20-μs Accuracy Time-Synchronization Method using Bluetooth Low Energy for Internet-of-Things Sensors

Harada, Masayasu ; Izumi, Shintaro ; Kozeni, Ryosuke ; Yoshikawa, Yukiko ; Ishii, Toru ; Kawaguchi, Hiroshi ; Uemura, Shohei ; Araki,…

(Citation)
2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC):181-186

(Issue Date)
2022-02-10

(Resource Type)
conference proceedings

(Version)
Accepted Manuscript

(URL)
https://hdl.handle.net/20.500.14094/0100483403

20-μs Accuracy Time-Synchronization Method using Bluetooth Low Energy for Internet-of-Things Sensors

Masayasu Harada, Shintaro Izumi, Ryosuke Kozeni, Yukiko Yoshikawa, Toru Ishii, Hiroshi Kawaguchi
the Graduate School of Science, Technology and Innovation
Kobe University
1-1 Rokkodai-cho, Nada-ku, Kobe, Hyogo 6578501, Japan
line 5: email address

Shohei Uemura, Kaname Araki
Technical Development Group
Kobe Steel, Ltd.
Kobe 6512271, Japan

***Abstract*— This paper presents a low-power and accurate time-synchronization method for Internet-of-Things (IoT) sensors. Time synchronization between the base station and sensor nodes is important for realizing synchronized measurement and data collection from multiple sensor nodes. The proposed method is implemented within the application layer of the Bluetooth Low Energy protocol, and it only requires a 32.768-kHz real-time clock and an active flag of a power amplifier in the transmission circuit and a low-noise amplifier in the receiver circuit. This limited hardware requirement allows for the implementation of commercially available communication modules. The synchronization performance was evaluated with eight peripheral nodes and one central node, and the measurement results indicate that a 20-μs synchronization error was achieved on average for all the eight peripherals.**

Keywords— time synchronization, sensor network, Bluetooth Low Energy

I. INTRODUCTION

Time synchronization is a key component of the recent Internet-of-Things (IoT) sensor systems and sensor networks. Many applications of these systems require time-series measurements and accurate time stamps for performing data analysis. The time stamps of the data measured at multiple points should be synchronized with the base station or server. Thus, the sensor nodes must have a real-time clock, and the system requires a time-synchronization method.

If the sensor is installed outdoors, highly accurate synchronization can be achieved with a global positioning system (GPS) or long-wave standard time radio wave. Unfortunately, these methods cannot be applied indoors or in heavily shielded areas. Alternatively, packet exchange through wireless communication and various other methods have been proposed [1-4]. For example, the network time protocol (NTP) is widely used on the Internet.

Packet-exchange schemes involve a trade-off between the synchronization accuracy and cost. When numerous sensor nodes are deployed, the manufacturing cost and power consumption of sensor nodes should be considered for practical use. Herein, we propose a time-synchronization method based on the Bluetooth 5.0 standard protocol to satisfy both accuracy and cost requirements.

Bluetooth is a communication standard defined for applications with limited power consumption (e.g., wearable devices and sensor networks). Bluetooth Low Energy (BLE), which has been added since Bluetooth 4.0, is based on the intermittent operation of the radio circuit to reduce its power consumption. The objective of this paper is to realize a low-power time synchronization method that can be implemented over the BLE. The BLE defines two communication schemes: Broadcast and Connection. Broadcast is a one-way communication, which means that data can be transmitted from one sender node to multiple receiver nodes simultaneously. In contrast, the connection mode defines bi-directional communication. Although the advantage of the broadcast mode is that it can transmit data to multiple nodes with a minimum control communication, the connection mode is employed in the proposed method because it requires bidirectional control communication.

For improving synchronization accuracy, conventional methods are often implemented in lower protocol layers. The lower the layer, the more accurate the timestamp can be since various delay factors are eliminated. The proposed method can be implemented in the application layer, whereas conventional methods are implemented in the lower protocol layer to improve the synchronization accuracy. It enables implementation using commercially available modules via a 32.768-kHz real-time clock and active flags of the transceiver circuit, which are provided by the wireless communication controller in the module.

This work is supported by Kobe Steel, Ltd.

II. TIME SYNCHRONIZATION USING WIRELESS COMMUNICATION

A. Issues of time synchronization using wireless communication

Each sensor node in the network has a different timer value driven by a built-in oscillator. In many cases, a timer is configured using a low-power real-time clock (RTC) to hold the time information. This timer must be synchronized to control the sensing timing for each node. Although there are various methods of synchronizing timer values between nodes, as mentioned earlier, we focus on the time-synchronization method using wireless communication and packet exchange.

By sending the timestamp value in the packet, the receiver can determine the gap between the sender's timer and its own timer. Although this is an effective method, regardless of the network topology, we assume a star network with one base station and multiple nodes connected by one hop. The sensor nodes send a timestamp to the base station, and the base station replies with the difference between that and its own timer. This operation synchronizes the time in the network and allows the base station to control the timing of sensing and communication.

As shown in Fig. 1, there are various error factors in time synchronization via wireless communication.

- Send : The time that elapses between the sending of a packet to the application layer and its passing to a lower layer
- Access : The time it takes at the MAC layer to convert a packet into a sequence of bits and make it available to the radio circuit
- Transmission : The time it takes to transmit a radio signal at the physical layer
- Propagation : The time it takes for a radio wave to propagate through the air
- Reception : The time it takes for a receiver to receive a radio signal at the physical layer
- Receive : The time it takes to reassemble the received bit string into a packet and pass it to the application layer

These error factors need to be addressed to improve the time-synchronization accuracy. As mentioned in the previous section, getting the timestamp at a lower layer closer to the physical layer improves the accuracy. However, it is difficult to change the lower layers of the protocol stack in commonly available communication modules, and cost is an issue when using dedicated communication devices. Developing a dedicated communication device for time synchronization is costly and impractical in many cases.

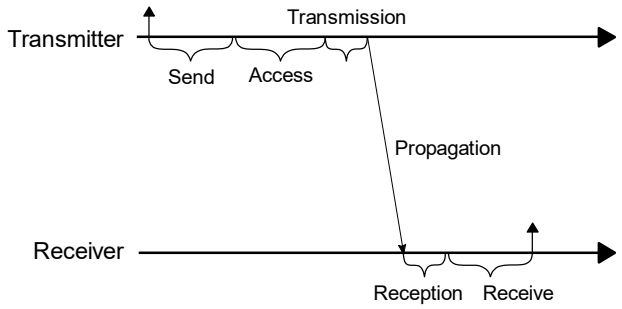

Fig. 1. Error factors in wireless communication.

The RTC performance also affects the time-synchronization accuracy. The upper limit of the synchronization accuracy is determined by the frequency of the RTC. Although the accuracy can be improved using a clock with a shorter period, 32.768-kHz RTC is generally used in applications where low power consumption is required.

The frequency variation and frequency drift of the RTC also affect the synchronization accuracy. Even if an RTC with the same specifications is used, the actual frequency will vary depending on the manufacturing variations and environmental conditions such as temperature differences. Even if the timestamp is exchanged and the offset corrected, the passage of time will cause another time shift. This problem is called clock drift. When the clock drift can be suppressed, the synchronization interval can be increased and the power dissipation can be suppressed.

B. Related works

Various time-synchronization methods have been proposed for applications that require low-power performance in IoT and sensor networks.

Global Positioning System (GPS) can be used for accurate time synchronization. GPS is a satellite-based positioning technology that is used in car navigation systems and smartphones. GPS satellites transmit positioning signals around the earth, and receivers can calculate its position using received signal from multiple satellites. The received signal and position calculation process are also used for time synchronization, because the satellites are equipped with atomic clocks, which can be synchronized with 10-ns order accuracy. However, positioning and time acquisition require data reception from at least four satellites, and it is difficult to be used indoors or in other environments with poor signal conditions. Another issue is the power consumption of the GPS receiver.

Reference broadcast synchronization (RBS) [1] uses bidirectional communication to synchronize the times of neighboring nodes. The base station broadcasts the packet to the entire network, and each node records the reception time. Subsequently, the nodes exchange the received times with each other to achieve time synchronization. As the number of nodes increases, the accuracy improves, but the power consumption increases.

The timing sync protocol for a sensor network (TSPN) [2] also realizes time synchronization using bidirectional communication, similar to the NTP, which is a time-synchronization protocol used in the Internet. WPTP [4] is a synchronization protocol for multi-hop wireless networks. This algorithm is an extension of the PTP and it can reduce the convergence time and number of packets required for synchronization.

WPTP [3] is a time synchronization protocol designed for wireless networks, based on the principles of PTP, a time synchronization protocol for computers. By taking advantage of

the characteristics of broadcast communication, WPTP reduces the number of packets required without compromising synchronization accuracy.

The flooding time-synchronization protocol (FTSP) [4] is a method that achieves highly accurate synchronization with only simple one-way communication. By obtaining timestamps at the sending and receiving MAC layers when packets are first sent and immediately after they are received, errors in the MAC layer are eliminated as much as possible, thereby improving accuracy. However, since FTSP requires access to low-level layers, it is often not feasible for commercial devices.

These protocols need to be implemented in layers lower than the media access control layer because the timestamp accuracy affects the accuracy. Therefore, these methods assume a dedicated protocol. It is necessary to process the data in the layer close to the physical layer to attach an accurate timestamp without the influence of fluctuation in processing time from the application layer to the physical layer.

Reference [5] shows that IoT devices have a serious clock drift problem, and it proposes a clock compensation method with 15-ms accuracy. Reference [6] reports that the accuracy of time synchronization using BLE connections is ±750 µs. CheepSync [7] is a time-synchronization method that uses an advertisement mode of Bluetooth 4.2 (BLE) protocol. It aims to synchronize the time between BLE beacons and smartphones and achieves an average time-synchronization accuracy of 10 µs. Another method [8] also uses the BLE advertising mode. The experimental results with two sensor nodes and one base station indicate that an average error of 3.2 µs was achieved without resynchronization for 10 min. However, it has a disadvantage in terms of power consumption because it requires a high-frequency (16 MHz) timer clock and additional control communications using proprietary protocols.

Reference [9] presents a method that uses the module's transceiver active flag and hardware timestamp to obtain an accurate timestamp. The shunt resistor is used to obtain the exact active time of the radio transmitter and receiver; it is used for time stamping to minimize the error factors mentioned earlier.

III. Proposed Method

We aimed to develop a time-synchronization method that can be implemented in commercially available modules. To solve the problems encountered in previous research, we propose a method that can be implemented in the application layer of the Bluetooth protocol and can work with a 32.768-kHz RTC.

A. Timestamp acquisition and wireless communication

The protocol stack processing time of the microcontroller depends on factors such as the timing of interrupt signals and memory accesses. To eliminate such effects, it is necessary for the microcontroller to accurately determine the operation timing of the transmitter and receiver circuits. There are several methods of achieving this; a previously developed method [9] uses an external shunt resistor. Some commercially available wireless microcontrollers can notify the user program of the active timing of the transmitter and receiver circuits [10]. This function allows the application layer to know the operation timing of the transmitter's PA and the receiver's LNA. This notification function is also used in our proposed method. In Fig. 2, TS_{PA} and TS_{LNA} respectively denote the time when the PA and LNA became active.

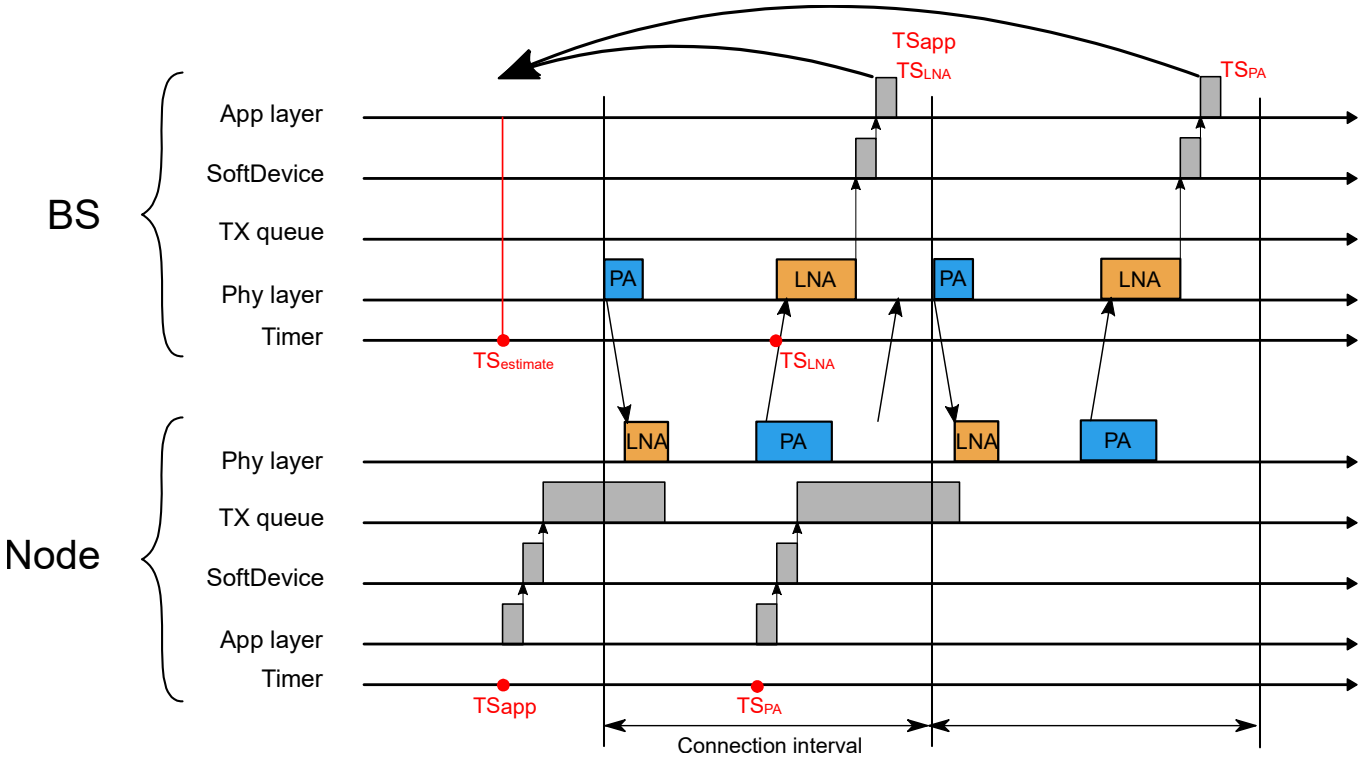

Fig. 2. Details of transmission delay and timer offset estimation sequence using wireless communication.

The next issue is to determine when the timestamp should be sent as a packet. Because the sender cannot obtain this timestamp until the physical layer is processed, the application-layer program cannot add a timestamp to the same packet. To address this problem, the proposed method divides communication into two stages. As shown in Fig. 2, in response to a synchronization request from the BS, the node first sends its own timestamp. When a node activates the radio circuit to transmit the first packet, it sends a notification to the application layer to obtain a timestamp. This timestamp is sent to the BS in the second packet. The BS can use these two timestamps to determine the delay caused by the node's first packet from the application layer to the physical layer. The following flow summarizes the BS and Node operations.

1. The timestamp when the Node adds a data packet to the queue is flagged as TS_{app}.
2. Node acquires TS_{PA} when PA goes High when the flag is on.
3. TS_{app} is sent in the first packet from Node to BS.
4. In BS, TS_{LNA} is updated every time LNA goes High.
5. Get TS_{LNA} when data is received in BS.
6. The Node sends TS_{PA} in the second packet.
7. In BS, these three timestamps can be used to estimate the $TS_{estimate}$.

B. *Timer offset calculation*

Next, the BS uses the obtained timestamp to synchronize the timer values of the nodes.

As mentioned earlier, the node sends two timestamps to the BS: TS_{app} and TS_{PA}. TS_{app} is a timestamp that serves as a reference for synchronization, and TS_{PA} is a correction timestamp that indicates when the transmitter's power amplifier (PA) is activated. Assuming that there is an offset between the BS and node timer values, this process aims to estimate the timing of TS_{app} at the BS timer value ($TS_{estimate}$ in Fig. 2).

In addition to the two timestamps received, the BS obtains the active time of the receiver's low-noise amplifier (LNA) as TS_{LNA} when TS_{app} is received. $TS_{estimate}$ can be estimated as follows:

$$TS_{estimate} = TS_{LNA} - (TS_{PA} - TS_{app}) - T_{prop} \quad (1)$$

Here, T_{prop} is a transmission delay including propagation delay, and it can be considered a fixed value obtained experimentally. Then, the timer offset, T_e, between the BS and node can be calculated from the difference between $TS_{estimate}$ and TS_{app}.

C. *Time synchronization and clock drift cancelation with multiple nodes*

Time synchronization of multiple nodes is performed using a time-division approach and the timer offset T_e obtained in the previous section. Fig. 3 presents a timing chart of the synchronization process with multiple nodes. In this synchronization process, the operation of the protocol stack of each node is the same as in Fig. 2. A time slot is set for each node, and the BS synchronizes each node sequentially. In each time slot, the BS first tells the target node to start the synchronization process. Next, the timer offset T_e is calculated according to the manner described earlier. At the end of each time slot, the BS returns the offset information to the node to complete the synchronization. Once the synchronization with all nodes is completed, sensing can be done simultaneously with synchronous timing.

Each node's timer runs on its own RTC, but the RTC frequency varies depending on the manufacturing process and the surrounding environment. The clock drift caused by frequency variation and fracture is the cause of the timer deviation after the completion of synchronization. Although this

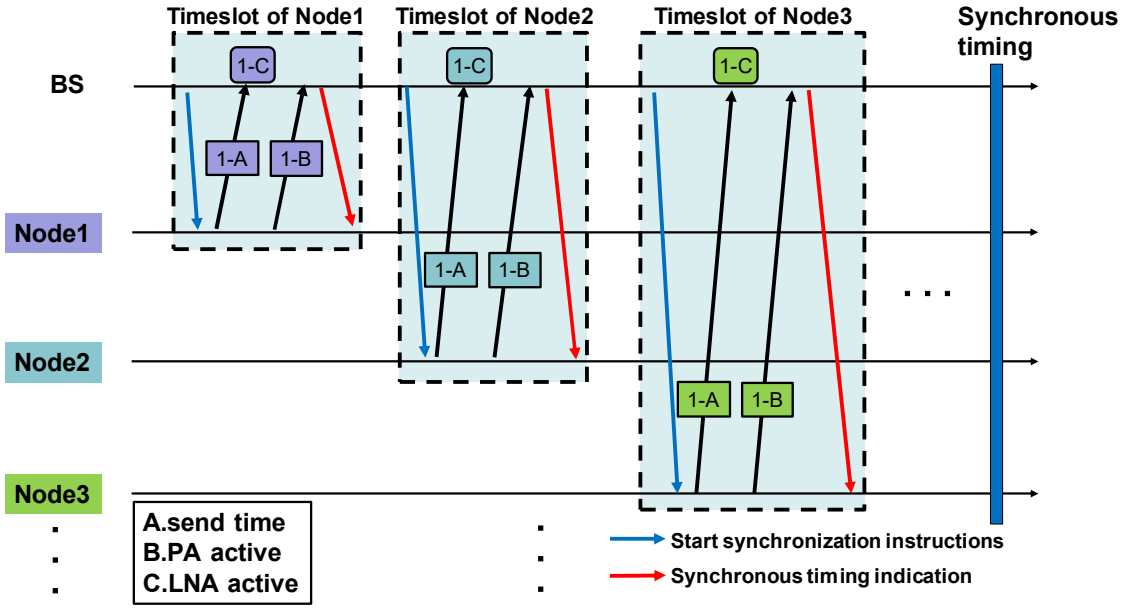

Fig. 3. Timing chart of synchronization process with multiple node.

clock drift cannot be solved by the first synchronization, it can be predicted from the variation in timer offset after the second synchronization. The offset value is basically constant, but due to the effect of clock drift, the derived offset value has a periodic deviation. The amount of device-specific drift can be determined by measuring the period of time until the displacement occurs. This can be compensated by adding the clock drift information to the packet that conveys the offset from the BS to the node.

IV. Performance Evaluation

To evaluate the accuracy of the proposed method, we conducted a practical experiment using a commercially available Bluetooth communication module (nRF52840 DK, Nordic Semiconductor). The proposed algorithm was implemented on one BS and eight nodes. A logic analyzer (Digiview, TechTools) was connected to all the nodes and the BS, and the timing of each operation was measured and used for evaluation. Fig. 4 presents the experimental setup.

First, we evaluated the transmission delay, T_{prop}, using a logic analyzer. The measurement results indicate that the average duration between TS_{PA} in the nodes and TS_{LNA} in the BS was 15 µs. Since this value is smaller than the LSB of the RTC, we decided not to correct it in this experiment. Next, we examined the accuracy of the delay to obtain $TS_{estimate}$. Fig. 5 shows the error of the delay value estimated by the prototype devices with the proposed method. The actual value of the transmission delay measured by a logic analyzer as shown in Fig. 5. The GPIOs of the BS and the nodes were toggled when sending and receiving, and these timings were measured with a logic analyzer to evaluate the precise delay value. Fig. 6 indicates that the error in delay estimation is generally within 0.03ms. This result depends on the resolution of the timestamp, because the nodes have 32.768-kHz RTC and the minimum resolution of the time stamp is 0.03ms.

Next, the effect of clock drift compensation was evaluated. Fig. 6 presents the measurement results with and without clock drift compensation. The GPIO is toggled every 300ms and resynchronized every 10 s. The actual error was measured by the logic analyzer according to the same manner as in Fig. 6. Error indicates the error in toggle timing. Without clock drift compensation, the synchronization error increases linearly until resynchronization. On the other hand, the increase of the synchronization error can be suppressed with clock drift compensation process.

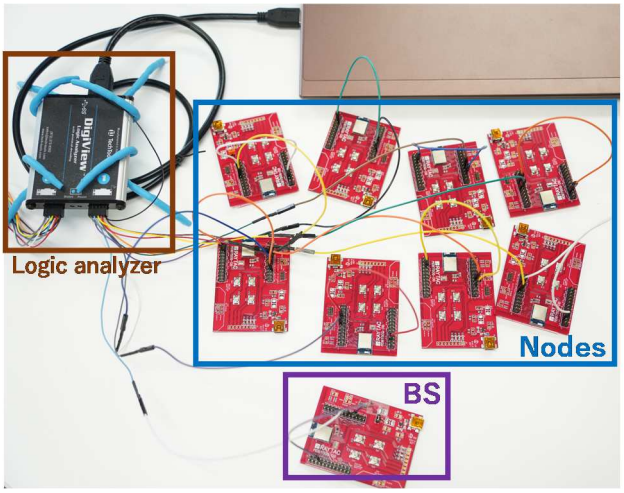

Fig. 4. Experimental setup.

The histogram of the synchronization errors of eight nodes is shown in Fig. 7. In this case, all nodes are synchronize periodically using the proposed method. Because of differences in oscillator characteristics at each node, the shape of the histogram is different. The Root Mean Square Error (RMSE) of each node was calculated, and the RMSE averaged over eight achieved 20 µs. It is small enough because the LSB of the timer was 0.03 ms, which was determined by the 32.768-kHz RTC frequency.

Table 1 compares the performances of the proposed method and those achieved in previous studies. Reference [7] communicates in the broadcast mode from the node to BS. Therefore, flexible sensor networks cannot be constructed and controlled due to one-way communication in this method. Reference [8] reported highly accurate synchronization using a high-frequency (16 MHz) timer clock, which requires large power consumption. It also requires not only BLE but also a dedicated communication protocol. This diminishes the

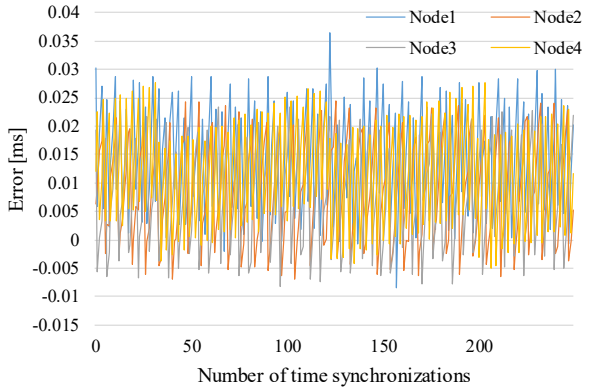

Fig. 5. Error of the estimated delay values. The actual delay is measured by logic analyzer

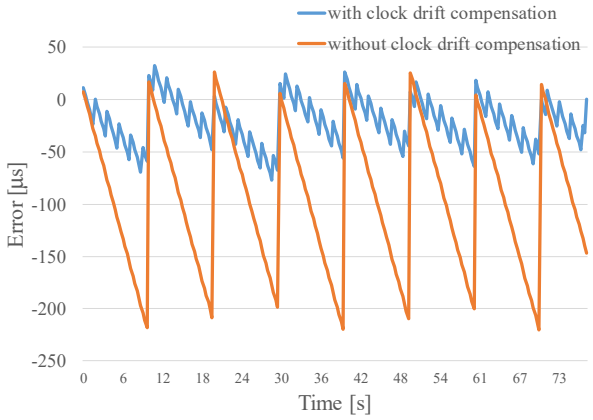

Fig. 6. Toggled timing error in each node with and without clock drift compensation. Resync every 10 seconds.

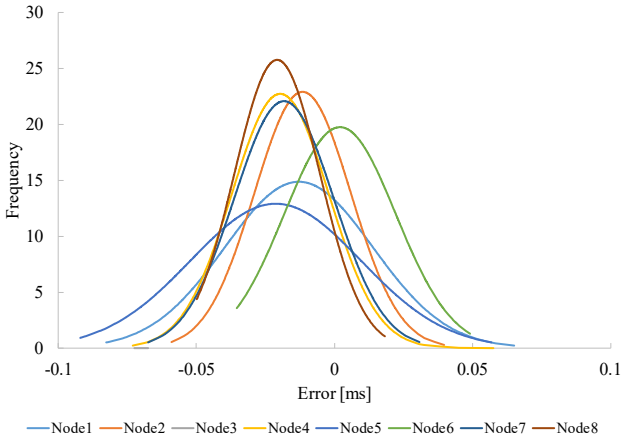

Fig. 7. Histogram of synchronization error with eight nodes.

advantage of BLE to reduce power consumption by deactivating the wireless circuit. Reference [9] used external hardware to improve the timestamp accuracy. This method has also been implemented with the advertised mode of BLE, and it has the same limitations as those reported in [7]. In contrast, the proposed method has an advantage that it can be implemented in the application layer of the Bluetooth protocol without dedicated lower-layer protocol. Furthermore, the proposed method only requires the low-power 32.768 kHz RTC, and it can minimize power dissipation during idle time.

V. Conclusion

Herein, we propose a time-synchronization method for IoT sensor nodes using the Bluetooth connection mode. The proposed method synchronized the timers of multiple nodes and compensates for clock drift. It could be implemented using only the application layer in the commercially available Bluetooth module with 32.768-kHz RTC. The experimental results indicate that the synchronization error was 20µs in the RMSE with eight nodes. A significant reduction in power consumption can be achieved by using a 32 kHz RTC while taking advantage of the BLE standard protocol. While using a low-frequency RTC, we achieved an accuracy determined by the LSB of RTC. As a result, we have achieved a low-power and low-cost synchronization method with the comparable accuracy as conventional methods.

References

[1] J. Elson, L. Girod, and D. Estrin, “Fine grained network time synchronization using reference broadcasts,” ACM SIGOPS Operating Systems Review, vol. 36, pp 147–163, 2002.

[2] S. Ganeriwal, R. Kumar, and M.B. Srivastava, “Timing-sync protocol for sensor networks,” in Proc. of the 1st international conference on Embedded networked sensor systems, pp. 138–149, 2003.

[3] A. Garg, A. Yadav, A. Sikora and A. S. Sairam, "Wireless Precision Time Protocol," in IEEE Communications Letters, vol. 22, no. 4, pp. 812-815, April 2018.

[4] M. Matori, B. Kusy, G. Simon, and A. Ledeczi, “The flooding time synchronization protocol,” in Proc. of the 2nd international conference on embedded networked sensor systems, pp. 39–49, 2004.

[5] SK. Mani, R. Durairajan, P. Barford, J.Sommers, “A System for Clock Synchronization in an Internet of Things,” in arXiv.org, 2018

[6] F. J. Dian, A. Yousefi and K. Somaratne, "A study in accuracy of time synchronization of BLE devices using connection-based event," 2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 595-601, 2017.

[7] S. Sridhar, P. Misra, and J. Warrior, “CheepSync: a time synchronization service for resource constrained Bluetooth low energy advertisers,” in *Proc. of the 14th International Conference on Information Processing in Sensor Networks, ACM*, pp. 364–365, 2015.

[8] F. Asgarian and K. Najafi, “Time Synchronization in a Network of Bluetooth Low Energy Beacons,” in *Proc. of the SIGCOMM Posters and Demos*, pp. 119–120, 2017.

[9] C.C. Rheinlander and N. When, “Precise Synchronization Time Stamp Generation for Bluetooth Low Energy,” in *Proc. of IEEE SENSORS, pp. 1-3*, 2016.

[10] N. Semiconductor, “S140 SoftDevice Specification ver2.1,” pp. 57-58, https://infocenter.nordicsemi.com/pdf/S140_SDS_v2.1.pdf

TABLE I. Comparison with prior works

Protocol		CheepSync [7]	BlueSync [8]	Proposed
Communication System		BLE (Bloadcast)+Android	BLE + Original Protocol	BLE (Connection)
Number of devices	*BS*	1(Android)	1	1
	Node	8(nRF24)	2	8
Timer clock		N/A, Android:Several GHz	16MHz	32kHz
Synchronization Accuracy		10 [µs]	42.84 [µs] / 6.22 [µs][*1]	20 [µs]
Timer current consumption		N/A	595 [µA]	0.25 [µA]
Communication Energy consumption		N/A	72.1 [mJ/sync][*2]	53.4 [mJ/sync][*2]
Sensor timing control		No	N/A	✓

*1 Depends on required timeslot: less than 2 seconds is required for 6.22-µs accuracy.

*2 Estimated from of required packets: 8 packets (8 Rx) in [7] and 4 packets (2 Tx and 2 Rx) in Prop.