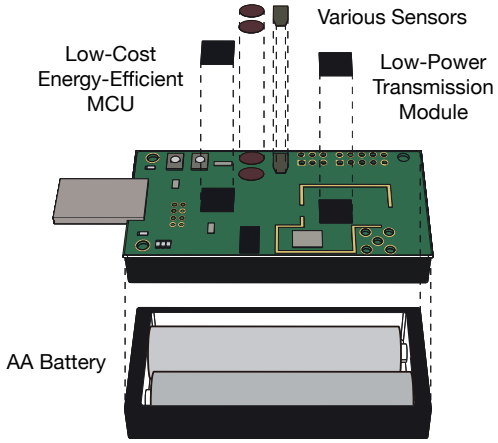


Figure 2.1: An exploded view of the typical WSN sensor node.

shifted its development focus from a single to a diverse direction riding the wave of WSN deployments proliferation. In this section, we review the up-to-date WSN time synchronization schemes based on the three major development focuses of energy-efficiency, computational complexity, and multi-hop synchronization accuracy. In particular, we put emphasis on the last focus since it is crucial to the prosperous large-scale deployments.

2.2.1 Energy Efficiency

A WSN sensor node usually equips limited power resources—e.g., AA batteries illustrated in Fig. 2.1—due to its applications in rather complex, unattended field domains. Not only should it prioritize energy efficiency for its essential components such as MicroController Unit (MCU) and transmission module in both their design and operation, but also for its fundamental services such as time synchronization.

A novel energy-efficient scheme is proposed in [2], it reverses the direction of the time synchronization based on two-way message exchange whereby the regular beacon messages could be employed as the synchronization beacon messages as exhibited in Fig. 1.3 (a); moreover, it proposes a data bundling method to embed the synchronization data into the application message for conserving more message transmissions. However, since the two-way message exchange it relies on conceptually requires more

message transmissions compared with the one-way message dissemination, there has room for further improvement. In succession, a beaconless asymmetric time synchronization scheme [1] based on the reverse one-way message dissemination as illustrated in Fig. 1.3 (b) has been proposed. The beacon messages broadcasting the reference time in the conventional schemes are no longer necessary, which reduces the majority of synchronization-related message transmissions. Moreover, the synchronization-related data are bundled in the application messages to further reduce the synchronization messages. Likewise, [20] proposes another energy-efficient scheme by employing the existing acknowledgment (ACK) messages for synchronization, therefore, reduces the synchronization message transmissions.

2.2.2 Computational Complexity

The computing power of the MCU has been significantly improved over the decades, which, however, still at the expense of high energy consumption. Mainly being limited by the size, sensor nodes could not equip abundant energy storage. Employing tailored energy-efficient MCUs as exhibited in Fig. 2.1 having limited computing power is rather the norm for those resource-constrained sensor nodes. Nevertheless, the limited computing power is overloaded with various functionalities from sensing, routing to data processing and transmission. WSN time synchronization building on complex mathematical computations such as linear regression [18] for its clock parameter estimation, on the other hand, has to develop towards lowering its computational complexity.

For reducing the computational complexity, there emerge schemes proposing simpler synchronization algorithms; one up-to-date representative is [21], which reduces the calculation operations—i.e., fewer calculation operations could lead to less computational complexity. However, those schemes still put computational pressure—though relatively less—brought by the time synchronization on the resource-constrained sensor nodes. A more thorough method is computation offloading, [3] reassigns all the synchronization-related computations from the sensor nodes to the head, the latter of which has abundant computing resources. The sensor nodes are thus completely free

from synchronization-related computations; thereby, the computational complexity is reduced significantly. Note that, the method of reassigning the synchronization-related computations has been maturely employed in other schemes such as [1].

2.2.3 Multi-Hop Time Synchronization Accuracy

The multi-hop nature of WSN puts a strict requirement on the WSN time synchronization for not merely its coverage over multiple hops but also its multi-hop synchronization accuracy. Multi-hop WSN time synchronization is demanded to provide common time for the sensor nodes locating tens of meters to kilometers away to accurately perform tasks from data ordering to collaborative actions. However, as WSN deployments scale in size and coverage, achieving comparable synchronization accuracy from nearest hop to farthest hop becomes more and more challenging, especially considering the previously discussed issues of energy consumption and computational complexity. Therefore, a major development focus of WSN time synchronization is to improve the multi-hop synchronization accuracy—especially for far hops—while keeping energy consumption and computational complexity as low as possible.

Unlike the aforementioned rather unified path in reducing the energy consumption and computational complexity, there bloom diverse methods for improving the multi-hop time synchronization accuracy. We review the up-to-date schemes in the following:

2.2.3.1 Rapid-Flooding

For decades, the flooding time synchronization schemes have been realizing multi-hop time synchronization through layer-by-layer rebroadcasting of synchronization messages, where the sensor node at each layer receives the synchronization message and synchronizes its time with the reference node. Through time translation at the gateway node—i.e., intermediate sensor nodes located in between sensor nodes and reference node, sensor nodes locating at far hops could synchronize themselves with the reference node. However, a well-known issue in the layer-by-layer synchronization is the per-hop cumulated synchronization error. As of writing this review, one successor [22]

of the flooding time synchronization proposes a rapid-flooding method to reduce the per-hop cumulated synchronization error, therefore, increases the multi-hop time synchronization accuracy. In the rapid-flooding method, more synchronization messages are flooded in one synchronization interval in each hop for reducing the synchronization error, however, at the expense of more message transmissions, which inevitably leads to more energy consumption.

2.2.3.2 Consensus-Based

The consensus-based time synchronization has gained much attention due to its distributed nature and robustness [32] in the past decade. Many contributions have been made to the development of consensus time synchronization with major concentrations on the mathematical model, convergence time, and delay handling. Of particular note is that, the fundamental idea of employing the consensus concept to achieve a common virtual logical clock among all sensor nodes naturally paves the way for covering multi-hop scenarios. Recently in [33], consensus-based time synchronization is employed for the multi-hop WSNs by combining the master node synchronization and medium access control (MAC)-layer timestamping. The potential of the consensus-based scheme in improving the multi-hop synchronization accuracy is well demonstrated through simulation: The average synchronization accuracy of the consensus-based scheme is comparable to that of the flooding scheme.

2.2.3.3 Pulse-Coupled

Initially inspired by the fireflies' behavior, the model of pulse-coupled oscillators is proposed to achieve time synchronization in wireless networks including WSN. During the development of pulse-coupled synchronization, its major drawbacks obstructing the realization of network-level time synchronization, such as the assumption of concurrent transmission, have been overcome through employing mechanisms like desynchronization [34]. One of the recent developments of pulse-coupled synchronization is the coverage of the multi-hop scenarios, e.g., [35] bases on the desynchronization and proposes

a state-space model for the pulse-coupled drifting oscillators to realize the multi-hop synchronization. Additionally in [35], pulse packets with timestamps are employed to estimate the offsets; and an attenuated clock correction scheme is proposed for correcting the drifts of the clocks. Its time synchronization performance in the multi-hop scenario is demonstrated through the evaluations on a simulated seven-hop linear network.

2.2.3.4 Per-Hop Delay Compensation

While many schemes improve the multi-hop synchronization accuracy by enhancing the estimation method and so on, few of them turn their attention to the foundation of multi-hop time synchronization—i.e., the multi-hop extension method, which, however, has a significant impact on the synchronization error accumulation at gateway nodes. As illustrated in [2], two methods, namely time translation and packet relaying, could be leveraged for extending the time synchronization to multi-hop. The packet relaying has the advantage of simple operation; however, it confronts the problem of delay deteriorating the multi-hop synchronization performance significantly. It is for this reason, most conventional multi-hop time synchronization schemes such as flooding schemes employ the time translation method for their multi-hop extension. Because of the simple nature of packet relaying that suits the resource-constrained sensor nodes, PHDC is proposed in [4] to effectively cure the delay that existed in relaying the packets, which paves the way for bringing the packet relaying method back into use.

Specifically, when the synchronization message being relayed in the gateway node, PHDC calculates the delay and compensates for it based on the head node's time, in which a simple ratio-based method is employed to calculate the frequency ratio between itself and the head node. Note that, the delay is calculated based on the MAC-layer timestamps in order to capture most delay components. Thus, a virtual direct connection could be established between the sensor node and head node whereby the reference time received in the sensor node is much closer to the actual time in the reference node, which leads to more accurate multi-hop time synchronization.

Chapter 3

An Asymmetric High-Precision Time Synchronization Scheme for Low Computational Complexity

In this chapter, we focus on the *computational complexity* of the WSN time synchronization; the work presented here is based on our publication of [3]. On a real testbed, we investigate the practical implementation of the state-of-the-art WSN time synchronization scheme EE-ASCFR that builds on the asynchronous source clock frequency recovery and the reverse two-way message exchange, which can compensate for both propagation delay and clock skew for higher precision and energy efficiency. Our investigation reveals that its performance on battery-powered, low-complexity sensor nodes is not up to that predicted from simulation experiments due to the limited precision floating-point arithmetic of sensor nodes, which draws attention to the lower computational capability of typical sensor nodes and its impact on time synchronization.

To reduce the computational complexity of WSN time synchronization on resource-constrained sensor nodes, we propose AHTS whose synchronization-related computations are all done at the head node equipped with abundant computing and power resources. The sensor nodes are responsible for timestamping only, and the computational

burdens of the sensor nodes are, therefore, significantly alleviated. Practical experimental results demonstrate that the proposed AHTS can avoid time synchronization errors resulting from the single-precision floating-point arithmetic of the resource-constrained sensor nodes and achieve microsecond-level time synchronization accuracy in multi-hop WSNs.

3.1 Introduction

High-precision time synchronization is essential to the runtime collaborative applications for wireless sensor networks (WSNs), including time-based channel sharing and media access control (MAC) protocols [36], coordinated duty cycling mechanisms [37], and device-free wireless sensing for human activity and gesture recognition [38, 39]. Considering the increasing number of WSN deployments for a variety of applications, most of which are based on multi-hop topologies with resource-constrained sensor nodes, achieving high-precision time synchronization in multi-hop networks while lowering the computational requirements and energy consumption at sensor nodes is crucial in designing WSN time synchronization schemes.

A novel energy-efficient time synchronization scheme EE-ASCFR has been proposed in [2], where the synchronization-related procedures are assigned to both head and sensor nodes to reduce the computational complexity of the latter. It reverses the direction of the conventional two-way message exchange as illustrated in Fig. 3.1 and employs data bundling to jointly conserve the synchronization message transmissions, which is conducive to energy-efficiency. Compared to the conventional WSN time synchronization schemes whose sensor nodes are responsible for most of the clock estimation procedures (e.g., [16, 17, 18, 40, 41, 42, 31, 43]), EE-ASCFR suits the resource-constrained WSNs more: (a) sensor nodes are relieved from the tasks of clock offset estimation which is done at the head node—also called a sink node in the literature; (b) the logical clocks of sensor nodes are synchronized to the reference clock of the head node in frequency through the asynchronous source clock frequency recovery (SCFR), but they could run

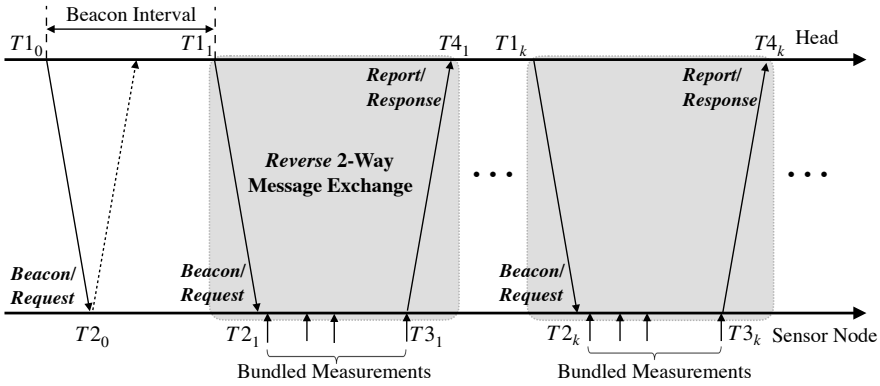


Figure 3.1: Reverse two-way message exchange with optional bundling of measurements introduced in [2].

possibly with different and independent offsets. Although [2] demonstrates through simulation that EE-ASCFR can provide sub-microsecond-level accuracy while significantly reducing the number of message exchanges required for time synchronization, its actual performance was not evaluated on the testbed. We start our investigation by implementing EE-ASCFR on a real WSN testbed in this chapter.

During implementing EE-ASCFR on TelosB [25] motes running TinyOS [44], we find that the limited computing capability of the sensor nodes could result in cumulative synchronization errors. The reason is that the estimation of the frequency ratio and the maintenance of the logical clock require floating-point divisions. However, the limited floating-point precision of the resource-constrained sensor nodes—i.e., 32-bit single-precision on TinyOS—hinders the calculation accuracy, which results in synchronization errors. Moreover, the 32-bit single-precision floating-point representation is the floating-point standard not only of most resource-constrained WSN platforms such as MicaZ [45], Iris [46], and TelosB [25] but also of most Arduino platforms [47], the latter platforms are extremely prevalent in IoT prototyping, and their computing and power resources are also quite limited. Therefore, reducing the computational complexity of the high-precision time synchronization schemes is essential for flourishing WSN deployments based on resource-constrained sensor nodes.

Building on EE-ASCFR, we propose an asymmetric high-precision time synchro-

nization scheme in which all synchronization-related computations are reassigned to the head with abundant computing and power resources; therefore, the investigated precision loss could be avoided. The contributions of our work presented in this chapter can be summarized as follows:

- We first investigate the precision loss issue resulting from the single-precision floating-point format at resource-constrained sensor nodes for WSN time synchronization EE-ASCFR. We further analyze the correlation between this precision loss issue and synchronization errors.

- We then base on the EE-ASCFR and propose an improved WSN time synchronization scheme called AHTS, which can achieve microsecond-level accuracy with resource-constrained sensor nodes through reassigning all synchronization-related computations to the head. We provide as well its multi-hop extension for covering large-scale scenarios.

- We finally present the evaluation results on a real WSN testbed composing of TelosB motes—one of the representative WSN platforms—running TinyOS. Experimental results demonstrate the effectiveness of the proposed scheme in both single-hop and multi-hop scenarios.

The rest of this chapter is organized as follows: The investigations of the impact of the limited precision floating-point arithmetic on time synchronization at resource-constrained sensor nodes are exhibited in Section 3.2. Our proposed AHTS and its multi-hop extension are presented in Section 3.3. Evaluation results performed on a real testbed for the comparative analysis of the performance of the proposed AHTS and EE-ASCFR in both single-hop and multi-hop topologies are illustrated in Section 3.4. The summary of this chapter is narrated in Section 3.5.

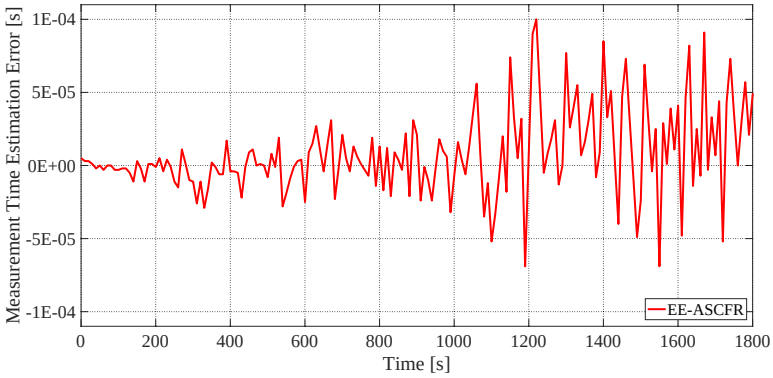


Figure 3.2: Measurement time estimation errors of EE-ASCFR with SI of 1 s evaluated on a real WSN [3].

3.2 Impact of Precision Loss on Time Synchronization

EE-ASCFR employs the first-order affine clock model (1.1) to represent the sensor node's hardware clock with respect to the head's reference clock, where the sensor node maintains a logical clock (1.2) for timestamping [2]. Based on the reverse two-way message exchange shown in Fig. 3.1, the clock frequency ratio in the kth synchronization—i.e., $1+\epsilon_k$ according to (1.1)—could be estimated as $(T2_k - T2_0)/(T1_k - T1_0)$ where the timestamps of Ti_j (i=1, 2 and $j \geq 0$) are recorded during the jth synchronization. As mentioned in Section 3.1, we begin our investigation by implementing EE-ASCFR on a real testbed.

We show in Fig. 3.2 the measurement time estimation errors from the experiment with the testbed for a period of 1800 s. The synchronization interval (SI) is set to 1 s, and 5 measurements are generated and bundled together in a "Report/Response" message to the head in each SI. The absolute value of measurement time estimation error of EE-ASCFR demonstrated in Fig. 3.2 gradually increases from around 2 µs to 100 µs over the period of 1200 s. It indicates that, as we will discuss shortly, the limited precision in floating-point arithmetic of the resource-constrained sensor nodes (i.e., 32-bit single-precision in this case) has negative impacts on the time synchronization performance. This finding is not isolated, however. In the ratio-based time synchronization protocol (RSP) [48] which is, in fact, a flooding time synchronization scheme employing a rather

simpler ratio-based clock estimation method; the impact of computational errors caused by the limited precision floating-point arithmetic on time synchronization has also been discussed. They reveal that a smaller synchronization time interval could lead to larger computational errors, while a larger synchronization time interval could also worsen the synchronization performance due to the clock drift.

To investigate the cause of the error increase in EE-ASCFR, we revisit the arithmetic computations involved with the logical clock updates in (1.2) at the sensor node. Specifically, the division of floating-point numbers (e.g., the division of $T_i(t) - T_i(t_k)$ by $1 + \hat{\epsilon}_{i,k}$ in (1.2)) in the simulation experiments in [2] does not incur much precision loss on the rather resource-abundant personal computers (PCs) and workstations providing 64-bit double-precision floating-point type[1]. In contrast, the floating-point type of most typical WSN platforms based on the low-cost MCU and limited memory space is limited to 32-bit single-precision, which may result in significant precision loss. [29] further confirms our investigation, they mention that implementing high-precision synchronization schemes requiring floating-point division in WSNs has to be discreet due to the hardware limitations of the underlying platforms. In the case of EE-ASCFR, an accurate floating-point division is required in the logical clock update (1.2) at the sensor node. Moreover, the logical clock update has a recursive nature which accumulates the computational errors.

We then make our attempt to avoid the recursive nature of (1.2) in EE-ASCFR and simplify the quantification of the impact of the precision loss induced errors by proposing the following improved logical clock update equation:

$$\mathscr{T}_i\Big(T_i(t)\Big) = \mathscr{T}_i\Big(T_i(t_0)\Big) + \frac{T_i(t) - T_i(t_0)}{1 + \hat{\epsilon}_{i,k}}, \tag{3.1}$$

where the update of the logical clock is based on the value of the logical clock—i.e., $\mathscr{T}_i(T_i(t_0))$—from the initial time synchronization round instead of the previous time synchronization round, i.e., $\mathscr{T}_i(T_i(t_k))$. The time duration since the first time synchro-

[1]The 64-bit double-precision floating-point type is also named as *double* type in common programming languages.

nization is divided by the estimated clock frequency ratio in the updated logical clock. In doing so, the recursive term is eliminated in (3.1). Unfortunately, the practical experiments with the remolded logical clock show still the cumulative errors which are caused by the precision loss involved with the division by $1 + \hat{\epsilon}_{i,k}$ (i.e., the second term in RHS of (3.1)).

In this case, the impact of the precision loss can be analyzed as follows: Because $\hat{\epsilon}_{i,k} \ll 1$ in general, the second term in RHS of (3.1) can be approximated by its first-order Taylor polynomial, i.e.,

$$\frac{T_i(t) - T_i(t_0)}{1 + \hat{\epsilon}_{i,k}} \approx \Big(T_i(t) - T_i(t_0)\Big) \times (1 - \hat{\epsilon}_{i,k}). \tag{3.2}$$

Let ϵ be the precision loss for the clock skew $\hat{\epsilon}_{i,k}$, i.e.,

$$\epsilon \triangleq \hat{\epsilon}_{i,k} - \hat{\epsilon}_{i,k}^{LP}, \tag{3.3}$$

where $\hat{\epsilon}_{i,k}^{LP}$ denotes the actual, imprecise value of the clock skew in implementation due to the limited precision. Therefore, the computational error Ψ caused by the precision loss can be described as follows:

$$\begin{aligned} \Psi &\triangleq \Big(T_i(t) - T_i(t_0)\Big) \times (1 - \hat{\epsilon}_{i,k}) - \Big(T_i(t) - T_i(t_0)\Big) \times (1 - \hat{\epsilon}_{i,k}^{LP}) \\ &= \Big(T_i(t) - T_i(t_0)\Big) \times (\hat{\epsilon}_{i,k}^{LP} - \hat{\epsilon}_{i,k}) \\ &= -\Big(T_i(t) - T_i(t_0)\Big)\epsilon. \end{aligned} \tag{3.4}$$

(3.4) demonstrates that, given the precision loss ϵ, the computational error Ψ is proportional to the time duration since the first time synchronization. This reveals that when the time duration since the first time synchronization, i.e., $T_i(t)$–$T_i(t_0)$, increases, the computational error becomes larger with the updating of logical clock. A further note is that, reducing the SI—e.g., $T_i(t_k)$–$T_i(t_{k-1})$ as in (1.2)—does not mitigate the impact of the precision loss since the computational error Ψ in (3.4) is independent of SI.

To quantify the impact of the precision loss, we turn back to the definition of the

floating-point formats in IEEE standard 754 [49]. In this standard, 7-digit precision is provided for decimal numbers based on the definitions of the 32-bit floating-point and decimal interchange format parameters. The nesC language [50]—the foundation of TinyOS, is basically the extension of the standard C language that follows the IEEE standard 754 as described in [51]. The decimal numbers in the 32-bit floating-point type of nesC, therefore, inherits limited precision of 7 digits. Noting the highest clock resolution of 1 µs in TinyOS with resource-constrained WSN platforms, we take SI of 10 s as an example, the actual value involved in the computation of logical clock in (1.2)—i.e., the difference between the two timestamps—is 10^7 µs. Therefore, in the worst case, the precision loss in the estimated frequency ratio, whose true value is very close to one, would be 10^{-7} (i.e., ϵ).

3.3 Asymmetric High-Precision Time Synchronization

To avoid the precision loss issue revealed in Section 3.2, we propose an asymmetric high-precision time synchronization AHTS whose synchronization-related computations are all reassigned to the head from the sensor nodes. In this section, we describe first the system architecture and basic operations of AHTS and then its multi-hop extension.

3.3.1 System Architecture and Basic Operations

Since typical WSNs are asymmetric where a number of resource-constrained sensor nodes report data to a resource-abundant head [2], we consider reassigning all time synchronization tasks except timestamping from the sensor nodes to head in AHTS to remedy the computation errors investigated in Section 3.2. Specifically, as shown in Fig. 3.3, the logical clock translator described in (3.1) and the frequency ratio estimator (i.e., the cumulative ratio (CR) estimator in [52]) which run at sensor nodes in EE-ASCFR are now part of the time synchronization tasks of the head.

As illustrated in Fig. 3.3, the redistribution of the synchronization-related tasks from the sensor nodes to the head in AHTS leaves just the timestamping procedure to the

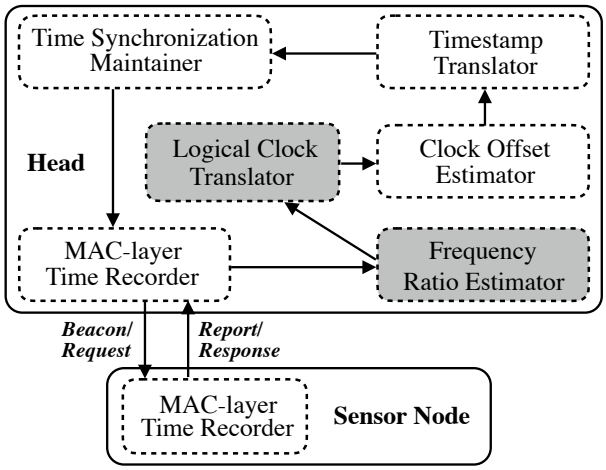


Figure 3.3: System architecture of AHTS for resource-constrained wireless sensor networks [3].

sensor node—i.e., the "MAC-layer Time Recorder". Therefore, the synchronization tasks requiring accurate floating-point arithmetic operations are then done at the head. In doing so, the head's abundant computing resources including the 64-bit double-precision floating-point alleviate the said computation errors caused by the precision loss.

AHTS operates as follows: The time synchronization maintainer initiates the time synchronization process at the head—a hardware clock timestamp $T1$ is recorded by the MAC-layer time recorder and sent to sensor nodes via a *Beacon/Request* message. Then the sensor node receives the *Beacon/Request* message and records its own hardware clock $T2$ at reception. Unlike in the reverse two-way message exchange of EE-ASCFR where the $T2_0$ shown in Fig. 3.1 is not required by the head since the estimation of the clock frequency ratio is completed at the sensor node, in AHTS, $T2_0$ is essential for the head, however. As a result, $T2_0$ has to be delivered from the sensor node to the head at the expense of message transmission. It could be carried either through one additional message after the initial *Beacon/Request* message (i.e., the dotted line in Fig. 3.1) or embedded in the first *Report/Response* message later.

On the other hand, the sensor node records a timestamp T_m based on its hardware clock when a measurement event triggers. Afterwards, a *Response/Report* message carrying the measurement timestamp T_m and the most recently generated $T2$ together

with the hardware clock timestamp $T3$ of its own transmission time is transmitted to the head. The head records a timestamp $T4$ using its MAC-layer time recorder when receiving the *Response/Report* message from the sensor node. Next, based on the differences of current $T2$ and $T1$ to the initial ones (i.e., $T2_0$ and $T1_0$), the head's frequency ratio estimator calculates the clock frequency ratio having a precision of 16 digits [49] by employing the 64-bit double-precision floating-point type. Finally, the clock offset is estimated as in EE-ASCFR based on the reverse two-way message exchange. As of completing the estimations of clock frequency and offset, the timestamp translator converts the measurement timestamp T_m that in sensor node's time into that in the reference time of the head.

Since AHTS inherits the reverse two-way message exchange from EE-ASCFR, the propagation delay could also be properly compensated for. This compensation moreover covers the interrupt delay between the transmission and reception interrupts of a message at the sender and receiver. Note that compensating for those delays is considered the major advantage of the two-way message exchange over one-way message dissemination.

3.3.2 Multi-Hop Extension

Here we provide the multi-hop extension of AHTS and its detailed implementation. As AHTS relies on the two-way message exchange, we first discuss the difference between multi-hop extensions for time synchronization schemes employing one-way message dissemination and two-way message exchange. Fig. 3.4 (a) and (b) exhibit respectively the multi-hop time synchronization schemes based on one-way message dissemination and two-way message exchange. Congenitally, time synchronization schemes based on two-way message exchanges (e.g., [16]) occupy more message transmissions. Therefore, in extending two-way schemes to multi-hop, one additional message at each hop is required to acquire four timestamps for time synchronization compared to one-way schemes. Considering the synchronization of a flat N-hop network as an example, N synchronization messages and $2N$ timestamps are required in one-way schemes as illus-

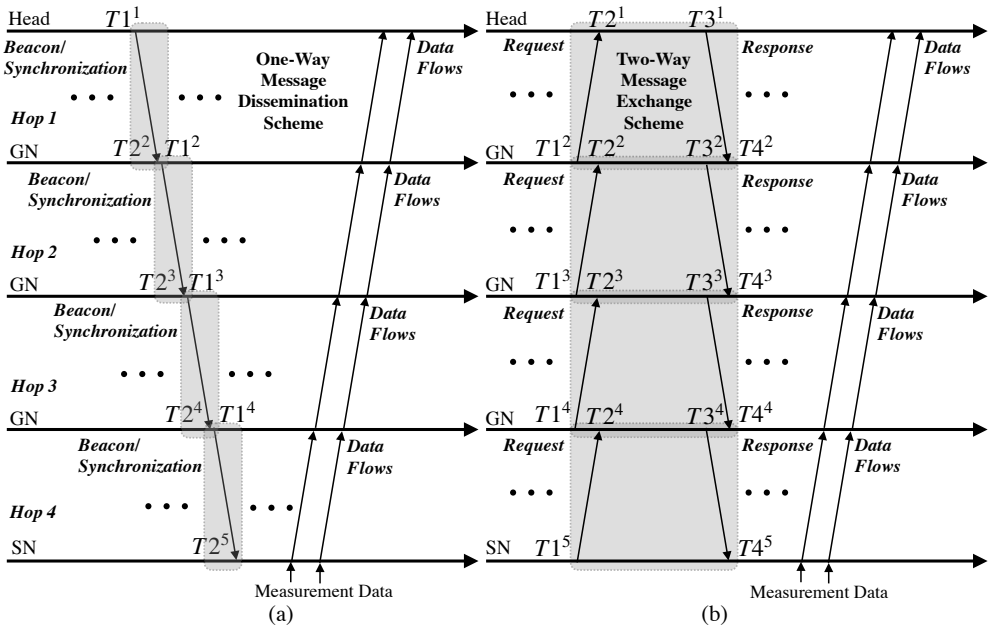


Figure 3.4: Multi-hop extension for time synchronization schemes based on (a) the one-way message dissemination and (b) the two-way message exchange [3].

trated in Fig. 3.4 (a), but $2N$ synchronization messages and $4N$ timestamps in two-way schemes as in Fig. 3.4 (b).

Although time synchronization schemes based on one-way message dissemination require fewer message transmissions, their synchronization accuracy would be deteriorated in a relatively long transmission distance due to their ignoring of the propagation delay. In the case of multi-hop, the impact of ignoring the propagation delay could be accumulated therefore no longer negligible. EE-ASCFR and AHTS thanks to their foundation of reverse two-way message exchange, could not only properly compensate for the propagation delay therefore provide more accurate time synchronization but also address the issue of the increasing number of message transmissions through bundling the measurement and synchronization messages. As shown in Fig. 3.5, the number of message transmissions is significantly reduced in the multi-hop extension based on reverse two-way message exchange compared to that based on the conventional two-way message exchange exhibited in Fig. 3.4 (b).

We adopt time-translating sketched in [2] however relocate it from sensor nodes to

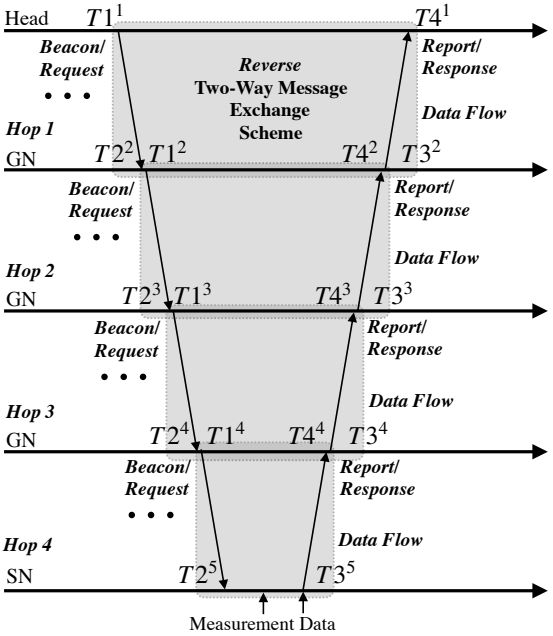


Figure 3.5: Multi-hop extension and data bundling for time synchronization schemes based on the reverse two-way message exchange [3].

the head. As such, sensor nodes serving as intermediate gateway nodes in the multi-hop are completely relieved from the synchronization-related computations that for not only themselves but also their offspring sensor nodes. As for timestamps in multi-hop time synchronization, each hop maintains the same timestamping procedure as the single-hop case discussed in Section 3.3.1 and shown in Fig. 3.5.

With the time translation and the reassignment of its location, the intermediate sensor nodes in AHTS—previously undertake the estimation procedures for the offspring sensor nodes in EE-ASCFR—are responsible for only transferring the timestamps from each hop to the head. The head handles all time translation procedures over hops for each sensor node, and eventually establishes time synchronization for all sensor nodes. In this way, we could also eliminate the impact of any extra delays on multi-hop time synchronization such as packet delays resulting from queueing and MAC operations, which are accumulated through per-hop forwarding in multi-hop networks.

Table 3.1: MAE and MSE of Measurement Time Estimation of EE-ASCFR and AHTS for the Single-Hop Scenario [3]

Synchronization Scheme		MAE [1]	MSE [1]
EE-ASCFR	SI = 100 s	2.7276E-05	1.0391E-09
	SI = 10 s	2.5182E-05	1.1559E-09
	SI = 1 s	2.4069E-05	1.0095E-09
AHTS	SI = 100 s	8.4225E-06	1.2524E-10
	SI = 10 s	2.3385E-06	9.1694E-12
	SI = 1 s	1.8166E-06	5.2094E-12

[1] The samples measured between 360 s (i.e., a tenth of the total observation period) and 3600 s are employed to avoid the effect of a transient period.

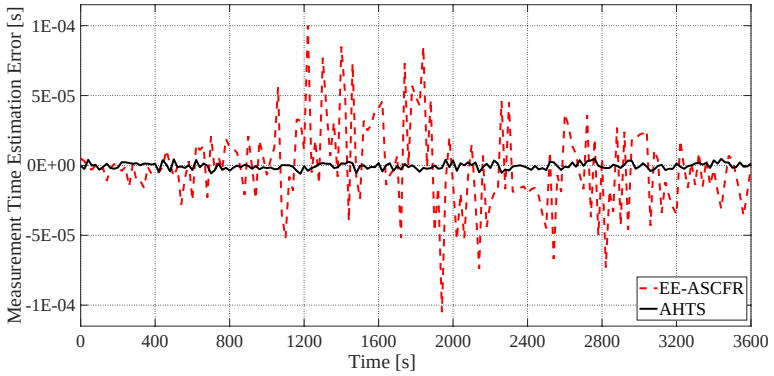
3.4 Experimental Results

To evaluate the actual performance of AHTS, we carry out a series of comparative experiments with a real WSN testbed consisting of TelosB motes running TinyOS. The focus of the evaluations in this section is to show how the proposed AHTS addresses the issue of precision loss which affects the actual performance of EE-ASCFR as revealed in Section 3.2, in both single- and multi-hop scenarios.

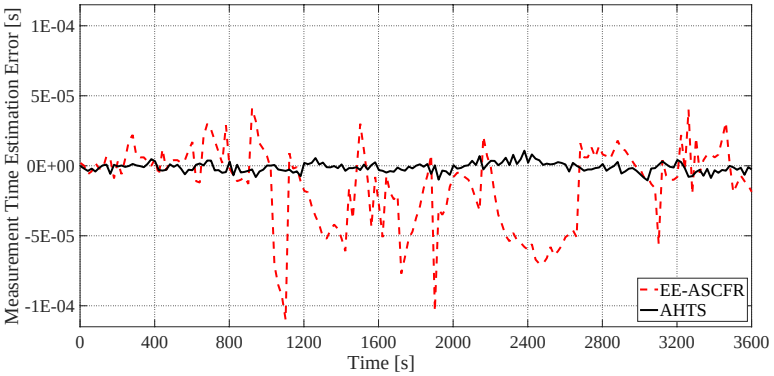
3.4.1 Single-Hop Scenario

We employ a simple single-hop scenario consisting of a head and a sensor node for our evaluation in this subsection. Three different SI values, i.e., 1 s, 10 s, and 100 s, are used in evaluating the said schemes. Each experiment runs over 3600 s. 5 measurements are bundled for all experiments. We show the measurement time estimation errors of AHTS and EE-ASCFR in Fig. 3.6. Table 3.1 summarizes the mean absolute error (MAE) and the mean squared error (MSE) of the measurement time estimation.

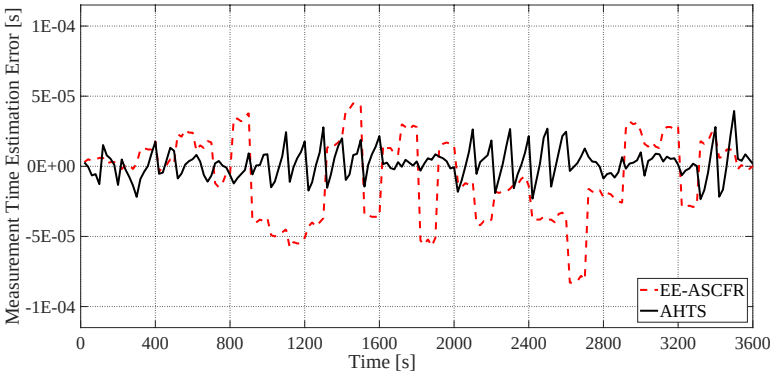
Fig. 3.6 demonstrates that the measurement time estimation errors of AHTS are stable and smaller than those of EE-ASCFR; AHTS performs consistently in all experiments employing three different SIs over their observation periods. Table 3.1 exhibits the performance of AHTS in terms of MAE and MSE, where the MAEs and MSEs



(a)



(b)



(c)

Figure 3.6: Measurement time estimation errors of EE-ASCFR and AHTS with SI of (a) 1 s, (b) 10 s and (c) 100 s for the single-hop scenario [3].

of AHTS are again much smaller than that of EE-ASCFR. The ensemble of Fig. 3.6 and Table 3.1 illustrates that the proposed AHTS could address the issue of precision

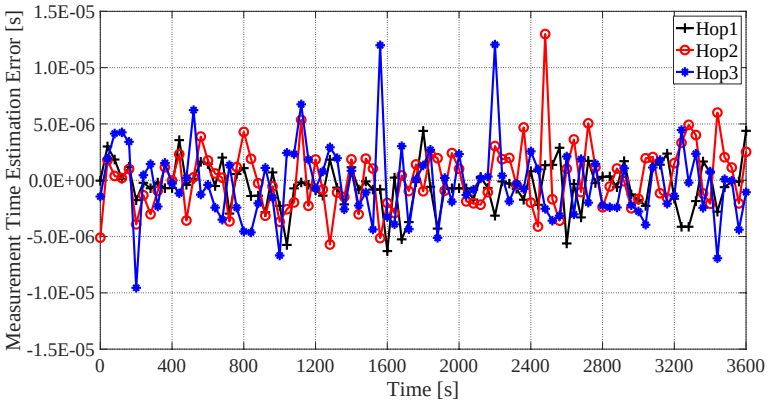


Figure 3.7: Measurement time estimation errors of AHTS with SI of 1 s for the multi-hop scenario [3].

loss and achieve microsecond-level time synchronization at resource-constrained sensor nodes in the single-hop scenario.

Note that, the value of SI affects more the AHTS, especially when SI increases to 100 s. This may because that the cheap crystal oscillator employed in the sensor nodes drifts more in a longer period of time. In contrast, the performance of EE-ASCFR is not much affected by the variation of SI. The reason is that the aforementioned precision loss issue eclipses the impact of SI change.

3.4.2 Multi-Hop Scenario

We evaluate the performance of AHTS in the multi-hop scenario by establishing a flat WSN composing of one head and three sensor nodes. Similar to the evaluation in the single-hop scenario, here our experiments run over the same period of 3600 s and the value of SI is fixed to 1 s. Each sensor node bundles only its own measurement and synchronization data, and the maximum bundling number is set to 5. The measurement time estimation errors of AHTS for three hops are exhibited in Fig. 3.7; their MAEs and MSEs are summarized in Table 3.2.

In Table 3.2, the MAE of measurement time estimation for hop 1 is 2.1774 µs, similar to the evaluated 1.8166 µs in the single-hop scenario. However, the MAEs of hops 2 and 3 become slightly larger as the hop count increases—about 0.2 µs per hop, which is

Table 3.2: MAE and MSE of Measurement Time Estimation of AHTS for the Multi-Hop Scenario [3]

Hop Number		MAE [1]	MSE [1]
Hop	3	2.5700E-06	1.2432E-11
	2	2.3648E-06	1.1056E-11
	1	2.1774E-06	9.4104E-12

[1] The samples measured between 360 s (i.e., a tenth of the observation period) and 3600 s are employed to avoid the effect of a transient period.

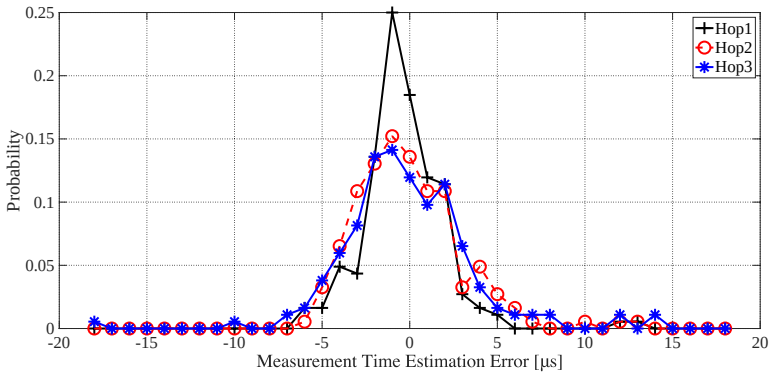


Figure 3.8: Probability distribution of the measurement time estimation errors of AHTS with SI of 1 s for the multi-hop scenario [3].

mainly due to the layer-by-layer time translation of the multi-hop extension. Likewise, in Fig. 3.7, more fluctuations could be found for hop 2 and hop 3 than hop 1. Similar behaviors could also be found in Fig. 3.8 that illustrates the probability distribution of the measurement time estimation errors of AHTS for three hops.

Specifically, it shows that 18% of the measurement time estimation errors of hop 1 are close to zero; and the percentages of hop 2 and hop 3 are decreased to around 14% and 13%, nevertheless. Moreover, most measurement time estimation errors of all three hops are within the range of −10 µs and 10 µs, where the performance of AHTS under the multi-hop scenario is clearly demonstrated. Overall, AHTS could provide microsecond-level time synchronization accuracy for multi-hop scenarios evaluated.

3.5 Summary

In this chapter, through investigating the actual performance of EE-ASCFR that builds on the reverse two-way message exchange and could simultaneously compensate for the propagation delay and clock skew, we reveal the impact of precision loss issue on the WSN time synchronization. We emphasize that time synchronization schemes should attach importance to their computational complexity for resource-constrained WSNs. To avoid the precision loss issue, we propose asymmetric high-precision time synchronization AHTS remolded from EE-ASCFR, where all synchronization-related computations are done at the head. Therefore, the only procedure related to time synchronization at the sensor node is timestamping, whereby the computational complexity is significantly reduced. Experimental results on a real WSN testbed demonstrate the performance of AHTS, i.e., microsecond-level synchronization accuracy in both single- and multi-hop scenarios.

As the foundation of AHTS, i.e., EE-ASCFR, is designed as well for energy-efficiency in the asymmetric WSNs, we will investigate the energy-efficiency issue of WSN time synchronization in the follow-up chapter. Note that, the design assumption that an asymmetric WSN comprises a resourceful head represents not only most WSN applications but also future IoT deployments [53].

Chapter 4

A Beaconless Asymmetric Time Synchronization Scheme for Low Energy Consumption

In this chapter, we focus on the *energy-efficiency* of the WSN time synchronization; the work presented here is based on our publication of [1]. Noting the massive number of battery-powered low-cost sensor nodes are being deployed for various applications from monitoring, detection to device-free wireless sensing; time synchronization that is essential to those applications but heavily relies on energy-consuming message transmissions, therefore, has to develop towards more energy-efficient. We introduce in this chapter the reverse asymmetric time synchronization framework for resource-constrained multi-hop WSNs. Based on this framework, we propose an energy-efficient beaconless asymmetric time synchronization scheme employing the reverse one-way message dissemination. On a real WSN testbed consisting of TelosB motes running TinyOS, we experimentally demonstrate that the proposed scheme could conserve significant energy consumption compared to the flooding time synchronization protocol while still achieving microsecond-level synchronization accuracy.

4.1 Introduction

Time synchronization for wireless sensor networks (WSNs) has been extensively studied in the last decades as the number of WSN deployments has been gradually increasing over the period [14, 13]. Because most of the WSN deployments are based on a large number of battery-powered, low-cost sensor nodes, which are limited in their computing and power resources, the focus of WSN time synchronization research has been shifted toward three major aspects of *accuracy*, *energy consumption*, and *computational complexity*.

In the literature, many conventional [18, 28, 16] and recent [30, 54, 22] time synchronization schemes have been proposed to address the aspect of synchronization accuracy due to its significance. Over the past few years, a few schemes [55, 31, 40, 56] tried to address the issues of energy consumption and computational complexity in time synchronization together with its accuracy. As the IoT thrives, many research efforts have been concentrated on conserving the energy of IoT systems, e.g., the conventional parameter adaptation [57, 58, 59] and the novel energy harvesting schemes [60]; in such an environment, the time synchronization schemes—e.g., [40, 31, 41, 61]—have been focusing on energy efficiency along with other requirements such as high synchronization accuracy and low computational complexity. In general, however, the attention received for the aspects of energy consumption and computational complexity is relatively less compared to that for the time synchronization accuracy. Especially in multi-hop WSNs, intermediate gateway nodes are overloaded with tasks for not only relaying messages but also a variety of computations for their offspring nodes as well as themselves. Therefore, not only minimizing the energy consumption but also lowering the computational complexity while maintaining the synchronization accuracy is crucial to the design of time synchronization schemes for resource-constrained sensor nodes.

In [2], unlike many existing WSN time synchronization schemes like FTSP [18] and TPSN [16], we put our focus on *asymmetric WSNs*—where a head is equipped with a powerful processor and supplied power from outlet and sensor nodes measuring data

and/or detecting events with sensors are limited in processing and battery-powered—and proposed a novel energy-efficient time synchronization scheme based on the reverse two-way message exchange and demonstrated through simulation experiments that sub-microsecond-level synchronization accuracy could be achieved.

Note that the computational precision required by the scheme proposed in [2] (i.e., the precise division of the floating-point numbers) is beyond the capability of most resource-constrained sensor nodes equipped with a low-cost MicroController Unit (MCU) providing only 32-bit floating-point as discovered in [3]. Also, its multi-hop extension through intermediate gateway nodes was discussed, but its performance under multi-hop scenarios was not analyzed at all. The idea of the reverse two-way message exchange together with message bundling for synchronization and measurement data, however, could be applied to the design of more advanced energy-efficient time synchronization schemes targeting resource-constrained sensor nodes, which could greatly reduce the number of message transmissions by sensor nodes.

Based on our prior work, therefore, in this chapter we present the reverse asymmetric time synchronization framework and propose the beaconless asymmetric energy-efficient time synchronization scheme BATS specifically based on the reverse one-way message dissemination, which can address in a more balanced way the three major challenges in WSN time synchronization on resource-constrained sensor nodes—i.e., achieving high synchronization accuracy, reducing energy consumption [31], and lowering computational complexity at sensor nodes [62]—as follows:

First, in the proposed BATS, all synchronization procedures but timestamping are moved from sensor nodes, including gateway nodes, to the head as in [3] in order to improve the accuracy of time synchronization by addressing the issue of precision loss resulting from the use of 32-bit single-precision floating-point numbers at sensor nodes: Because all procedures but timestamping are carried out at the head with plenty of computing and power resources including 64-bit floating-point precision, numerical computational errors due to precision loss could be avoided.

Second, the movement of synchronization procedures to the head greatly reduces the

computational complexity of sensor nodes as well. In the proposed scheme, sensor nodes are responsible for only the timestamping procedure whose computational complexity is $O(1)$; the computational complexity for sensor nodes under the conventional schemes using linear regression (e.g., FTSP [18]), for example, could be as high as $O(CG^2)$ in case the least squares method is used for G training samples and C variables [63].

Third, noting that the propagation delay compensation through the two-way message exchange is not required for microsecond-level synchronization [18], we also design BATS based on the reverse one-way message dissemination to lower the energy consumption of battery-powered sensor nodes; due to the one-way message dissemination under the reverse asymmetric time synchronization framework, BATS does not rely on the "Beacon/Request" messages and therefore saves the energy for their receptions and transmissions at sensor nodes, the former of which often consume more energy than the latter [25]. As in [2], application messages (e.g., for reporting measurement data to the head) can also embed and carry synchronization-related data, which further reduces the number of message transmissions.

This *beaconless* time synchronization of BATS is a major advantage compared to many existing time synchronization schemes relying on sensor nodes' broadcasting synchronization messages received from a root node (i.e., *flooding*) to achieve network-level time synchronization (e.g., [18, 64]), because it eliminates beacon-related computation and energy consumption imposed on the resource-constrained sensor nodes which are already loaded with tasks including medium access control (MAC) protocol, message scheduling and routing, and data measurement. Note that the actual energy consumption and synchronization accuracy of the proposed scheme are evaluated and analyzed through experiments on a real WSN testbed.

The rest of this chapter is organized as follows: In Section 4.2, the proposed BATS based on the reverse one-way message dissemination under the reverse asymmetric time synchronization framework are presented. Section 4.3 illustrates our multi-hop extension of BATS with discussing the issue of communication overhead. Section 4.4 exhibits the practical evaluation results on a real WSN testbed with a focus on energy consumption

and synchronization accuracy. Section 4.5 concludes the work of this chapter.

4.2 Energy-Efficient Time Synchronization Tailored for Resource-Constrained Sensor Nodes

When the propagation delay is not significant (e.g., sub-microsecond delays for WSNs with a communication range of 300 m or less), time synchronization schemes based on the one-way message dissemination have a clear advantage over those based on the two-way message exchange in terms of the number of message transmissions at sensor nodes. Still, the schemes based on the one-way message dissemination have issues in their implementation on resource-constrained sensor nodes, we will discuss in the following. Afterwards, we introduce BATS—i.e., the energy-efficient time synchronization scheme based on the reverse one-way message dissemination—which addresses the implementation issues of the one-way-message-dissemination-based time synchronization schemes on resource-constrained sensor nodes.

4.2.1 Impact of Precision Loss on the Performance of One-Way-Message-Dissemination-Based Time Synchronization Schemes

We take RSP [48] as an example—which represents the flooding time synchronization, however, has simplified estimations of its clock skew and offset—to analyze the impact of the precision loss on the performance of the time synchronization schemes based on one-way message dissemination.

As the conventional one-way message dissemination illustrated in Fig. 1.2 (a) is employed in RSP, the relationship between its head's hardware clock t and its sensor node i's hardware clock $T_i(t)$ can be modeled as follows:

$$t = R_i T_i(t) + \theta_i, \tag{4.1}$$

where R_i denotes the clock frequency ratio and θ_i the clock offset of the reference clock

with respect to the hardware clock of the sensor node. During the broadcastings of the Beacon/Request messages, the sensor node could collect a set of timestamp pairs, i.e., $T1_k$ and $T2_k$ (k=1, ...). Based on those timestamps, the sensor node i, therefore, could maintain its logical clock $\mathscr{T}_i^{\mathrm{RSP}}$ which is the estimated reference time with reference to the sensor node i's hardware clock $T_i(t)$ as follows [48]:

$$\mathscr{T}_i^{\mathrm{RSP}}\Big(T_i(t)\Big) = \hat{R}_{i,k}T_i(t) + \hat{\theta}_{i,k}, \tag{4.2}$$

where $\hat{R}_{i,k}$ and $\hat{\theta}_{i,k}$ represent the estimated clock frequency ratio and clock offset based on the linear interpolation employing a set of timestamps, i.e.,

$$\hat{R}_{i,k} = \frac{T1_k - T1_{k-1}}{T2_k - T2_{k-1}}, \tag{4.3}$$

$$\hat{\theta}_{i,k} = \frac{T1_{k-1}\cdot T2_k - T2_{k-1}\cdot T1_k}{T2_k - T2_{k-1}}. \tag{4.4}$$

In this case, the impact of the precision loss on the estimated reference clock of RSP in (4.2) can be analyzed similarly to the investigation presented in [3]: We define ε_R and ε_θ as the precision loss in the estimation of the clock frequency ratio and clock offset respectively:

$$\varepsilon_R \triangleq \hat{R}_{i,k} - \hat{R}_{i,k}^{\mathrm{LP}}, \tag{4.5}$$

$$\varepsilon_\theta \triangleq \hat{\theta}_{i,k} - \hat{\theta}_{i,k}^{\mathrm{LP}}, \tag{4.6}$$

where $\hat{R}_{i,k}^{\mathrm{LP}}$ and $\hat{\theta}_{i,k}^{\mathrm{LP}}$ are the value of the clock frequency ratio and the clock offset from the practical implementation that are affected by the limited-precision floating-point arithmetic in (4.3) and (4.4). Thus, we can derive the computational error $\Psi(t)$ in the estimated reference clock at the sensor node at time t as follows:

$$\begin{aligned}\Psi(t) &= \Big(\hat{R}_{i,k}T_i(t) + \hat{\theta}_{i,k}\Big) - \Big(\hat{R}_{i,k}^{\mathrm{LP}}T_i(t) + \hat{\theta}_{i,k}^{\mathrm{LP}}\Big),\\ &= \varepsilon_R T_i(t) + \varepsilon_\theta.\end{aligned} \tag{4.7}$$

(4.7) shows that two components exist in the computational error, the first of which is proportional to the current time of the sensor node's hardware clock (i.e., $T_i(t)$), however.

We further trace the error based on the IEEE standard for floating-point arithmetic [49]. In that standard, a non-zero floating-point number x can be represented in the binary format as follows:

$$x = \sigma \cdot \bar{b} \cdot 2^e, \tag{4.8}$$

where σ is the *sign* taking the value of +1 or −1, $\bar{b}$ is the *binary fraction* whose value is within the range of $[1, 2)$, and e is the *integer exponent*.

Because the IEEE 32-bit single-precision floating-point format assigns 1 bit to σ, 8 bits to e and 23 bits to $\bar{b}$, the *machine epsilon* [65] becomes 2^{-23}. If the rounding arithmetic is chopping (i.e., rounding towards 0), the precision loss ε_R is within the range of $[-2^{-23}, 0]$, and the maximum absolute precision loss in this case is $2^{-23} \approx 1.19 \times 10^{-7}$. This implies that the first component of (4.7) alone could result in about 0.1 µs and 1 µs computational errors for $T_i(t)$ of 1 s and 10 s, respectively, in the worst case.

Note that the analysis of the computational error above is based on the worst-case scenario for simplicity. As discussed in [48], however, in reality, the precision loss ε_R itself is inversely proportional to SI (i.e., the time difference between two consecutive beacon messages), which could more or less relax the dependency of the computational error $\Psi(t)$ on $T_i(t)$. In fact, setting an optimal value of SI is quite complicated because the value of SI not only affects the computational error but also determines the impact of the sensor node's hardware clock drift due to the changes in the ambient temperature and the battery voltage.

4.2.2 Beaconless Asymmetric Energy-Efficient Time Synchronization

As illustrated in Fig. 4.1, the proposed BATS scheme based on the reverse one-way message dissemination does not rely on beacon messages but utilizes only measurement data messages to carry the reversed synchronization timestamp $T1$, which results in the

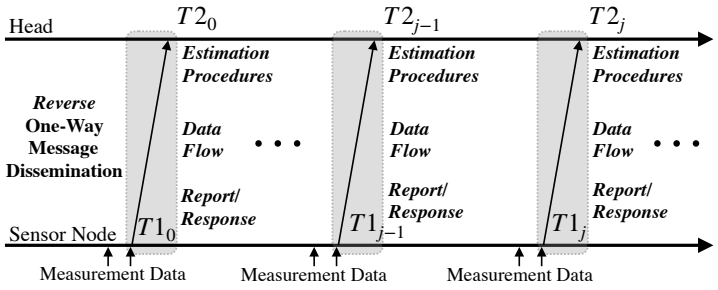


Figure 4.1: Asymmetric energy-efficient time synchronization based on reverse one-way message dissemination with optional bundling of measurement data [1].

movement of all time synchronization procedures from sensor nodes to the head except timestamping of $T1$. As shown in Fig. 4.2, the timestamping of $T1$ and $T2$ is triggered by an interrupt generated by the radio chip immediately after the Start Frame Delimiter (SFD) byte has been sent and received by the MAC layer of the sender and the receiver, respectively, which is similar to that of RTSP [17]. This timestamping approach, which is based on a pair of timestamps generated during the transmission and reception of one message, is simpler and faster than that of FTSP [18] which reduces the jitter of interrupt handling through recording, normalizing and averaging multiple timestamps at both the sender and the receiver.

In the single-hop scenario shown in Fig. 4.1, the head's direct offspring nodes employ their measurement data messages to transmit their timestamps $T1_j$ during the j-th synchronization to the head for maintaining the time synchronization. Using the abundant computing and power resources including the 64-bit floating-point precision at the head, the numerical computational errors analyzed in Section 4.2.1 could be avoided in the proposed scheme. For modeling the linear relationship between the hardware clocks of the head and the sensor node, the first-order affine clock model (1.1) is employed; the affine clock model is widely used in many existing WSN time synchronization schemes as a clock model for a relatively short time period, during which the effect of environmental conditions (e.g., temperature) and voltage variation on the clock oscillator hardly changes.

In this regard, the use of the linear regression for clock parameter estimation does

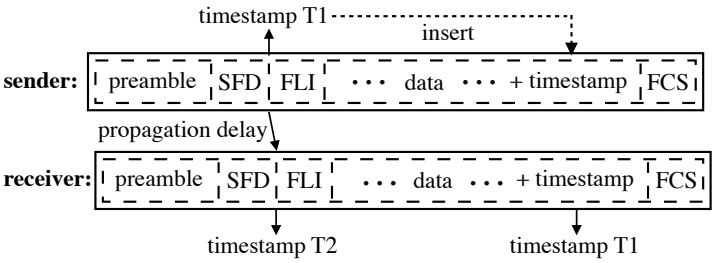


Figure 4.2: MAC-layer timestamping adopted in BATS [1].

make sense in the context of WSNs due to its simplicity, and, in fact, most representative WSN time synchronization schemes, including FTSP and RBS, rely on this method. For a fair comparison with those schemes, we also employ the linear regression method in the proposed scheme to estimate the parameters of sensor nodes' hardware clocks. Note that the performance of the linear regression in the proposed scheme, which is running at the head, is not constrained by the limited sample size unlike FTSP using the past 8 samples only due to the resource constraints of sensor nodes.

The following linear least squares method is employed in BATS to model the linear relationship between the hardware clocks of the sensor node i and the head based on m samples during the j-th synchronization: For $i \in [0, 1, \ldots, N-1]$ and $j = m, m+1, \ldots$,

$$\mathbf{\Phi}_i(j) = \left\{\mathbf{T2}_i(j)^\top \mathbf{T2}_i(j)\right\}^{-1} \mathbf{T2}_i(j)^\top \mathbf{T1}_i(j), \tag{4.9}$$

where

$$\begin{aligned} \mathbf{\Phi}_i(j) &= [1 + \hat{\epsilon}_i(j), \hat{\theta}_i(j)], \\ \mathbf{T1}_i(j) &= [T1_i(j - m + 1), ..., T1_i(j)], \\ \mathbf{T2}_i(j) &= [T2_i(j - m + 1), ..., T2_i(j)], \end{aligned}$$

where $\mathbf{\Phi}_i(j)$ is a vector of the estimated clock frequency ratio and offset and $(\cdot)^\top$ and $(\cdot)^{-1}$ denote vector transpose and matrix inverse, respectively.

As we will see in Section 4.4, employing the complex linear regression solution with no limitation on the sample size for solving the linear equation of the time synchroniza-

tion could improve the performance of time synchronization compared with the conventional simpler solutions such as [48]. Furthermore, thanks to the reverse asymmetric framework that BATS is based on, we can use a more complex estimation technique with higher computational complexity (e.g., machine learning [66]) as all the estimation procedures now run at the head with abundant computing capability instead of the resource-constrained sensor nodes.

As exhibited in Fig. 4.1, the direction of the conventional one-way message dissemination is reversed, where the estimation of the hardware clocks between the sensor node and the head is also reversed compared to the conventional one. On the one hand, when timestamps along with measurement data from sensor nodes are gathered in the head, the corresponding hardware clock timestamps of the sensor nodes should be translated based on the head's reference clock to estimate the exact time for the measurement events. On the other hand, when the head issues commands to notify sensor nodes to perform collaborative operations (e.g., sleep and wake-up for energy-efficient MAC protocols and application of coherent sampling), the timestamps carried in the commands have to be translated to the target sensor nodes' hardware clock time. Using the estimated frequency ratio $1 + \epsilon_i$ and offset θ_i between a sensor node i and the head, in the single-hop scenario, the translation between their hardware clock times is done as follows:

$$t = \frac{T_i(t) - \theta_i}{1 + \epsilon_i} \Longleftrightarrow T_i(t) = (1 + \epsilon_i)t + \theta_i, \tag{4.10}$$

where the translation from sensor node's clock time $T_i(t)$ to head's clock time t involved the floating-point division, which is similar to the clock time translation in EE-ASCFR. As mentioned in [29], such calculation involved with floating-point division and microsecond-level timestamp should be carefully handled in the resource-constrained sensor nodes, but those are not major issues at all in BATS.

4.3 Extension of BATS to Multi-Hop WSNs

Independent of the network topologies which are typically established through routing protocols (e.g., Collection Tree Protocol (CTP) [67]), the proposed BATS scheme in principle could be extended to any existing multi-hop routing protocols with the addition of the required timestamps in the application messages. There are several issues to consider, however, in its extension to the multi-hop scenario.

In the multi-hop scenario, gateway sensor nodes located in intermediate layers process not only their own data as regular sensor nodes but also the data from their offspring sensor nodes. The presence of gateway nodes makes it complicated for the head to directly handle all timestamps required for time synchronization in multi-hop WSNs. Note that in practice, the role of a sensor node is not fixed but can be changed to either a gateway node or a leaf node—i.e., a node without relaying packets from other nodes—depending on the topology and the arrangement by the routing protocol.

4.3.1 Multi-Hop Extension of The Time Synchronization Based On Reverse One-Way Message Dissemination

As described in [2], two possible approaches for the multi-hop extension of WSN time synchronization schemes based on the reverse two-way message exchange (including EE-ASCFR) are those of *time-relaying* and *time-translation* at the gateway nodes. Of the two approaches, the time-relaying one could introduce more random delays, including queueing delays, as the messages from the sensor nodes are being forwarded to the head through multiple gateway nodes, which are not properly compensated for unlike the time-translation one. To maximize the advantage of MAC-layer timestamping and avoid random queueing delays cumulated through multiple gateway nodes, therefore, we use *per-hop time synchronization* for the multi-hop extension of BATS, which is similar to the aforementioned time-translation approach in which the hardware clock time of a sensor node goes through layer-by-layer translation in order to estimate its time with respect to the reference clock of the head. The fundamental difference of

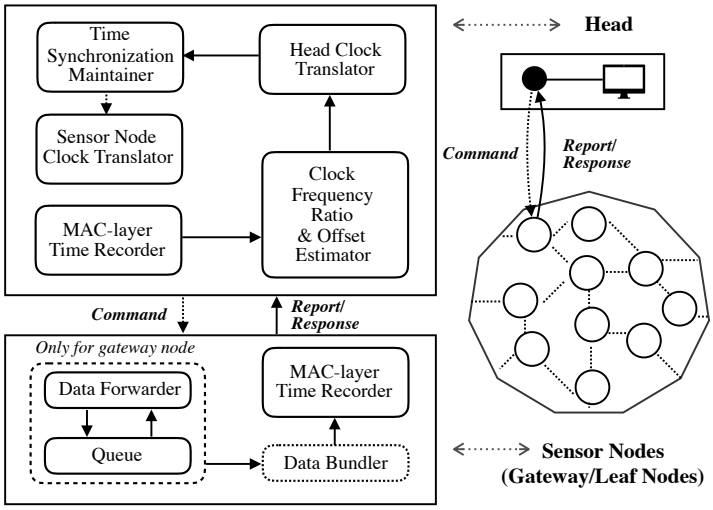


Figure 4.3: A system architecture of the proposed time synchronization scheme [1].

the per-hop time synchronization employed in BATS compared to the time-translation approach, however, is that all layer-by-layer translations are again moved from the gateway nodes to the head to relieve the burden of the gateway nodes under the original time-translation approach. The system architecture is illustrated in Fig. 4.3, where the "command" is optional and the "sensor nodes" refer to both gateway and leaf sensor nodes.

In the following, we focus on one end-to-end path shown in Fig. 4.4 for ease of description and notation simplicity; in this case, the index of a node becomes its layer index as well, with 0 indicating both the head and layer 0 it belongs to.

As in (1.1), we still use the affine clock model to describe the relationship between the hardware clocks of two neighbor nodes: For $i=1, 2, \ldots, N$,

$$T_i(t) = \left(1 + \epsilon_{i,i-1}\right) T_{i-1}(t) + \theta_{i,i-1}, \tag{4.11}$$

where $\left(1+\epsilon_{i,i-1}\right) \in \mathbb{R}_+$ and $\theta_{i,i-1} \in \mathbb{R}$ are the clock frequency ratio and the clock offset of $T_i(t)$ with respect to $T_{i-1}(t)$, respectively, and $T_0(t)=t$. The clock frequency ratio and offset could be estimated by applying the linear least squares method of (4.9) to (4.11). We further use a recursive equation to translate the hardware clock time to the reference

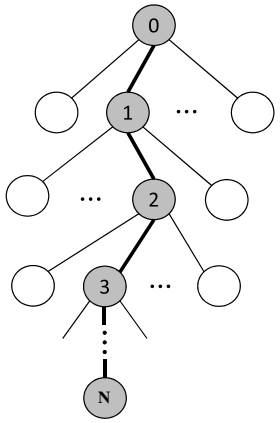


Figure 4.4: N-hop end-to-end path in a multi-hop WSN [1].

clock time: For i=$N, N-1 \ldots, 1$,

$$T_{i-1}(t) = \frac{T_i(t) - \theta_{i,i-1}}{1 + \epsilon_{i,i-1}}. \tag{4.12}$$

Note that, in the proposed per-hop time synchronization, timestamps required for the synchronization of a sensor node based on the reverse one-way message dissemination are recorded at the sensor node and its gateway node (working as a reference) as shown in Fig. 4.1. Unlike the time translation approach described in [2], however, the time translation operation is moved from the gateway to the head, and the gateway node just sends the pair of timestamps obtained from the reverse one-way message dissemination (i.e., $T1_i$ and $T2_{i-1}$) all the way to the head. Based on these pairs of timestamps from gateway nodes, the head can establish the relationships between sensor nodes and their gateway nodes and eventually translate sensor nodes hardware clock times to those based on the reference clock at the head recursively based on (4.11) and (4.12).

Using the aforementioned per-hop time synchronization, BATS could be extended to cover the multi-hop scenario as demonstrated in Fig. 4.5 (b), which is different from the conventional multi-hop scheme based on one-way message dissemination shown in Fig. 4.5 (a) where the synchronization timestamps are carried by standalone beacon messages and the measurement data are transported by the standalone measurement

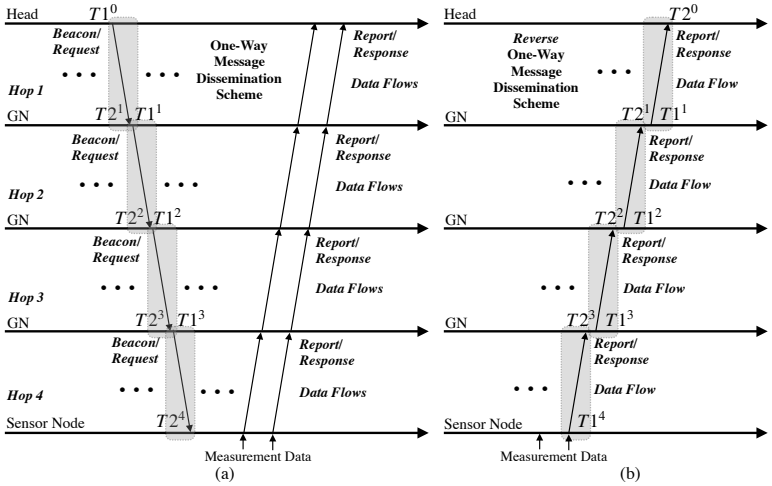


Figure 4.5: Multi-hop extension of (a) a conventional (e.g., FTSP) and (b) a reverse asymmetric time synchronization scheme based on the one-way message dissemination with all-data bundling procedure [1].

messages. The proposed BATS does not rely on broadcasting beacon messages and embeds the synchronization timestamps into the measurement messages as in [2]. In case of the conventional schemes, there are two separate flows of messages with different directions (i.e., one for synchronization and the other for measurement); we cannot embed synchronization timestamps into the measurement messages as in the extended BATS. Pseudocodes for the details of the event handling at a sensor node under BATS with and without bundling are provided in Algorithm 1 and 2.

4.3.2 Comparison to Other Multi-Hop WSN Time Synchronization Schemes

Having discussed the multi-hop extension of BATS, which does not rely on beacon messages but exploits the bundling of synchronization timestamps with the measurements, we compare the extended BATS to other multi-hop WSN time synchronization schemes.

Since the synchronization timestamps are bundled with measurement data in the same message, the major cost of the proposed scheme is the increase of the payload of the measurement message by the synchronization timestamps $T1$ and $T2$. As shown in

```
Data: The node maintains the following data and variables:
    • e: Event object including a timestamp;
    • node_status: Variable indicating the status of node (i.e., GATEWAY or LEAF);
    • p: Packet object (optionally) including timestamps from MAC-layer timestamping;
    • Q_M: FIFO queue for measurement data;
    • Q_P: FIFO queue for packets;
    • Q_T2: FIFO queue for timestamp T2.
1  On detecting an event e:
2  switch e.type do
3      case MEASUREMENT do // its own measurement
4          d ← Q_M.dequeue()                          // measurement data from the queue
5          ts ← e.getTimestamp()                      // for measurement, not for synchronization
6          p ← Packet(d, ts)                          // create a packet object
7          send(p, MAC-LAYER_TIMESTAMPING = ON)        // for T1
8      case PACKET do
9          if p.getDestAddress() ≠ HEAD then // packet received from other sensor nodes
10             if node_status == GATEWAY then
11                 p ← Q_P.dequeue()                  // packet from the queue
12                 if p.getT2() == NULL then // direct offspring
13                     T2 ← Q_T2.dequeue()            // from mac-layer timestamping
14                     p.setT2(T2)
15                 send(p, MAC-LAYER_TIMESTAMPING = OFF)  // no need for another T1
16         else
               // Process the packet from the head ...
17     otherwise do
           // Process other event ...
```

Algorithm 1: Event handling at a sensor node under BATS without bundling [1].

Fig. 4.5 (b), for a node j hops away from the head, 2 timestamps—i.e., $T1^j$ and $T2^{j-1}$—are required for the head to achieve network-wide time synchronization. Therefore, for a flat N-hop network, $2N-1$ timestamps[1] are to be transmitted through the measurement messages to the head. Note that, though the same pair of timestamps (i.e., $T1$ and $T2$) are used in the conventional schemes based on one-way message dissemination such as FTSP, the cost of the transmissions of the standalone synchronization messages would be much higher; even considering that the timestamp $T2$ could be kept locally in the sensor node to perform the synchronization procedure, for a flat N-hop network, N synchronization messages carrying the timestamp $T1$ have to be broadcasted for achieving network-wide time synchronization.

[1]Because $T2^0$ is recorded at the head, there is no need of transmission.

Data: The node maintains the following data and variables:
- e: Event object including a timestamp;
- *node_status*: Variable indicating the status of node (i.e., GATEWAY or LEAF);
- p: Packet object (optionally) including timestamps from MAC-layer timestamping;
- b: Bundling counter for counting the buffered bundles.
- bf: Bundling buffer for carrying packets;
- Q_M: FIFO queue for measurement data;
- Q_P: FIFO queue for packets;
- Q_{T2}: FIFO queue for timestamp $T2$.

```
1  On detecting an event e:
2  switch e.type do
3      case MEASUREMENT do // its own measurement
4          d ← Q_M.dequeue()                          // measurement data from the queue
5          ts ← e.getTimestamp()                      // for measurement, not for synchronization
6          bf ← bf + Packet(d, ts)                    // kick packet into bundling buffer
7          b ← b + 1                                  // update the bundling number
8      case PACKET do
9          if p.getDestAddress() ≠ HEAD then // packet received from other sensor nodes
10             if node_status == GATEWAY then
11                 p ← Q_P.dequeue()                  // packet from the queue
12                 if p.getT2() == NULL then // direct offspring
13                     T2 ← Q_T2.dequeue()            // from mac-layer timestamping
14                     p.setT2(T2)
15                 if SELF_DATA_BUNDLING == TRUE then // in case of self-data
                     bundling
16                     send(p, MAC-LAYER_TIMESTAMPING = OFF)  // no need for another
                         T1
17                 else // in case of all-data bundling
18                     bf ← bf + p                    // kick packet into bundling buffer
19                     b ← b + p.bundles              // update the bundling number
20         else
               // Process the packet from the head ...
21     otherwise do
           // Process other event ...
22 if b >= MAX_BUNDLING then // number of bundled data equals or exceeds maximum
     bundling number
23     send(bf, MAC-LAYER_TIMESTAMPING = ON)          // for T1
24     bf ← NULL                                      // clear bundling buffer
```

Algorithm 2: Event handling at a sensor node under BATS with bundling [1].

Using the self-data and all-data bundling procedures detailed in Algorithm 1 and 2 with the payload and data structure of a message shown in Fig. 4.6, for a flat N-hop network with each node generating M measurements, the number of message transmissions and receptions for conventional ($\mathbf{N}_{msg}^{conv}$) and proposed ($\mathbf{N}_{msg}^{prop}$) scheme can be

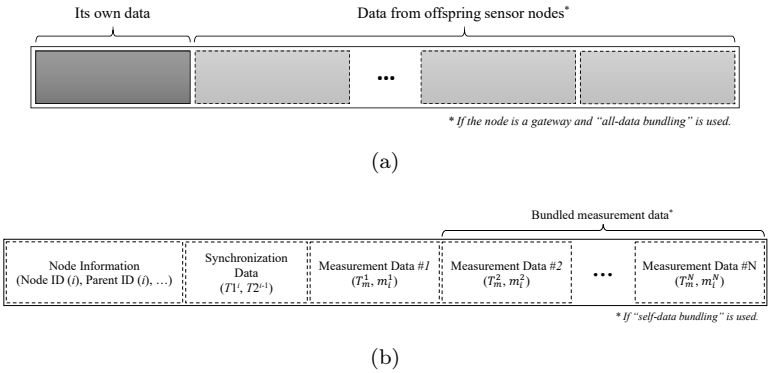


Figure 4.6: Payload and data structure of a message generated at sensor node i: (a) Payload with optional "all-data bundling" and (b) data structure with optional "self-data bundling" [1].

obtained as follows:

$$\mathbf{N}_{msg}^{conv} = 2(N-1) + 1 + M \sum_{i=1}^{N} \Big(2(i-1)+1\Big), \tag{4.13}$$

$$\mathbf{N}_{msg}^{prop} = \begin{cases} \sum_{i=1}^{N} (2(i-1)+1), & \text{self-data bundling,} \\ 2(N-1)+1, & \text{all-data bundling.} \end{cases} \tag{4.14}$$

Note that $\mathbf{N}_{msg}^{conv}$ includes not only synchronization messages but also measurement messages, while $\mathbf{N}_{msg}^{prop}$ just includes measurement messages with synchronization timestamps embedded. For a flat 4-hop network generating 2 measurements per node, the number of message transmissions and receptions for the conventional scheme and the proposed scheme with self-data and all-data bundling procedures are 39, 16 and 7, respectively. These results suggest that the proposed scheme with all-data bundling could reduce more than 80% message transmissions and receptions compared to the conventional one. Consequently, the proposed scheme could achieve much higher energy efficiency.

Compared to the multi-hop extension of the time synchronization scheme based on reverse two-way message exchange proposed in [2], the multi-hop extension of BATS

based on one-way message dissemination can greatly lower communication overheads by reducing the number of synchronization messages required for network-wide synchronization, which is a significant advantage when used for large-scale WSNs. If the end-to-end communication range of a multi-hop WSN is over a kilometer, however, the propagation delay cannot be ignored any longer, and the time synchronization schemes based on reverse two-way messages exchange would be a better option in such a case.

4.3.3 Discussions

Though the proposed scheme could save the energy consumption by reducing the number of message transmissions and receptions and significantly lower the computational complexity of sensor nodes, those advantages come at the expense of the increase in the computational complexity of the head. The increase in the computational complexity of the head, however, is intentional because, as discussed in Section 4.1, the head in an asymmetric WSN is assumed to have plenty of computing and power resources. Note that the computation offloading for sensor nodes in the proposed framework could be extended to the head as well by exploiting mobile edge and fog computing as discussed in [68].

One major issue encountered in large-scale deployments is the additional message transmissions in bundling: Because of the reassignment of the time synchronization procedure from sensor nodes to the head, timestamps for time synchronization have to be delivered to the head. As the number of hops and sensor nodes increases, the number of timestamps to be delivered to the head also increases; the increasing number of timestamps, by the way, does not directly result in more message transmissions thanks to the proposed bundling procedure. Because the maximum number of bundled data is limited by the maximum payload size of a message, however, a gateway node has to generate additional messages when the measurement and/or synchronization data either generated by themselves or received from its offspring nodes cannot be bundled in one message, which reduces the energy efficiency of the proposed time synchronization scheme in large-scale deployments.

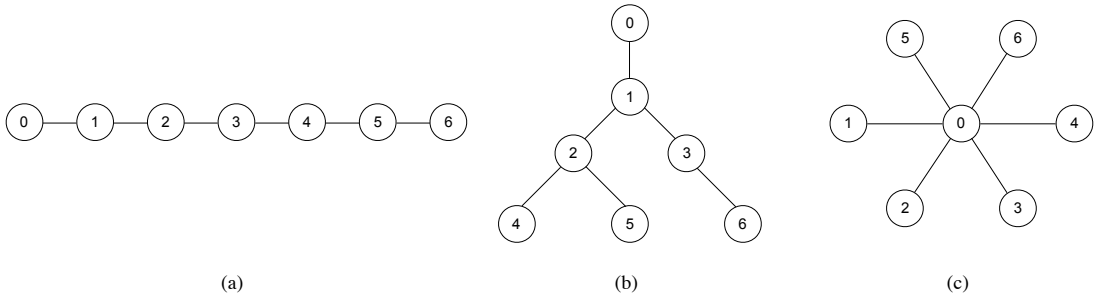


Figure 4.7: Multi-hop and single-hop topologies with one head and six sensor nodes: (a) Multi-hop chain topology, (b) multi-hop tree topology and (c) single-hop star topology [1].

Due to the use of computation offloading, BATS also faces the well-known trade-off between energy consumption and execution delay [68]: Bundling a large number of data would increase the end-to-end delay while saving the energy by reducing the number of message transmissions. In this regard, we have proposed an optimal bundling scheme to address this trade-off [6].

4.4 Experimental Results

The proposed BATS scheme is implemented on a real WSN testbed consisting of one head and six sensor nodes as exhibited in Fig. 4.7, all of which are based on TelosB motes running TinyOS [44]. Note that the timer resolution of TelosB motes running TinyOS is 1 µs since the resolutions of its hardware clock running on a 32-kHz crystal oscillator and software timer are 30.5 µs and 1 µs, respectively. The accuracy of time synchronization schemes tested on the testbed, therefore, is limited to a microsecond level.

4.4.1 Energy Efficiency

To evaluate the energy efficiency of the proposed scheme, we adopt both indirect and direct methods, namely, counting the number of message transmissions and receptions and measuring the energy consumption using the equipment.

Table 4.1: The Numbers of Message Transmissions and Receptions at Sensor Node for Different SIs During The Period of 3600 s [1]

Type of Synchronization Scheme		$\mathbf{N}_{\mathrm{TX}}$[1]	$\mathbf{N}_{\mathrm{RX}}$
Conventional Two-way	SI = 100 s	136	36
	SI = 10 s	460	360
	SI = 1 s	3700	3600
Conventional One-way	SI = 100 s	100	36
	SI = 10 s	100	360
	SI = 1 s	100	3600
Reverse Two-way	SI = 100 s	100	36
	SI = 10 s	100	360
	SI = 1 s	100	3600
Reverse One-way	SI = 100 s	100	0
	SI = 10 s	100	0
	SI = 1 s	100	0

[1] Both synchronization and measurement messages are counted.

4.4.1.1 The Number of Message Transmissions/Receptions

We first count the numbers of message transmissions and receptions of conventional and reverse asymmetric time synchronization schemes based on one-way message dissemination and two-way message exchange. A single-hop scenario where one sensor node connects to the head is considered. We assume that there are 100 measurements in total at the sensor node over the period of 3600 s, which are reported to the head through measurement messages without measurement bundling, and the SI is set to 1 s. With counting both synchronization and measurement messages, the numbers of messages transmitted ($\mathbf{N}_{\mathrm{TX}}$) and received ($\mathbf{N}_{\mathrm{RX}}$) by the sensor node are counted in Table 4.1. Of particular note is that, the type listed in Table 4.1 categorize *the ways of message exchange in time synchronization*, and are not specific to certain time synchronization schemes; examples of the "Conventional Two-way", "Conventional One-way", and "Reverse Two-way" types are TPSN, FTSP, and EE-ASCFR, respectively, while BATS belongs to the "Reverse One-way" type.

From the comparison, we observe that the reverse one-way scheme could save the energy consumed by both transmissions and receptions of the synchronization messages;

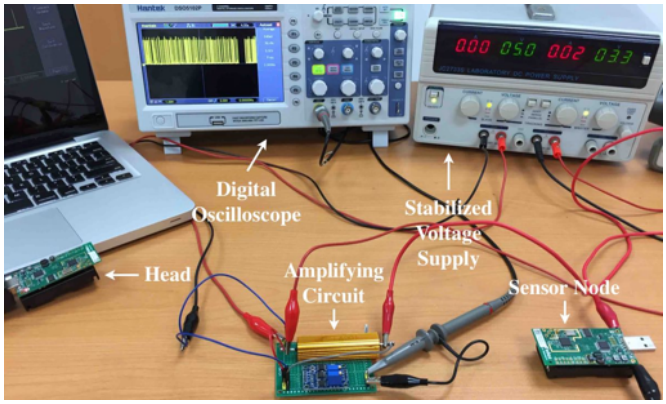


Figure 4.8: Experiment setup for the measurement of the energy consumption on a WSN sensor node [1].

this is because the reverse one-way scheme is free from transmissions and receptions of beacon messages and synchronization timestamps are embedded into measurement messages.

In the reverse two-way scheme, the "Request" synchronization message including timestamp $T1$ can be embedded into beacon messages, but still, sensor nodes consume energy to receive the beacon messages. On the other hand, the conventional two-way scheme requires the most message transmissions due to the sending the "Request" messages from the sensor node to the head. In addition, the conventional one-way scheme requires the same number of synchronization message receptions as in the reverse two-way scheme.

4.4.1.2 Direct Measurement of Energy Consumption

To measure the actual power consumption of a time synchronization scheme on the resource-constrained sensor nodes, we employ a stabilized voltage supply and a digital storage oscilloscope (DSO) to power the sensor node that synchronizes with the head and log the actual power consumption as shown in Fig. 4.8. In the experiment setup, one $1\,\Omega$ resistor is connected in series between the power supply and the sensor node. The voltage of the power supply is set to $3.3\,\mathrm{V}$, a slightly higher voltage than that received from the battery holder of the TelosB sensor board [25], to provide the sensor

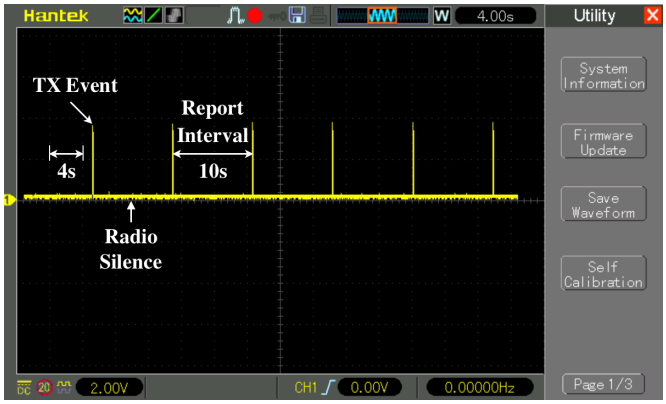


Figure 4.9: Measuring and logging the energy consumption using DSO [1].

node sufficient power. The two pins of the resistor are connected to the inputs of the amplifying circuit[2] which is used to enlarge the mA-level current input value with 150 times. The outputs of the amplifying circuit are connected to the DSO for logging the power consumption. The power consumption logged in the DSO is illustrated in Fig. 4.9. Because the voltage is stabilized in the measurement experiment which is $3.3V$ constantly, so the actual power consumption could be compared through comparing the logged current value sets.

During the experiments, the measurement is generated every 2 s and the bundling procedure with bundling 5 measurements is employed in both FTSP and BATS for a fair comparison. FTSP broadcasts and receives the synchronization beacon messages per second (i.e., the default setting) and reports the bundling messages every 10 s; BATS reports the bundling messages with the measurements and synchronization data every 10 s without broadcasting or receiving—i.e., radio listening deactivated—the synchronization beacons. The power and energy consumptions are computed as follows:

$$P(t) = \mathcal{V}(t) \times I(t), \tag{4.15}$$

$$E = \int_{t_s}^{t_e} P(t)dt, \tag{4.16}$$

where $P(t)$, $\mathcal{V}(t)$ and $I(t)$ are the instant power, voltage and current for the sensor node

[2]The amplifying circuit is based on the analog devices designer guidebook [69, chapter 4].

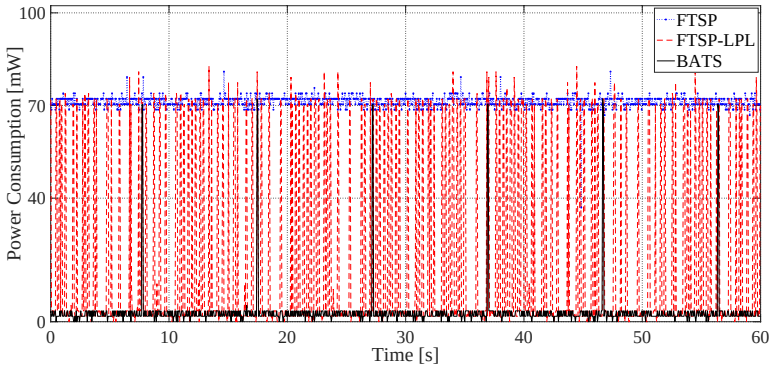


Figure 4.10: Power consumptions of different time synchronization schemes over 60 s [1].

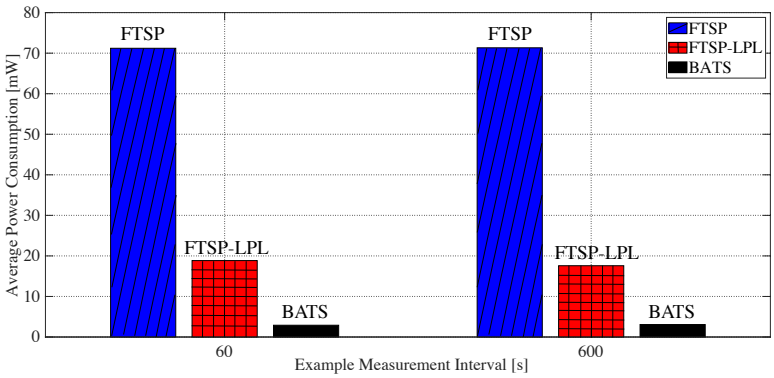


Figure 4.11: Average power consumptions of time synchronization schemes over the measurement interval of 60 s and 600 s [1].

at time t, respectively, and E is the energy consumed by the sensor node over the time period of $[t_s, t_e]$. Because the voltage $\mathcal{V}(t)$ is fixed to 3.3 V during the experiments, we only measure the current $I(t)$ over the two different time intervals of 60 s and 600 s (i.e., $t_e - t_s$ in (4.16)) to obtain power and energy consumptions; Fig. 4.10 shows the power consumption for different time synchronization schemes over the period of 60 s. Since the broadcasting of beacon messages and the reporting of the (bundled) measurement message are periodic, so the two aforementioned example measurement intervals could represent the long-term experiments. As illustrated in Fig. 4.11, the average power consumption for BATS is the lowest, while the original FTSP—i.e., without using low-power mode—consumes the most power. Specifically, the FTSP employing low-power

mode, i.e., FTSP-LPL, consumes less than one-third power of the original FTSP but more than BATS, which consumes less than 5% and 16% power of the original and the low-power FTSP, respectively.

4.4.2 Time Synchronization Accuracy

4.4.2.1 Single-Hop Scenario

We first evaluate the time synchronization accuracy of the proposed BATS scheme in comparison to that of FTSP with a single-hop scenario such as the star topology illustrated in Fig. 4.7 (c). For the experiments, we assume that the sensor node periodically sends measurement data to the head via measurement messages, where 5 measurements are bundled in each measurement message for both FTSP and BATS together with synchronization data in case of the latter. We run experiments for 3600 s.

Note that BATS uses the measurement messages to carry both measurement and synchronization data, while conventional one-way time synchronization schemes like FTSP rely on beacon messages for synchronization data and use measurement messages only for measurement data. For a fair comparison between BATS and FTSP, therefore, we define SI as the interval between two consecutive measurement messages carrying synchronization data for BATS and the interval between two consecutive beacon messages for FTSP, respectively, and use the same SI value for both schemes for each experiment.

Before carrying out a comparative analysis with FTSP, we need to decide the values of BATS system parameters affecting time synchronization performance. During the preliminary experiments, we found that the time synchronization accuracy of BATS is affected by the sample size for the linear regression for clock frequency ratio estimation. Note that, for a reasonable comparison, to find the best value of the sample sizes for different SIs, we implemented the offline re-running program in which the same raw data trace produced in the online experiment could be reused to run different offline experiments with different sample sizes. With this program, the possible sample sizes (e.g., 2, . . . , 10, . . . , 100, . . . , all) for different values of SI are evaluated. In particular,

the performance of the experiment based on all samples does not outperform the one using the most recent timestamps with a certain sample size value. This may result from, as time goes on, due to the aggravating clock drift, the very past sample data—i.e., very past timestamps—do not have positive contributions for the estimations of the timestamps in the most current synchronization interval. Instead, the most recent samples could provide relatively better contributions to the estimation of the most recent timestamps.

Fig. 4.12 shows the effect of sample size on the MSE and the MAE of measurement time estimation of BATS with different values of SI. From the results, we find that the MSE and MAE of measurement time estimation are minimal when the sample size is 19 and 5 for SI of 1 s and 10 s, respectively. As for the experiments with SI of 100 s, the size of total samples is quite limited (i.e., only 36 samples from the experiment over 3600 s), which means, it is quite difficult to apply large value for the aforementioned sample size. Anyhow, we still tuned the value of sample size in the range of 2–30, and the results showed that, 2 is the best sample size for the experiment with SI of 100 s.

With the best empirical values of the sample sizes for various values of SI, the performance of our proposed scheme embedding the linear regression method is demonstrated. The performance of the method 2 of linear regression with the best parameter value outperformed the conventional methods such as the method 1 based on the calculation of the cumulative frequency ratio proposed in [48] as shown in Fig. 4.13. In this figure, the measurement time estimation errors of our proposed time synchronization scheme based on both traditional ratio-based method and linear regression method are demonstrated, in which the linear regression method outperforms the ratio-based method in all three report intervals at different levels. In addition, Fig. 4.13 demonstrated that our proposed reverse asymmetric framework could be employed on different conventional time synchronization schemes with diverse estimation methods.

Table 4.2 summarizes the MAE and MSE of measurement time estimation from the experiments during the period of 3600 s, where “method 1” and “method 2” denote the ratio-based and the linear regression estimation methods, respectively, as in Fig. 4.13.

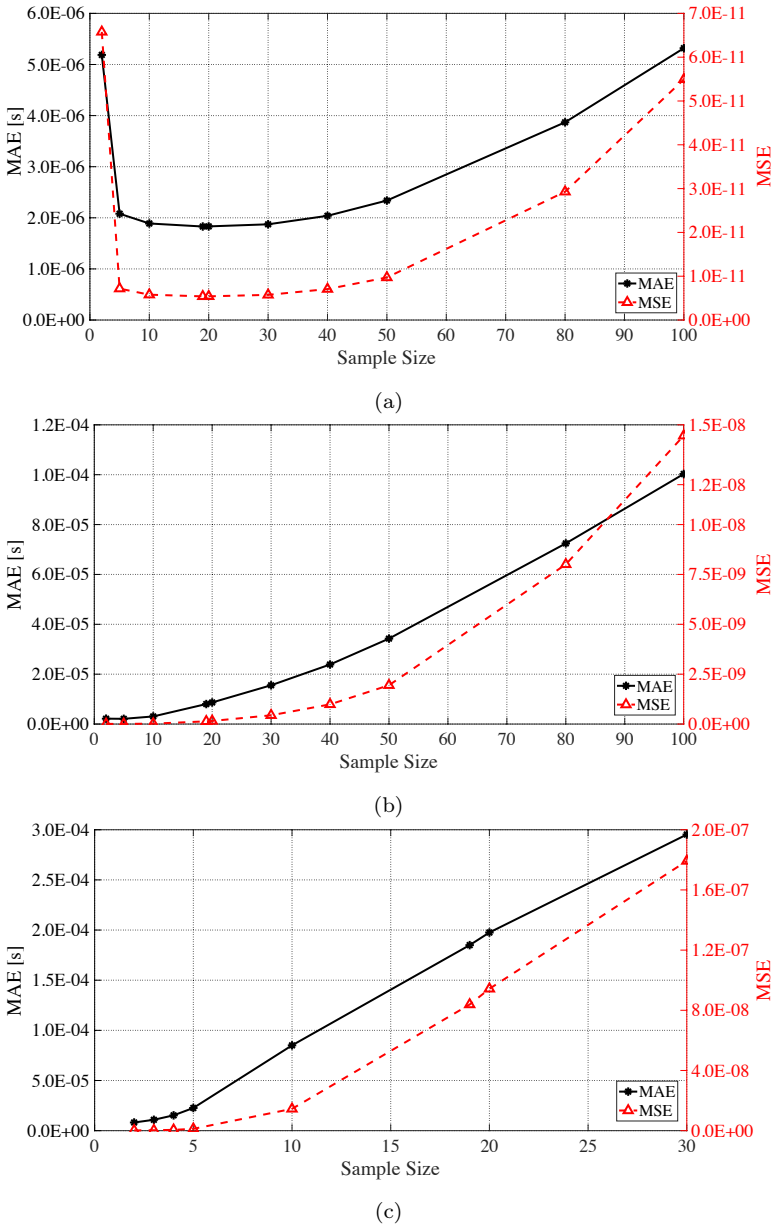
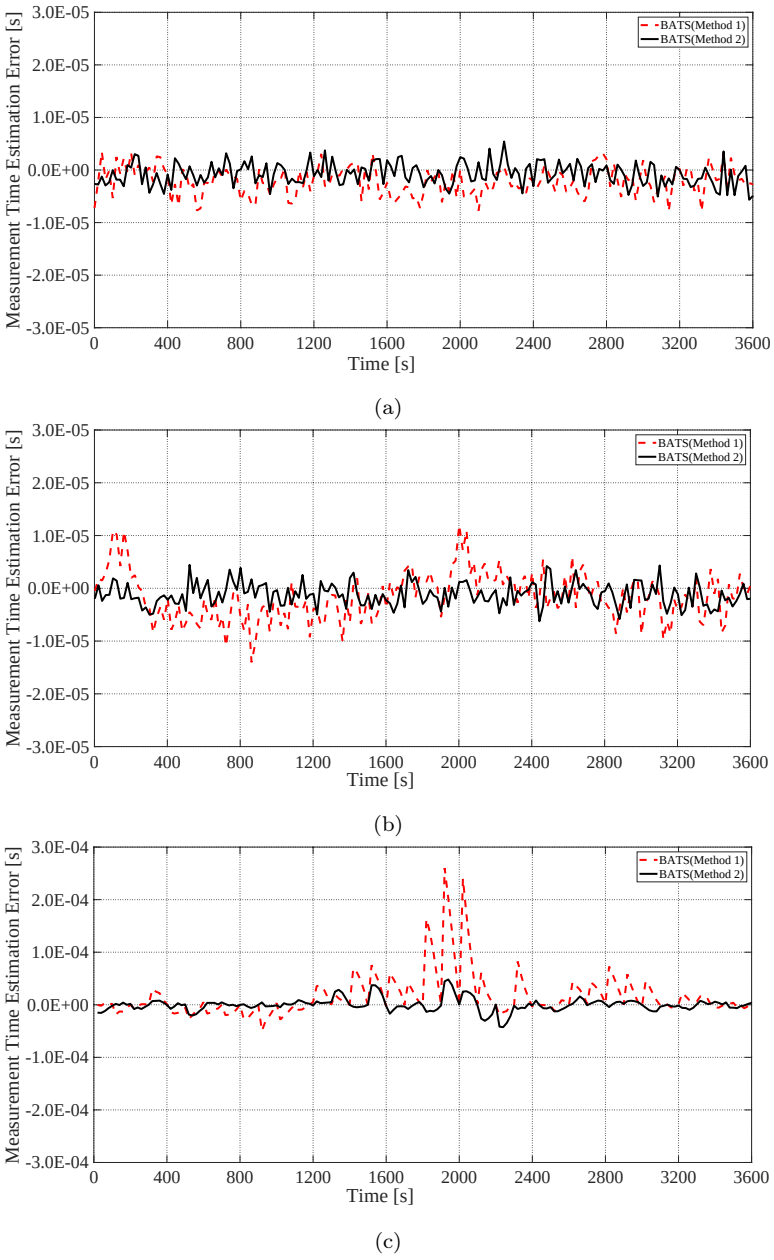


Figure 4.12: The effect of sample size on the measurement time estimation of BATS: (a) SI=1 s; (b) SI=10 s and (c) SI=100 s [1].

In the single-hop scenario, one sensor node is synchronized to one head in the experiments. Note that, as the standard FTSP implementation provided in TinyOS library is employed in our experiments, so the synchronization accuracy of FTSP is limited to



(a)

(b)

(c)

Figure 4.13: Measurement time estimation errors of BATS with ratio-based and linear regression methods with SI of (a) 1 s, (b) 10 s, and (c) 100 s [1].

millisecond-level [70]. The MAE of measurement time estimation of all three different SIs show that the proposed scheme with two proposed methods provides satisfactory precision with a minimum of 1.8299 µs which is competitive with consideration to the

Table 4.2: MAE and MSE of Measurement Time Estimation of FTSP and BATS for the Single-Hop Scenario [1]

Synchronization Scheme		MAE[1][s]	MSE[1]	$\mathbf{N}_{TX}$	$\mathbf{N}_{RX}$
FTSP[2]	SI = 100 s	0.2892E-03	0.3614E-06	36	36
	SI = 10 s	0.3164E-03	0.3983E-06	360	360
	SI = 1 s	0.3173E-03	0.4038E-06	3600	3600
BATS with the method 1	SI = 100 s	2.4837E-05	2.1194E-09	36	0
	SI = 10 s	3.2770E-06	1.7492E-11	360	0
	SI = 1 s	2.4903E-06	1.0120E-11	3600	0
BATS with the method 2	SI = 100 s	8.1524E-06	1.5805E-10	36	0
	SI = 10 s	2.1016E-06	7.3933E-12	360	0
	SI = 1 s	1.8299E-06	5.4018E-12	3600	0

[1] Based on the measurement time estimation obtained from 3600 s such that the actual performance in real deployment is represented.
[2] The standard FTSP implementation provided in TinyOS library offers limited millisecond-level time synchronization [70].

results presented in other papers—e.g., conventional two-way scheme TPSN and one-way schemes FTSP and RSP—which are also evaluated through real testbeds, however, with drastically fewer message transmissions as shown in Table 4.2.

In addition, Fig. 4.13 and Table 4.2 illustrate that a relatively smaller SI results in better performance with smaller MSE, which means the performance of the proposed scheme with both methods is related to the value of report intervals; this may result from the drifting of the low-cost crystal oscillator with up to tens of ppm of clock skew in the resource-constrained sensor node, in which the drifting range is typically larger in a longer interval.

4.4.2.2 Multi-Hop Scenario

Here we investigate the effect of the number of hops on time synchronization with a 6-hop chain topology as exhibited in Fig. 4.7 (a). We set the SI to 1 s and employ the optimal sample size of 19 for the experiment. Note that we apply the self-data bundling only in order to mainly focus on the effect of the number of hops, rather than that of bundling, and therefore make the results more consistent with those of conventional

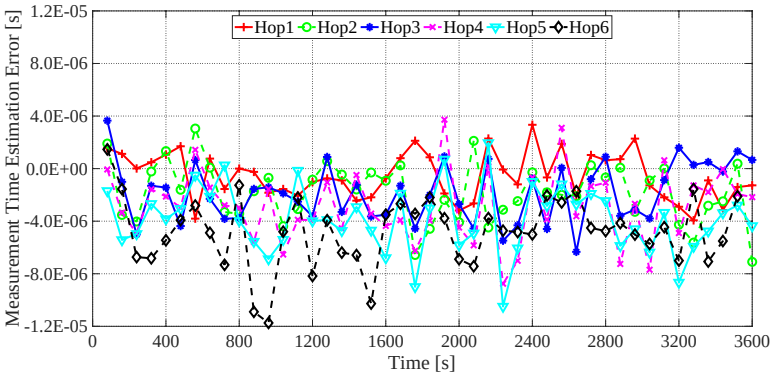


Figure 4.14: Measurement time estimation errors of BATS for the multi-hop scenario [1].

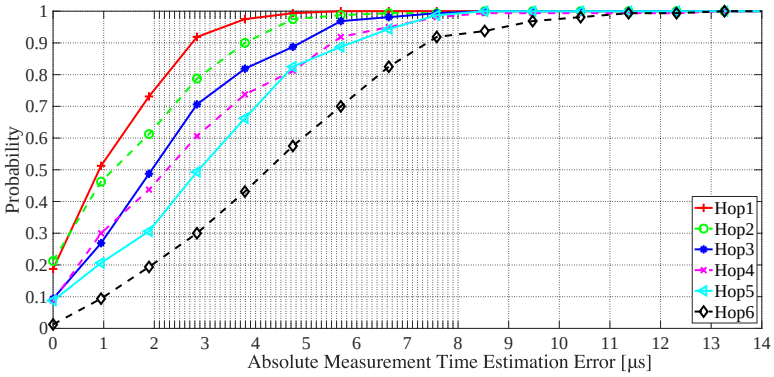


Figure 4.15: Cumulative distribution functions of the absolute measurement time estimation errors of BATS for the multi-hop scenario [1].

schemes reported in the literature.

As shown in Fig. 4.14, the level of fluctuations of the measurement time estimation errors are roughly proportional to the hop counts; for instance, the measurement time estimation errors of the node 6 hops away from the head show the highest fluctuations, while those of the node 1 hop away from the head show the least fluctuations. This is due to the per-hop synchronization strategy employed in BATS, where the estimation of the hardware clock time of a sensor node with respect to the reference clock relies on those of its upper-layer sensor nodes.

Fig. 4.15 shows the effect of the number of hops on time synchronization in a clearer way through the cumulative distribution functions (CDFs) of *absolute measurement*

Table 4.3: MAE and MSE of Measurement Time Estimation of BATS for the Multi-Hop Scenario [1]

Sensor Node		MAE(Chain)[1]	MSE(Chain)[1]	MAE(Tree)[1]	MSE(Tree)[1]
ID	6	4.2580E-06	2.4586E-11	2.6301E-06	1.0573E-11
	5	3.6149E-06	1.8405E-11	2.5140E-06	1.0240E-11
	4	3.1341E-06	1.4519E-11	2.7873E-06	1.1414E-11
	3	2.4847E-06	9.4813E-12	2.0985E-06	7.0036E-12
	2	1.9455E-06	6.0164E-12	2.1368E-06	7.1055E-12
	1	1.6764E-06	4.4735E-12	1.6932E-06	4.5415E-12

[1] Based on the measurement time estimation obtained from 3600 s such that the actual performance in real deployment is represented.

time estimation errors. 90th-percentile absolute measurement time estimation errors for sensor nodes 1 to 6 hops away from the head are 2.8 µs, 3.8 µs, 4.9 µs, 5.5 µs, 5.9 µs and 7.4 µs, respectively. The MAEs and MSEs of measurement time estimation are also summarized as MAE(Chain) and MSE(Chain) in Table 4.3. The results of Fig. 4.15 and Table 4.3 demonstrate that the proposed scheme can provide microsecond-level time synchronization accuracy for all the sensor nodes in the 6-hop WSN, even though the time synchronization error is cumulative over the hop count. In addition, we established one comparison experiment with the tree topology illustrated in Fig. 4.7 (b) to demonstrate the stable performance of BATS under various topologies. The results are summarized as MAE(Tree) and MSE(Tree) in Table 4.3.

The practical evaluation results in Section 4.4.1 and Section 4.4.2 have jointly proved that the proposed scheme could drastically conserve more energy consumptions while maintaining the same level of synchronization accuracy compared to the conventional schemes.

4.5 Summary

In this chapter, we focus on the energy-efficiency of the WSN time synchronization. The major contribution of this chapter is three-fold: First, the reverse asymmetric time synchronization framework is presented, whose synchronization-related computations—

except timestamping—are completely reassigned from the resource-constrained sensor nodes to the head. In doing so, we can not only avoid the computational errors caused by the limited precision floating-point arithmetic at the sensor nodes but also bring further potential to use more complex estimation methods at the head, e.g., machine learning techniques such as neural networks [71]. Second, building on the reverse asymmetric time synchronization framework, we proposed BATS basing on the reverse one-way message dissemination where the transmissions and receptions of the beacon messages are avoided. BATS can, therefore, significantly lower the energy consumption while achieving high time synchronization accuracy. Third, we provide the actual energy consumption measurement and the time synchronization accuracy of BATS through extensive experiments on the real WSN testbed. Experimental results demonstrate that BATS conserves up to 95% of the energy consumption compared to FTSP and provides 1.8299 µs synchronization accuracy in the single-hop scenario. In the case of the multi-hop scenario, the synchronization accuracy for 1-hop and 6-hop sensor nodes are respectively 1.6764 µs and 4.2580 µs, which results in 0.5163 µs per-hop synchronization error on average.

In addition, we have briefly outlined the message bundling procedure; it affects both the end-to-end delay and time synchronization performance, however. In the following chapter, therefore, we will provide our further investigation on the effect of the message bundling.

Chapter 5

A Per-Hop Delay Compensation Scheme For Improving Multi-Hop Time Synchronization Performance

In this chapter, we focus on improving the *multi-hop synchronization accuracy* of the WSN time synchronization; the work presented here is based on our publication of [4, 5]. We propose a per-hop delay compensation scheme based on the packet-relaying gateway nodes. We analyze the effect of timestamping and clock skew compensation on multi-hop extensions based on our proposed PHDC and conventional time translation (TT). We further implement PHDC and TT on three representative time synchronization schemes for conducting a comparison study. Experimental results on a real testbed demonstrate that the multi-hop extension based on the proposed PHDC greatly improves the multi-hop time synchronization performance of all the schemes considered compared to the multi-hop extension based on the conventional TT.

5.1 Introduction

The advent of the Internet of Things (IoT) ushers in large-scale monitoring and sensing [9, 72], which, combined with the artificial intelligence/machine learning (AI/ML), would bring a variety of intelligent applications enabling future smart homes to smart factories to smart cities [11, 73]. It is the *multi-hop extension* that enables a wireless sensor network (WSN)—i.e., one of the foundational components of IoT—to cover vast areas (e.g., farms [74], forests [75], or even metropolitan areas like Shanghai and New York City [11]) for such large-scale monitoring and sensing through long-range transmissions and flexible energy balancing among the sensor nodes. Providing time synchronization to WSN applications, which is indispensable to not only ordering the sensor data but also guaranteeing the collaboration of the sensor nodes, becomes more and more challenging as WSNs scale in size and coverage through multi-hop extension, especially considering the issues of the energy consumption and computational complexity of a large number of battery-powered, low-cost WSN sensor nodes as investigated in [2, 1].

With a major focus on the *energy efficiency and computational complexity* of a large number of battery-powered, low-cost WSN sensor nodes, we have also proposed a series of WSN time synchronization schemes based on the *reverse asymmetric time synchronization framework* [2, 3, 1]: First, we have proposed schemes based on the reverse two-way message exchange—i.e., energy-efficient time synchronization based on asynchronous source clock frequency recovery (EE-ASCFR) [2] and asymmetric high-precision time synchronization (AHTS) [3]—to reduce the energy consumption of the conventional two-way schemes while maintaining the synchronization accuracy through propagation delay compensation; initiating the two-way message exchange process from the head instead of sensor nodes, we could move most of the tasks related with time synchronization from sensor nodes to the head and thus relieve the sensor nodes from the computational burden of time synchronization. In addition, bundling the upstream synchronization messages together with measurement data could reduce the energy

consumption for message transmissions at the sensor nodes. As for one-way schemes, we have proposed beaconless asymmetric energy-efficient time synchronization scheme (BATS) [1] to reduce the energy consumption and computational burden at both gateway and leaf sensor nodes in multi-hop WSNs, where we can avoid broadcasting of beacons including time synchronization messages and their forwarding at each gateway and sensor nodes to achieve higher energy efficiency while maintaining microsecond-level time synchronization accuracy.

While extending the proposed time synchronization schemes to multi-hop WSNs based on the time translation (TT) method outlined in [2], we observed that multi-hop time synchronization faces a cumulative synchronization error caused by its per-hop synchronization strategy, which results from the recursive TT at either gateways or the head [3, 1]. Note that the multi-hop extension based on TT is quite popular, especially among one-way schemes including FTSP. Since the sensor nodes in one-way flooding schemes are synchronized with the reference node through layer-by-layer TT, the closer the sensor node is to the reference node (i.e., the head), the better is its synchronization accuracy. In other words, the multi-hop synchronization accuracy in flooding schemes, as well as BATS, is curbed by the TT method during the multi-hop extension through gateway nodes. The two-way schemes—i.e., the novel reverse two-way schemes like EE-ASCFR/AHTS and the conventional ones like TPSN—could also suffer from the cumulative errors in TT. Moreover, the multi-hop extension based on packet-relaying gateways described in [2] is no exception in this regard due to the cumulative errors caused again by the aforementioned processing delay during the packet-relaying process at the gateway nodes.

To address the cumulated multi-hop synchronization error induced by the processing delays at gateway nodes, therefore, we have recently proposed a novel per-hop delay compensation (PHDC) method [4] that laying its foundation on the packet-relaying gateways [2, 76], where we demonstrate that PHDC is capable of alleviating the cumulative synchronization errors through a preliminary investigation with experimental results. To further extend the investigation based on coarse-level mathematical anal-

ysis and experiments only with one-way schemes in [4], in this chapter, we carry out an extensive comparative analysis of the performance of multi-hop extension based on TT and PHDC for both one-way and two-way WSN time synchronization schemes in the context of energy-efficient multi-hop WSN time synchronization with low computational complexity. The major contributions of our work in this chapter are summarized as follows:

- First, we systematically and mathematically analyze the feasibility of compensating for the processing delay at gateway nodes with consideration on the effect of timestamping and clock skew compensation over multiple hops for both TT and PHDC methods.
- Second, we describe two implementation options for the multi-hop extension of WSN time synchronization schemes based on PHDC and discuss the details of PHDC implementation specific to the representative one-way schemes, i.e., BATS and FTSP, and the two-way scheme, i.e., EE-ASCFR.
- Third, we extend BATS, FTSP, and EE-ASCFR for the multi-hop time synchronization based on both TT and PHDC, and implement them for linear and tree topologies on a real WSN testbed consisting of TelosB sensor nodes [25] running TinyOS [44].
- Finally, we comprehensively demonstrate the experimental evaluation results, where the improvement brought by PHDC in multi-hop time synchronization performance is elucidated over the combination of three time synchronization schemes and two network topologies.

The rest of this chapter is organized as follows: The proposed PHDC is discussed in Section 5.2. The systematic analysis of the effect of timestamping and clock skew compensation on PHDC in comparison to TT is carried out in Section 5.3. The implementation of PHDC on both one-way and two-way schemes is exhibited in Section 5.4. The experimental results evaluated on a real testbed are demonstrated in Section 5.5. Section 5.6 concludes our work and outlines our future work.

5.2 Per-Hop Delay Compensation in Multi-Hop WSNs Based on Packet-Relaying Gateways

5.2.1 Packet-Relaying Gateways For Multi-Hop Extension

Packet relaying and TT gateways are two options advocated in [2] for EE-ASCFR extending to multi-hop time synchronization. Unlike TT that is adopted for multi-hop extension by most time synchronization schemes (e.g., [18, 22, 1]), packet relaying is relatively simpler and provides a transparent end-to-end connection between the head and a leaf sensor node as far as time synchronization is concerned. Therefore, packet relaying could conceptually reduce the problem of multi-hop time synchronization to that of single-hop time synchronization.

However, as pointed out in [2, 4], the performance of time synchronization based on packet relaying could be affected by rather large and random per-hop processing delay resulting from queueing/scheduling and MAC operations at each gateway. Anyhow, due to its simplicity, packet relaying provides a more attractive option of multi-hop extension to time synchronization schemes highlighting energy efficiency and low computational complexity as their major advantages. Especially for large-scale deployments, the simplicity of packet relaying could relieve a large number of gateway nodes from the burden of multi-hop time synchronization tasks in terms of both energy and computation.

5.2.2 Delay Compensation at a Packet-Relaying Gateway

We explain the basic idea of PHDC using a 2-hop WSN shown in Fig. 5.1, where for ease of notation we ignore subindexes of timestamps denoting hop numbers unlike the analysis in following Section 5.3. The processing-delay-compensated timestamp $\widehat{T1}(i)$ for ith synchronization (i=2, 3...) can be expressed as follows:

$$\widehat{T1}(i) = T1(i) + \left\lfloor \Delta(i) \times \frac{T1(i) - T1(i-1)}{TA(i) - TA(i-1)} \right\rfloor, \tag{5.1}$$

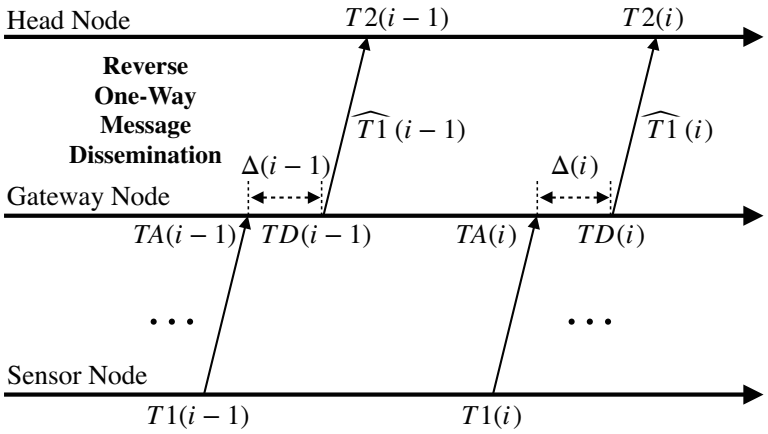


Figure 5.1: A WSN with one gateway node and one sensor node [4].

where $T1(i)$, $TA(i)$, and $TD(i)$ are the MAC-layer timestamps recorded at transmission, reception and forwarding of the ith synchronization message on the sensor node and the gateway node, and $\Delta(i)$ is the processing delay defined as $TD(i)-TA(i)$. The *floor function* (i.e., $\lfloor\cdot\rfloor$) is used to convert the processing delay scaled by the estimated clock frequency ratio between the gateway and the sensor nodes to an integer for a timestamp.

The integer conversion in (5.1) could eliminate the effect of the precision loss in the floating-point division: Let r be the true value of the floating-point division $\frac{T1(i)-T1(i-1)}{TA(i)-TA(i-1)}$ and ε be the error resulting from the precision loss. In this case, the second term in (5.1) can be expressed as follows:

$$\begin{aligned}\lfloor\Delta(i)\times(r+\varepsilon)\rfloor &= \lfloor\Delta(i)r+\Delta(i)\varepsilon\rfloor \\ &= n+\lfloor\delta+\Delta(i)\varepsilon\rfloor,\end{aligned}$$

where n and δ are the integer and the fractional part of $\Delta(i)r$, respectively (i.e., $n=\lfloor\Delta(i)r\rfloor$ and $\delta=\Delta(i)r-n$). Therefore, if $\delta+\Delta(i)\varepsilon$ is less than one (i.e., $\lfloor\delta+\Delta(i)\varepsilon\rfloor=0$), the effect of the precision loss is eliminated during the integer conversion.

Note that, because $\Delta(i)$ is an integer (i.e., a difference of timestamps), if the clock frequency ratio between the gateway and the sensor nodes is close to one, the fractional part of $\Delta(i)r$ becomes negligible (i.e., $\delta\approx0$). In such a case, we can obtain the following

upper bound for the precision loss:

$$\Delta(i)\varepsilon < 1 \Rightarrow \varepsilon < \frac{1}{\Delta(i)}. \tag{5.2}$$

If the precision loss is less than the upper bound given in (5.2), it does not affect the processing delay compensation.

5.3 Effect of Timestamping and Clock Skew Compensation on Multi-Hop Extension Based on PHDC and TT

The preliminary analysis based on a single gateway in Section 5.2 shows that the clock skew could be selectively compensated for depending on the processing delay and relative skews of the sensor nodes. In this section, we present a more comprehensive analysis of the effect of timestamping and clock skew compensation on PHDC in comparison with widely-employed TT.

5.3.1 Per-Hop Delay Compensation

We begin our analysis with the simplest case of the 2-hop WSN over a single gateway node shown in Fig. 5.2. During the kth (k=1, 2, ...) synchronization, the timestamp $\widehat{T1_2}^k$ of Fig. 5.2 (a), whose processing delay is compensated for at gateway node 1, can be described based on Section 5.2.2 as follows:

$$\widehat{T1_2}^k = T1_2^k + \left\lfloor R_{2,1}^k \times \Delta_1^k \right\rfloor, \tag{5.3}$$

where Δ_1^k is the processing delay at gateway node 1 estimated by $T1_1^k - T2_1^k$, and $R_{2,1}^k$ is the clock frequency ratio between gateway node 1 and sensor node 2 which could be estimated by either simple ratio-based method [48] or more advanced ones like linear regression [1]. Note that $T1_2^k$ is the timestamp recorded at sensor node 2 during the transmission of kth synchronization message and that $T2_1^k$ and $T1_1^k$ are the timestamps recorded at gateway node 1 during the reception and forwarding of that message, re-

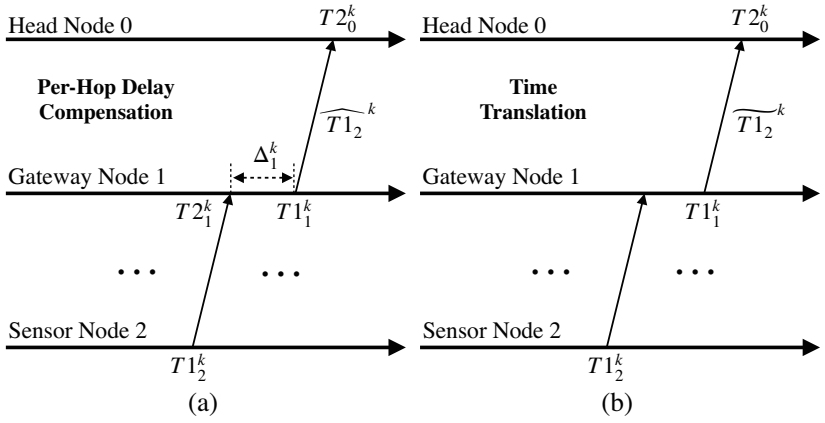


Figure 5.2: A 2-hop WSN over a single gateway with multi-hop extension based on (a) per-hop delay compensation and (b) time translation [5].

spectively. The floor function is again used to convert the compensated delay to an integer number.[1]

Now we extend (5.3) by including the errors in timestamping of $T2_1^k$ and $T1_1^k$ and in clock skew compensation by $R_{2,1}^k$ at gateway node 1 to investigate their effect on PHDC[2]: First, we define a timestamping error of a timestamp as the differences between the continuous hardware clock time and the discrete timestamp as shown in Fig. 5.3, where we assume the first-order affine clock model [78] for nodes' hardware clocks; for instance, $\delta_{T2_1^k}$—i.e., the timestamping error of $T2_1^k$—is defined as $T2_1(k)-T2_1^k$ or $\mathrm{frac}(T2_1(k))$ where $\mathrm{frac}(x)\triangleq x-\lfloor x\rfloor$ for $x\geq 0$; the timestamping errors could result from not only the integer conversion but also the remainder of interrupt delay compensation in MAC-layer timestamping [17, 1]. Second, we denote by $\delta_{\epsilon_{1,2}^k}$ the error in clock skew compensation including precision loss, where $\epsilon_{1,2}^k$ is a clock skew (i.e., $1+\epsilon_{1,2}^k=R_{1,2}^k$).

Let us consider the processing delay based on the discrete timestamps and compen-

[1] Alternatively, the *ceiling function* could be used as in [77]

[2] We do not include the timestamping error for $T1_2^k$, which occurs at sensor node 2, in the following analyses as we focus on PHDC at gateway nodes.

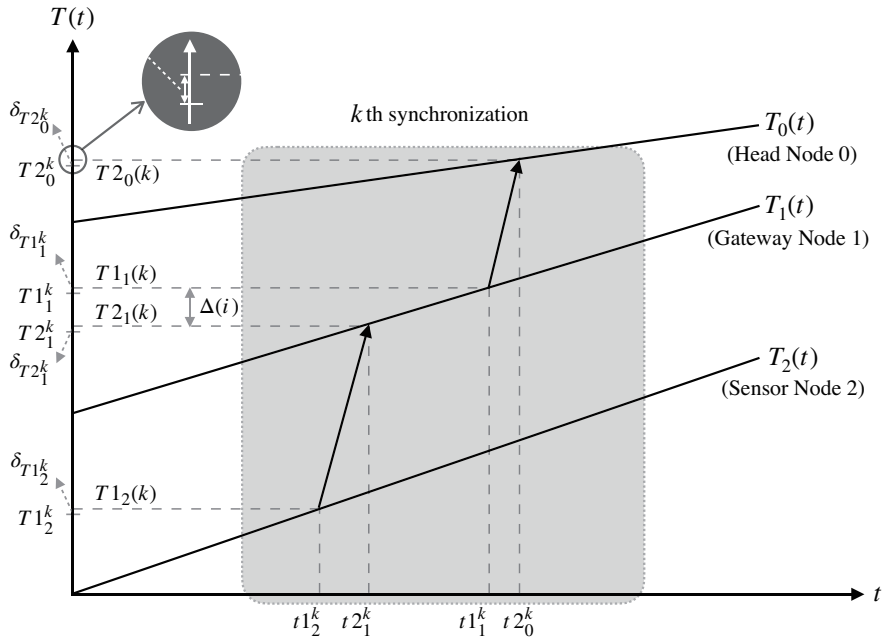


Figure 5.3: Relationship between global reference times (e.g., $t1_2^k$) and corresponding continuous hardware clock times (e.g., $T1_2(k)$) and timestamps (e.g., $T1_2^k$) during the kth synchronization of the 2-hop WSN shown in Fig. 5.2 [5].

sated by the estimated value of clock skew in (5.3), i.e.,

$$\begin{aligned}\left\lfloor R_{2,1}^k \times \Delta_1^k \right\rfloor &= \left\lfloor \left(1 + \epsilon_{1,2}^k + \delta_{\epsilon_{1,2}^k}\right) \times \left(T1_1^k - T2_1^k\right) \right\rfloor \\ &= T1_1^k - T2_1^k + \left\lfloor \left(\epsilon_{1,2}^k + \delta_{\epsilon_{1,2}^k}\right) \times \left(T1_1^k - T2_1^k\right) \right\rfloor,\end{aligned} \tag{5.4}$$

and that based on the continuous hardware clock times and compensated by the true value of clock skew, i.e.,

$$\begin{aligned}&R_{2,1}^k \times (T1_1(k) - T2_1(k)) \\ &\quad = \left(1 + \epsilon_{1,2}^k\right) \times \left(\left(T1_1^k + \delta_{T1_1^k}\right) - \left(T2_1^k + \delta_{T2_1^k}\right)\right).\end{aligned} \tag{5.5}$$

Subtracting (5.4) from (5.5) and rearranging it, we can obtain the *error in PHDC over*

a single gateway node as follows:

$$\begin{aligned}&\left(\epsilon_{1,2}^k\Delta_1^k-\left\lfloor\left(\epsilon_{1,2}^k+\delta_{\epsilon_{1,2}^k}\right)\Delta_1^k\right\rfloor\right)+\left(\delta_{T1_1^k}-\delta_{T2_1^k}\right)\times\left(1+\epsilon_{1,2}^k\right)\\&\quad\approx\left(\epsilon_{1,2}^k\Delta_1^k-\left\lfloor\epsilon_{1,2}^k\Delta_1^k\right\rfloor\right)+\left(\delta_{T1_1^k}-\delta_{T2_1^k}\right)\\&\quad=\mathrm{frac}(\epsilon_{1,2}^k\Delta_1^k)+\left(\delta_{T1_1^k}-\delta_{T2_1^k}\right),\end{aligned}\tag{5.6}$$

where the approximation is done on the basis of $\delta_{\epsilon_{1,2}^k}\ll\epsilon_{1,2}^k$ and $\epsilon_{1,2}^k\ll1$ because the clock skew compensation error (mainly precision loss) is less than 10^{-7} in 32-bit single-precision floating point arithmetic [3] and a typical frequency tolerance of a crystal over the manufacturing process (hence the clock skew) is ±100 ppm [79].

Now we can consider the effect of timestamping and clock skew compensation on PHDC over multiple gateway nodes based on (5.6). Let X_i and Y_i be random variables modeling the first and the second component of the error in PHDC in (5.6) at gateway node i. Then, we can model the *total error in PHDC for N-hop WSN* (i.e., over $N-1$ gateway nodes) as follows:

$$\sum_{i=1}^{N-1}(X_i+Y_i),\tag{5.7}$$

which gives the average of the total error in PHDC as follows:

$$\mathrm{E}\left[\sum_{i=1}^{N-1}(X_i+Y_i)\right]=\sum_{i=1}^{N-1}\left(\mathrm{E}\left[X_i\right]+\mathrm{E}\left[Y_i\right]\right).\tag{5.8}$$

Because timestamping errors (i.e., fractional part of continuous hardware clocks) are likely to be uniformly distributed in the range of $[0,1)$, Y_i modeling $\delta_{T1_i^k}-\delta_{T2_i^k}$ can be considered uniformly distributed as well in the range of $(-1,1)$ (i.e., $Y_i\sim U(-1,1)$), and $\mathrm{E}\left[Y_i\right]=0$. This means that the effects of timestamping on PHDC at multiple gateway nodes could be canceled one another. In such a case, the average of the total error in PHDC reduces to

$$\sum_{i=1}^{N-1}\mathrm{E}\left[X_i\right],\tag{5.9}$$

which is solely determined by X_i modeling $\mathrm{frac}(\epsilon_{i,i+1}^k\Delta_i^k)$.

The variance of the total error can be obtained, too, on the condition that X_i's and Y_i's are independent of one another, i.e.,

$$\begin{aligned}\mathrm{Var}\left(\sum_{i=1}^{N-1}(X_i+Y_i)\right) &= \sum_{i=1}^{N-1}\mathrm{Var}(X_i)+\sum_{i=1}^{N-1}\mathrm{Var}(Y_i)\\ &= \sum_{i=1}^{N-1}\mathrm{Var}(X_i)+\frac{N-1}{3},\end{aligned} \tag{5.10}$$

because

$$\mathrm{Var}(Y_i)=\frac{(1-(-1))^2}{12}=\frac{1}{3}$$

for $Y_i \sim U(-1,1)$.

5.3.2 Time Translation

We also begin our analysis with the 2-hop WSN over a single gateway node shown in Fig. 5.2. During the kth time synchronization (k=1, 2, ...), the timestamp $\widetilde{T1_2}^k$ of Fig. 5.2 (b), which is translated at gateway node 1, can be expressed as follows:

$$\widetilde{T1_2}^k = \left\lfloor R_{2,1}^k \times T1_2^k + \theta_{1,2}^k \right\rfloor, \tag{5.11}$$

where $\theta_{1,2}^k$ is the clock offset between gateway node 1 and sensor node 2. Note that the translation of the received timestamp in (5.11) is not involved with timestamping unlike PHDC.

Let $\delta_{\theta_{1,2}^k}$ be the fractional part of the clock offset (i.e., $\mathrm{frac}(\theta_{1,2}^k)$), which also takes into account the error in clock offset estimation including precision loss. Let us consider the translated timestamp based on the estimated value of clock skew, i.e.,

$$\begin{aligned}&\left\lfloor \left(1+\epsilon_{1,2}^k+\delta_{\epsilon_{1,2}^k}\right) T1_2^k + \left(\left\lfloor\theta_{1,2}^k\right\rfloor+\delta_{\theta_{1,2}^k}\right)\right\rfloor\\ &\quad = T1_2^k + \left\lfloor \left(\epsilon_{1,2}^k+\delta_{\epsilon_{1,2}^k}\right) T1_2^k + \delta_{\theta_{1,2}^k}\right\rfloor + \left\lfloor\theta_{1,2}^k\right\rfloor,\end{aligned} \tag{5.12}$$

and the translated time based on the true value of clock skew without integer conversion,

i.e.,

$$(1+\epsilon_{1,2}^k)T1_2^k + \left(\left\lfloor \theta_{1,2}^k \right\rfloor + \delta_{\theta_{1,2}^k}\right). \tag{5.13}$$

As in Section 5.3.1, we can obtain the *error in TT over a single gateway node* by subtracting (5.12) from (5.13) as follows:

$$\begin{aligned} &\epsilon_{1,2}^k T1_2^k + \delta_{\theta_{1,2}^k} - \left\lfloor \left(\epsilon_{1,2}^k + \delta_{\epsilon_{1,2}^k}\right) T1_2^k + \delta_{\theta_{1,2}^k} \right\rfloor \\ &\quad \approx \epsilon_{1,2}^k T1_2^k + \delta_{\theta_{1,2}^k} - \left\lfloor \epsilon_{1,2}^k T1_2^k + \delta_{\theta_{1,2}^k} \right\rfloor \\ &\quad = \mathrm{frac}(\epsilon_{1,2}^k T1_2^k + \delta_{\theta_{1,2}^k}), \end{aligned} \tag{5.14}$$

where the approximation is done on the basis of $\delta_{\epsilon_{1,2}^k} \ll \epsilon_{1,2}^k$ as discussed in Section 5.3.1.

As in Section 5.3.1, we can consider the effect of timestamping and clock skew compensation on TT over multiple gateway nodes based on (5.14). Let Z_i be a random variable modeling the error in TT in (5.14) at gateway node i—i.e., $\mathrm{frac}(\epsilon_{i,i+1}^k T1_{i+1}^k + \delta_{\theta_{i,i+1}^k})$—that is in the range of $[0, 1)$. Then, we can model the *total error in TT for N-hop WSN* as follows:

$$\sum_{i=1}^{N-1} Z_i, \tag{5.15}$$

which gives the average of the total error in TT as follows:

$$\mathrm{E}\left[\sum_{i=1}^{N-1} Z_i\right] = \sum_{i=1}^{N-1} \mathrm{E}\left[Z_i\right]. \tag{5.16}$$

As in Section 5.3.1, the variance of the total error in TT can be obtained on the condition that Z_i's are independent of one another, i.e.,

$$\mathrm{Var}\left(\sum_{i=1}^{N-1} Z_i\right) = \sum_{i=1}^{N-1} \mathrm{Var}\left(Z_i\right). \tag{5.17}$$

5.3.3 Comparison: PHDC vs. TT

Comparing the total error in PHDC and TT over multiple gateway nodes analyzed in the previous sections, we can find a couple of major differences between the two: First,

in the case of PHDC, though the error over a single gateway node depends on both timestamping and clock skew compensation, the effect of timestamping can be canceled out, which leaves only the effect of clock skew compensation in the average of the total error over multiple gateway nodes in (5.9). In the case of TT, on the other hand, the effect of both clock skew and offset compensation controls the average of the total error over multiple gateway nodes.

Second, the actual value of X_i in the total error in PHDC, which can be in the range of $[0, 1)$ by definition of frac$(\cdot)$, could be much smaller than one unless the traffic load of a gateway node and thereby its processing delay (i.e., Δ_i^k) is comparable to or larger than the inverse of the clock skew (i.e., $1/\epsilon_{i,i+1}^k$) because a typical value of clock skew is very small as discussed in Section 5.3.1. This is not the case, however, for the two components of the error in TT, i.e., $\epsilon_{i,i+1}^k T1_{i+1}^k$ and $\delta_{\theta_{i,i+1}^k}$ in (5.14). As for the first component, unlike the processing delay in PHDC (i.e., $\Delta_i^k = T1_i^k - T2_i^k$), $T1_{i+1}^k$ in TT can take any integer value in the range of $[0, 2^S)$, where S is the size of a timestamp in bits, so its value can be large even after multiplied by the clock skew. Given that $\delta_{\theta_{i,i+1}^k}$ is the fractional part of the clock offset in the range of $[0, 1)$, it is likely that Z_i can take any value in the range of $[0, 1)$ unlike X_i.

In summary, we can conclude that the cumulative error in multi-hop time synchronization is well under control for PHDC in comparison to TT, because the major component in the error[3] in PHDC (i.e., timestamping error which is the fractional part of continuous hardware clock) can be canceled out over multiple gateway nodes, while that in the error in TT (i.e., the fractional part of the clock offset) cannot.

5.4 On The Implementation of PHDC

In this section, we provide a general overview of PHDC implementation. We also discuss the details of the implementation specific to the two energy-efficient time synchronization schemes based on the reverse asymmetric framework (i.e., BATS and EE-ASCFR)

[3]On the condition that the error in clock skew compensation is negligible compared to the clock skew itself (i.e., $\delta_{\epsilon_{i,i+1}^k} \ll \epsilon_{i,i+1}^k$) as discussed in Section 5.3.1.

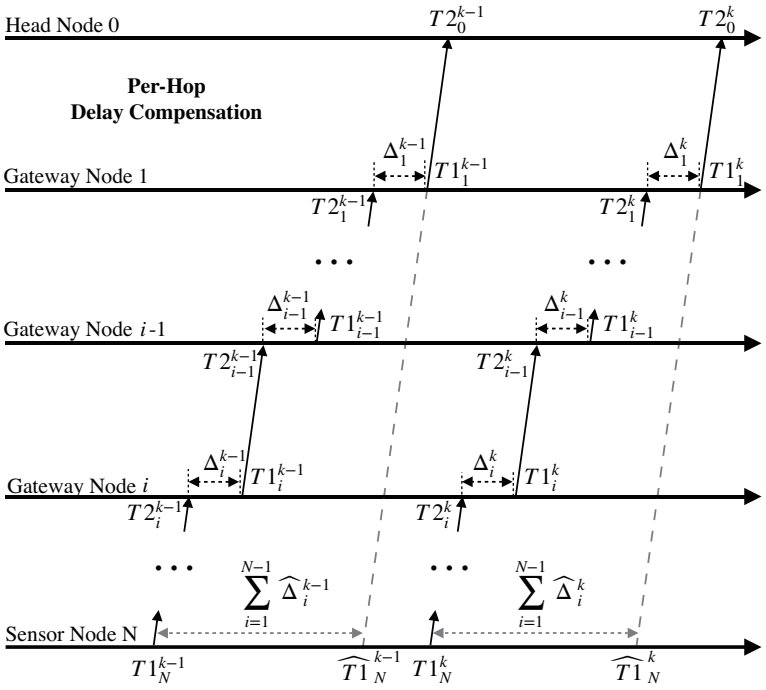


Figure 5.4: PHDC implementation for one-way N-hop WSN time synchronization based on the reverse asymmetric framework [5].

as well as one of the most popular conventional schemes (i.e., FTSP).

5.4.1 Delay Estimation in PHDC

Consider two neighbor nodes—i.e., gateway nodes $i-1$ and i—in Fig. 5.4. During the kth synchronization, a pair of timestamps for the departure time $T1_i^k$ at gateway node i and the arrival $T2_{i-1}^k$ at gateway node $i-1$ are recorded through MAC-layer timestamping. The clock frequency ratio can be estimated based on a set of those timestamps using a simple ratio-based method or more advanced methods such as linear regression. Here, we consider the ratio-based estimation, where we can estimate the clock frequency ratio $R_{i,i-1}^k$ as follows[4]: For $k>l\geq 1$,

$$R_{i,i-1}^k = \frac{T1_i^k - T1_i^l}{T2_{i-1}^k - T2_{i-1}^l}. \tag{5.18}$$

[4]The clock skew compensation could be ignored when the clock frequencies of all gateway and sensor nodes are synchronized to the head [2] or the processing delay is controlled within a certain bound [4].

Note that, if we fix l to 1, (5.18) becomes the cumulative ratio method adopted in EE-ASCFR [2]; for simplicity, we set l to $k-1$ in the following. Then, the skew-compensated processing delay $\widehat{\Delta}_{i-1}^{k}$ based on (5.18) with $l=k-1$ is given by

$$\begin{aligned}\widehat{\Delta}_{i-1}^{k} &= R_{i,i-1}^{k} \times \Delta_{i-1}^{k} \\ &= \frac{T1_{i}^{k} - T1_{i}^{k-1}}{T2_{i-1}^{k} - T2_{i-1}^{k-1}} \times (T1_{i-1}^{k} - T2_{i-1}^{k}),\end{aligned} \tag{5.19}$$

where $\Delta_{i-1}^{k}=T1_{i-1}^{k}-T2_{i-1}^{k}$ as discussed in Section 5.3.1.

One implementation option discussed in [4] is the centralization of PHDC in the head at the expense of the increased communication overhead, which can address the impact of limited precision floating-point arithmetic of gateway and sensor nodes on time synchronization. In this option, we could more accurately compensate for the processing delays by doing the following calculation at the head based on all the timestamps transferred from gateway and sensor nodes:

$$\widehat{T1}_{N}^{k} = T1_{N}^{k} + \sum_{i=1}^{N-1} \left(\left(\prod_{j=i+1}^{N} R_{j,j-1}^{k} \right) \times \Delta_{i}^{k} \right). \tag{5.20}$$

An alternative, distributed implementation option for gateway and sensor nodes is to compensate for the processing delay and replace the original timestamp $T1_{N}^{k}$ from sensor node N with a new compensated timestamp *on the fly* through gateway nodes[5]: At gateway node i ($i=1,\ldots,N-1$), we replace a received timestamp for the message departure time with a compensated timestamp $\widehat{T1}_{N,i}^{k}$ as follows:

$$\widehat{T1}_{N,i}^{k} = \begin{cases} \widehat{T1}_{N,i-1}^{k} + \widetilde{\Delta}_{i}^{k} & \text{if } i < N-1, \\ T1_{N}^{k} + \widehat{\Delta}_{N-1}^{k} & \text{if } i = N-1, \end{cases} \tag{5.21}$$

where

$$\widetilde{\Delta}_{i}^{k} = \frac{\widehat{T1}_{N,i+1}^{k} - \widehat{T1}_{N,i+1}^{k-1}}{T2_{i}^{k} - T2_{i}^{k-1}} \times (T1_{i}^{k} - T2_{i}^{k}). \tag{5.22}$$

[5] Algorithm 3 in Section 5.4.2.1 explains this option in more detail.

Note that the estimation of the clock frequency ratio in (5.22) is now based on the updated timestamps. In this way, the compensation of processing delay based on clock skew can be done independently at each gateway node. This option could be readily implemented at the resource-constrained gateway and sensor nodes due to its simplicity and is our choice for PHDC implementation on the three representative schemes, which will be discussed in the next subsection.

With either of the PHDC implementation options, the head can finally estimate clock parameters based on the final pair of timestamps $(\widehat{T1}_N^k,\ T2_0^k)$ during the kth synchronization as if the head and sensor node N are directly connected to each other.

5.4.2 Case Studies

Here we discuss the details of our own implementation of PHDC on three representative WSN time synchronization schemes for their performance analysis through real testbed experiments, whose results are reported in Section 5.5.

5.4.2.1 BATS

Algorithm 3 describes the details of the lightweight PHDC implementation at gateway nodes for BATS, where a gateway node keeps track of timestamps $T1$ and $T2$ from the initial phase to the current and then estimates the most recent clock frequency ratio based on the current timestamps.

PHDC running on gateway node is formed with two parts locating respectively in the application and the MAC layer. The relatively complex processes, i.e., collecting timestamp pairs and calculating the frequency ratio, are done at the application layer. The frequency ratio is calculated following the illustration in Section 5.4.1, which is based on the simple ratio-based method leveraging the timestamp pairs of $T1$ and $T2$. Afterwards, the packet that contains the $T1$ together with the calculated frequency ratio and current $T2$ will be sent out to the MAC layer. The only operation which is done at the MAC layer besides the MAC-layer timestamping is the updating of the timestamp $T1$. Using the MAC-layer timestamp stored in $T1'$, we could update the $T1$

Data: The node maintains the following data and variables:
- e: Event object including a timestamp;
- $packet\_status$: Variable indicating the status of packet (i.e., FIRST_PACKET or NON_FIRST_PACKET);
- d: Measurement data;
- ts: Measurement timestamp;
- p: Packet object (optionally) including timestamps from MAC-layer timestamping;
- $T1, T2, T1\_last, T2\_last, T1'$: Timestamps;
- R: Frequency ratio;
- Q_M: FIFO queue for measurement data;
- Q_P: FIFO queue for packets;
- Q_{T2}: FIFO queue for timestamp $T2$.

```
1  On detecting an event e:
2  switch e.type do
3      case MEASUREMENT do // its own measurement
4          d ← Q_M.dequeue()                          // measurement data from the queue
5          ts ← e.getTimestamp()                      // for measurement, not for synchronization
6          p ← Packet(d, ts, T1)                      // create a packet object
7          sendToMAC(p)                               // send to MAC layer
8          p.T1 ← getMACTimestamp()                   // get MAC-layer timestamp
9          send(p)                                    // send packet out
10     case PACKET do
11         if p.getDestAddress()≠HEAD then // packet received from other sensor nodes
12             p ← Q_P.dequeue()                      // packet from the queue
13             T2 ← Q_T2.dequeue()                    // from MAC-layer timestamping
14             T1 ← p.T1                              // get T1 from the packet
15             if packet_status==FIRST_PACKET then
16                 R ← 1.0f                           // initialize frequency ratio variable
17             else
18                 R ← (T1 − T1_last)/(T2 − T2_last)  // calculate current frequency ratio
19             T1_last ← T1                           // update last T1
20             T2_last ← T2                           // update last T2
21             sendToMAC(p, R, T2)                    // send to MAC layer
22             T1' ← getMACTimestamp()                // get MAC-layer timestamp
23             p.T1 ← p.T1 + R ∗ (T1' − T2)           // update T1
24             send(p)                                // forward the packet
25         else
               // Process the packet from the head ...
26     otherwise do
           // Process other event ...
```

Algorithm 3: PHDC at a gateway node in BATS.

in the packet as exhibited in line 23 of Algorithm 3. Note that, though this calculation is simple, it may delay the actual transmission of the packet due to its being done at the MAC layer. This delay is system-specific, which could be counted in the interrupt delay and later handled by the receiver through interrupt delay compensation as illustrated

in [17]. When the delay is relatively large, however, we could alternatively skip the $T1$ update at the MAC layer and send the corresponding frequency ratio and delay to the upper gateway node for post-updating the $T1$ at the application layer, which of course, at the expense of payload overhead.

Compared to the original BATS based on TT, the multi-hop extension of BATS based on PHDC not only enhances the synchronization performance through processing delay compensation but also reduces the communication overhead, i.e., timestamps occupying the payload: For instance, $2N$ timestamps are required for synchronizing all the sensor nodes of a flat N-hop WSN in the case of the original BATS (e.g., refer to Fig. 8 (b) of [1]); for the same network, on the other hand, N is enough for BATS with PHDC as discussed in Section 5.4.1. Also, we could directly establish the time synchronization between the head node and sensor node in BATS with PHDC due to the update procedure of $T1$ on the gateway nodes as illustrated in Fig. 5.2 unlike that between the sensor nodes in BATS's per-hop synchronization strategy. Thanks to the end-to-end time synchronization between the sensor node and the head, the time translation in the multi-hop scenario of BATS with PHDC becomes as straightforward as in the single-hop scenario [1], which could be established between sensor node i and the head as follows:

$$t = \frac{T_i(t) - \theta_i}{1 + \epsilon_i}, \tag{5.23}$$

where $T_i(t)$ and t denote the time of sensor node i and that of the head, and $1+\epsilon_i$ and θ_i are the estimated clock frequency ratio and offset between the two.

5.4.2.2 FTSP

Because the publicly-available implementation of FTSP is incomplete[6][70, 81], we need to remold FTSP to achieve microsecond-level time synchronization accuracy: Considering that the performance of multi-hop extension of FTSP is limited by the accuracy of the calculation in linear regression, which could be affected by many factors such as the

[6]For instance, the time synchronization accuracy of TinyOS public FTSP implementation [80] is limited to milliseconds.

sample size and the precision of floating-point arithmetic. We first employ the MAC-layer timestamping suggested in [1] which is simpler and more prevalent now. We then adopt the specific linear regression method provided in public FTSP implementation [80], this method unlike other methods which do the linear regression directly using the pairs of reference and local timestamps [1], it carries out the linear regression from local timestamps to time offsets, i.e., time differences of reference time and local time.

Afterwards, we extend FTSP for multi-hop time synchronization based on PHDC following Algorithm 3 but with the direction from the head to the gateway and sensor nodes, which could provide microsecond-level time synchronization accuracy.

5.4.2.3 EE-ASCFR

We have investigated the multi-hop extension of EE-ASCFR based on TT in [3], where we identify the issue of precision loss in time synchronization due to the recursive nature of TT and propose AHTS to address it by moving all the time synchronization tasks except timestamping from gateway and sensor nodes to the head with higher computing and power resources. Note that, although AHTS could provide microsecond-level time synchronization accuracy, it is centralized implementation of EE-ASCFR with increased communication overhead.

Thanks to PHDC, now we can extend EE-ASCFR for multi-hop time synchronization while keeping its distributed nature, i.e., carrying out in parallel the synchronization of clock frequency at gateway and sensor nodes and clock offset at the head, respectively. Given the time t of a reference clock (i.e., the hardware clock of the head), we convert the logical clock time $\mathscr{T}_i\Big(T_i(t)\Big)$ of sensor node i based on its hardware clock time $T_i(t)$ presented in [2] as follows: For $t_k<t\leq t_{k+1}$ $(k=0,1,\ldots)$,

$$\mathscr{T}_i\Big(T_i(t)\Big) = \mathscr{T}_i\Big(T_i(t_k)\Big) + R_{i,0}^k \times \Big(T_i(t) - T_i(t_k)\Big) \tag{5.24}$$

where t_k represents the time of a reference clock when a kth synchronization occurs, and $R_{i,0}^k$ is the clock frequency ratio between the head node 0 and sensor node i esti-

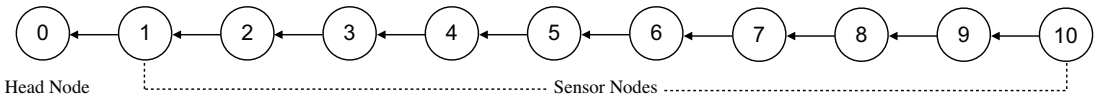


Figure 5.5: 10-hop flat WSN testbed employed in the experiments [5].

mated as $(t_k - t_{k-1})/(T_i(t_k) - T_i(t_{k-1}))$ based on the timestamp pairs from kth and $(k-1)$th synchronization, which is slightly different from and simpler than CR used in [3].

Note that in the case of two-way time synchronization schemes like EE-ASCFR, PHDC is used for timestamps in both directions, i.e., from the head to sensor nodes or vice versa to establish the virtual two-way end-to-end connection between the head and sensor nodes.

5.5 Performance Evaluation

We have extended the three representative WSN time synchronization schemes discussed in Section 5.4.2—i.e., BATS, FTSP, and EE-ASCFR—for multi-hop time synchronization based on PHDC as well as TT; and implemented them on a flat WSN testbed consisting of 11 TelosB motes running TinyOS as shown in Fig. 5.5 for a comparative analysis of their multi-hop time synchronization performance. The TelosB motes in our testbed embed with a 32-kHz crystal oscillator (CO) with a resolution of 30.5 µs and could provide a minimum resolution of 1 µs through running a digitally-controlled oscillator (DCO). Hence the time synchronization accuracy of the evaluated schemes is limited to microseconds, even though it has been shown that the schemes under reverse asymmetric framework could theoretically provide the possibility of sub-microsecond-level time synchronization accuracy [2].

In the following, we set the synchronization interval to 1 s for all the schemes and assume the self-data bundling option for BATS and EE-ASCFR for a fair comparison with FTSP.

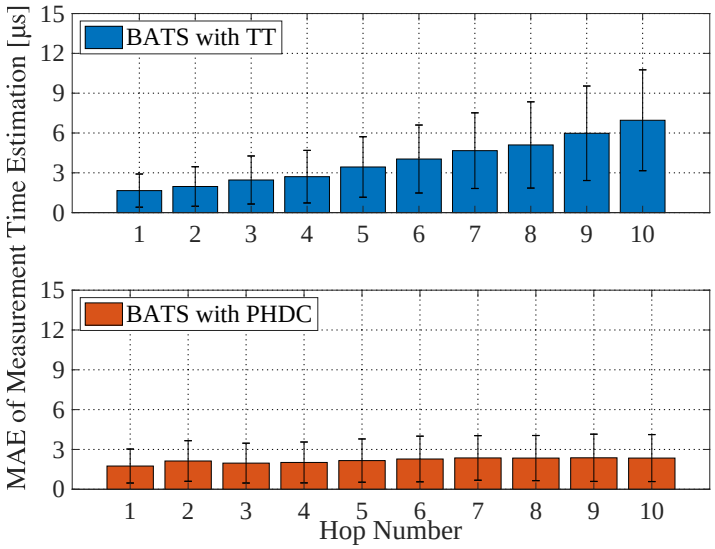


Figure 5.6: MAE of measurement time estimation of BATS with TT and PHDC [5].

5.5.1 BATS with TT and PHDC

Fig. 5.6 shows the MAE of measurement time estimation at each hop of BATS with TT and PHDC with its standard deviation (i.e., the errorbar). BATS with TT has a per-hop cumulative synchronization error of about 0.58 µs, which verifies the results of the analysis in Section 5.3 that the time translation process at gateway nodes of multi-hop extension based on TT could induce cumulative error. In contrast, BATS with PHDC demonstrates that the cumulative synchronization error over ten hops is 0.62 µs, which results in a much smaller per-hop cumulative synchronization error of 0.069 µs; as a result, the MAE of measurement time estimation of the farthest hop is maintained around 2 µs, i.e., much smaller than 7 µs for BATS with TT.

The cumulative distribution functions of absolute measurement time estimation error shown in Fig. 5.7 provide a further evidence. For instance, the 90th-percentile absolute measurement time estimation error of BATS with TT is cumulatively increasing over ten hops from 2.9 µs to 11.4 µs, while that of BATS with PHDC hardly depends on the hop number.

An additional view of the measurement time estimation errors of BATS with PHDC

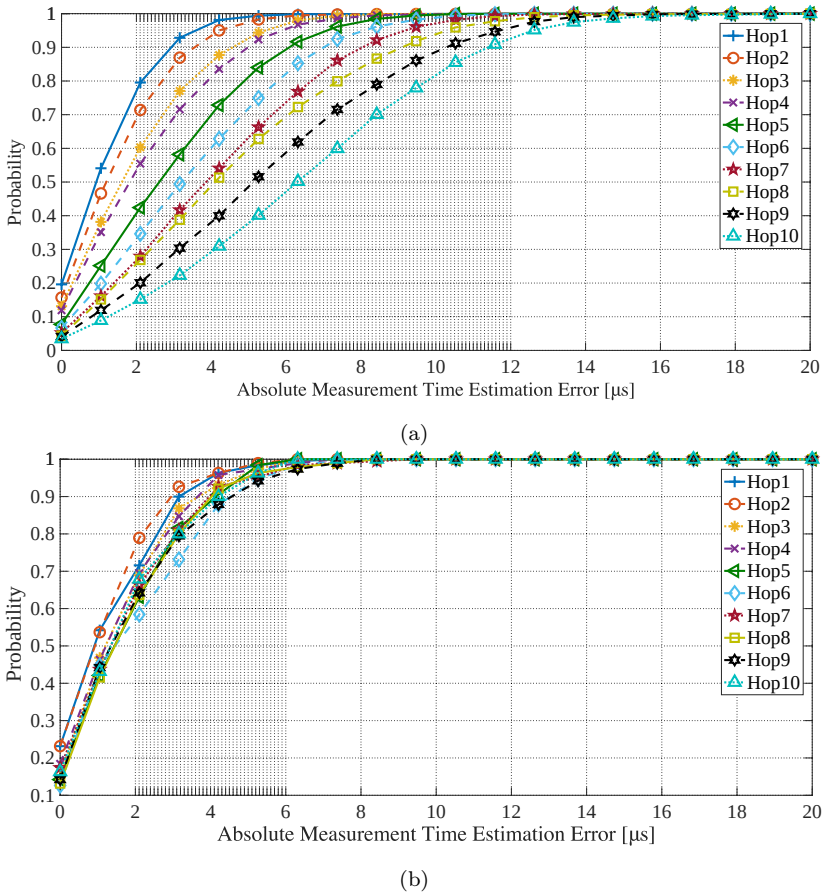


Figure 5.7: Cumulative distribution function of absolute measurement time estimation error for BATS based on (a) TT and (b) PHDC [5].

over time is shown in Fig. 5.8 for a period of 3600 s; all sensor nodes across 10 hops could achieve approximately the same performance—i.e., the fluctuations of the measurement time estimation errors are similar from the first to the last hop.

5.5.2 FTSP with TT and PHDC

To demonstrate the wider applicability of PHDC to and its performance on conventional one-way schemes in addition to the reverse one-way schemes, we take FTSP—i.e., the representative conventional one-way flooding time synchronization scheme—as an example and extend it for multi-hop time synchronization based on both TT and PHDC.

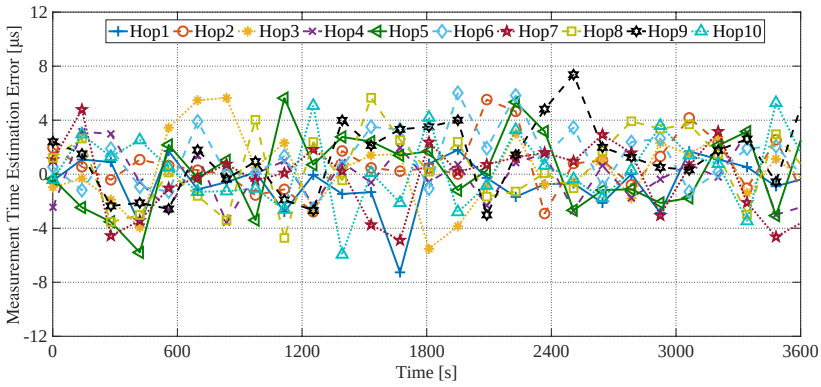


Figure 5.8: Measurement time estimation errors of BATS with PHDC over 3600 s [5].

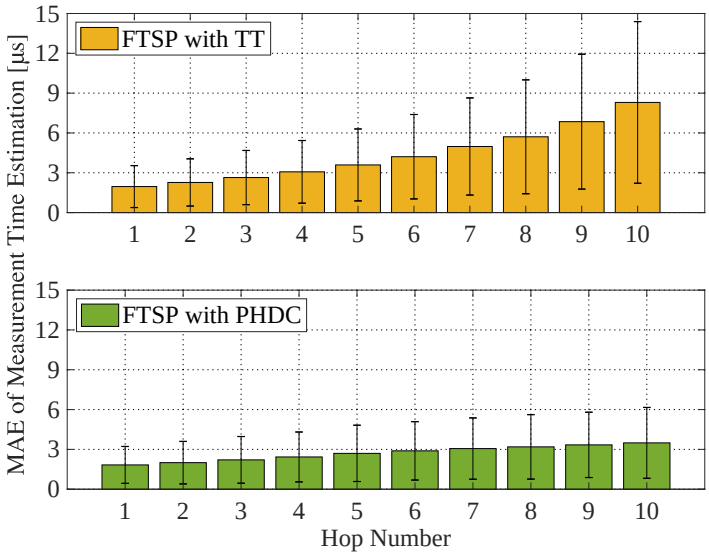


Figure 5.9: MAE of the measurement time estimation of FTSP with TT and PHDC [5].

Fig. 5.9 shows the MAE of measurement time estimation and its standard deviation of FTSP with both TT and PHDC, where we can observe that the MAE of measurement time estimation at hop 10 with TT is more than double that with PHDC; FTSP with PHDC achieves 0.18 µs error per hop over 10 hops, which is more than 70% improvement over 0.7 µs error per hop for FTSP with TT.

The CDFs of absolute measurement time estimation error in Fig. 5.10 illustrate the nature of multi-hop time synchronization performance of FTSP with TT and PHDC in

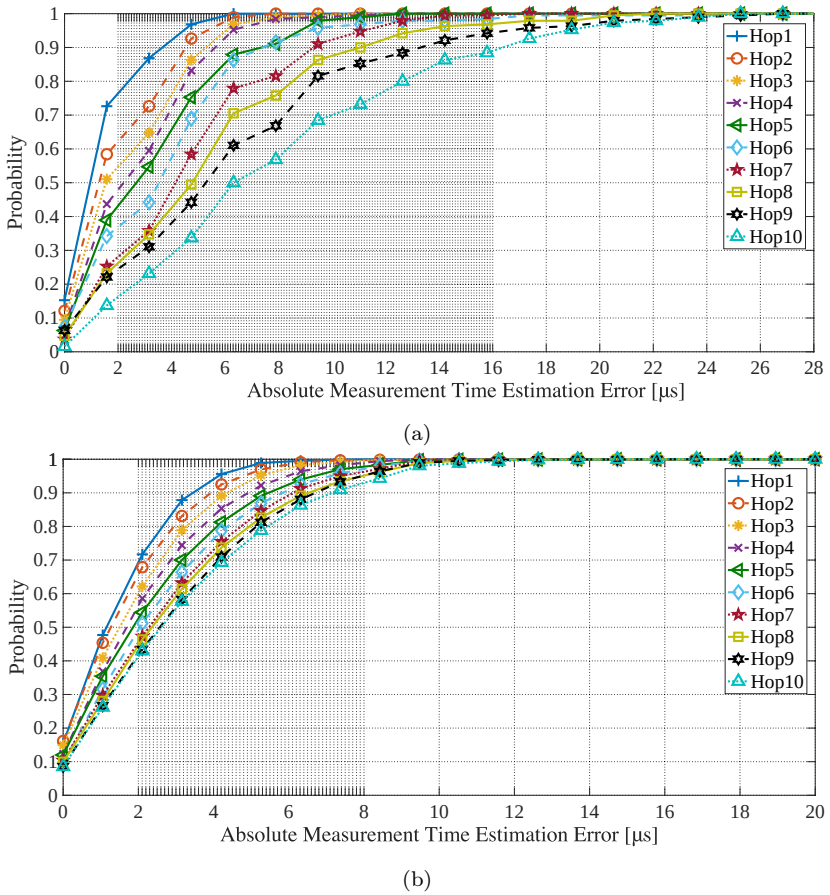


Figure 5.10: Cumulative distribution function of absolute measurement time estimation error for FTSP based on (a) TT and (b) PHDC [5].

a clearer way. FTSP with TT has larger fluctuations in its absolute estimation errors where the maximum value exceeds 20 µs. In comparison, FTSP with PHDC has smaller fluctuations, and the 90%-percentile absolute estimation error at the last hop is smaller than 7.1 µs.

Although the multi-hop time synchronization performance of FTSP could be greatly enhanced by PHDC, it is still not as good as that of BATS with PHDC. This is because, unlike FTSP, the head in BATS receives timestamps from sensor nodes for synchronization due to its reverse asymmetric framework and can increase the sample size and the complexity of the estimation algorithms due to its ample computational and

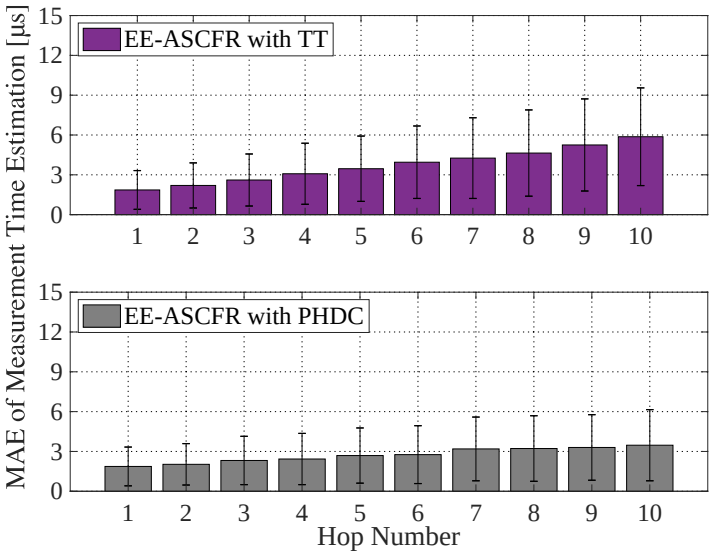


Figure 5.11: MAE of the measurement time estimation of EE-ASCFR with TT and PHDC [5].

power resources [1]. The experimental results in Sections 5.5.1 and 5.5.2 demonstrate that PHDC could effectively alleviate the cumulated synchronization error over multiple gateway nodes for both conventional and reverse one-way time synchronization schemes.

5.5.3 EE-ASCFR with TT and PHDC

In addition to the one-way time synchronization schemes, we also take EE-ASCFR as an example and demonstrate the effectiveness of PHDC on two-way schemes as well. Fig. 5.11 shows the MAE of measurement time estimation and its standard deviation of EE-ASCFR with TT and PHDC where the per-hop cumulative error is more clearly visible for TT, which is also confirmed by its more rapidly increasing standard deviation.

Note that the MAE of measurement time estimation at the tenth hop of EE-ASCFR with TT is 5.87 µs, which is noticeably lower than those of the one-way schemes with TT, i.e., 6.96 µs in BATS with TT and 8.3 µs in FTSP with TT. This is because the reverse two-way message exchange procedure in EE-ASCFR can provide a better estimation of the clock offset due to its two-way nature and the clock skew estimation at each gateway

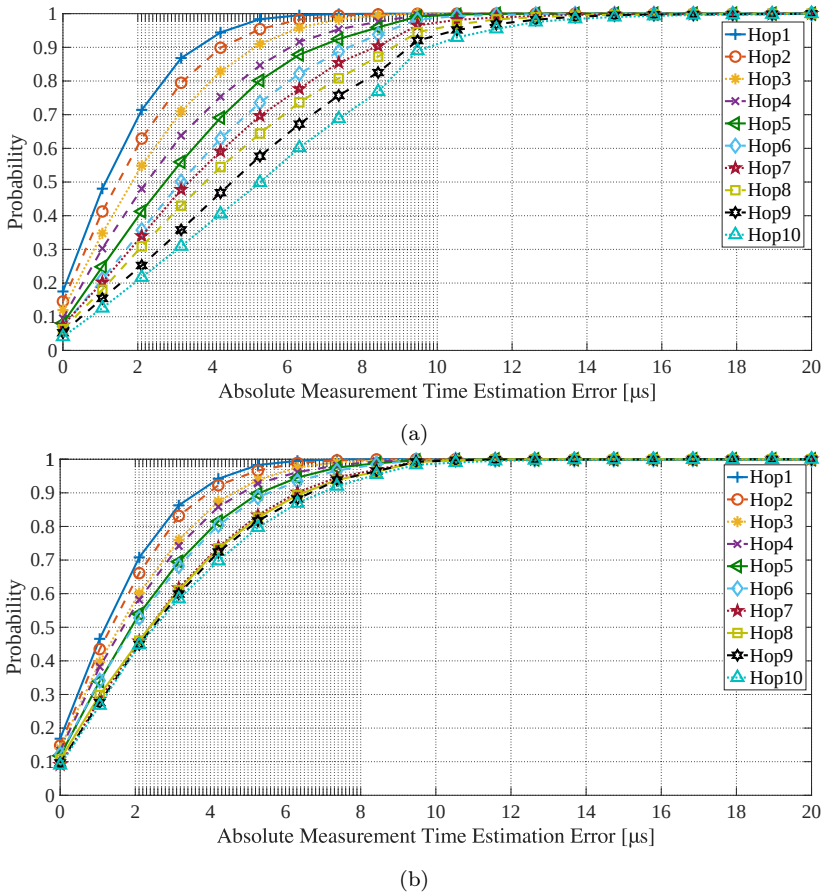


Figure 5.12: Cumulative distribution function of absolute measurement time estimation error for EE-ASCFR based on (a) TT and (b) PHDC [5].

or sensor node is not affected by TT. As for EE-ASCFR with PHDC, it also shows a slight increase in its MAE of measurement time estimation over the hops, which results from the clock skew estimation at each gateway and sensor node affected by PHDC. EE-ASCFR with PHDC, however, still performs 60% better than EE-ASCFR with TT in terms of per-hop error—i.e., 0.18 µs vs 0.45 µs, which interestingly is quite similar to that of FTSP with PHDC discussed in Section 5.5.2.

Like BATS and FTSP with TT, the CDFs of absolute measurement time estimation error of EE-ASCFR with TT in Fig. 5.12 (a) show huge fluctuations, and the maximum absolute measurement time estimation error is close to 14 µs; only 65% of the errors at

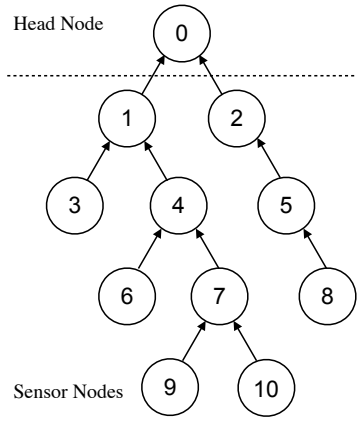


Figure 5.13: 4-hop tree topology employed in the experiments [5].

the tenth hop are distributed within ±7 µs. On the contrary, the CDFs of EE-ASCFR with PHDC in Fig. 5.12 (b) show that even the 90%-percentile at the last hop is less than 7 µs, which again is similar to that of FTSP with PHDC. Compared to BATS with PHDC, EE-ASCFR with PHDC has a slight per-hop error like FTSP with PHDC. This indicates that, as discussed in Section 5.5.1, the centralized clock parameter estimation in BATS has advantages over the distributed estimation of clock skew at gateway and sensor nodes in EE-ASCFR.

5.5.4 Impact of Network Topology on PHDC Performance

We have demonstrated so far the effectiveness of PHDC in multi-hop extension of various time synchronization schemes with a 10-hop flat WSN shown in Fig. 5.5. To investigate the impact of network topology on the performance of PHDC, we set up our testbed with the 4-hop tree topology consisting of 10 sensor nodes and 1 head shown in Fig. 5.13 for further evaluations. Of the three time synchronization schemes considered, we select BATS for the experiments with the 4-hop tree WSN, which provides the best performance for the flat WSN and therefore could better demonstrate the performance of PHDC.

Table 5.1: MAE of Measurement Time Estimation and Its Standard Deviation of BATS-PHDC for the Multi-Hop Tree Scenario [5]

Hop Number		MAE [1]	STD [1]
Hop	1	1.7018E-06	1.3445E-06
	2	1.9617E-06	1.4774E-06
	3	1.9266E-06	1.4781E-06
	4	1.7668E-06	1.3734E-06
	5	1.7664E-06	1.4023E-06
	6	2.0559E-06	1.5960E-06
	7	2.0270E-06	1.5804E-06
	8	1.8606E-06	1.4593E-06
	9	2.0569E-06	1.5954E-06
	10	2.0083E-06	1.5696E-06

[1] Based on the measurement time estimation obtained from 3600 s such that the actual performance in real deployment is represented.

Table 5.1 summarizes the MAE of measurement time estimation and its standard deviation for BATS with PHDC with the 4-hop WSN. From the results, we can observe that the maximum (i.e., that of node 9) and the minimum (i.e., that of node 1) MAE values are kept quite close to each other, i.e., with the difference of 0.3551 µs; a similar observation can be made in regard to the standard deviation of MAE, which shows the difference between the maximum and the minimum values of 0.2515 µs. Interestingly, the MAE of measurement time estimation with the tree topology does not show strict dependency on the hop count unlike that with the linear topology of Fig. 5.5: For instance, sensor node 8, which is 3 hops away from the head, achieves slightly better time synchronization performance than sensor nodes 2 and 3, which are one hop and two hops away from the head. These experimental results demonstrate the effectiveness of PHDC in multi-hop extension with tree topology as well as linear topology.

5.6 Summary

In this chapter, we have proposed a per-hop delay compensation scheme based on the packet-relaying gateways for improving the multi-hop time synchronization performance. We have also systematically analyzed the feasibility of PHDC over single and multiple gateway nodes, especially the effect of timestamping and clock skew compensation including the precision loss in the skew estimation. Based on the analysis, we have extended both reverse and conventional one-way schemes—i.e., BATS and FTSP—and one reverse two-way scheme—i.e., EE-ASCFR—for multi-hop time synchronization based on TT as well as PHDC. Note that unlike the centralized multi-hop extension reported in [3], we have provided in this chapter the distributed multi-hop extension of EE-ASCFR for the first time.

We have demonstrated the effectiveness of PHDC on both one-way and two-way schemes in alleviating the cumulative multi-hop time synchronization error through practical experiments on a real 10-hop WSN testbed. Specifically, BATS with PHDC can achieve nearly flat multi-hop synchronization accuracy; PHDC, compared to TT, also reduces more than 70% and 60% per-hop synchronization errors for FTSP and EE-ASCFR, respectively. From our observation of the results, we have also identified that, besides the multi-hop extension methods like PHDC and TT, there are other factors affecting the per-hop synchronization error in multi-hop time synchronization, including clock parameter estimation methods & sample sizes and the computational capability of the underlying platform (i.e., gateway and sensor nodes or the head). Note that our investigation of the centralized and distributed implementation options suggests that there is a research potential in centralized two-way schemes; while leveraging PHDC, more advanced clock parameter estimation methods could be employed in the centralized two-way time synchronization schemes to achieve better multi-hop time synchronization accuracy.

Chapter 6

An Optimal Message Bundling Scheme with Delay and Synchronization Constraints

In this chapter, we focus on the application of our reverse asymmetric time synchronization schemes for the *message bundling* of WSN; the work presented here is based on our publication of [6]. Message bundling is considered an efficient method for conserving energy consumption. Bundling more messages leads to more energy conservation at the expense of a larger end-to-end delay, the latter of which, however, is a major requirement for monitoring and detection applications based on WSNs. With the thriving of novel reverse asymmetric time synchronization schemes for energy-efficiency, synchronization data are often embedded into the application messages together for bundling. The bundling number, therefore, affects not only the end-to-end delay but also the synchronization accuracy. In this chapter, we propose an optimal message bundling approach for reducing the number of message transmissions while maintaining the required end-to-end delay and time synchronization accuracy. We formulate the optimal bundling problem as an integer linear programming, where an optimal bundling number is computed for each sensor node in the WSN. Extensive experimental results based on a real

WSN testbed consisting of TelosB sensor nodes running TinyOS demonstrate that the proposed optimal bundling approach could significantly reduce the number of message transmissions while achieving the required end-to-end delay and time synchronization accuracy.

6.1 Introduction

Minimizing energy consumption is crucial to practical WSN deployments. As the radio activities consume the majority of the total energy of the typical battery-powered sensor nodes. Therefore, reducing the number of message transmissions is a key to saving energy [2, 82]. In the literature, message bundling[1] is considered a major approach for achieving that [83]; and many research efforts have been centering around the message bundling, e.g., [84, 85, 86].

A major tradeoff of message bundling is the increase of E2E delay as illustrated in Fig. 6.1, which has been put under the spotlight for decades [87, 88, 89]. By E2E delay, we mean the difference between the measurement time of a message at an originating sensor node and the reception time of the resulting measurement data by the head node. Many research have taken into account the ensemble of energy consumption and E2E delay for message bundling [86, 90]; however, the synchronization accuracy has hardly been considered together prior to the advent of reverse asymmetric time synchronization schemes [2, 3, 1]. Thanks to the reverse asymmetric time synchronization, the energy-efficiency, E2E delay, and synchronization accuracy could be jointly tuned as a whole. This is often required by a group of long-term monitoring and detection WSN applications such as human activity and gesture recognition based on novel device-free wireless sensing [38, 39].

In this chapter, based on the pioneer of reverse asymmetric time synchronization schemes—i.e., EE-ASCFR [2] & AHTS [3]—whose time synchronization is done at the head node and the corresponding synchronization data are bundled together with mea-

[1]The terms of "message bundling" and "data bundling" are used interchangeably in this chapter.

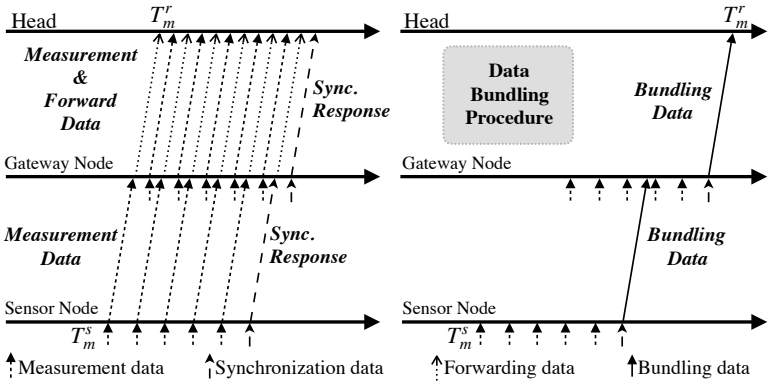


Figure 6.1: Comparison of the message transmissions of the time synchronization schemes with and without data bundling procedure [6].

surements into a bundled message at a sensor node, we propose a novel optimal message bundling scheme which formulates the aforementioned bundling optimization problem as integer linear programming (ILP) with jointly satisfying the user-defined performance requirements on E2E delay and time synchronization accuracy.

The rest of this chapter is organized as follows: Section 6.2 provides the preliminaries, where we introduce the time synchronization scheme based on the data bundling procedure and discuss the related conflicts of performance metrics. Section 6.3 presents the details of our proposed optimal bundling with delay and synchronization constraints and its formulation as ILP. Section 6.4 exhibits our system design at both head and sensor nodes. Section 6.5 demonstrates the performance of the proposed approach through experimental results on a real WSN testbed. Section 6.6 concludes our work and discusses future works.

6.2 Preliminaries

In this section, we present relevant preliminaries of the proposed optimal bundling approach from the foundation of the proposed approach—asymmetric time synchronization—to the conflicts of performance metrics.

Table 6.1: MAE and MSE of Measurement Time Estimation of EE-ASCFR and AHTS with Different SIs Provided in [2] and [3] Described in [6]

Synchronization Scheme		MAE [1]	MSE [2]
EE-ASCFR	SI = 100 s	5.8990E-19	8.8811E-25
	SI = 1 s	5.4210E-19	9.1748E-25
	SI = 10 ms	4.7684E-19	1.0887E-24
AHTS	SI = 100 s	8.4225E-06	1.2524E-10
	SI = 10 s	2.3385E-06	9.1694E-12
	SI = 1 s	1.8166E-06	5.2094E-12

[1] MAE is the mean absolute error of measurement time estimation.
[2] MSE is the mean square error of measurement time estimation.

6.2.1 Energy-Efficient Time Synchronization Schemes Using Data Bundling

Many time synchronization schemes are proposed for achieving the ensemble of energy efficiency and synchronization accuracy such as [2, 31, 40, 41]. Among those, EE-ASCFR and AHTS proposed in [2, 3] thanks to both their reverse synchronization nature and application of data bundling, particularly suit the proposed optimal bundling scheme where E2E delay and synchronization accuracy are maintained at the head.

We omit the synchronization details here and focus on the impact of message bundling on its synchronization which is the foundation of the proposed scheme. As illustrated in Table 6.1 showing the simulation results of EE-ASCFR and practical evaluation results of AHTS, the synchronization accuracy (SA) is heavily affected by synchronization interval. For achieving the desired synchronization accuracy, a proper SI should be maintained. Therefore, the relationship ($\mathscr{R}$) between SI and SA could be represented as follows:

$$SI = \mathscr{R}(SA), \tag{6.1}$$

where the requirement of SA could be translated to the requirement of SI. Of particular note is that, the bundling number is not discussed either in EE-ASCFR or AHTS since it directly affects the E2E delay and synchronization accuracy, which makes its value selection very complicated.

6.2.2 Conflicts of Performance Metrics: Energy Efficiency, E2E Delay and Synchronization Accuracy

Achieving simultaneous high energy-efficiency, low E2E delay, and high synchronization accuracy is barely possible. Bundling more messages could lead to less energy consumption caused by message transmission, the E2E delay of the messages is however increased, which results in the *conflict between energy efficiency and E2E delay calculation*. Specific to the delay calculation illustrated in Fig. 6.1, the E2E delay—i.e., T_m^r - T_m^s—of measurement m, would be relatively small (e.g., multiples of forwarding delay which typically in milliseconds) in the regular direct forwarding method. However, due to the bundling procedure in the intermediate sensor nodes, the E2E delay could be as large as multiples of measurement interval. As for synchronization accuracy, SI should be shorter for achieving higher accuracy [2], but shorter SI results in more message transmissions, which again leads to higher energy consumption. This draws forth the *conflict between energy efficiency and synchronization accuracy*.

6.3 ILP Model for Optimal Bundling Problem

For node i in a WSN consisting of N sensor nodes, its energy consumption e_i^t for the message transmissions could be modeled as follows: For $i \in [0, 1, \ldots, N-1]$,

$$e_i^t = a_i e_i^m + b_i e_i^s + c_i e_i^f, \tag{6.2}$$

where e_i^m denotes the energy consumption for transmission of the measurement message and e_i^s that of the synchronization message at sensor node i, and e_i^f stands for the energy consumption for forwarding either the synchronization or measurement message from offspring sensor nodes at sensor node i. a_i, b_i, and c_i are coefficients of the number of transmissions for corresponding messages. When the data bundling is considered, (6.2)

can be updated as follows:

$$e_i^t = \begin{cases} \delta_i e_i^b, & \text{all data bundling} \\ \delta_i e_i^b + c_i e_i^f, & \text{self data bundling} \end{cases} \tag{6.3}$$

where $\delta_i e_i^b$ is the total energy consumption for transmitting bundling messages. Note that, the self data bundling option bundles only the data generated from the sensor node itself while the all data bundling option bundles all data from both the sensor node itself and its offspring sensor nodes into one bundling message. Therefore, the term $c_i e_i^f$ still remains in the case of the former of which.

From a network-level perspective, the total energy consumption for message transmissions in the network could be described as follows:

$$\mathbf{E}^t = \sum_{i=0}^{N-1} e_i^t. \tag{6.4}$$

6.3.1 Maximization of Bundling Number for Energy Efficiency

From (6.3), a larger number of bundled message transmissions (i.e., δ_i) leads to more energy consumption; in contrast, bundling more messages reduces the said number therefore lessens energy consumption. Let Γ^i be the number of bundled messages at sensor node i, a larger Γ^i results in a smaller δ_i, which reduces the total energy consumption for message transmissions. We can minimize $\mathbf{E}^t$ by maximizing Γ^i in a certain range due to the increasing E2E delay and payload size, however. The maximization of bundling number for energy efficiency can be formulated as follows:

$$\begin{aligned} &\textbf{maximize } \boldsymbol{\Gamma} \\ &\textbf{subject to } \chi^{min} \leq \Gamma^i \leq \chi^{max}, \forall i \in [0, \ldots, N-1], \end{aligned} \tag{6.5}$$

where

$$\boldsymbol{\Gamma} = \sum_{i=0}^{N-1} \Gamma^i.$$

$\mathbf{\Gamma}$ is the total bundling number in the network, χ^{min} and χ^{max} denote the lower and upper bounds of the measurement bundling number which are user-defined application-specific parameters.

6.3.2 Constraining E2E Delay

Based on the illustration in Fig. 6.1, the E2E delay of sensor node i could be defined as the time difference between the transmission time of the measurement m at the sensor node i and the reception time of that at the head:

$$\mathbf{D}^{i}_{e2e} \triangleq \sum_{l=0}^{L-1} D^{i}_{l} = T^{i,r}_{m} - T^{i,s}_{m}, \tag{6.6}$$

where D^{i}_{l} is the link delay at link l of L links from sensor node i to the head, $T^{i,r}_{m}$ and $T^{i,s}_{m}$ are the receiving time at the head and the measuring time at the sensor node respectively, the latter of which is a time with respect to the reference clock at the head translated by a time synchronization scheme. Moreover, the $\mathbf{D}^{i}_{e2e}$ is a path-level delay that consists of several link-level delays in the network. Considering the bundling procedure, the link delay at link l for sensor node i could be described as follows:

$$D^{i}_{l} = D^{i,l}_{prop} + D^{i,l}_{serv} + D^{i,l}_{bund}, \tag{6.7}$$

where $D^{i,l}_{prop}$ denotes the propagation delay that is typically in nanosecond level in WSN; $D^{i,l}_{bund}$ is the delay caused by the bundling procedure, which is in multiples of the measurement interval (e.g., 5×1 s for the measurement interval of 1 s and the bundling number of 5). According to the service time model [91] for TinyOS which running on the TelosB sensor node, we can model $D^{i,l}_{serv}$ as follow:

$$D^{i,l}_{serv} = \begin{cases} D^{i,l}_{SPI} + D^{i,l}_{succ} + (N^{i,l}_{try} - 1) \cdot D^{i,l}_{retry}, & N^{i,l}_{try} \leq N^{i,l}_{max} \\ D^{i,l}_{SPI} + D^{i,l}_{fail} + (N^{i,l}_{max} - 1) \cdot D^{i,l}_{retry}, & N^{i,l}_{try} > N^{i,l}_{max} \end{cases} \tag{6.8}$$

where

$$D^{i,l}_{succ} = D^{i,l}_{MAC} + D^{i,l}_{frame} + D^{i,l}_{ACK},$$
$$D^{i,l}_{fail} = D^{i,l}_{MAC} + D^{i,l}_{frame} + D^{i,l}_{waitACK},$$
$$D^{i,l}_{retry} = T^{i,l}_{retry} + D^{i,l}_{frame} + D^{i,l}_{waitACK}.$$

The delay parameters—i.e., one-time serial-peripheral interface (SPI) bus loading delay $D^{i,l}_{spi}$, MAC layer delay $D^{i,l}_{MAC}$, frame transmission delay $D^{i,l}_{frame}$, acknowledgment (ACK) transmission delay $D^{i,l}_{ACK}$ and ACK waiting delay $D^{i,l}_{waitACK}$—in the above equations are platform-dependent values, and their values are typically in the order of milliseconds. Besides, $N^{i,l}_{try}$ and $N^{i,l}_{max}$ are the current and the maximum allowed number of transmissions for a successful delivery, and $T^{i,l}_{retry}$ is the user-defined backup time of retransmission. For simplicity and energy efficiency, the packet retransmission is not taken into account in our proposed scheme since there are usually not many packet retransmissions in the network with lower traffic. Therefore, we can simplify (6.8) as follows:

$$D^{i,l}_{serv} = D^{i,l}_{SPI} + D^{i,l}_{MAC} + D^{i,l}_{frame} + D^{i,l}_{ACK} \tag{6.9}$$

where the value of $D^{i,l}_{serv}$ is around 10 ms based on the reference values in [91].

The link delay in (6.7), on the other hand, could be further simplified when the measurement interval for an application ($\mathbf{I}^{i,l}_{meas}$) is much larger than the service delay ($D^{i,l}_{serv}$): Because $D^{i,l}_{bund} > \mathbf{I}^{i,l}_{meas}$, $\mathbf{I}^{i,l}_{meas} \gg D^{i,l}_{serv}$ also implies $D^{i,l}_{bund} \gg D^{i,l}_{serv}$, hence

$$D^{i,l}_{link} \approx D^{i,l}_{bund}, \quad \text{if } \mathbf{I}^{i,l}_{meas} \gg D^{i,l}_{serv}. \tag{6.10}$$

Here the service delay $D^{i,l}_{serv}$ should not be ignored in case the application requires frequent measurements, i.e., $\mathbf{I}^{i,l}_{meas}$ is comparable to $D^{i,l}_{serv}$. As sensor nodes located in different layers have to serve different numbers of offspring sensor nodes thus face different amounts of traffics. Specifically, the gateway node in the upper layer has to handle the message traffic from its offspring sensor nodes as well as itself; the higher

layer it is located in, the more message traffic it has to handle. Therefore, even two sensor nodes with the same bundling number (i.e., Γ^i) could have different bundling delays due to the variance in their message traffics. With introducing a message traffic coefficient ($\frac{1}{1+\lambda^i}$) for each sensor node that periodically measures data with the same measurement interval (i.e., $\mathbf{I}^i_{meas}$), the bundling delay at sensor node i (D^i_{bund}) could be represented as follows:

$$D^i_{bund} = \frac{\Gamma^i}{1+\lambda^i} \cdot \mathbf{I}^i_{meas}, \tag{6.11}$$

where λ^i denotes the number of offspring sensor nodes. Then the D^i_{e2e} for the applications with normal measurement interval could be modeled as follows:

$$\mathbf{D}^i_{e2e} = \sum_{l=0}^{L-1} D^i_{bund}. \tag{6.12}$$

With above (6.12), we can constrain the E2E delay in the optimal bundling problem in (6.5) with the user-defined E2E delay requirement ($\mathbf{D}^{max}_{e2e}$): For $i \in [0, 1, \ldots, N-1]$,

$$\mathbf{D}^i_{e2e} \leq \mathbf{D}^{max}_{e2e}. \tag{6.13}$$

6.3.3 Constraining Synchronization Accuracy

As the proposed optimal bundling approach lays its foundation on the reverse asymmetric time synchronization, the synchronization accuracy thus depends on the report interval of the synchronization message, the latter of which is carried by the bundling message, however. In such a case, the user-required synchronization accuracy ($\mathbf{SA}^{min}$) could be maintained through constraining the E2E delay of the bundling message when sensor nodes generate measurement data periodically. Based on the empirical sets—i.e., $\mathscr{R}$—of the relationships between synchronization accuracy and SI previously provided in Section 6.2, the user-required synchronization accuracy could be translated to the delay requirement as follows:

$$\mathbf{D}^{SA}_{e2e} = \mathscr{R}(\mathbf{SA}^{min}). \tag{6.14}$$

By combining (6.12) and (6.14), the synchronization accuracy could be achieved through constraining the E2E delay as follows: For $i \in [0, 1, \ldots, N-1]$,

$$\mathbf{D}_{e2e}^{i} \leq \mathbf{D}_{e2e}^{SA}. \tag{6.15}$$

6.3.4 ILP model

Considering the objective function (6.5) and the two constraint sets (6.13) and (6.15), we can formulate the optimal bundling problem as the following ILP:

$$\begin{aligned} &\textbf{maximize } \boldsymbol{\Gamma} = \sum_{i=0}^{N-1} \Gamma^{i} \\ &\textbf{subject to} \\ &\quad \chi^{min} \leq \Gamma^{i} \leq \chi^{max}, \ \forall i \in [0, \ldots, N-1], \\ &\quad \mathbf{D}_{e2e}^{i} \leq \min\left(\mathbf{D}_{e2e}^{max}, \mathscr{R}(\mathbf{SA}^{min})\right), \ \forall i \in [0, \ldots, N-1], \end{aligned} \tag{6.16}$$

where the E2E delays of all sensor nodes are jointly constrained by the user-defined E2E delay $\mathbf{D}_{e2e}^{max}$ and the synchronization accuracy $\mathbf{SA}^{min}$ requirements, and the bundling number of each sensor node is constrained by the user-defined lower χ^{min} and upper χ^{max} bounds, respectively. Applying the set of optimal bundling numbers computed from this ILP model to sensor nodes, the required E2E delay and synchronization accuracy could be maintained while the bundled message transmissions are minimized.

6.4 System Design

We show the system architecture of the proposed optimal bundling based on the ILP model formulated in Section 6.3.4 in Fig. 6.2. Two subsystems—i.e., the *performance maintainer* at the head and the *parameter adapter* at each sensor node—are built to achieve the optimization target. In the following, we narrate our system separately.

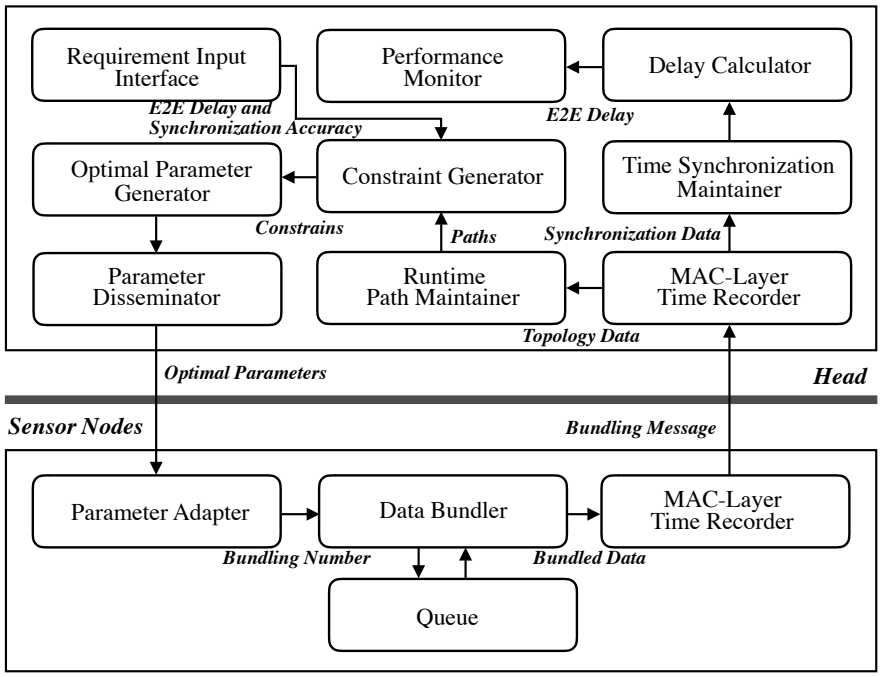


Figure 6.2: System architecture of the proposed optimal bundling [6].

6.4.1 Performance Maintainer at Head

As the proposed optimal bundling system adopts the reverse asymmetric time synchronization schemes of EE-ASCFR and AHTS, the centralized nature is inherited, i.e., the head undertakes both time synchronization and optimization. Note that time synchronization is achieved through the *MAC-layer Time Recorder* and the *Time Synchronization Maintainer*, the latter of which translates timestamps between the head and a sensor node. A measurement timestamp recorded at the sensor node—i.e., T_m^s in Fig. 6.1—could be translated to a timestamp based on the hardware clock of the head.

The E2E delay is obtained as a difference between the translated measurement timestamp and the receiving timestamp of T_m^r through the *Delay Calculator*, and the runtime E2E delay is monitored through the *Performance Monitor*. According to the topology data carried in each bundled message, the routing paths for all sensor nodes are further recovered in the *Runtime Path Maintainer*, and one set of paths is generated and

delivered to the *Constraint Generator*. The user-defined requirements of E2E delay and synchronization accuracy, on the other hand, are captured through the user interface of *Requirement Input Interface*. With combining the performance requirements and the current path information, the *Constraint Generator* generates a set of constraints and passes it to the *Optimal Parameter Generator*, where the optimal bundling number for each sensor node is calculated as a solution of the ILP model. Eventually, the optimal bundling numbers from the *Optimal Parameter Generator* are delivered to sensor nodes by the *Parameter Disseminator*.

Note that, when the topology is updated or the user-defined requirements are changed, the system will repeat the above procedures to provide up-to-date optimal bundling numbers. By the way, as the proposed system is in a centralized manner, the major trade-off is the communication overhead for disseminating the optimal results to the sensor nodes. This overhead would not be a major issue in practice unless topology updates and user requirements changes are too frequent.

6.4.2 Parameter Adapter at Sensor Nodes

Three lightweight components (including the *MAC-layer Time Recorder* from the time synchronization) are implemented at sensor nodes. Specifically, the *Parameter Adapter* receives the optimal bundling number and delivers it to the *Data Bundler* that bundles the measurement data temporarily stored in the *Queue* plus timestamps into one message. The bundled message will be timestamped for $T3$ by the *MAC-layer Time Recorder* as shown in Fig. 3.1 and Fig. 6.2.

6.5 Experimental Results

We implement the proposed optimal bundling approach on a real three-hop WSN testbed consisting of five TelosB sensor nodes as illustrated in Fig. 6.3. By employing a reference node—similar to the approach of reference-broadcast synchronization (RBS) [28]—as illustrated in Fig. 6.4, we could verify the time synchronization accuracy of

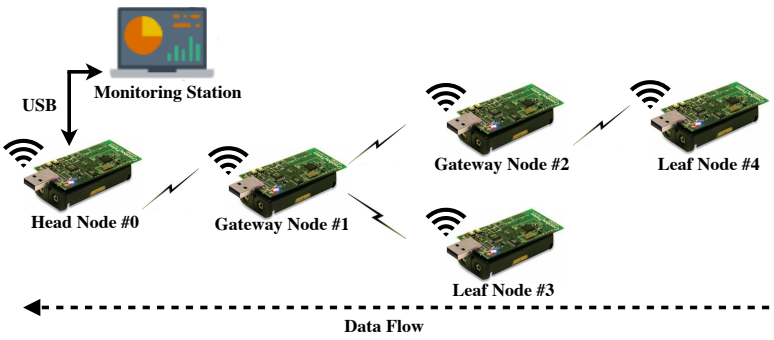


Figure 6.3: Experiment setup of a real three-hop WSN testbed consisting of five TelosB sensor nodes [6].

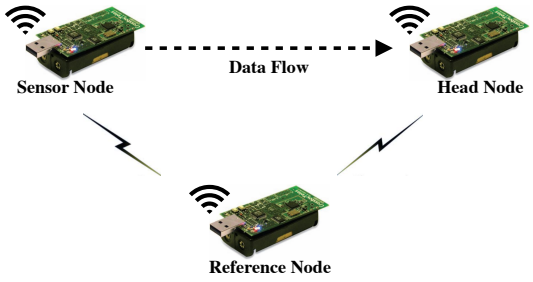


Figure 6.4: Evaluation of time synchronization accuracy using one additional reference node [6].

the time synchronization approach embedded in the proposed optimal bundling. Note that, when neglecting the propagation delay, those receiving timestamps should be at the same time, thanks to the leveraging of the MAC-layer timestamping [1]. Therefore, the synchronization errors of each sensor node under evaluation could be computed in the head through comparing the differences of the head's recorded timestamp and the translated one using time synchronization.

In our experiments, each sensor node generates one measurement per second. The E2E delays of the latest measurements in bundled messages from all sensor nodes are collected and stored in a time sequence in order of their arrivals at the head. We show the E2E delay performance in Fig. 6.5, Fig. 6.6 and Fig. 6.7. The red horizontal dotted lines indicate the E2E delay requirements for corresponding time periods, and the requirement of synchronization accuracy is set to 5 µs for all experiments.

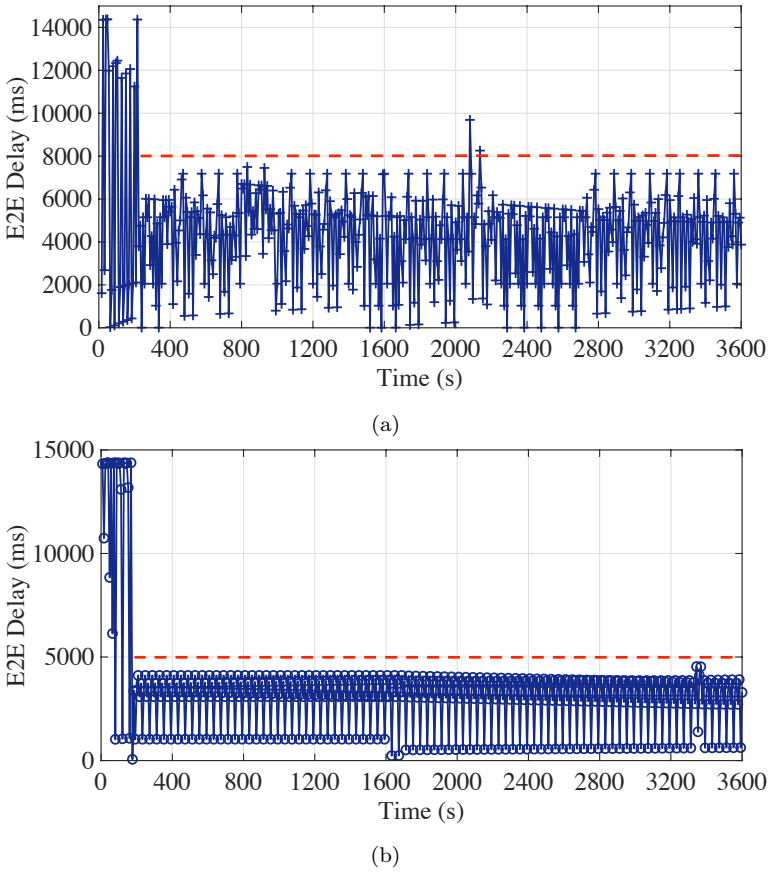


Figure 6.5: E2E delay performance of the optimal bundling under static E2E delay requirement setting of (a) 8 s and (b) 5 s with the maximum bundling number of 15 [6].

6.5.1 Delay Performance Under Static Requirement Setting

We first evaluate the E2E delay performance of the optimal bundling under static requirement setting—e.g., E2E delay requirement of 8 s and 5 s—with different maximum bundling numbers—i.e., χ^{max} in (6.16)—of 15 and 10. We run the experiment for 3600 s to demonstrate the *long-term maintenance capability*.

We demonstrate in Fig. 6.5 the performance of the proposed approach under static E2E delay requirement setting through a group of experiments employing different parameters and requirements. Fig. 6.5 (a) and (b) show that the E2E delays can exceed 14 s before the optimal bundling is applied. However, once the optimal bundling is

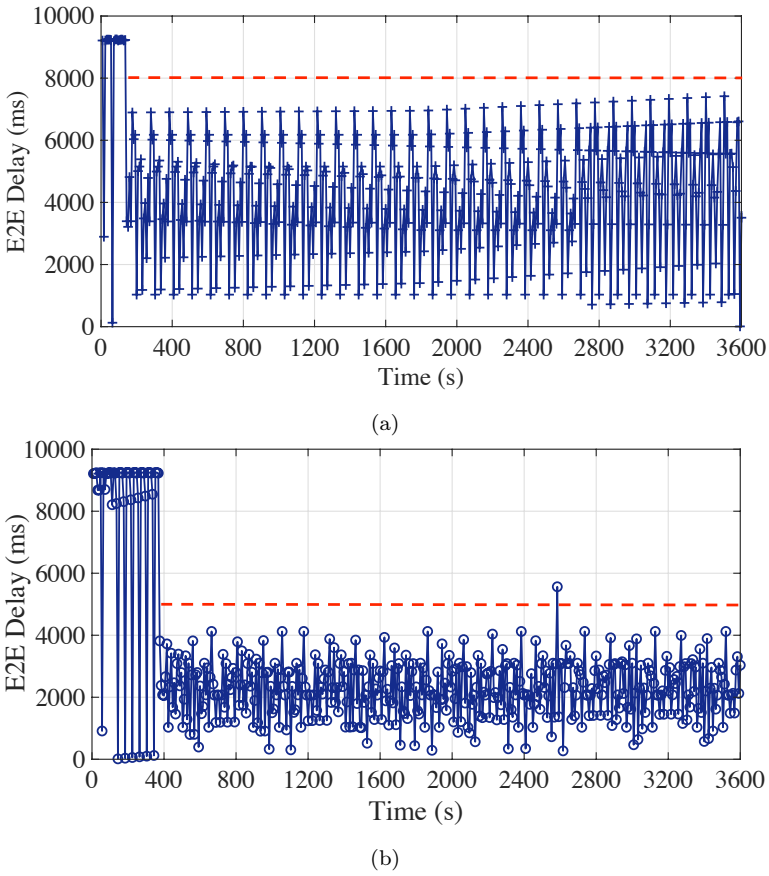


Figure 6.6: E2E delay performance of the optimal bundling under static E2E delay requirement setting of (a) 8 s and (b) 5 s with the maximum bundling number of 10 [6].

applied with the requirement of 8 s and 5 s, we can see that the E2E delay is controlled and kept under the requirements for most of the time. Note that there are few data points that crossed the requirement line; because the gateway node serves its own measurement data first, the data from its offspring nodes, sometimes, could be buffered in the queue and sent by the next available message, which would increase the E2E delay of the corresponding message. We further change the maximum bundling number from 15 to 10 to evaluate the proposed optimal bundling approach since the maximum bundling number is often limited by the measurement data length and the maximum payload size of the underlying protocol. Fig. 6.6 (a) and (b) illustrate that the E2E delay requirement can be well fulfilled with applying the proposed optimal bundling

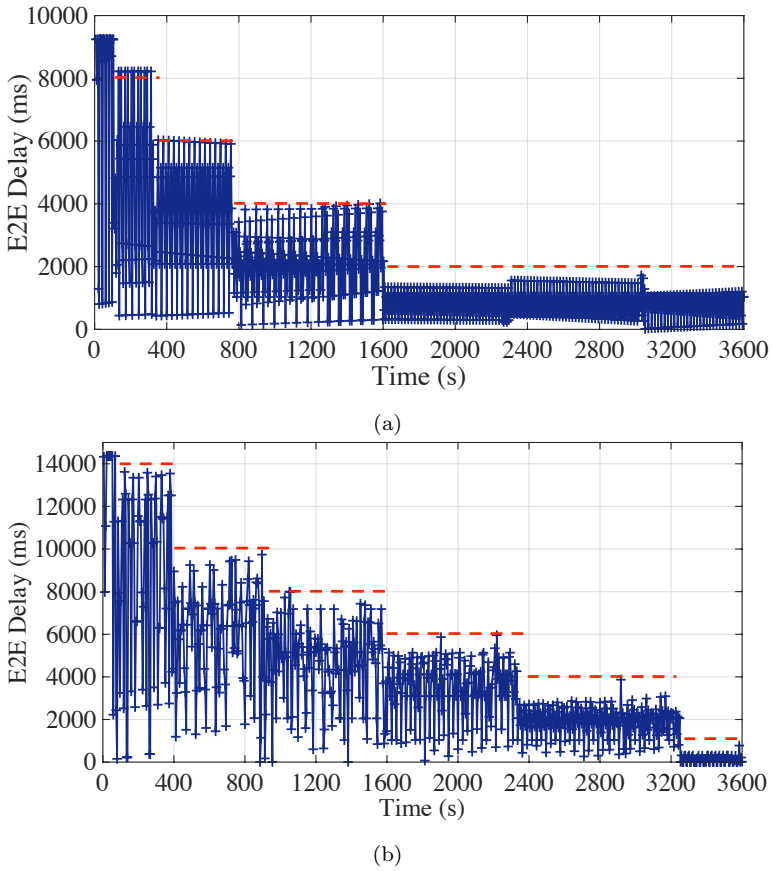


Figure 6.7: E2E delay performance of the optimal bundling under dynamic E2E delay requirement setting and the maximum bundling number of (a) 10 and (b) 15 [6].

approach.

6.5.2 Delay Performance Under Dynamic Requirement Setting

We next evaluate the E2E delay performance of the optimal bundling under dynamic requirement setting with the maximum bundling number of 10 and 15 to demonstrate its *run-time maintenance capability*.

During the evaluation with the maximum bundling number of 10, the E2E delay requirement is dynamically changed from 8 s to 2 s step-by-step as illustrated in Fig. 6.7 (a). For the evaluation with the maximum bundling number of 15 shown in Fig. 6.7 (b), we extend the range of E2E delay requirement to 14 s to 1 s. Experimental

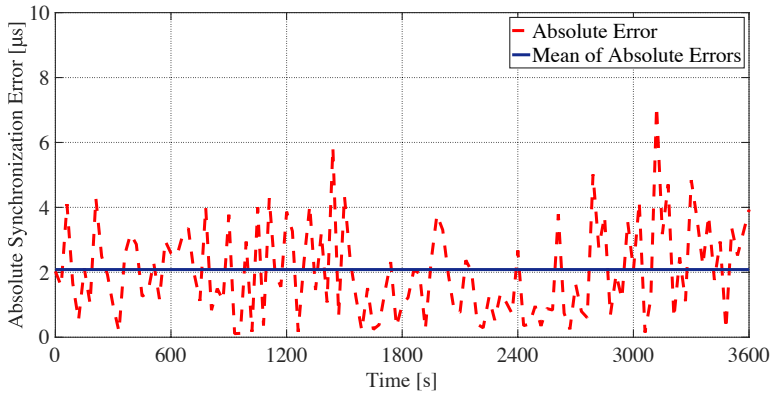


Figure 6.8: Absolute time synchronization error of node 2 during the experiment of static requirement setting with maximum bundling number of 10 and E2E delay requirement of 8 s illustrated in Fig. 6.6 (a) [6].

results exhibit that the proposed optimal bundling could well handle multiple dynamic requirements of E2E delay throughout the experiments. As discussed in Section 6.5.1, however, some data points slightly cross the requirement line, which is due to the neglect of the service time (i.e., $D_{serv}^{i,l}$) in equation (6.7). When the optimal bundling numbers are too strict that just fulfill the requirement, the influence of the neglect of the small service time would be notable. Note that, in the extreme case, when the E2E delay is strict to 1 s as shown in Fig. 6.7 (b), which reaches the measurement interval of our experiment as previously described in Section 6.5, the proposed optimal bundling could still fulfill the requirement but with assigning no larger optimal bundling number than 1 to all sensor nodes as shown in Table. 6.2. Nevertheless, when the E2E delay requirement is much stricter, i.e., smaller than the measurement interval, which may lead to an unsatisfactory situation since there is no room for the proposed optimal bundling to play.

6.5.3 Synchronization Accuracy

In the reverse asymmetric time synchronization schemes of EE-ASCFR and AHTS, the synchronization accuracy could be fulfilled as far as the E2E delay of the bundled message could satisfy the synchronization interval: The synchronization interval—i.e., E2E

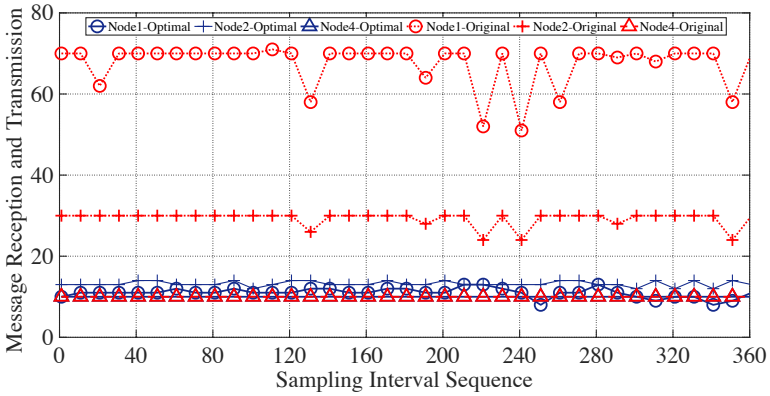


Figure 6.9: Number of message receptions and transmissions of sensor nodes with and without optimal message bundling for 3600 s with sampling interval of 10 s [6].

delay—demonstrated in the evaluation results shown in Fig. 6.5, Fig. 6.6 and Fig. 6.7, could overfulfill the SI requirement (e.g., 10 s SI could lead to 2.3385 µs synchronization accuracy as illustrated in Table. 6.1), the synchronization accuracy, therefore, could be strictly followed.

We take node 2 in the static requirement setting experiment—i.e., experiment shown in Fig. 6.6 (a) with maximum bundling number of 10 and E2E delay requirement of 8 s—as an example, to evaluate the time synchronization accuracy. As illustrated in Fig. 6.8, most of the absolute synchronization errors (i.e., the dotted line) are under 5 µs, and the mean of absolute synchronization errors is 2.0849 µs, which overfulfills the 5 µs synchronization accuracy requirement.

6.5.4 Energy Efficiency

We indirectly estimate the energy efficiency by comparing the number of message receptions and transmissions with and without optimal bundling. In the experiment exhibited in Fig. 6.5 (a), we take the path of 0↔1↔2↔4 as an example, and show the number of message receptions and transmissions of the sensor nodes $1, 2, 4$ in Fig. 6.9.

As illustrated in Fig. 6.9, we count the number of message receptions and transmissions of each sensor node every 10 s over the period of 3600 s. In the case of using

Table 6.2: Optimal bundling number under different E2E delay requirements during the dynamic experiment as shown in Fig. 6.7 (b) [6]

Max Bundling Number	E2E Delay Requirement (s)	Optimal Bundling Number			
		Node 1	Node 2	Node 3	Node 4
15	14	15	14	8	1
15	10	15	10	6	1
15	8	15	6	4	1
15	6	15	2	2	1
15	4	14	1	1	1
15	1	1	1	1	1

the proposed optimal bundling, all sensor nodes could maintain their message receptions and transmissions around the number of 10. In contrast, when without the optimal bundling, the numbers of their message receptions and transmissions are relatively large, whose average of 35 is over triple of that with the optimal bundling. In other words, around 70% message transmissions can be reduced using the proposed optimal bundling in this specific case. Note that, with the increase of the network size and the maximum bundling number, the performance of the proposed optimal bundling would be even better.

6.5.5 Discussions

In Table. 6.2, the optimal bundling numbers that our proposed approach assign to the leaf nodes (i.e., node 3 and 4 in our experiment as shown in Fig. 6.3) are relatively small regardless of the variation of the E2E delay requirements. This is because the proposed optimal bundling treats both the gateway nodes and leaf nodes equally since they are most likely the same resource-constrained sensor nodes in the real deployment and the gateway nodes undertake more traffics than the leaf ones. However, when the gateway node is rather resource-abundant, the proposed approach should incline the optimization to the leaf nodes.

In addition, as briefly mentioned in Section 6.5.4, the maximum energy efficiency performance of the proposed optimal bundling is not fully exhibited due to the joint

constraints of the network size and various parameters such as maximum bundling number. In the follow-up work, a further study will employ a relatively larger testbed and proper parameters to evaluate the extreme performance of the proposed optimal bundling.

6.6 Summary

In this chapter, we have proposed an approach to optimize the number of bundled messages at sensor nodes in WSNs under the constraints of time synchronization accuracy and E2E delay. We formulate the optimal bundling problem as an ILP model and employ the novel asymmetric time synchronization scheme. To the best of the authors' knowledge, this is the first work to optimize message bundling for energy efficiency under the joint constraints of synchronization accuracy and E2E delay in the context of WSN. The practical evaluation results on a real testbed composed of TelosB sensor nodes demonstrate the long-term and runtime maintenance capability of the proposed approach in terms of E2E delay. We also exhibit the energy efficiency of the proposed approach through a case study, which shows about 70% reduction on the overall message transmissions.

Note that bundling a larger number of messages could result in a longer payload, which may lead to possible link degradation such as the packet reception ratio degradation. In this regard, link quality requirements could be introduced to the optimization model as additional constraints to take into account more impacts on the overall performance by the bundling procedure. In addition, the proposed approach should be tested with a larger testbed to investigate its scalability and network-wide energy efficiency.

Chapter 7

NISA: Node Identification and Spoofing Attack Detection Based on Clock Features and Radio Information for Wireless Sensor Networks

In this chapter, we focus on the application of our reverse asymmetric time synchronization schemes for the *node identification* of WSN; the work presented here is based on our publication of [7]. We propose a new node identification scheme called *node identification against Spoofing attack* with novelly taking clock features and radio information together into account for simultaneous node identification and spoofing attack detection. Node identification based on unique hardware features has been considered an efficient technique in wireless networks; spoofing attacks imitating unique hardware features, however, could significantly impair or break down the node identification system. The same applies to the conventional clock-skew-based node identification that is also vulnerable to these attacks due to clock information exposed through broad-

casting. To defend against Spoofing attacks, our proposed NISA utilizes the reverse time synchronization framework for estimating the sensor nodes' clock skews and the spatially-correlated radio link information to achieve simultaneous node identification and attack detection. We further provide centralized and distributed NISA for covering both single- and multi-hop scenarios. On a real WSN testbed consisting of TelosB sensor nodes, we investigate the identifiability of clock skews under temperature and voltage variations; implement and evaluate both centralized and distributed NISA. In particular, the centralized NISA employs a single-input and multiple-output convolutional neural network. Experimental results demonstrate that both centralized and distributed NISA could provide accurate node identification and Spoofing attack detection.

7.1 Introduction

The thriving technologies of wireless communications and networking expedite the reliance of our daily lives on smart devices such as laptops, smartphones, and nowadays Internet of Things (IoT) devices [92]. The security of those devices, therefore, becomes a serious concern. An efficient security measure for wireless networks is node identification, i.e., the identification of legitimate devices often in the presence of attackers. Conventional techniques based on pre-defined identifiers (IDs) or media access control (MAC) addresses have been widely used for node identification. Malicious devices, however, could impersonate legitimate ones so that attackers join a network, intercept data exchanged, and even launch attacks—like denial of service (DoS)—on the network [93, 94].

Device fingerprinting (DF) techniques utilizing hardware features as device-specific fingerprints are considered a promising technique that could alleviate the vulnerability of the conventional node identification techniques [95]. Of many hardware features, radio frequency (RF) and clock features—i.e., related with wireless modules and crystal oscillators (COs) of devices, respectively—attract much attention for a variety of application scenarios from wireless local area networks (WLANs) [96] to cloud [97] and

controller area networks (CANs) [98] to IoT [99]. The RF features, however, often require additional equipment [99]—e.g., universal software radio peripheral (USRP)—for their measurement, so they couldn't be employed in normal gateway nodes (also called cluster heads) in multi-hop wireless networks. The clock features such as clock skews, on the other hand, are readily available as part of the time synchronization service for wireless communication and networking. Therefore, clock-skew-based node identification (CSNI) becomes an attractive option, especially for large-scale, multi-hop wireless sensor networks (WSNs).

In the literature, clock skews as device-specific fingerprints could be estimated by various methods from simple ratio-based one [48] to complex linear regression [1], and node identification could be achieved through simple thresholding [100] to advanced machine learning (ML) techniques [99, 101]. Depending on the location of clock skew estimation and node identification, CSNI schemes can be categorized as centralized (i.e., at the head[1]) or distributed (i.e., at the gateway and/or sensor nodes) ones. In [100], a preliminary investigation of differentiating sensor nodes based on clock skews was studied using the flooding time synchronization protocol (FTSP) [18] as a skew estimator; sensor nodes locally estimate their clock skews and transmit them back to the head for centralized node identification by employing a simple thresholding method. In [102], the authors experimentally verified that different sensor nodes—i.e., MICAz [45] and TelosB [25]—have different and unique clock skews, which can be easily distinguished at a centralized monitoring station even in a multi-hop WSN. They also identified and discussed the issue of exposing the estimated clock skews through a *non-covert* channel from the security perspective; the transmission of clock skews over the network as in [100] is vulnerable to security attack [102]. The utilization of CSNI for defending some common attacks such as Sybil [103], Replication [104], and Wormhole [105] has also been studied: The combination of the continuity of clock skews and node IDs can be used to detect Sybil and Replication attacks [100], while the immutable characteristics

[1]A head is typically an ensemble of a head/sink node and a monitoring station such as a PC or a server connected to it.

of clock skews can be employed to defend Wormhole and Sybil attacks [106].

However, CSNI fails when malicious nodes can imitate the clock skews of legitimate ones to the point that fake and true clock skews cannot be differentiated from one another, which we call *Spoofing attack*[2] throughout this chapter. The Spoofing attacker described in [108] could pretend itself as a legitimate sensor node by estimating the clock skew and offset based on the timestamps intercepted from the legitimate sensor node. To defend CSNI against Spoofing attack, therefore, we propose an approach called Node Identification against Spoofing Attack (NISA) based on the *reverse asymmetric time synchronization framework* [1, 3, 2]. Through unilaterally collecting the spatially-correlated radio information of received signal strength (RSS) and link quality indicator (LQI)[3] together with clock features extracted from the reverse time synchronization, NISA could simultaneously detect Spoofing attacks and achieve node identification. Note that radio information such as RSS is available as part of the transmission services for wireless communication and networking, which does not require additional equipment for its measurement; the spatial correlation of radio information has been widely employed for various applications such as indoor localization [109, 110], key generation [111], and attack detection and localization [112, 113]. To the best of the authors' knowledge, this is the first CSNI scheme simultaneously addressing the node identification and spoofing attack detection for both single-hop and multi-hop WSNs.

Our major contributions in this chapter are summarized as follows:

- We propose centralized NISA that lays its foundation on BATS—i.e., the state-of-the-art reverse asymmetric time synchronization scheme—which can significantly improve the secrecy and identifiability of sensor nodes' clock skews. A single-input and multiple-output (SIMO) convolutional neural network (CNN) is employed in centralized NISA for simultaneous node identification and attack detection by taking the time series of clock features and radio information as its input.

- We also propose distributed NISA which can run at normal gateway nodes cover

[2]This attack is also called *clock skew replication attack* in [107].
[3]LQI is a vendor-specific value which is currently available in IEEE 802.15.4 standard.

multi-hop scenarios, which most conventional RF fingerprint-based identification schemes [99, 114, 115] couldn't be applied to. As node identification and attack detection are done locally at gateway nodes in distributed NISA without the involvement of the head, it could immediately filter out attacks nearby and remove unnecessary packet transmissions upstream from the gateways to the head.

- We carry out a systematic investigation of the identifiability of sensor nodes' clock skews based on the high-precision centralized NISA. Unlike the existing investigations [55, 56], ours demonstrates the concurrent behaviors of the clock skews of a group of sensor nodes under temperature and voltage variations, which result from the drifts of digitally-controlled oscillators (DCOs) calibrated by COs.

- We present the design, implementation, and practical evaluation of the performance of both centralized and distributed NISA on a real WSN testbed consisting of TelosB [25] sensor nodes running TinyOS [44]. Experimental results demonstrate the effectiveness of both centralized and distributed NISA in node identification while defending against Spoofing attacks.

The rest of the chapter is organized as follows: The overview of NISA is provided in Section 7.2. The centralized NISA is presented in Section 7.3. Section 7.4 narrates the distributed NISA. Our investigations on clock skews and experimental evaluation results of both centralized and distributed NISA are demonstrated in Section 7.5. Section 7.6 reviews the related work in comparison to our work. The summary of this chapter is given in Section 7.7.

7.2 NISA: Node Identification and Spoofing Attack Detection Based on Clock Features and Radio Information

In this section, we first discuss the acquisition of the clock features for node identification and the characteristics of radio information for spoofing attack detection. Based on these foundations, then we provide an overview of the proposed NISA system.

7.2.1 Clock Features For Node Identification

Clock skew and offset represent the fundamental relationship between two hardware clocks in time synchronization. The accurate estimation and update of them are essential not only for high synchronization accuracy but also for reliable node identification based on the uniqueness of hardware clocks. Due to the imperfect manufacturing of the low-cost COs in common WSN devices, the clock frequencies of any two sensor nodes are hardly identical to each other [102]. Hence a different and unique clock frequency specific to each sensor node.

As a clock skew is one of the two parameters in modeling the relationship between the hardware clocks in the first-order affine *hardware clock* model that most time synchronization schemes rely on (e.g., [116, 2]), the clock skew between the hardware clock T_i of a sensor node i and the reference clock t of the head node can be defined as follows: For $i \in [0, 1, \ldots, N-1]$,

$$T_i(t) = (1 + \epsilon_i)t + \theta_i \rightarrow \epsilon_i = \frac{T_i(t) - \theta_i}{t} - 1, \tag{7.1}$$

where N denotes the number of sensor nodes, $\epsilon_i \in \mathbb{R}$ and $\theta_i \in \mathbb{R}$ respectively represent the clock skew and offset between the sensor node i's hardware clock and the reference clock.

For convenience, we often use clock frequency ratio R_i—also called *slope*—instead of the clock skew ϵ_i, which is given by:

$$R_i = 1 + \epsilon_i = \frac{T_i(t) - \theta_i}{t}. \tag{7.2}$$

The clock frequency ratio is calculated based on the clock time acquired from the internal DCO calibrated by the external CO, which represents the behaviors of both internal DCO and external CO. Note that, since DCO has up to ten times larger ppm than CO [4], our investigation is conceptually different from those based only on external CO as in [55, 56].

The clock offset θ_i, also called *intercept*, represents the hardware clock at $t=0$ (i.e.,

$T_i(0)$). Unlike clock skew, it cannot be used as a device-specific fingerprint because its value does not depend on hardware features and changes whenever power cycling any of the two nodes. During the normal operation, however, its value is fixed & likely unique and rather stable for a short period of time, so it can be used as *an auxiliary variable* for CSNI, especially when fine-grained node identification is needed in large-scale WSNs as we will discuss in Section 7.3.

The clock parameters are estimated based on the linear regression in BATS [1]: During the kth synchronization ($k \geq m$), the clock parameters of the sensor node i are estimated based on the latest m timestamp pairs in a sliding window as follows:

$$\mathbf{\Phi}_i(k) = \left\{\mathbf{t}(k)^\top \mathbf{t}(k)\right\}^{-1} \mathbf{t}(k)^\top \mathbf{T}_i(k), \tag{7.3}$$

where

$$\begin{aligned} \mathbf{\Phi}_i(k) &= \left[\hat{R}_i(k), \hat{\theta}_i(k)\right], \\ \mathbf{t}(k) &= [t_{k-m+1}, \ldots, t_k], \\ \mathbf{T}_i(k) &= [T_i(t_{k-m+1}), \ldots, T_i(t_k)], \end{aligned}$$

and $\hat{R}_i(k)$ and $\hat{\theta}_i(k)$ are the clock parameters of frequency ratio and offset, $(\cdot)^\top$ and $(\cdot)^{-1}$ denote vector transpose and matrix inverse, respectively. Note that the estimation of clock skew can be obtained from $\hat{R}_i(k)$, i.e., $\hat{\epsilon}_i(k) = \hat{R}_i(k) - 1$.

It is worthwhile to mention the advantages of the clock parameter estimation of BATS when applied to CSNI. First, the reverse one-way time synchronization framework of BATS estimates the clock parameters of sensor nodes at the head and, therefore, does not expose estimated clock skews to other nodes in the network unlike conventional schemes where the clock skews estimated at sensor nodes have to be transmitted back to the head for node identification. Second, the sliding window size of the linear regression (i.e., m) can be adjusted for the operation environment of a WSN; a larger window size can improve the accuracy of the estimation for a static environment, while a smaller

one can adapt to a dynamic environment more quickly at the expense of the estimation accuracy. Third, the high-precision estimation of clock parameters based on 64-bit double-precision floating-point at the head—providing up to 16 significant digits after a decimal point [49]—enables finer-grained CSNI, which is critical for large-scale WSNs [102].

7.2.2 Radio Information For Spoofing Attack Detection

There are existing measures for CSNI against the Spoofing attack such as actively altering the synchronization interval [107]. In contrast to those measures, we novelly adopt the radio information in defending against Spoofing attacks, which could be passively measured to be consistent with the *passive nature* of the CSNI method.

Radio information such as RSS and LQI from a received packet indicates the signal strength and the quality of packet transmission. Specifically, RSS is correlated in space and, therefore, varies with location, which has been well exploited in many techniques such as indoor localization and key generation. For instance, the spatial variation of the RSS has been extensively studied in the key generation [111, 117], where they revealed that an attacker located more than one half-wavelength away from any existing legitimate device faces uncorrelated multipath fading in most scenarios. Note that RSS is not the instantaneous power of the received signal, which is not available at typical receivers, but its time average. In common WSN platforms, the average power of the received signal is referred to as the received signal strength indicator (RSSI).

LQI is another parameter on the radio information. In [118], it is suggested that LQI should be employed as an indicator for intermediate quality links after averaging over multiple readings due to its high variance over time; time-averaged LQI could be a better indicator for packet receive ratio (PRR) than RSSI. In contrast, when the link quality is good whereby the LQI is insensitive, RSSI could be a better indicator for distinguishing attackers. Note that LQI is platform-specific; LQI is highly correlated in space and can be used for key generation like RSS on some platforms [119] but not on other platforms. As such, LQI at least could be used for node identification

for distinguishing attackers from legitimate devices when intermediate quality links are available [118]. Because the ensemble of RSSI and LQI is less sensitive to instantaneous link fluctuations, it can better represent the link status [120].

Note that, due to its spatial correlation, radio information is hard to be impersonated by attackers and, hence, could complement clock features that are vulnerable to the Spoofing attack. In the following, we provide an overview of NISA which leverages both radio information and clock features to reinforce CSNI against Spoofing attacks.

7.2.3 Overview of NISA System

Two distinctive aspects of our design of NISA systems are *a covert channel* [102] and *passive nature*. For the former, the transmission of estimated clock features and radio information over a network, which is vulnerable to eavesdropping, cannot be allowed; for the latter, the system should be able to perform node identification and attack detection without the modification of standard procedures requiring the active involvement of sensor nodes like the frequent changes of synchronization time period suggested in [107]. For these reasons, we found that BATS, i.e., the reverse one-way time synchronization scheme whose estimations of clock features are all done at the head, is a perfect candidate for CSNI part of NISA; unlike conventional CSNI schemes relying on transmissions of clock features from sensor nodes to the head (or gateway nodes in the case of multi-hop WSNs) for node identification (e.g., [100]), BATS enables the clock features to be estimated and used for node identification at the same place (i.e., the head or gateway nodes) without exposing it over a network. Likewise, the head or gateway nodes can directly measure the spatially-correlated radio information for attack detection without exposing it over a network. Therefore, NISA could passively achieve node identification and attack detection through a covert channel.

In Fig. 7.1, we provide an overview of the proposed NISA system. When the packets from sensor nodes arrive, the head extracts the clock features—i.e., skew and offset—using the reverse time synchronization and measures the radio information of RSSI and LQI. Then a database is constructed based on them for node identification and attack

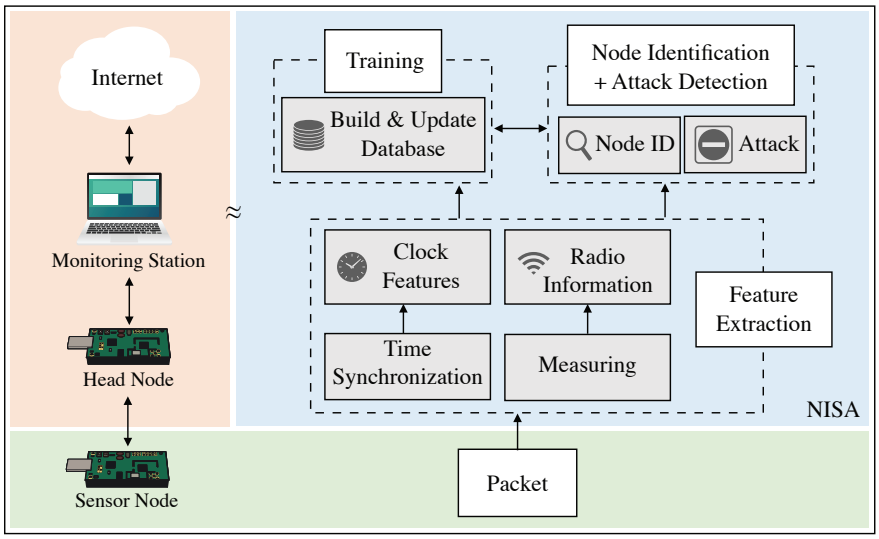
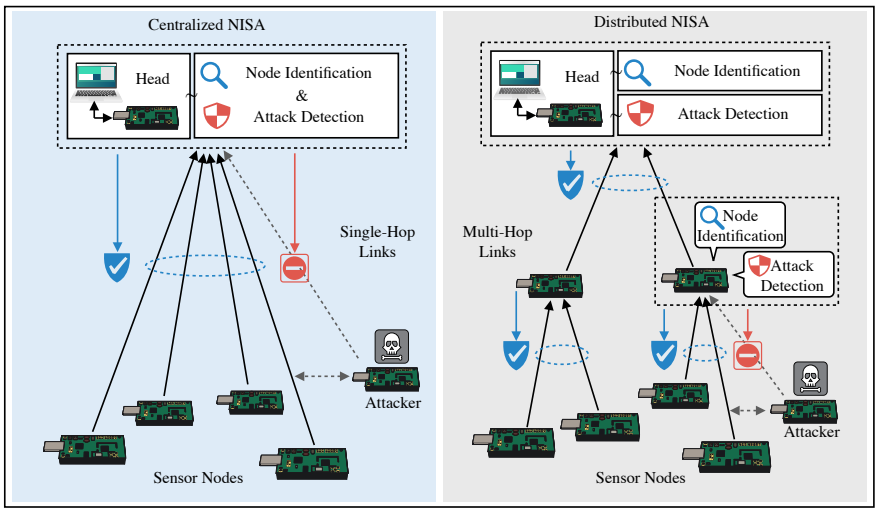


Figure 7.1: Overview of NISA system [7].

detection later. However, as WSNs often require multi-hop configurations to cover vast areas, we provide two variations in NISA system design, namely *centralized NISA* and *distributed NISA*, for respectively addressing single-hop and multi-hop scenarios. As illustrated in Fig. 7.2 (a), the node identification and attack detection are done at the head consisting of a head node and a powerful monitoring station in centralized NISA, which makes it possible to employ advanced techniques for better performance. Because it would be impractical to implement a rather heavy centralized NISA system at gateway nodes, however, we propose distributed NISA in order to cover multi-hop scenarios as shown in Fig. 7.2 (b). Note that the system design is intentionally simplified in this case so that it can be easily implemented at gateway nodes that are often normal battery-powered sensor nodes.

The two system designs provide different options in performance and computational complexity tradeoff: Simple classification or thresholding schemes could suffice for distributed NISA but at the expense of relatively lower performance, which we discuss in Section 7.5. Centralized NISA, on the other hand, could provide better performance through advanced classification/detection methods like those based on neural networks [99], which can take all parameters into account and run node identification and attack



(a) (b)

Figure 7.2: NISA system designs: (a) Centralized NISA for single-hop WSNs; (b) distributed NISA for multi-hop WSNs [7].

detection in an integrated manner, at the expense of computational complexity.

7.3 Centralized NISA

As we discussed in Section 7.2.3, centralized NISA is designed to provide better performance based on the integrated processing of clock features and radio information for single-hop WSNs. Here we provide more details of centralized NISA, including its workflow and implementation example.

7.3.1 Workflow of Centralized NISA

As shown in Fig. 7.3, there are two phases in the workflow of centralized NISA, i.e., the initial phase and detection/identification phase. From the packets received from sensor nodes, centralized NISA extracts clock features through time synchronization and collects radio information during the initial phase. The clock features and the radio information are then combined and fed into a database. After gathering enough data for all the sensor nodes in the network, a training process is performed based on the

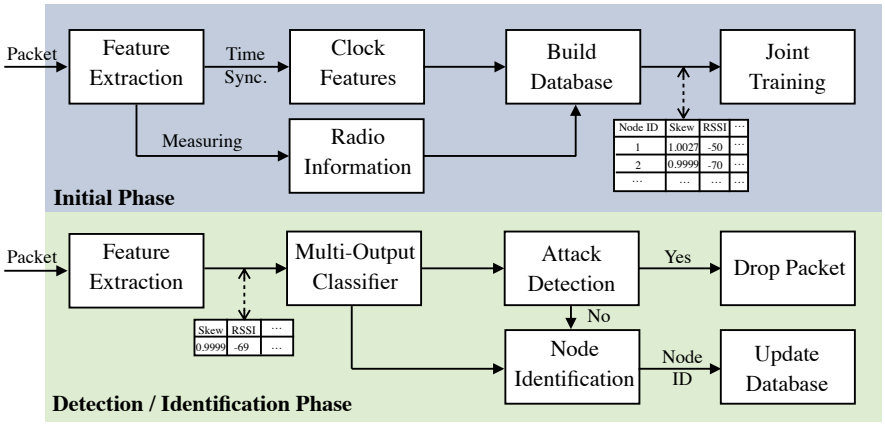


Figure 7.3: Workflow of centralized NISA [7].

database for classification. The system switches to the detection/identification phase when the training process is completed.

During the detection/identification phase, a common functional block—i.e., the "Multi-Output Classifier" shown in Fig. 7.3—carries out both node identification and attack detection simultaneously. When an attack is detected, the corresponding packet is dropped; otherwise, the identified node ID is reported as a final result of CSNI. This group of data—i.e., a node ID and its corresponding clock features and radio information—is used to continuously update the database.

The major challenge in the workflow of centralized NISA is the implementation of the common functional block that should be able to process the clock features and the radio information as a whole for simultaneous node identification and attack detection through time-series classification (TSC); solutions based on advanced neural networks like CNN, recurrent neural network (RNN), and their many variations (e.g., long short-term memory (LSTM) and gated recurrent unit (GRU)) could be employed for TSC. Among them, CNN is rather popular for node identification (i.e., device fingerprinting) due to its capability of not only classifying nodes' fingerprints as image data [99, 121] but also solving the TSC problem as demonstrated in [122] and [123], which is why we choose it for our sample implementation described in the next subsection.

Data: The head node maintains the following data and variables:
- e: Event object;
- p: Packet object;
- $T2$: MAC-layer Timestamp;
- $RSSI$: RSSI value;
- LQI: LQI value;
- Q_P: FIFO queue for packets;
- Q_{T2}: FIFO queue for timestamp $T2$.

```
1  On detecting an event e:
2  switch e.type do
3      case PACKET do
4          if p.getDestAddress()==HEAD then
5              p ← Q_P.dequeue()
6              T2 ← Q_T2.dequeue()
7              RSSI ← getRSSI(p)
8              LQI ← getLQI(p)
9              p ← Packet(p, T2, RSSI, LQI)
10             send(p)                          // forward the packet to monitoring station
11         else
               // Packets not for the head ...
12     otherwise do
           // Process other events ...
```

Algorithm 4: Procedures of centralized NISA at the head node.

7.3.2 Implementation of Centralized NISA Based on SIMO CNN

Here we provide the details of our sample implementation of centralized NISA based on SIMO CNN. As discussed in Section 7.2.1, BATS—the reverse one-way time synchronization scheme—is used for time synchronization and CSNI due to its capability of providing fine-grained identifiability of clock skews and offsets. At sensor nodes, no processes are running for NISA except MAC-layer timestamping required by BATS for achieving time synchronization. At the head node, it measures RSSI and LQI of each received packet and forwards them—together with the associated timestamps from MAC-layer timestamping—to a monitoring station which not only estimates clock skew and offset based on BATS but also performs NISA based on SIMO CNN. The details of the procedures at the head node for centralized NISA are given in the pseudocode in Algorithm 4.

Fig. 7.4 shows the architecture of SIMO CNN working as the multi-output classifier in the implementation of centralized NISA. The input layer of SIMO CNN takes time

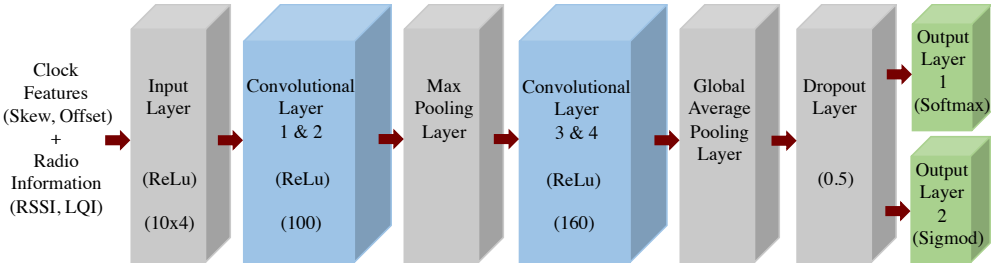


Figure 7.4: SIMO CNN architecture employed as the multi-output classifier of centralized NISA [7].

series of combined clock features and radio information—i.e., clock skew, clock offset, RSSI, and LQI—as its input. Convolutional layers 1 & 2 connecting the input layer are followed by a max pooling layer preventing overfitting, convolutional layers 3 & 4, and a global average pooling layer again preventing overfitting by using the average value. The output from the global average pooling layer goes through a dropout layer with dropout rate of 0.5, which is connected to output layers 1 and 2. All the convolutional layers use the rectified linear unit (ReLu) as their activation function. The output layer 1 for node identification uses softmax as its activation function for multiclass classification, while the output layer 2 for attack detection uses sigmoid as its activation function for binary classification. We use ADAM optimizer [124] for training and loss functions of categorical cross-entropy and binary cross-entropy for multi-class classification (node identification) and binary classification (attack detection). The batch size and the number of epochs are set to 10 and 100, respectively.

The proposed SIMO CNN is implemented in Python with Keras [125] and TensorFlow [126]. The subsystems at the head and the sensor nodes are implemented in nesC on TinyOS, and those in the monitoring station in Java and Python on macOS.

7.4 Distributed NISA

Centralized NISA can provide better performance based on the integrated processing of clock features and radio information, but it would be impractical to implement a rather heavy centralized NISA system at gateway nodes in multi-hop WSNs. To support multi-

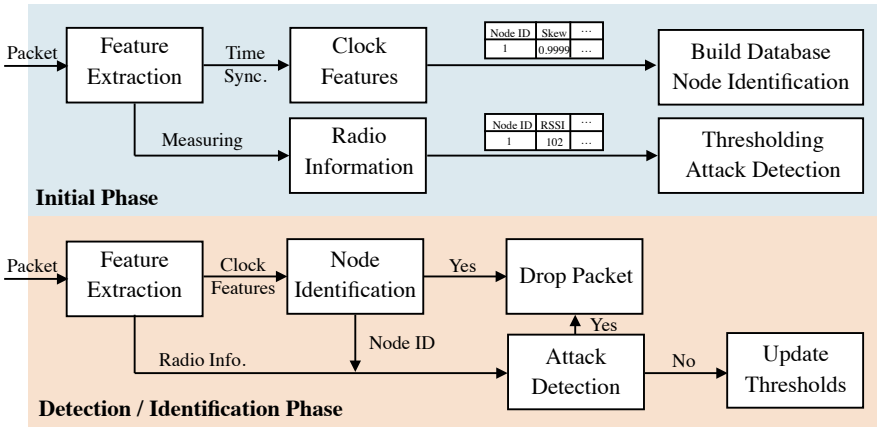


Figure 7.5: Workflow of distributed NISA [7].

hop WSNs, therefore, we separate the processing of node identification and that of attack detection in the design of distributed NISA, which not only lowers the computational complexity but also enables further simplification of each of the processing. In this section, we discuss the details of the workflow and the implementation of distributed NISA.

7.4.1 Workflow of Distributed NISA

Fig. 7.5 shows the workflow of distributed NISA. Using the packets received from sensor nodes, distributed NISA extracts clock features through time synchronization and collects radio information during the initial phase. Unlike centralized NISA, however, the clock features and the radio information are used separately for building a clock database for node identification and a radio database for attack detection, respectively.

During the detection/identification phase, the clock features are fed into a classifier for node identification, which operates on the built clock database consisting of pairs of node ID and clock features. The classifier could be implemented based on a simple thresholding method with upper and lower bounds for a clock skew [100] or a relatively complex classification algorithm like k-nearest neighbors (KNN) [127]. If the node identification is successful, the resulting node ID is used in further processing of attack detection; otherwise, the received packet is dropped.

Then, the radio information, together with the node ID, is provided as an input to the attack detection block, where potential Spoofing attacks are detected based on the node ID by comparing the received radio information with that already in the radio database; multiple data points are used for the processing in order to mitigate the effects of instantaneous fluctuations in the radio information like RSSI. It turns out that the anomaly detection based on RSSI with methods like thresholding is a well-studied problem [128]; there are various thresholding solutions with different complexities. Finally, if a Spoofing attack is detected, the received packet is dropped; otherwise, distributed NISA returns the node ID as a final result and updates the databases based on the new legitimate data samples.

7.4.2 Implementation of Distributed NISA on Gateway Nodes

Based on the above workflow, we implement distributed NISA on gateway nodes which are ordinary sensor nodes in this work as discussed in Section 7.1; our implementation of distributed NISA on an ordinary sensor node as a gateway node could demonstrate its practicality for multi-hop scenarios.

For time synchronization, we implement a simplified version of BATS based on simple linear regression for the estimation of clock parameters as in [18], which is also used for extracting clock features. We apply the reverse one-way synchronization framework of BATS to the time synchronization between a gateway node and its offspring sensor nodes in a multi-hop WSN, which was originally proposed for the time synchronization between the head and sensor nodes in a single-hop WSN [1]. For node identification, we compare the clock skew of a received packet with the average clock skews of legitimate nodes in the database and find the ID of a legitimate node that matches best; during the comparison, we use a threshold-based prefiltering of the clock skew from a fake node as suggested in [100]. We use a similar approach for attack detection using RSSI for comparison.

Note that the gateway nodes in distributed NISA can decide whether to accept or drop received packets based on the results of node identification and attack detection

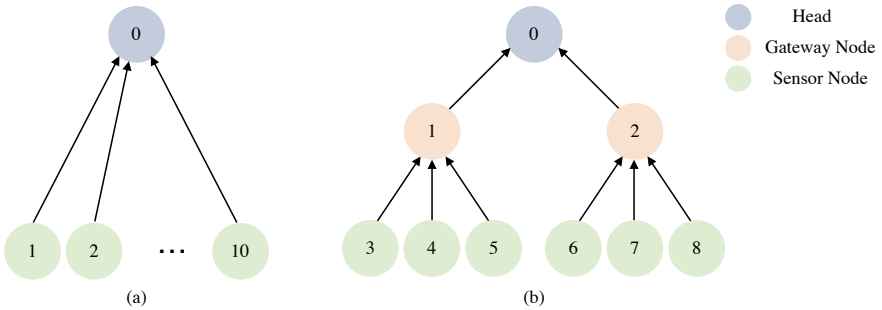


Figure 7.6: Network topologies for (a) a single-hop WSN and (b) a multi-hop WSN [7].

locally without transmitting the clock skews and the radio information for the processing at the head, which prevents attackers from eavesdropping them.

7.5 Experimental Results

In this section, we present the experimental results for the investigation on the identifiability of clock skews under various operating conditions and the performance evaluation of centralized and distributed NISA with single-hop and multi-hop WSNs.

7.5.1 Experimental Setup

Unlike conventional node identification schemes (e.g., [99]), the proposed NISA can be implemented and evaluated on a common WSN testbed without special hardware like USRP or modification of the existing infrastructure. The WSN testbed for our investigation of clock skews and evaluation of the performance of NISA systems, therefore, consists of ordinary head, gateway, and sensor nodes, all of which are based on TelosB motes running TinyOS. We use a single-hop star topology with 1 head and 10 sensor nodes for the investigation on clock skews & the evaluations of the performance of centralized NISA and a multi-hop tree topology with 1 head, 2 gateway, and 6 sensor nodes for the evaluation of the performance of distributed NISA, which are shown in Fig. 7.6 (a) and (b), respectively.

7.5.2 Investigation on Clock Skew Identifiability

We carry out a series of experiments with the single-hop WSN shown in Fig. 7.6 (a) to investigate the fine-grained identifiability of clock skews provided by BATS under various operating environments, where we set the time synchronization interval and the sliding window size to 1 s and 19, respectively, as in [1].

7.5.2.1 Under Fixed Temperature and Voltage

We recorded the clock skews of each sensor node with respect to the head node for an hour in a controlled environment where the effect of temperature and voltage variations on the clock skews can be ignored. Fig. 7.7 (a) shows the clock frequency ratios of 10 sensor nodes, where we can observe that the frequency ratios of the sensor nodes are quite stable for the whole period and that most frequency ratios are distinguishable even in the low-precision view except those of sensor nodes 1 & 8 and 3 & 6. Thanks to the high-precision estimation of the clock parameters of BATS, we can zoom in those indistinguishable frequency ratios with higher precision as shown in Fig. 7.7 (b) and (c), which reveals that they can be clearly identified, too.

7.5.2.2 Under Temperature and Voltage Variations

It has been demonstrated by many (e.g., [55, 56, 102]) that hardware clocks, unless equipped with compensation circuits, are affected by temperature and voltage variations, resulting in time-varying clock skews; this is the case for the hardware clocks of most WSN devices, which are based on low-cost COs [56]. However, the concurrent behaviors of the clock skews of a group of sensor nodes resulting from the drifts of both DCOs and COs under temperature and voltage variations are yet to be investigated, which, in fact, is the main focus of the experiments described in this subsection.

According to the empirical study in [55], the effect of temperature and voltage variations on the clock skew could be modeled as a linear function for a period of time during which the amount of changes in temperature and voltage is rather small: Given

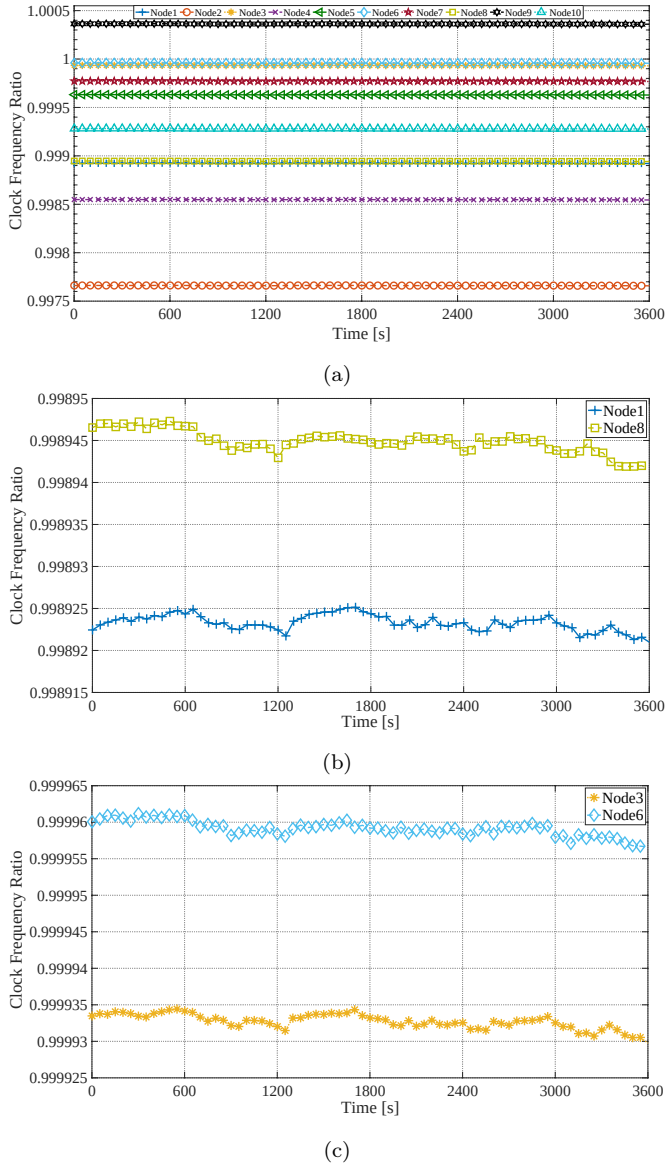


Figure 7.7: Estimated clock frequency ratios of (a) all 10 sensor nodes and zoomed-in (b) sensor nodes 1 & 8 and (c) sensor nodes 3 & 6 over 3600 s [7].

the initial time t_0, the skew for $t>t_0$ is given by

$$\epsilon_i(t) = \epsilon_i(t_0) + \Delta\epsilon_i(t_0, t), \tag{7.4}$$

$$\Delta\epsilon_i(t_0, t) = \alpha(t)(\mathcal{T}(t) - \mathcal{T}(t_0)) + \beta(t)(\mathcal{V}(t) - \mathcal{V}(t_0)), \tag{7.5}$$

where $\Delta\epsilon_i(t_0, t)$ is the difference of the skew value between time t_0 and t, $\mathcal{T}(t)$ and $\mathcal{V}(t)$ are the temperature and the voltage at time t, and $\alpha(t)$ and $\beta(t)$ are the temperature-skew sensitivity factor and the voltage-skew sensitivity factor at time t; the sensitivity factors are to be estimated through experiments. Alternatively, we could feed as input data the time series of timestamps, temperatures, and voltages into a *neural network* so that it can learn the complex relationship between the clock skew and those environment variables [71].

Instead of directly modeling the effect of temperature and voltage variations on the clock skew, we could adjust the sliding window size mentioned in Section 7.5.2 so that the periodic updates of the clock skew reflect the effect, which is practical for finding the current clock skew under a dynamic environment and also sufficient for the purpose of node identification. Based on this, we experimentally investigate the identifiability of the clock skews under voltage and temperature variations in the following experiments.

We use the onboard temperature sensor to read the environment temperature that reduces gradually from 32 °C to the turn-over temperature 25 °C. We equip each sensor node with two AA batteries as in actual deployments and use the onboard voltage sensor to measure the voltage level. Since not all the batteries are brand new, their starting voltages are quite different from one another, which enables us to investigate the clock skew behavior under the same temperature change but different voltage levels. During the experiments, we observe that the voltage decrease is not significant, i.e., an average of 0.05 V decrease per each node during the 1-hour period.

Fig. 7.8 shows as an example the changes in the clock skew of sensor node 10 under temperature and voltage variations, where the range of the voltage level (0.05 V) is much smaller compared to that of the temperature (7 °C) as mentioned. The ranges of temperature and voltage variations in this experiment are typical for actual deployments where the batteries live for months and the temperature changes day and night.

Fig. 7.9 summarizes the effect of temperature variation on the clock skews of all 10 sensor nodes over 3600 s, where the left and right y-axes denote the clock frequency ratio and the environment temperature, respectively.

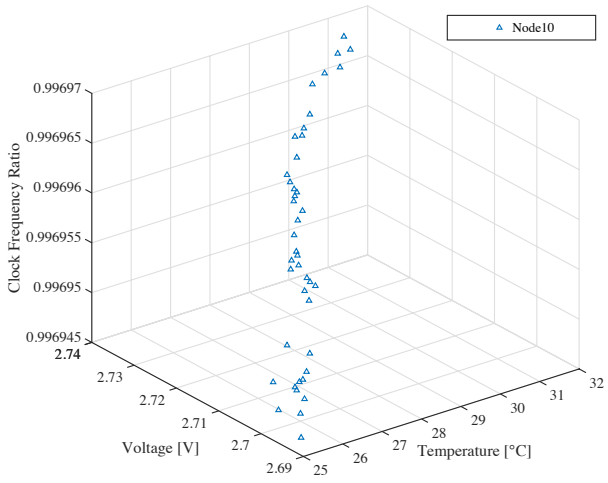


Figure 7.8: Clock frequency ratio of sensor node 10 under temperature and voltage variations over 3600 s [7].

Note that, though the starting voltage levels of all sensor nodes are different, their consumptions during the experiment are little and similar to one another, whose effect on the clock skews of all sensor nodes are relatively smaller compared to that of the temperature. Specifically, all the variations of the clock frequency ratios shown in Fig. 7.9 follow the similar trend of the varying temperature; Fig. 7.9 (a) and (c) in particular show that, though the clock frequency ratios of the sensor nodes are different, their changes are quite similar to each other in terms of both trend and the amount of changes. These results demonstrate that the clock skews of sensor nodes are still distinguishable as far as they are deployed under the same environment, which is of critical importance to CSNI.

7.5.3 Performance of Centralized NISA under Single-Hop Scenarios

We evaluate the performance of centralized NISA in a single-hop WSN shown in Fig. 7.6 (a), where each sensor node sends its packet to the head node every second as in [1]. We collect 1,000 and 500 packets for each sensor node respectively for training and validation on the first day, and another 1,000 packets for testing on the second day. Since there is no existing CSNI scheme carrying out simultaneous node identification

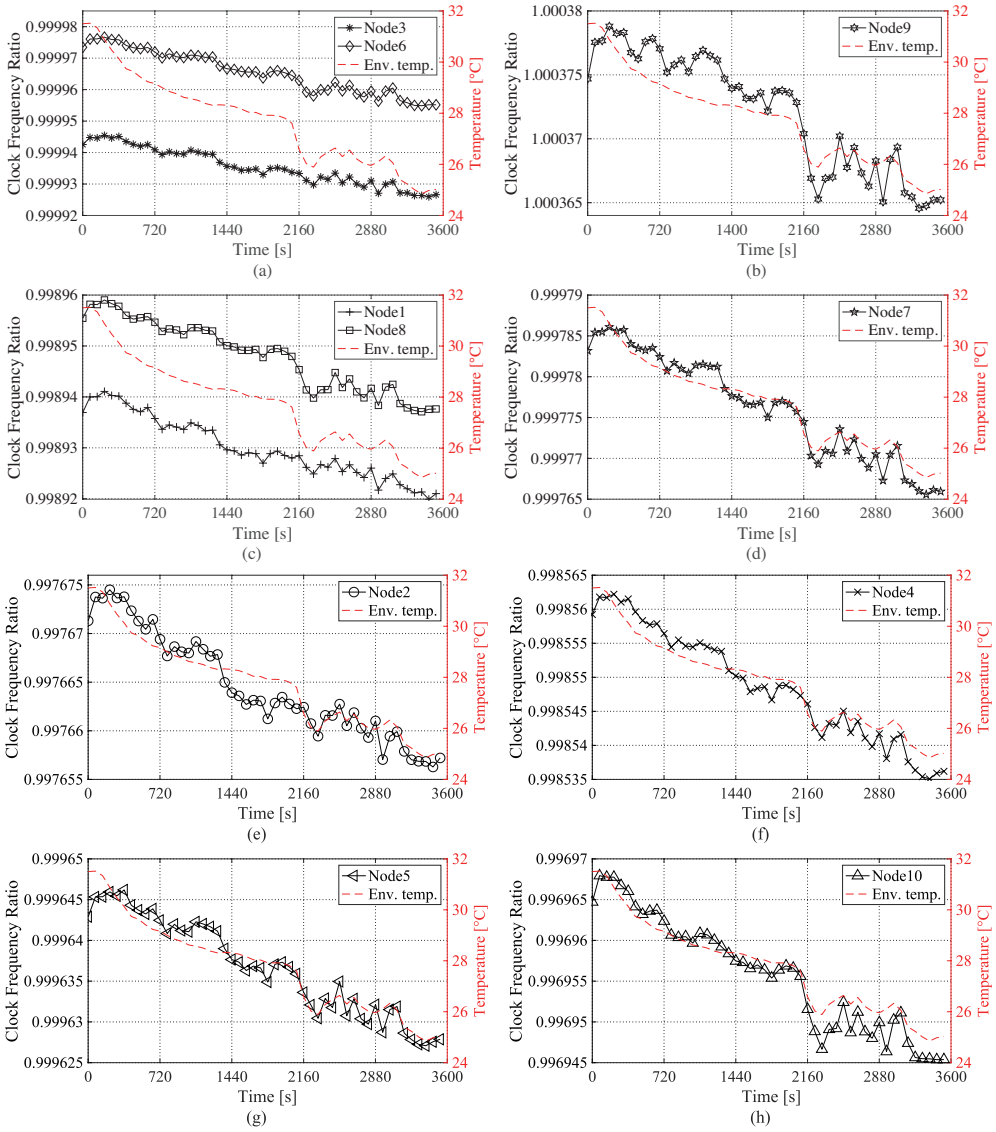


Figure 7.9: Clock frequency ratios of sensor nodes under temperature variations over 3600 s: (a) Sensor nodes 3 & 6, (b) sensor node 9, (c) sensor nodes 1 & 8, (d) sensor node 7, (e) sensor node 2, (f) sensor node 4, (g) sensor node 5, and (h) sensor node 10 [7].

and spoofing attack detection for WSNs, we put the focus of our evaluation on the effectiveness of the proposed centralized NISA as such using node identification accuracy and attack detection rate as performance measures. We carry out the performance evaluation in two steps: First, we focus on the effectiveness of the node identification

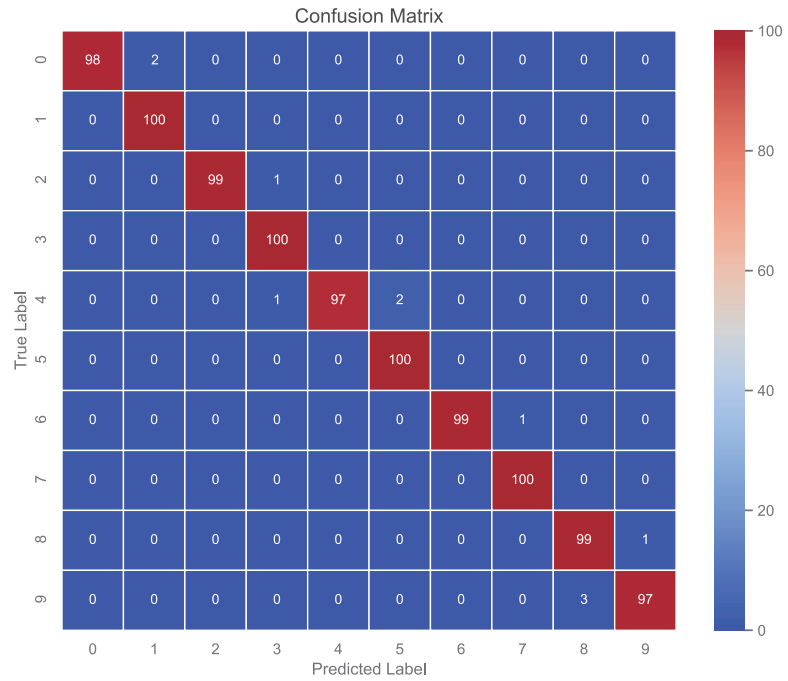


Figure 7.10: Experimental results of centralized NISA for identifying 10 sensor nodes each for 100 times without Spoofing attacks [7].

without Spoofing attacks. Second, we evaluate the effectiveness of the attack detection under Spoofing attacks.

7.5.3.1 Without Spoofing Attacks

As discussed in Section 7.3, the measurements from 10 consecutive packets are formed into time series in order to handle the variations of clock skew and offset and the fluctuations of RSS and LQI, resulting in 100 testing time series for each sensor node. Another crucial step towards TSC is the scaling of the datasets—we transform the value of our data into the range of $[0, 1]$, which is of importance for the classifier to achieve satisfactory results.

We trained the SIMO CNN of the system based on the training dataset, tuned its hyperparameters based on the validation dataset, and finally evaluated the performance of node identification based on the test dataset, whose confusion matrix is shown in Fig. 7.10. Labels 0 to 9 stand for sensor node IDs from 1 to 10 as shown in Fig. 7.6 (a).

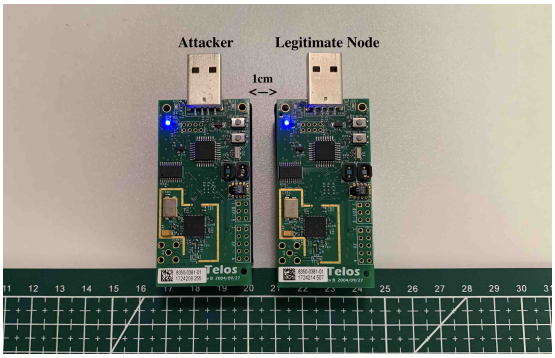


Figure 7.11: Illustration of the distance between the attacker and the legitimate sensor node in our experiments [7].

We identify each sensor node 100 times, resulting in 1,000 identifications in total, and obtain the node identification accuracy of 98.9%, which is well in line with our prior investigation on the identifiability of clock skews in Section 7.5.2. In spite of such a high node identification accuracy, some testing segments are falsely classified to wrong labels, which results from the limited coverage of the training datasets for the clock behaviors represented by those segments. In contrast, other sensor nodes, e.g., the labels 1 and 3, are identified with 100% accuracy, which implies that their testing segments provide enough information to the system so that they can be uniquely identified based on the trained SIMO CNN model.

From the experimental results discussed above, we found that the integrated system design of centralized NISA could clearly identify sensor nodes thanks to its use of combined clock features and radio information for most cases.

7.5.3.2 Under Spoofing Attacks

For the evaluation of the performance of centralized NISA under Spoofing attacks, we add one sensor node working as an attacker to the testbed described in Section 7.5.1. We deploy the attacker next to a legitimate sensor with just 1-cm gap between the two as shown in Fig. 7.11 as a worst-case scenario.

As in [102, 107], the attacker estimates the clock parameters of the legitimate sensor node—node 10 with the label 9 in our case—by eavesdropping on the time synchroniza-

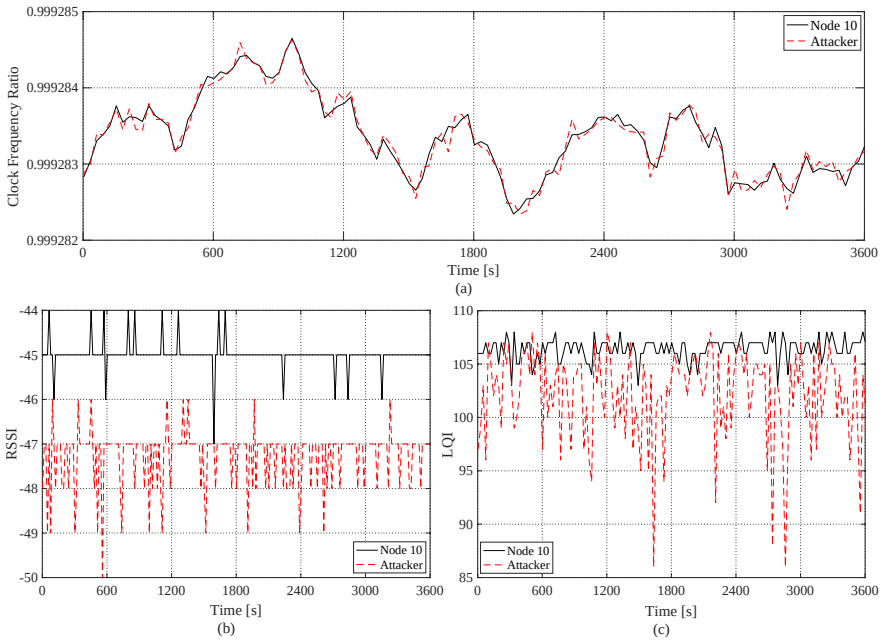


Figure 7.12: Illustrations of (a) clock feature of skew, and radio information of (b) RSSI and (c) LQI between the attacker and the legitimate sensor node in our attack experiments [7].

tion packets. The attacker then generates timestamps in its own time synchronization packets based on the legitimate sensor node's clock parameters through time synchronization so that the calculated clock skew at the head based on the received packets from the attacker would be very close to that of the legitimate sensor node. Fig. 7.12 (a) shows how close the attacker could imitate node 10 in its clock skew, where the attacker could very closely imitate the clock skew as investigated in [108]. Fig. 7.12 (b) and (c), on the other hand, show the radio information of RSSI and LQI of the attacker does not very close to that of node 10 due to its spatial correlation.

The results shown in Fig. 7.12 demonstrate that conventional CSNI schemes may not detect the attacker solely based on the clock skews as discussed in [108]. However, the difference in RSSI between the attacker and node 10 enables centralized NISA, which takes into account both clock features and radio information during the node identification, to detect the attacks. Note that the LQI difference is not evident as that of RSSI, where there are many intersections; this suggests that LQI alone is not enough

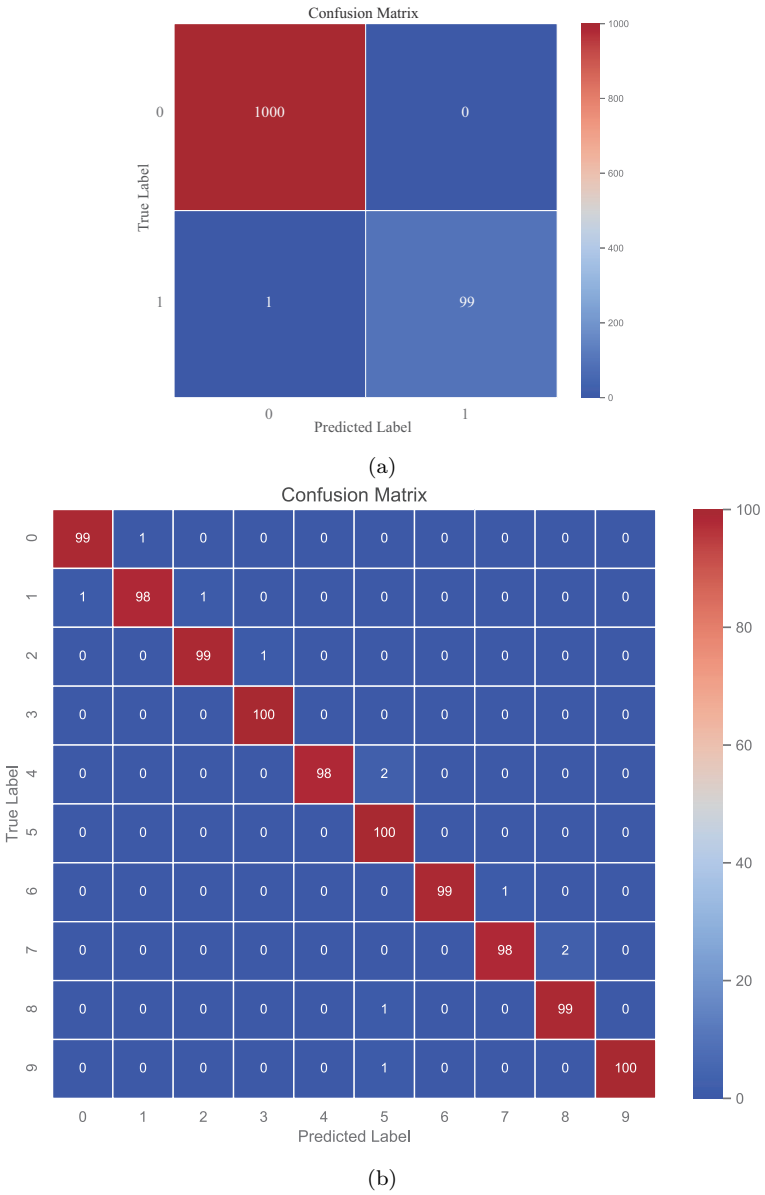
for attack detection, though it could supplement RSSI for further improvement of its performance.

The confusion matrix shown in Fig. 7.13 (a) summarizes the results of 1,100 trials of detecting attacks, 100 of which are true attacks originated by the attacker targeting at sensor node 10 (label 9). The x- and y-axis list the predicted and true labels where 0 and 1 indicate "no attack" and "under attack", respectively; when a predicted label matches a true label, we consider it as correct detection. The lower right corner of Fig. 7.13 (a) shows our detection for actual attacks, in which 99 out of 100 attacks are correctly detected (i.e., attack detection rate of 99%). The reason for this high detection rate is the high radio information difference between the attacker and the legitimate sensor node, even though the attacker is located very close to the legitimate sensor node (i.e., just 1-cm gap between them), which could demonstrate the effectiveness of the proposed system for detecting the attackers.

The confusion matrix shown in Fig. 7.13 (b) is for node identification which is carried out simultaneously with the attack detection. As discussed with Fig. 7.13 (a), there is one case where the fake data segment received from the attacker has been considered as a legitimate one due to a false attack detection. As a consequence, one additional node identification is done for node 10 of label 9 (i.e., 101 times in total unlike 100 times for other legitimate sensor nodes), while 99 out of 100 attacks are filtered out for node identification based on the results from the attack detection. Compared to the results shown in Fig. 7.10, we found that the effect of Spoofing attacks on node identification has been negligible thanks to the highly-successful attack detection.

7.5.4 Performance of Distributed NISA under Multi-Hop Scenarios

For the evaluation of the performance of distributed NISA, we change the network topology to that of the multi-hop WSN shown in Fig. 7.6 (b). As in Section 7.5.3, we also carry out the performance evaluation without and under Spoofing attacks.



(a)

(b)

Figure 7.13: Experimental results of centralized NISA for simultaneous (a) attack detection and (b) node identification for 10 sensor nodes and 1 attacker each for 100 times [7].

7.5.4.1 Without Spoofing Attacks

We evaluate the performance of distributed NISA in the multi-hop WSN shown in Fig. 7.6 (b), where the NISA system is implemented at the head and gateway nodes,

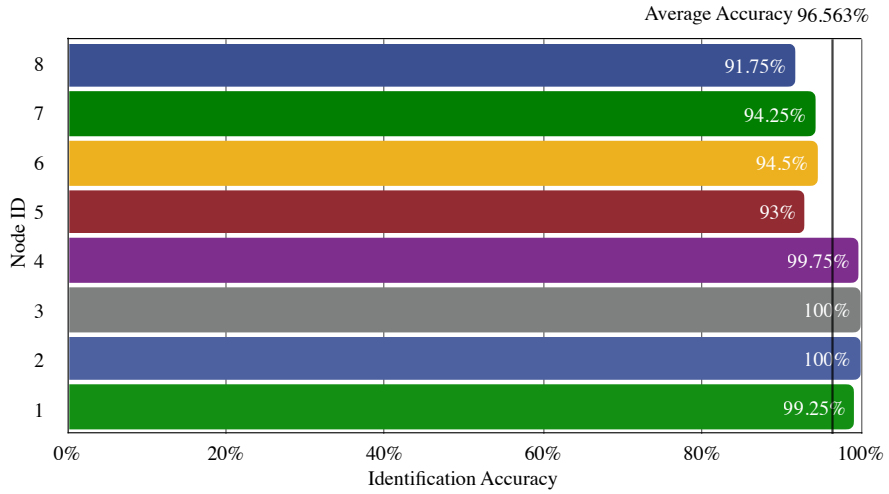


Figure 7.14: Experimental results of distributed NISA for identifying 8 gateway and sensor nodes without Spoofing attacks [7].

i.e., nodes 0, 1, and 2. The head node 0 and gateway nodes 1 and 2 are responsible for the node identification and attack detection for gateway nodes 1-2, sensor nodes 3-5, and sensor nodes 6-8, respectively.

As discussed in Section 7.4.1, distributed NISA drops the packet at either case of failed node identification or attack detection. A packet drop without the existence of attackers, therefore, indicates failed node identification. Fig. 7.14 demonstrates the evaluation results where distributed NISA could achieve then overall node identification accuracy of 96.563%. The identification of node 8 is the worst with the accuracy of 91.75% and that of nodes 2 and 3 are the best with the accuracy of 100%. This performance difference may result from the fluctuations in the radio information, which the simple thresholding cannot handle well.

7.5.4.2 Under Spoofing Attacks

To evaluate the performance of distributed NISA under Spoofing attacks, we add one sensor node working as an attacker as in Section 7.5.3.2. To comprehensively demonstrate the performance of distributed NISA under Spoofing attacks, we focus on nodes 8 and 3 in the experiments, which provide the worst and best node identification accuracy

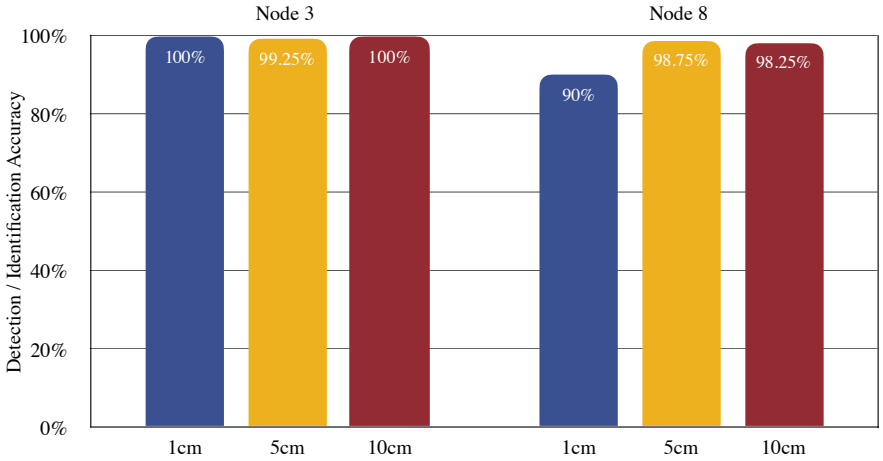


Figure 7.15: Experimental results of distributed NISA running at gateway nodes: Detection/identification accuracy of node 3 and node 8 with spoofing attacks at distances of 1 cm, 5 cm, and 10 cm [7].

from the experiments without Spoofing attacks. In addition, we launch the attacks from different distances—i.e., 1 cm, 5 cm, and 10 cm—to the legitimate sensor nodes. Fig. 7.15 illustrates the combined accuracy of attack detection and node identification—called "detection/identification accuracy" in the following—of distributed NISA for nodes 8 and 3 under Spoofing attacks. For node 3, the detection/identification accuracy of above 99% is achieved for all the distances. For node 8, on the other hand, the detection/identification accuracy increases from 90% to above 98% as the attack distance increases from 1 cm to 10 cm; this is because the channel condition of the attacker differs more from that of the legitimate sensor node as the distance between the two increases.

7.6 Comparison to Related Work

Changing Time Synchronization Interval: In [107], based on their investigations on the Spoofing attack countering CSNI, the authors observe that two main factors should be considered in spoofing clock skew, namely the difference between two consecutive offsets and the synchronization interval, which we call $\Delta\theta$ and SI, respectively. Changing $\Delta\theta$ could puzzle the Spoofing attacker in predicting the correct difference in the next pe-

riod; the clock skew in this circumstance, however, becomes unstable & fluctuating and therefore being unpredictable, which further causes false identification. Consequently, they shift their focus on the latter factor of SI whose value in most time synchronization schemes (e.g., [18, 1]) is fixed. Based on their observations that changing SI could cause an immediate impact on the precision of attacker's imitating approach—i.e., wrong calculation of fake timestamps induced by miscalculation of SI—while the effectiveness of identification is maintained, they employ this method for defending against Spoofing attack in CSNI. The performance is evaluated through the experiments on a real testbed based on Taroko motes [129], which demonstrates that the proposed method reduces the success rate of an attack to less than 2.4%. Note that, however, this method requires frequent changing of SI; it is in fact changed every 7 rounds of synchronizations in their underlying time synchronization scheme. This changing period method requires an empirical value that demanding experiments in advance for its acquisition. In addition, it has certain assumptions on the clock skew calculation capability of the attacker, e.g., the attacker could not resume its accuracy in estimating the clock skew in certain synchronization rounds. Note that, unlike NISA, this method requires changes in existing synchronization schemes on the resource-constrained sensor nodes.

RSS Distribution (RSSD)-Based Fingerprinting: In [108], the authors experimentally demonstrate the vulnerability of CSNI to the Spoofing attack and, as a result, change the base of node identification from clock skews to spatially-correlated RSSs instead of reinforcing CSNI itself as in [107]. Specifically, a set of RSSs measured at a sensor node for its neighboring nodes becomes a fingerprint for the sensor node. Note that, employing RSS for detecting the Spoofing attack is not new in the literature; for instance, the effectiveness of detecting and localizing the Spoofing attackers has been systematically investigated in [112, 113]. Therefore, the RSSD-based fingerprinting proposed in [108] could identify sensor nodes based on their RSS fingerprints while defending the Spoofing attack.

Note that RSSD-based fingerprinting is different from NISA in that its node identification is solely based on radio information: In NISA, on the other hand, the radio

information is used only for detecting the Spoofing attack. Due to the random fluctuation of a signal, the noise from multi-path effects, and the device dependency in RSS measurements [130], the RSSD-based fingerprinting has an innate limit in its identifiability compared to CSNI. The use of radio information in NISA, on the other hand, is dedicated for attack detection, which is binary classification and, therefore, does not require fine-grained identifiability. As we discussed in Section 7.1, the use of RSS for node identification would make the RSSD-based fingerprinting unsuitable for large-scale scenarios due to its coarse-grained identifiability.

7.7 Summary

In this chapter, we have proposed a new CSNI framework called NISA, where node identification is protected from Spoofing attacks based on spatially-correlated radio information. We have also provided two variations of NISA system implementation—called *centralized NISA* and *distributed NISA*, respectively—to cover both single- and multi-hop scenarios.

In distributed NISA, node identification and attack detection are done independently with a dedicated algorithm for each, which could significantly lower the implementation complexity and, hence, enables distributed NISA to be implemented and run at gateway nodes (i.e., cluster heads) as well as the head. Centralized NISA, on the other hand, could provide higher performance through its integrated processing of node identification and attack detection using a common algorithm for both based on the time-series of combined clock features and radio information. This synergy of the two processes that are separately handled in distributed NISA, however, comes at the expense of the higher implementation complexity and would limit its application to single-hop scenarios, i.e., running only at the head.

To investigate the identifiability of clock skews under various working conditions and the performance/complexity tradeoff between centralized and distributed NISA in single-hop and multi-hop WSNs, we have carried out a series of experiments on a

real WSN testbed consisting of 11 TelosB motes running TinyOS. The experimental results demonstrate the identifiability of clock skews and their concurrent behaviors under temperature and voltage variations, which are enabled by the high-precision clock parameter estimation of BATS. We could also observe from the experimental results that the feasibility and effectiveness of node identification and Spoofing attack detection of centralized and distributed NISA are proper for single-hop and multi-hop scenarios, respectively.

Although we have provided the initial design, implementation, and performance evaluation of both centralized and distributed NISA in this chapter, there is still room for further improvements: Centralized NISA has the potential to use not only other neural network architectures like RNN (including its variations of LSTM and GRU) and hybrid of CNN and RNN but also conventional machine learning algorithms such as random forest, which could have different performance and computational complexity; distributed NISA, on the other hand, could be improved by adopting more advanced thresholding and parameter fusion techniques.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we have presented our work on the development of WSN time synchronization schemes and their applications. Specifically, we have proposed three WSN time synchronization schemes to address the important aspects of energy-efficiency, computational complexity, and multi-hop synchronization accuracy. We have also applied the proposed schemes to optimal data bundling and node identification.

In Chapter 3, we investigated the *computational complexity* of WSN time synchronization with a major focus on the precision loss in floating-point arithmetic and its impact on time synchronization and proposed the asymmetric high-precision time synchronization. Specifically, the proposed scheme reassigned all synchronization-related computations from the resource-constrained sensor nodes to the head equipped with abundant computing and power resources and, therefore, significantly relieved the computational complexity of time synchronization at sensor nodes.

In Chapter 4, we concentrated on the *energy-efficiency* of WSN time synchronization and proposed the reverse asymmetric time synchronization framework and a beaconless asymmetric energy-efficiency time synchronization based on it. In particular, the proposed scheme completely eliminates the use of beacon messages for time synchronization and bundles synchronization-related data with regular application data in the

same message to reduce the number of message transmissions. The results of performance evaluation on a real WSN testbed demonstrate that the proposed scheme could achieve microsecond-level time synchronization accuracy compared to the benchmark scheme while significantly reducing energy consumption.

In Chapter 5, focusing the investigation on the *multi-hop synchronization accuracy* of WSN time synchronization, we proposed the per-hop delay compensation to enhance the performance and systematically analyzed the synchronization errors caused by the multi-hop extension methods of time translation and per-hop delay compensation. We evaluated the two methods with three representative WSN time synchronization schemes. Extensive experimental results demonstrated the effectiveness of the proposed scheme in improving the multi-hop time synchronization performance.

In Chapter 6, we applied our proposed time synchronization scheme to the optimal data bundling with delay and synchronization constraints. Formulating the optimal bundling as an integer linear programming problem, we could maximize the number of bundled data to minimize the energy consumption under the constraints of user-defined delay and synchronization requirements. By implementing and evaluating the optimal bundling scheme on a real WSN testbed, we showed that the number of message transmissions could be greatly reduced.

In Chapter 7, we applied our proposed time synchronization scheme to node identification, which is considered an efficient security measure for WSNs. Based on clock features and radio information, we proposed the novel clock-skew-based node identification scheme against Spoofing attacks. We provided both centralized and distributed design and implementation of the proposed scheme for covering both single- and multi-hop scenarios. Performance evaluation results for identifying legitimate sensor nodes and defending against a Spoofing attacker clearly demonstrate the effectiveness of the proposed scheme.

8.2 Future Work

The work presented in this thesis could be extended in the aspects of WSN time synchronization and its application. For WSN time synchronization, the following topics could be further investigated:

- The scalability of the proposed WSN time synchronization schemes could be studied through simulations or an actual testbed with a large number of sensor nodes.
- More advanced clock parameter estimation methods could be proposed to enhance the performance of the centralized time synchronization at the head.
- The use of machine learning techniques (e.g., artificial neural networks) could be employed for WSN time synchronization.

For the application of WSN time synchronization, the following extensions could be considered:

- The proposed optimal bundling scheme could be extended with more advanced optimization algorithms or covering more metrics such as link quality.
- The proposed node identification against Spoofing attack scheme could be extended with more advanced classification methods for both its centralized and distributed variations.

Bibliography

[1] X. Huan, K. S. Kim, S. Lee, E. G. Lim, and A. Marshall, "A beaconless asymmetric energy-efficient time synchronization scheme for resource-constrained multi-hop wireless sensor networks," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1716–1730, Mar. 2020.

[2] K. S. Kim, S. Lee, and E. G. Lim, "Energy-efficient time synchronization based on asynchronous source clock frequency recovery and reverse two-way message exchanges in wireless sensor networks," *IEEE Trans. Commun.*, vol. 65, no. 1, pp. 347–359, Jan. 2017.

[3] X. Huan and K. S. Kim, "On the practical implementation of propagation delay and clock skew compensated high-precision time synchronization schemes with resource-constrained sensor nodes in multi-hop wireless sensor networks," *Comput. Netw.*, vol. 166, p. 106959, Jan. 2020.

[4] X. Huan and K. S. Kim, "Per-hop delay compensation in time synchronization for multi-hop wireless sensor networks based on packet-relaying gateways," *IEEE Commun. Lett.*, vol. 24, no. 10, pp. 2300–2304, Oct. 2020.

[5] X. Huan, K. S. Kim, S. Lee, E. G. Lim, and A. Marshall, "Improving multi-hop time synchronization performance in wireless sensor networks based on packet-relaying gateways with per-hop delay compensation," *IEEE Trans. Commun.*, 2021, under 2nd round review.

[6] X. Huan, K. S. Kim, S. Lee, and M. K. Kim, "Optimal message bundling with delay and synchronization constraints in wireless sensor networks," *Sensors*, vol. 19, no. 18, Sep. 2019.

[7] X. Huan, K. S. Kim, and J. Zhang, "NISA: Node identification and spoofing attack detection based on clock features and radio information for wireless sensor networks," *IEEE Trans. Commun.*, pp. 1–1, 2021, Early Access.

[8] M. Carminati, O. Kanoun, S. L. Ullo, and S. Marcuccio, "Prospects of distributed wireless sensor networks for urban environmental monitoring," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 34, no. 6, pp. 44–52, Jun. 2019.

[9] B. Maag, Z. Zhou, and L. Thiele, "A survey on sensor calibration in air pollution monitoring deployments," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4857–4870, Dec. 2018.

[10] F. Luna, J. F. Valenzuela-Valdes, S. Sendra, and P. Padilla, "Intelligent wireless sensor network deployment for smart communities," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 176–182, Aug. 2018.

[11] R. Du, P. Santi, M. Xiao, A. V. Vasilakos, and C. Fischione, "The sensable city: A survey on the deployment and management for smart city monitoring," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1533–1560, Secondquarter 2019.

[12] Y. Zhong, Y. Yang, X. Zhu, Y. Huang, E. Dutkiewicz, Z. Zhou, and T. Jiang, "Impact of seasonal variations on foliage penetration experiment: A WSN-based device-free sensing approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5035–5045, Sep. 2018.

[13] Y. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 124–138, Jan. 2011.

[14] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Netw.*, vol. 3, no. 3, pp. 281–323, Feb. 2005.

[15] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Netw.*, vol. 18, no. 4, pp. 45–50, Jul. 2004.

[16] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. SenSys'03*, Nov. 2003, pp. 138–149.

[17] M. Akhlaq and T. R. Sheltami, "RTSP: An accurate and energy-efficient protocol for clock synchronization in WSNs," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 3, pp. 578–589, Mar. 2013.

[18] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi, "The flooding time synchronization protocol," in *Proc. 2nd Int. Conf. SenSys*, Nov. 2004, pp. 39–49.

[19] C. Lenzen, P. Sommer, and R. Wattenhofer, "Pulsesync: An efficient and scalable clock synchronization protocol," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 717–727, Jun. 2015.

[20] H. Wang, F. Yu, M. Li, and Y. Zhong, "Clock skew estimation for timestamp-free synchronization in industrial wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 90–99, Jan. 2021.

[21] G. S. S. Chalapathi, V. Chamola, S. Gurunarayanan, and B. Sikdar, "E-SATS: An efficient and simple time synchronization protocol for cluster-based wireless sensor networks," *IEEE Sensors J.*, vol. 19, no. 21, pp. 10 144–10 156, Nov. 2019.

[22] F. Shi, X. Tuo, S. X. Yang, J. Lu, and H. Li, "Rapid-flooding time synchronization for large-scale wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 1581–1590, Mar. 2020.

[23] T. Qiu, X. Liu, M. Han, M. Li, and Y. Zhang, "SRTS : A self-recoverable time synchronization for sensor networks of healthcare IoT," *Comput. Netw.*, vol. 129, pp. 481 – 492, May 2017.

[24] A. M. Joshi, U. P. Shukla, and S. P. Mohanty, "Smart healthcare for diabetes during COVID-19," *IEEE Consum. Electron. Mag.*, vol. 10, no. 1, pp. 66–71, Jan. 2021.

[25] TelosB datasheet. http://www2.ece.ohio-state.edu/~bibyk/ee582/telosMote.pdf. Accessed: 2021-04-15.

[26] K. S. Yildirim and A. Kantarci, "Time synchronization based on slow-flooding in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 244–253, Jan. 2014.

[27] ——, "External gradient time synchronization in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 633–641, Mar. 2014.

[28] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, p. 147–163, Dec. 2003.

[29] D. Djenouri, N. Merabtine, F. Z. Mekahlia, and M. Doudou, "Fast distributed multi-hop relative time synchronization protocol and estimators for wireless sensor networks," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2329–2344, Jun. 2013.

[30] G. Cena, S. Scanzio, A. Valenzano, and C. Zunino, "Implementation and evaluation of the reference broadcast infrastructure synchronization protocol," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 801–811, Jun. 2015.

[31] F. Gong and M. L. Sichitiu, "CESP: A low-power high-accuracy time synchronization protocol," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2387–2396, Apr. 2016.

[32] J. He, P. Cheng, L. Shi, J. Chen, and Y. Sun, "Time synchronization in WSNs: A maximum-value-based consensus approach," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 660–675, Mar. 2014.

[33] A. Saïah, C. Benzaïd, and N. Badache, "CMTS: Consensus-based multi-hop time synchronization protocol in wireless sensor networks," in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, Oct. 2016, pp. 232–236.

[34] T. Anglea and Y. Wang, "Phase desynchronization: A new approach and theory using pulse-based interaction," *IEEE Trans. Signal Process.*, vol. 65, no. 5, pp. 1160–1171, Mar. 2017.

[35] Y. Zong, X. Dai, Z. Gao, K. Busawon, R. Binns, and I. Elliott, "Synchronization of pulse-coupled oscillators for IEEE 802.15.4 multi-hop wireless sensor networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018, pp. 1–7.

[36] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, "The evolution of MAC protocols in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 101–120, First 2013.

[37] R. C. Carrano, D. Passos, L. C. S. M. aes, and C. V. N. Albuquerque, "Survey and taxonomy of duty cycling mechanisms in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 181–194, First 2014.

[38] Y. Liu, D. Zhang, X. Guo, M. Gao, Z. Ming, L. Yang, and L. M. Ni, "RSS-based ranging by leveraging frequency diversity to distinguish the multiple radio paths," *IEEE Trans. Mobile Comput.*, vol. 16, no. 4, pp. 1121–1135, Apr. 2017.

[39] D. Zhang, Y. Liu, X. Guo, and L. M. Ni, "RASS: A real-time, accurate, and scalable system for tracking transceiver-free objects," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 5, pp. 996–1008, May 2013.

[40] J. P. B. Nadas, R. D. Souza, M. E. Pellenz, G. Brante, and S. M. Braga, "Energy efficient beacon based synchronization for alarm driven wireless sensor networks," *IEEE Signal Process. Lett.*, vol. 23, no. 3, pp. 336–340, Mar. 2016.

[41] A. Al-Shaikhi and A. Masoud, "Efficient, single hop time synchronization protocol for randomly connected wsns," *IEEE Wireless Commun. Lett.*, vol. 6, no. 2, pp. 170–173, Apr. 2017.

[42] L. Tavares Bruscato, T. Heimfarth, and E. Pignaton de Freitas, "Enhancing time synchronization support in wireless sensor networks," *Sensors*, vol. 17, no. 12, Dec. 2017.

[43] H. Wang, L. Shao, M. Li, and P. Wang, "Estimation of frequency offset for time synchronization with immediate clock adjustment in multihop wireless sensor networks," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2239–2246, Dec 2017.

[44] TinyOS. https://github.com/tinyos/tinyos-main. Accessed: 2021-04-15.

[45] MicaZ datasheet. http://courses.ece.ubc.ca/494/files/MICAz_Datasheet.pdf. Accessed: 2021-04-15.

[46] IriS datasheet. http://www.nr2.ufpr.br/~adc/documentos/iris_datasheet.pdf. Accessed: 2021-04-15.

[47] Arduino platforms. https://www.arduino.cc/. Accessed: 2021-04-15.

[48] J.-P. Sheu, W.-K. Hu, and J.-C. Lin, "Ratio-based time synchronization protocol in wireless sensor networks," *Telecommunication Systems*, vol. 39, no. 1, pp. 25–35, Sep. 2008.

[49] IEEE Computer Society, *IEEE Std 754^{TM}-2008, IEEE Standard for floating-point arithmetic*, Std., 2008.

[50] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," in *Proc. PDLI 2003*, Jun. 2003.

[51] Float format in nesC. https://github.com/tinyos/nesc/blob/master/libiberty/floatformat.c. Accessed: 2021-04-15.

[52] K. S. Kim, "Asynchronous source clock frequency recovery through aperiodic packet streams," *IEEE Commun. Lett.*, vol. 17, no. 7, pp. 1455–1458, Jul. 2013.

[53] H. Yiğitler, B. Badihi, and R. Jäntti, "Overview of time synchronization for iot deployments: Clock discipline algorithms and protocols," *Sensors*, vol. 20, no. 20, Oct. 2020.

[54] H. Wang, L. Shao, M. Li, B. Wang, and P. Wang, "Estimation of clock skew for time synchronization based on two-way message exchange mechanism in industrial wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4755–4765, Nov. 2018.

[55] M. Jin, T. Xing, X. Chen, X. Meng, D. Fang, and Y. He, "Dualsync: Taming clock skew variation for synchronization in low-power wireless networks," in *Proc. IEEE INFOCOM 2016*, Apr. 2016, pp. 1–9.

[56] K. S. Yildirim and O. Gurcan, "Efficient time synchronization in a wireless sensor network by adaptive value tracking," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3650–3664, Jul. 2014.

[57] S. Lin, F. Miao, J. Zhang, G. Zhou, L. Gu, T. He, J. A. Stankovic, S. Son, and G. J. Pappas, "ATPC: Adaptive transmission power control for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 12, no. 1, pp. 6:1–6:31, Mar. 2016.

[58] A. Pal and A. Nasipuri, "Joint power control and routing for rechargeable wireless sensor networks," *IEEE Access*, vol. 7, pp. 123 992–124 007, 2019.

[59] W. Dong, C. Chen, X. Liu, Y. He, Y. Liu, J. Bu, and X. Xu, "Dynamic packet length control in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 3, pp. 1172–1181, Mar. 2014.

[60] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[61] B. Zhou and M. C. Vuran, "Cortis: Correlation-based time synchronization in internet of things," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–7.

[62] S. el Khediri, N. Nasri, M. Samet, A. Wei, and A. Kachouri, "Analysis study of time synchronization protocols in wireless sensor networks," *arXiv e-prints*, pp. 1–15, Jun. 2012.

[63] M. V. Barel, G. Heinig, and P. Kravanja, "A superfast method for solving toeplitz linear least squares problems," *Linear Algebra and its Applications*, vol. 366, pp. 441–457, 2003, special issue on Structured Matrices: Analysis, Algorithms and Applications.

[64] B. Kusý, P. Dutta, P. Levis, M. Maróti, Ákos Lédeczi, and D. Culler, "Elapsed time on arrival; a simple and versatile primitive for canonical time synchronisation services," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 1, no. 4, pp. 239–251, Jul. 2006.

[65] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Comput. Surv.*, no. 1, pp. 5–48, Mar. 1991.

[66] G. Cena, S. Scanzio, and A. Valenzano, "A neural network clock discipline algorithm for the rbis clock synchronization protocol," in *Proc. 14th IEEE International Workshop on Factory Communication Systems*, Jun. 2018, pp. 1–10.

[67] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 10, no. 1, pp. 16:1–16:49, Dec. 2013.

[68] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, third quarter 2017.

[69] A Designer's Guide to Instrumentation Amplifiers, 3RD Edition, 2006. https://www.analog.com/en/education/education-library/dh-designers-guide-to-instrumentation-amps.html. Accessed: 2021-04-15.

[70] J. M. Castillo-Secilla, J. M. Palomares, and J. Olivares, "Temperature-compensated clock skew adjustment," *Sensors*, vol. 13, no. 8, pp. 10 981–11 006, Aug. 2013.

[71] G. Cena, S. Scanzio, and A. Valenzano, "A neural network clock discipline algorithm for the RBIS clock synchronization protocol," in *Proc. 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, Jun. 2018, pp. 1–10.

[72] P. Giri, K. Ng, and W. Phillips, "Wireless sensor network system for landslide monitoring and warning," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 4, pp. 1210–1220, Apr. 2019.

[73] F. Cirillo, D. Gómez, L. Diez, I. Elicegui Maestro, T. B. J. Gilbert, and R. Akhavan, "Smart city IoT services creation through large-scale collaboration," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5267–5275, Jun. 2020.

[74] T. Ojha, S. Misra, and N. S. Raghuwanshib, "Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges," *Computers and Electronics in Agriculture*, vol. 118, pp. 66–84, Oct. 2015.

[75] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, J. Zhao, and Xiang-Yang Li, "Does wireless sensor network scale? a measurement study on GreenOrbs," in *Proc. 2011 IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 873–881.

[76] L. Xiao, X. Lu, D. Xu, Y. Tang, L. Wang, and W. Zhuang, "UAV relay in VANETs against smart jamming with reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4087–4097, May 2018.

[77] B. Etzlinger, N. Palaoro, W. Haselmayr, B. Rudić, and A. Springer, "Timestamp free synchronization with sub-tick accuracy in the presence of discrete clocks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 771–783, Feb. 2017.

[78] R. T. Rajan and A.-J. van der Veen, "Joint ranging and clock synchronization for a wireless network," in *Proc. CAMSAP 2011*, Dec. 2011, pp. 297–300.

[79] Texas Instruments: Selection and specification of crystals for Texas Instruments USB 2.0 devices. https://www.ti.com/lit/an/slla122/slla122.pdf. Accessed: 2021-04-15.

[80] Public FTSP implementation in TinyOS. https://github.com/tinyos/tinyos-main/blob/master/tos/lib/ftsp. Accessed: 2021-04-15.

[81] D. Djenouri and M. Bagaa, "Implementation of high precision synchronization protocols in wireless sensor networks," in *Proc. WOCC 2014*, May 2014, pp. 1–6.

[82] J. Cui and F. Valois, "Data aggregation in wireless sensor networks: Compressing or forecasting?" in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2014, pp. 2892–2897.

[83] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 8, no. 4, pp. 48–63, Fourth 2006.

[84] L. A. Villas, A. Boukerche, H. S. Ramos, H. A. B. F. de Oliveira, R. B. de Araujo, and A. A. F. Loureiro, "Drina: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks," *IEEE Trans. Comput.*, vol. 62, no. 4, pp. 676–689, Apr. 2013.

[85] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proceedings of IEEE 22nd International Conference on Distributed Computing Systems*, Jul. 2002, pp. 457–458.

[86] X. Xu, X. Y. Li, X. Mao, S. Tang, and S. Wang, "A delay-efficient algorithm for data aggregation in multihop wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 1, pp. 163–175, Jan. 2011.

[87] X. Deng and Y. Yang, "Online adaptive compression in delay sensitive wireless sensor networks," *IEEE Trans. Comput.*, vol. 61, no. 10, pp. 1429–1442, Oct. 2012.

[88] M. Doudou, D. Djenouri, and N. Badache, "Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 528–550, Second 2013.

[89] J. Wang, W. Dong, Z. Cao, and Y. Liu, "On the delay performance in a large-scale wireless sensor network: Measurement, analysis, and implications," *IEEE/ACM Trans. Netw.*, vol. 23, no. 1, pp. 186–197, Feb. 2015.

[90] X. Li, W. Liu, M. Xie, A. Liu, M. Zhao, N. N. Xiong, M. Zhao, and W. Dai, "Differentiated data aggregation routing scheme for energy conserving and delay sensitive wireless sensor networks," *Sensors*, vol. 18, no. 7, Jul. 2018.

[91] S. Fu, Y. Zhang, Y. Jiang, C. Hu, C. Shih, and P. J. Marrón, "Experimental study for multi-layer parameter configuration of WSN links," in *Proceedings of IEEE 35th International Conference on Distributed Computing Systems*, Jun. 2015, pp. 369–378.

[92] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[93] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proc. IEEE*, vol. 104, no. 9, pp. 1727–1765, Sep. 2016.

[94] L. Xiao, G. Sheng, X. Wan, W. Su, and P. Cheng, "Learning-based phy-layer authentication for underwater sensor networks," *IEEE Commun. Lett.*, vol. 23, no. 1, pp. 60–63, Jan. 2019.

[95] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 94–104, Firstquarter 2016.

[96] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," *IEEE Trans. Mobile Comput.*, vol. 9, no. 3, pp. 449–462, Mar. 2010.

[97] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 2, pp. 93–108, Apr. 2005.

[98] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the clock: Emulating clock skew in controller area networks," in *Proc. ACM/IEEE 9th International Conference on Cyber-Physical Systems*, Apr. 2018, pp. 32–42.

[99] J. Yu, A. Hu, G. Li, and L. Peng, "A robust RF fingerprinting approach using multisampling convolutional neural network," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6786–6799, Aug. 2019.

[100] D. Huang, W. Teng, C. Wang, H. Huang, and J. M. Hellerstein, "Clock skew based node identification in wireless sensor networks," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, Nov. 2008, pp. 1–5.

[101] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "GTID: A technique for physical device and device type fingerprinting," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 5, pp. 519–532, Sep. 2015.

[102] M. B. Uddin and C. Castelluccia, "Toward clock skew based wireless sensor node services," in *Proc. The 5th Annual ICST Wireless Internet Conference*, Mar. 2010, pp. 1–9.

[103] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Proc. Third International Symposium on Information Processing in Sensor Networks (IPSN)*, Apr. 2004, pp. 259–268.

[104] H. Fu, S. Kawamura, M. Zhang, and L. Zhang, "Replication attack on random key pre-distribution schemes for wireless sensor networks," in *Proc. Sixth Annual IEEE SMC Information Assurance Workshop*, Jun. 2005, pp. 134–141.

[105] Y.-C. Hu, A. Perrig, and D. Johnson, "Wormhole attacks in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 2, pp. 370–380, Feb. 2006.

[106] R. Maheshwari, J. Gao, and S. R. Das, "Detecting wormhole attacks in wireless networks using connectivity information," in *Proc. 26th IEEE International Conference on Computer Communications (INFOCOM)*, May 2007, pp. 107–115.

[107] D. Huang and W. Teng, "A defense against clock skew replication attacks in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 39, pp. 26 – 37, Mar. 2014.

[108] X. Mei, D. Liu, K. Sun, and D. Xu, "On feasibility of fingerprinting wireless sensor nodes using physical properties," in *Proc. IEEE 27th International Symposium on Parallel and Distributed Processing (IPDPS)*, May 2013, pp. 1112–1121.

[109] A. Achroufene, Y. Amirat, and A. Chibani, "RSS-based indoor localization using belief function theory," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 3, pp. 1163–1180, Jul. 2019.

[110] S. Tomic, M. Beko, and R. Dinis, "RSS-based localization in wireless sensor networks using convex relaxation: Noncooperative and cooperative schemes," *IEEE Trans. Veh. Technol.*, vol. 64, no. 5, pp. 2037–2050, May 2015.

[111] J. Zhang, T. Q. Duong, A. Marshall, and R. Woods, "Key generation from wireless channels: A review," *IEEE Access*, vol. 4, pp. 614–626, Jan. 2016.

[112] Y. Chen, J. Yang, W. Trappe, and R. P. Martin, "Detecting and localizing identity-based attacks in wireless and sensor networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 5, pp. 2418–2434, Jun. 2010.

[113] J. Yang, Y. Chen, W. Trappe, and J. Cheng, "Detection and localization of multiple spoofing attackers in wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 44–58, Jan. 2013.

[114] D. A. Knox and T. Kunz, "Wireless fingerprints inside a wireless sensor network," *ACM Trans. Sen. Netw.*, vol. 11, no. 2, Mar. 2015.

[115] B. W. Ramsey, B. E. Mullins, M. A. Temple, and M. R. Grimaila, "Wireless intrusion detection and device fingerprinting through preamble manipulation," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 5, pp. 585–596, Sep. 2015.

[116] N. M. Freris, S. R. Graham, , and P. R. Kumar, "Fundamental limits on synchronizing clocks over networks," *Proc. IEEE*, vol. 56, no. 6, pp. 1352–1364, Jun. 2011.

[117] H. Liu, J. Yang, Y. Wang, Y. Chen, and C. E. Koksal, "Group secret key generation via received signal strength: Protocols, achievable rates, and implementation," *IEEE Trans. Mobile Comput.*, vol. 13, no. 12, pp. 2820–2835, Dec. 2014.

[118] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: A survey," *ACM Trans. Sen. Netw.*, vol. 8, no. 4, Sep. 2012.

[119] N. Kuruwatti, Y. N. Nayana, N. Sarole, G. Revadigar, and C. Javali, "LQI-Key: Symmetric key generation scheme for Internet-of-Things (IoT) devices using wireless channel link quality," in *2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, Feb. 2018, pp. 1–6.

[120] W. Liu, Y. Xia, R. Luo, and S. Hu, "Lightweight, fluctuation insensitive multi-parameter fusion link quality estimation for wireless sensor networks," *IEEE Access*, vol. 8, pp. 28 496–28 511, Feb. 2020.

[121] K. Sankhe, M. Belgiovine, F. Zhou, L. Angioloni, F. Restuccia, S. D'Oro, T. Melodia, S. Ioannidis, and K. Chowdhury, "No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 1, pp. 165–178, Mar. 2020.

[122] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *J. Syst. Eng. Electron.*, vol. 28, no. 1, pp. 162–169, Feb. 2017.

[123] C. Liu, W. Hsaio, and Y. Tu, "Time series classification with multivariate convolutional neural network," *IEEE Trans. Ind. Electron*, vol. 66, no. 6, pp. 4788–4797, Jun. 2019.

[124] D. Kingma and J. Ba, "ADAM: A method for stochastic optimization," *ArXiv e-prints*, Jan. 2017.

[125] Keras. https://keras.io. Accessed: 2021-04-15.

[126] TensorFlow™. https://www.tensorflow.org/. Accessed: 2021-04-15.

[127] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[128] S. Fu, M. Ceriotti, Y. Jiang, C. Shih, X. Huan, and P. J. Marron, "An approach to detect anomalous degradation in signal strength of IEEE 802.15.4 links," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Jun. 2018, pp. 1–9.

[129] S.-Y. Lau, T.-H. Chang, S.-Y. Hu, H.-J. Huang, L. de Shyu, C.-M. Chiu, and P. Huang, "Sensor networks for everyday use: the bl-live experience," in *Proc.*

IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06), vol. 1, Taichung, Taiwan, Jun. 2006, pp. 1–7.

[130] K. S. Kim, R. Wang, Z. Zhong, Z. Tan, H. Song, J. Cha, and S. Lee, "Large-scale location-aware services in access: Hierarchical building/floor classification and location estimation using wi-fi fingerprinting based on deep neural networks," *Fiber and Integrated Optics*, vol. 37, no. 5, pp. 277–289, Apr. 2018.