Wearables Clock Synchronization Using Skin Electric Potentials

Zhenyu Yan, Rui Tan, *Senior Member, IEEE,* Yang Li, *Member, IEEE,* and Jun Huang, *Member, IEEE*

Abstract—Design of clock synchronization for networked nodes faces a fundamental trade-off between synchronization accuracy and universality of heterogeneous platforms, because a high synchronization accuracy generally requires platform-dependent hardware-level network packet timestamping. This paper presents *TouchSync*, a new indoor clock synchronization approach for wearables that achieves millisecond accuracy while preserving universality in that it uses standard system calls only, such as reading system clock, sampling sensors, and sending/receiving network messages. The design of TouchSync is driven by a key finding from our extensive measurements that the skin electric potentials (SEPs) induced by powerline radiation are salient, periodic, and synchronous on a same wearer and even across different wearers. TouchSync integrates the SEP signal into the universal principle of Network Time Protocol and solves an integer ambiguity problem by fusing the ambiguous results in multiple synchronization rounds to conclude an accurate clock offset between two synchronizing wearables. With our shared code, TouchSync can be readily integrated into any wearable applications. Extensive evaluation based on our Arduino and TinyOS implementations shows that TouchSync's synchronization errors are below 3 and 7 milliseconds on the same wearer and between two wearers 10 kilometers apart, respectively.

Index Terms—Clock synchronization, skin electric potential, wearables

1 Introduction

The annual worldwide shipments of consumer wearables (e.g., smart watches, wristbands, eyewears, clothing, etc) grew by 17% in 2017 [1]. This rapid growth is expected to continue, projected to be 504 million units shipped in 2021 [1]. Along with the proliferation of consumer wearables, specialized domains such as clinical/home healthcare [2] and exercise/sport analysis [3] are also increasingly adopting smart wearable apparatuses. In the body-area networks formed by these wearables, a variety of system functions and applications depend on tight clock synchronization among the nodes. For instance, motion analysis [4] and muscle activity monitoring [3], [5] require sensory data from multiple tightly synchronized nodes.

While current wearable systems adopt customized, proprietary clock synchronization approaches [6], we envisage a wide spectrum of interoperable wearables that can synchronize with each other to enable more novel applications. In the envisaged scheme, an application developer can readily synchronize any two communicating wearables using high-level and standard system calls provided by their operating systems (OSes), such as reading system clock, transmitting and receiving network messages. However, the design of clock synchronization approaches faces a fundamental trade-off between the synchronization accuracy and the universality for heterogeneous platforms. This is because a high synchronization accuracy generally requires low-level timestamping for the synchronization packets, which may be unavailable on the hardware platforms or inaccessible to the application developer.

We illustrate this accuracy-universality trade-off using the Network Time Protocol (NTP) [7] and the Precision Time Protocol (PTP) [8]. NTP synchronizes a slave node and a master node by recording their clock values when a UDP synchronization packet is passed to and received from the sender's and receiver's OSes, respectively. Thus, NTP is universal in that it can be applied to any host that speaks UDP. However, as the application-layer timestamping cannot capture the details of the nondeterministic OS and network delays, NTP may yield significant synchronization errors up to hundreds of milliseconds (ms) in a highly asymmetric network. To solve this issue, PTP uses hardware-level timestamping provided by PTP-compatible Ethernet cards at the end hosts and all the switches on the network path to achieve microsecond (μs) accuracy. However, the need of the special hardware inevitably negates its universality and restricts PTP's adoption to time-critical local-area networks only, e.g., those found in industrial systems.

In wireless networks, due to the more uncertain communication delays caused by media access control (MAC), NTP performs worse. Thus, similar to PTP, most existing clock synchronization approaches for wireless sensor networks (WSNs) (e.g., RBS [9], TPSN [10], and FTSP [11]) have resorted to MAC-layer timestamping provided by the nodes' radio chips to pursue synchronization accuracy. The study [12] uses MAC-layer access of Bluetooth called the Host Controller Interface (HCI) to synchronize Bluetooth devices. The classic Bluetooth devices can use the Bluetooth Clock Synchronization Protocol (CSP) included in the Bluetooth Health Device Profile (HDP). However, Bluetooth Low Energy (BLE) does not support HDP and CSP. The need of

- *Z. Yan and R. Tan are with School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798. E-mail: {zyan006, tanrui}@ntu.edu.sg*
- *Y. Li is with Additive Manufacturing Institute, College of Mechatronics & Control Engineering, Shenzhen University, Shenzhen, China 518060. This work was completed while Yang Li was with Advanced Digital Sciences Center, Illinois at Singapore. E-mail: yli@szu.edu.cn*
- *J. Huang is with Center for Energy Efficient Computing and Applications, Peking University, Beijing, China 100871. E-mail: jun.huang@pku.edu.cn*

Manuscript received January 4, 2018; revised August XX, XXXX.

the MAC-layer timestamping or specific radio links present a barrier for the wide adoption of these approaches to the broader Internet of Things (IoT) domain, where the IoT platforms use diverse radios and OSes, and in general they do not provide an interface for the MAC-layer timestamping.

In this paper, we aim at developing a new clock synchronization approach for wearables that establishes a desirable accuracy-universality trade-off point between the two extremes represented by NTP and PTP to well address the momentum of IoT platform heterogenization. In particular, we will stem from the sensor nature of wearables to explore ambient signals that can assist clock synchronization. Recent studies exploited external periodic signals such as powerline radiation [13] and Radio Data System (RDS) [14] to calibrate the clocks of WSN nodes. However, these approaches need non-trivial extra hardware to capture the external signals. Moreover, they focus on *clock calibration* that ensures different clocks advance at the same speed, rather than synchronizing the clocks to have the same value. But they inspire us to inquire (i) the existence of a periodic and synchronous signal that can be sensed by different wearables without adding non-trivial hardware to preserve universality and (ii) how to exploit the signal to synchronize the wearables without using hardware-level packet timestamping.

For the first inquiry, we conduct extensive measurements to explore such signals. Our measurements based on Adafruit's Flora, an Arduino-based wearable platform, show that by simply sampling an onboard analog-to-digital converter (ADC), a Flora can capture powerline electromagnetic radiation that oscillates at the power grid frequency (e.g., 50 Hz). When the Flora's ADC has a physical contact with the wearer's skin, the sampled skin electric potential (SEP) is a significantly amplified version of the powerline radiation, because the human body acts as an effective antenna. Although the SEP's amplitude is dynamic due to the human body movements, its frequency is highly stable. Moreover, the SEPs on the same wearer and even different wearers in a same indoor environment exhibit desirable synchrony. The time displacement between the SEPs at different positions of a still wearer is generally less than 1 ms. These results suggest that SEP is a promising basis for synchronizing the wearables.

For the second inquiry, we integrate the periodic and synchronous SEP signal into the universal principle of NTP to deal with the major source of NTP's error, i.e., asymmetric communication delays. In the original NTP, the problem of estimating the offset between the slave's and master's clocks is a *real-domain* underdetermined problem that has infinitely many solutions. NTP chooses a solution by assuming symmetric communication delays, which does not hold in many scenarios, however. Assisted with the periodic SEP, the problem reduces to an *integer-domain* underdetermined problem that has a finite number of solutions. However, it is challenging to infer which solution is correct. The integer ambiguity can be resolved by jointly considering multiple synchronization rounds with dynamic and asymmetric communication delays. Thus, the clock offset between a pair of wearables can be estimated with ms accuracy due to the ms synchrony between their SEP signals.

Based on the above two key results, we design a novel clock synchronization approach for wearables, which we call *TouchSync*. It runs at the application layer in that the needed SEP sampling, the network message exchange and timestamping can be implemented using standard wearable OS calls. Thus, by introducing a rather simple skin contact, we can readily achieve ms synchronization accuracy across heterogeneous wearable platforms, without resorting to the hardware-level packet timestamping that is extremely difficult to be standardized. To simplify the adoption of TouchSync by application developers, we design and release a `touchsync.h` header file [15] that implements TouchSync's signal processing algorithms and the integer ambiguity solver. With this header, the implementations of TouchSync in Arduino and TinyOS need about 50 and 150 lines of code, respectively, which manage sensor sampling, synchronization message exchange, and application-layer timestamping only. All these tasks can be easily implemented by Arduino and TinyOS developers. We also conduct extensive experiments in various indoor environments to show the pervasive availability of the SEP signals. On the same wearer, TouchSync mostly achieves sub-ms accuracy and the largest error is 2.9 ms. We also conduct a TouchSync-over-Internet proof-of-concept experiment that yields errors below 7 ms between two wearers 10 km apart.

The ADC-skin contact needed by TouchSync can be easily integrated into the wearable designs with near-zero cost. Although our experiments show that, in the absence of the contact, TouchSync can still work with graceful performance degradation, SEP is a new and valuable sensing modality for integration consideration, since it provides accurate timing and is indicative of other information about the wearer (e.g., body orientation, movements, and indoor location) as suggested by our measurements in this paper.

2 Related Work

Highly stable time sources are ill-suited for wearables. Chip-scale atomic clocks are too expensive ($1,500 per unit [16]). GPS receivers are power-consuming and do not work in indoor environments. The studies [17] and [18] propose synchronization protocols for IEEE 802.15.4 networks. The studies [19] and [20] improve the synchronization accuracy by compensating the propagation delays. However, these studies [17], [18], [19], [20] need the MAC-layer access, which is usually not provided by the heterogeneous wearable platforms. Recent studies exploited external signals available in indoor environments to synchronize or calibrate the clocks of distributed nodes. In [21], an AM radio receiver is designed to decode the global time information broadcast by timekeeping radio stations. In [14], a mote peripheral is designed to capture the periodic RDS signals of FM radios for clock calibration. In [13], a mote peripheral called syntonistor can receive the periodic electromagnetic radiation from powerlines to calibrate wireless sensors' clocks, where some clock synchronization approach is still needed for the initial synchronization. In [22], [23], voltage sensors plugged in to wall power outlets are used to secure clock synchronization against malicious network delays. In particular, the voltage cycle length fluctuations are exploited to implement a data-based clock synchronization approach [22]. In [24], such fluctuations extracted from the powerline

radiation are used as *natural timestamps*. However, the error of the natural timestamps can be up to hundreds of ms. Moreover, the clock synchronization based on the natural timestamps needs to transmit a considerable amount of cycle length data and a compute-intensive matching process to decode the fluctuations to time information [24]. Thus, the natural timestamping approach is ill-suited for tight clock synchronization among resource-constrained wearables. In [25], a smartphone captures ultrasonic beacons from pre-deployed synchronized beacon nodes to synchronize its own clock. All the above approaches [13], [14], [21], [22], [23], [24], [25] need non-trivial customized hardware and infrastructures, which reduce their universality.

Two recent studies leverage built-in sensing modalities to capture external periodic signals for clock calibration. In [26], a 802.15.4 radio is used to capture the Wi-Fi beacons to calibrate motes' clocks. Although this approach requires no peripherals, it uses the received signal strength indication (RSSI) register of the radio chip, which makes it hardware specific and nonuniversal for IoT platforms that use diverse radios. In [27], motes use light sensors to capture the fluorescent light that flickers at a frequency twice of the power grid frequency to calibrate their clocks. Although light sensors are widely available, the required fluorescent lighting may not be available in natural lighting environment. In contrast, the powerline radiation that induces the SEP signal used by our approach pervades civil infrastructures.

The studies [13], [14], [26], [27] mentioned above, including the two [13], [27] that are power grid related, focus on *clock calibration* that involves no message exchanges among nodes. Though continuous clock calibration maintains the nodes synchronized once they are initially synchronized, the initial synchronization and the resynchronizations needed upon clock calibration faults are not addressed in these studies. Thus, these studies and ours are complementary, in that the principle of TouchSync can be used for the initial synchronization of the systems adopting these clock calibration approaches [13], [14], [26], [27].

In [28], a distributed clock synchronization approach based on consensus is proposed. Note that TouchSync addresses the clock synchronization between two nodes only. The noise estimation algorithm in [28] can be integrated with TouchSync to improve performance. TouchSync can be applied to both centralized and distributed clock synchronization systems.

3 Objective

NTP, though universal, gives unacceptably low accuracy. The detailed performance analysis for NTP and its evaluation over a Bluetooth connection can be found in Appendix A.1.[1] On the other extreme, existing WSN clock synchronization approaches, though achieving μs accuracy, may not be universally applicable to the diversified IoT platforms with different radio chips and OSes. In this paper, by introducing the readily available SEP signal, we aim at developing a new clock synchronization approach for wearables that (i) uses application-layer timestamping as NTP does to preserve universality and (ii) achieves ms accuracy that meets the requirements of a range of applications. For instance, in seismic sensing based motion analysis [4] and muscle activity monitoring [3] that generally adopt sampling rates up to 500 Hz, the ms synchronization accuracy can enable us to discriminate any readings sampled by different sensors at different time instants. To this end, we need to understand the properties of SEP, which is the subject of Section 4.

1. Due to space limitations, all appendixes are omitted and can be found in the supplementary file of this paper.

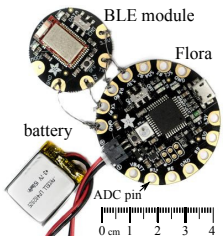


Fig. 1. Flora.

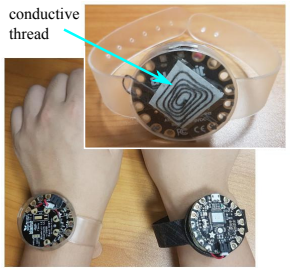


Fig. 2. TouchSync Prototypes.

4 Measurement Study

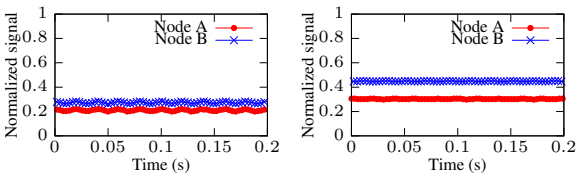
In this section, we conduct measurements to gain insights for guiding the design of TouchSync.

4.1 Measurement Setup

Our measurement study uses two Adafruit Flora nodes and a Raspberry Pi (RPi) 3 Model B single-board computer. The Flora is an Arduino-based wearable platform. Each Flora node, as shown in Fig. 1, consists of a main board with an ATmega32u4 MCU (8 MHz, 2.5 KB RAM), a BLE 4.1 module, and a 150mAh lithium-ion polymer battery. The RPi has a built-in BLE 4.1 module and runs Ubuntu MATE 16.04 with BlueZ 5.37 as the BLE driver. Each node is powered by an independent battery and has no electrical connection to any ground or grounded appliance. We use Adafruit's nRF51 Arduino library and BluefruitLE Python library on the Floras and the RPi, respectively, to send and receive data over BLE in the UART mode. The Floras and RPi operate as BLE peripheral (slave) and central (master), respectively. To obtain the ground truth clocks, in each experiment, we synchronize the Floras with the RPi as follows. At the beginning of the experiment, we wire a general-purpose input/output (GPIO) pin of the RPi with a digital input pin of each Flora. Then, the RPi issues a rising edge through the GPIO pin and records its clock value t_{master}. Upon detecting the rising edge, a Flora records its clock value t_{slave} and sends it to the RPi. The RPi computes the ground-truth offset between the Flora's and the RPi's clocks as $\delta_{GT} = t_{slave} - t_{master}$. Then, we remove the wiring and conduct experiments.

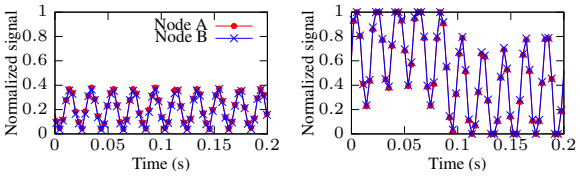
4.2 Skin Electric Potential (SEP)

In this set of measurements, we explore i) whether a human body is an effective antenna for receiving the powerline radiation and ii) whether the SEP signals induced by the radiation on the same wearer or different wearers are synchronous. The Flora's MCU has a 10-bit ADC that supports



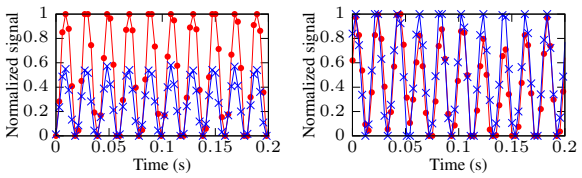
(a) Shared ground (b) Independent grounds

Fig. 3. No human body contact.



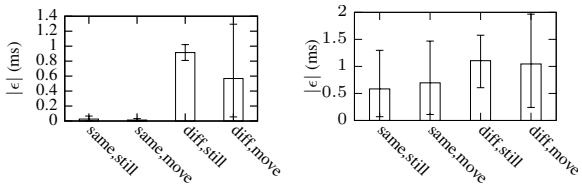
(a) Still body (b) Moving body

Fig. 4. SEPs on a same wearer (shared ground).



(a) Still bodies (b) Moving bodies

Fig. 5. SEPs on different wearers (shared ground).



(a) Shared ground (b) Independent grounds

Fig. 6. Absolute time displacement $|\epsilon|$ between the EMR signals captured by the two Floras in various scenarios. Error bar represents $(5\%, 95\%)$ confidence interval in one minute of data.

a sampling rate of up to 15 kHz. To facilitate experiments, we have made two Flora-based prototypes as shown in Fig. 2. We place the Flora into a 3D-printed insulating wristband and use a thin stainless steel conductive thread to create a connection between Flora's ADC pin and the wearer's skin. The Flora samples the ADC at 333 Hz continuously for two minutes and streams the timestamped raw data to the RPi for offline analysis. The sampling rate of 333 Hz is sufficient to capture the powerline radiation or SEP with a frequency of 50 Hz in our region. All samples are normalized using the reference voltage of the ADC. As only the ADC pin is connected to the researcher, the grounding of the Flora may affect the sampling result. To understand the impact of grounding, we conduct two sets of comparative experiments, where the two Floras have shared and independent grounds, respectively. During the experiment, neither Flora nor human body has contact with grounded appliance. In each experiment set, there are two scenarios: still and moving. For the moving scenario, the researcher keeps changing the body orientation, movement, and location. The experiments are conducted in a computer science laboratory with various appliances such as lights, computers, and printers.

4.2.1 Shared ground

We wire the ground pins of the two Floras, such that they have a shared ground. We conduct three tests.

First, Fig. 3(a) shows the signals captured by the two Floras when they have no physical contact with any human body. The signals have small fluctuations with a normalized peak-to-peak amplitude of 0.024. The signals fluctuate at a frequency of 50 Hz. This suggests that the Floras can pick up the powerline radiation. However, the signals are weak.

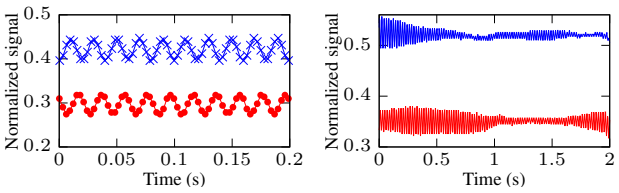
Second, a researcher touches the ADC pins of the two Floras with his two hands, respectively. Figs. 4(a) and 4(b) show the signals captured by the two nodes during the same time duration, when the researcher stands still and walks, respectively. Under the two scenarios (still and moving), the two nodes yield salient and almost identical signals. The peak-to-peak amplitudes in the two figures are around 0.4 and 0.8, which are 17 and 33 times larger than that of the signal shown in Fig. 3(a). This suggests that the human body can effectively receive the powerline radiation.

Third, two researchers touch the ADC pins of the two Floras separately. Figs. 5(a) and 5(b) show the signals captured by the two nodes when the two researchers stand still and walk, respectively. The two nodes yield salient signals with different amplitudes. We note that several factors may affect the reception of powerline radiation, e.g., human body size, position and facing of the body in the electromagnetic field generated by the powerlines.

We evaluate the synchrony between the signals captured by the two Floras shown in Figs. 4 and 5. We condition the signals by first applying a band-pass filter (BPF) to remove the direct current (DC) component that may fluctuate as seen in Fig. 4(b) and then detect the zero crossings (ZCs) of the filtered signals. More details of the BPF and ZC detection will be presented in Section 5.2. We use the *time displacement* between the two signals' ZCs as the metric to evaluate their synchrony. Specifically, the time displacement, denoted by ϵ, is given by $\epsilon = t_A^{ZC} - t_B^{ZC}$, where t_A^{ZC} and t_B^{ZC} represent the ground-truth times of Node A's ZC and the corresponding ZC at Node B, respectively. Fig. 6(a) shows the error bars for $|\epsilon|$, which correspond to the scenarios in Figs. 4(a), 4(b), 5(a), and 5(b), respectively. In Fig. 6(a), "same" and "diff" mean the same wearer and different wearers, respectively; "still" and "move" mean standing still and walking, respectively. On the same wearer, the SEPs captured by the two Floras are highly synchronous, with an average $|\epsilon|$ of 0.9 μs. On different wearers, the $|\epsilon|$ increases to about 1 ms. When the two wearers move, the average $|\epsilon|$ is 0.35 ms smaller than that when they stand still. Note that as shown shortly in Fig. 9(b), the body orientation affects the time displacement. In Fig. 6(a), the two wearers stand still facing certain orientations. The time displacement under such body orientation setting is consistently large. Thus, the average time displacement when they move is smaller. However, consistent with intuition, the human body movements increase the variance of $|\epsilon|$, since they create more signal dynamics as seen in Fig. 4(b).

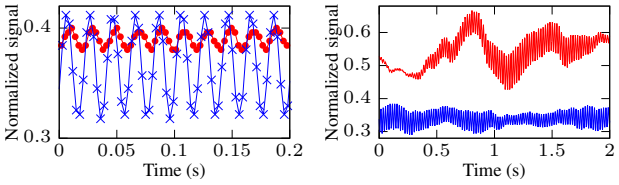
4.2.2 Independent grounds

Then, we remove the connection between the two Floras' ground pins, such that they have independent grounds. This



(a) Still body (b) Moving body

Fig. 7. SEPs on a same wearer (independent grounds).



(a) Still bodies (b) Moving bodies

Fig. 8. SEPs on different wearers (independent grounds).

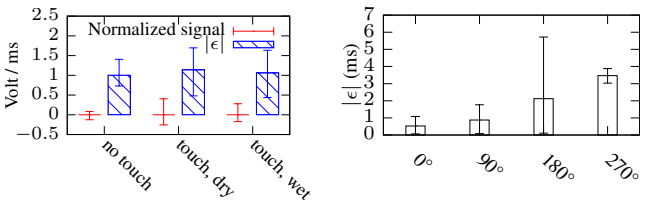
setting is consistent with real scenarios, where the wearables are generally not wired. We conduct three tests.

Fig. 3(b) shows the two Floras' signals when they have no physical contact with any human body. The signals have small oscillations with a frequency of 50 Hz.

Figs. 7(a) and 7(b) show the signals of the two Floras worn on two wrists of a researcher when he stands still and walks, respectively. Compared with the results in Fig. 4(a) based on a shared ground, the two signals in Fig. 7(a) have an offset in their values. This offset is the difference between the electric potentials at the two Floras' grounds. Fig. 7(b) shows the signals over two seconds that contain about 100 SEP cycles to better illustrate the changing signal envelopes over time due to the human body movements. Compared with Fig. 4(b), the two signals in Fig. 7(b) have different signal envelopes. This is because the electric potentials at the two Floras' grounds, which are also induced by the powerline radiation, are not fully correlated in the presence of human body movements.

Figs. 8(a) and 8(b) show the signals of the two Floras worn by two researchers each when they stand still and walk, respectively. Salient EMR signals can be observed. Moreover, the human movements cause significant fluctuations of DC lines and the signal amplitudes, as seen in Fig. 8(b).

We also evaluate the synchrony between the two Floras' signals. Fig. 6(b) shows the time displacement's error bars that correspond to the scenarios in Figs. 7(a), 7(b), 8(a), and 8(b). The average $|\epsilon|$ is below 2 ms. Compared with the results in Fig. 6(a) that are based on a shared ground, the time displacements increase. This is because of the additional uncertainty introduced by the independent floating grounds of the two Floras. Nevertheless, on the same wearer, the average $|\epsilon|$ is about 0.5 ms only. The body movements increase the 95%-percentile of $|\epsilon|$ to 1.5 ms. In Section 5.2, we will use a phase-locked loop to reduce the variations of ϵ.



(a) Skin conditions (b) Orientations

Fig. 9. Normalized range and time displacement $|\epsilon|$ of SEPs under different skin conditions and body orientations.

We further evaluate the impact of skin condition and body orientation on the SEPs when the two Floras are worn by two hands of the same wearer. Fig. 9(a) shows the normalized SEP signal range and their time displacement on the same still person, when the skin is dry or wet. The SEP signal is amplified when the sensor has direct contact with the human body. The skin moisture condition does not affect the amplitude and synchrony ($|\epsilon|$) of SEP much. We have also tested SEP sensing using two conductive threads for a Flora. Results show that the signal reception is not affected. However, using multiple conductive threads reduces the chance that the wearable loses direct skin contact. Fig. 9(b) shows the time displacements when the wearer is in different orientations. We can see that the body orientation affects the time displacement. However, the average time displacement $|\epsilon|$ in all tested orientations is below 4ms.

4.2.3 *Summary*

From the above measurements, we obtain the following three key observations. First, the human body can act as an antenna that effectively improves the powerline radiation reception. Second, during the human body movements, the SEP amplitude changes. However, the synchrony between the two SEP signals captured by the two nodes on the same wearer or different wearers is still acceptably preserved. In Section 5.2, we will condition the SEP signals to improve the synchrony. Third, the floating ground of a node introduces additional uncertainty, because the powerline radiation can also generate a varying electric potential at the ground pin. However, the floating ground does not substantially degrade the synchrony between the two nodes' SEP signals. All experiments in the rest of this paper are conducted under the floating ground setting. The above three observations suggest that the SEP induced by powerline radiation is a good periodic signal that can be exploited for synchronizing wearables.

5 Design of TouchSync

This section presents the design of TouchSync. Section 5.1 overviews the workflow. Section 5.2 presents the signal processing algorithms to generate stable, periodic, and synchronous impulses trains (i.e., Dirac combs) from the SEP signals. Section 5.3 presents a synchronization protocol assisted with the Dirac combs. Section 5.4 solves the integer ambiguity problem to complete synchronization.

5.1 TouchSync Workflow

TouchSync synchronizes the clock of a slave to that of a master. This paper focuses on the synchronization between a slave-master pair, which is the basis for synchronizing a network of nodes. A *synchronization process*, as illustrated in Fig. 10, is performed periodically or in an on-demand

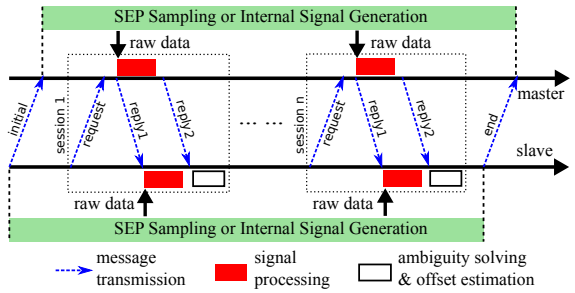


Fig. 10. A synchronization process of TouchSync.

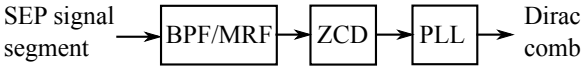


Fig. 11. SEP signal processing pipeline.

fashion. For instance, the wearer(s) may push some buttons on two wearables to start a synchronization process. The period of the synchronization can be determined by the needed clock accuracy and the clock drift rate. During the synchronization process, both the slave and the master continuously sample the SEP signals and store the timestamped samples into their local buffers. Note that the SEP signal strength may become too weak to support accurate clock synchronization in certain scenarios. The TouchSync node will switch to a mode that generates an internal periodic signal to drive the clock synchronization, which will be presented in Section 6. At the beginning of the synchronization process, the slave node sends a message to the master to signal the start of the sensor sampling. As shown in Fig. 10, a synchronization process has multiple *synchronization sessions*. In each session, the slave and the master exchange three messages: `request`, `reply1`, and `reply2`. The `request` and `reply1` are used to measure the communication delays. After transmitting the `reply1`, the master retrieves a segment of SEP signal from its buffer to process and transmits the processing results using the `reply2` to the slave. Upon receiving the `reply1`, the slave retrieves a segment of SEP signal from its buffer to process. Upon receiving the `reply2`, the slave tries to solve the integer ambiguity problem to estimate the offset between the slave's and the master's clocks. If the ambiguity cannot be solved, another synchronization session is initiated; otherwise, the two nodes stop sampling SEPs and the slave uses the estimated offset to adjust its clock and complete the synchronization process.

5.2 SEP Signal Processing

This section presents TouchSync's signal processing pipeline as illustrated in Fig. 10 that aims to generate a highly stable, periodic, and synchronous Dirac comb from a SEP signal with fluctuating DC component and jitters. It has three steps:

Band-pass filter (BPF) or mean removal filter (MRF): We apply a 5th-order/6-tap infinite impulse response (IIR) BPF with steep boundaries of a $(45\,\text{Hz}, 55\,\text{Hz})$ passband to remove the fluctuating DC component and high-frequency noises of the SEP signal. For too resource-limited wearables, a MRF that subtracts the running average from the original signal can be used instead of the BPF for much lower compute and storage complexities. Its effect is similar to high-pass filtering.

Zero crossing detector (ZCD): It detects the ZCs, i.e., the time instants when the filtered SEP signal changes from negative to positive. It computes a linear interpolation point between the negative and the consequent positive SEP samples as the ZC to mitigate the impact of low time resolution due to a low SEP sampling rate.

Phase-locked loop (PLL): We apply a software PLL to deal with the ZC jitters and miss detection caused by significant dynamics of the SEP signal. The PLL generates an impulse train using a loop and uses an active proportional integral (PI) controller to tune the interval between two consecutive impulses according to the time differences between the past impulses and the input ZCs. The controller skips the time differences larger than 25 ms to deal with ZC miss detection.

5.3 NTP Assisted with Dirac Combs

TouchSync uses the synchronous Dirac combs at the slave and the master to achieve clock synchronization through multiple synchronization sessions. This section presents the protocol for a single synchronization session.

5.3.1 Protocol for a synchronization session

A synchronization session of TouchSync is illustrated in Fig. 12. We explain it from the following two aspects.

Message exchange and timestamping: The session consists of the transmissions of three messages: `request`, `reply1`, and `reply2`. The `request` and `reply1` messages are similar to the two UDP packets used by NTP. Their transmission and reception timestamps, i.e., t_1, t_2, t_3, and t_4, are obtained upon the corresponding messages are passed/received to/from the OS. The master transmits the auxiliary `reply2` message to convey the results of its signal processing, which is detailed below.

Signal processing and clock offset estimation: After the master has transmitted the `reply1` message, the master (i) retrieves from its signal buffer a SEP signal segment that covers the time period from t_2 to t_3 with some safeguard ranges before t_2 and after t_3, (ii) feeds the signal processing pipeline in Section 5.2 with the retrieved SEP signal segment to produce a Dirac comb as illustrated in Fig. 12, and (iii) identifies the last impulses (LIs) in its Dirac comb that are right before the time instants t_2 and t_3, respectively. The LIs are illustrated by thick red arrows in Fig. 12. Then, the master computes the elapsed times from t_2's LI to t_2 and t_3's LI to t_3, which are denoted by ϕ_2 and ϕ_3, respectively. The ϕ_2 and ϕ_3 are the *phases* of the t_2 and t_3 with respect to the Dirac comb. After that, the master transmits the `reply2` message that contains t_2, t_3, ϕ_2, and ϕ_3 to the slave. After receiving the `reply1` message, the slave retrieves a SEP signal segment that covers the time period from t_1 to t_4 with some safeguard ranges, executes the signal processing pipeline, identifies the LIs right before t_1 and t_4, and computes the phases ϕ_1 and ϕ_4, as illustrated in Fig. 12. After receiving the `reply2` message, based on $\{t_1, t_2, t_3, t_4\}$ and $\{\phi_1, \phi_2, \phi_3, \phi_4\}$, the slave uses the approach in Section 5.3.2 to analyze the offset between the slave's and master's clocks.

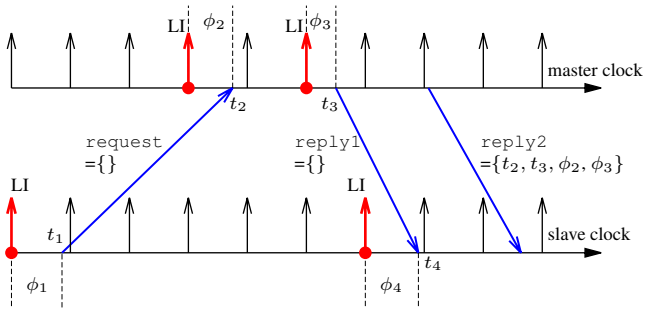


Fig. 12. A synchronization session of TouchSync. The vertical arrows represent the Dirac comb impulses generated from the SEP.

Now, we discuss several design considerations for the protocol described above. TouchSync uses the `request` and `reply1` messages to measure the clock offset, while the `reply2` is an auxiliary message to convey the timestamps t_2, t_3 and the measurements ϕ_2, ϕ_3. With this auxiliary message, we can decouple the task of timestamping the reception of `request` and the transmission of `reply1` from the signal processing task of generating the Dirac comb and computing ϕ_2 and ϕ_3. On many platforms (e.g., Android Wear and watchOS), continuously sampled sensor data is passed to the application block by block. With the decoupling, the master can compute ϕ_2 and ϕ_3 after the `reply1` is transmitted and the needed SEP data blocks become available.

5.3.2 Clock offset analysis

We now analyze the offset between the slave's and the master's clocks based on $\{t_1, t_2, t_3, t_4\}$ and $\{\phi_1, \phi_2, \phi_3, \phi_4\}$. Denote by T the period of the Dirac comb. In our region served by a 50 Hz grid, the nominal value for T is 20 ms. To capture the small deviation from the nominal value, T can be also easily computed as the average interval between consecutive impulses of the Dirac comb. We define the *rounded phase differences* θ_q and θ_p (which correspond to the `request` and `reply1` messages, respectively) as

$$\theta_q = \begin{cases} \phi_2 - \phi_1, & \text{if } \phi_2 - \phi_1 \geq 0; \\ \phi_2 - \phi_1 + T, & \text{otherwise.} \end{cases} \quad (1)$$

$$\theta_p = \begin{cases} \phi_4 - \phi_3, & \text{if } \phi_4 - \phi_3 \geq 0; \\ \phi_4 - \phi_3 + T, & \text{otherwise.} \end{cases} \quad (2)$$

As ϕ_k is the elapsed time from t_k's LI to t_k, we have $0 \leq \phi_k < T$, for $k \in [1, 4]$. From Eqs. (1) and (2), we can verify that $0 \leq \theta_q < T$ and $0 \leq \theta_p < T$. From our measurements in Appendix A.2, the times for transmitting the `request` and `reply1` messages can be longer than T. Thus, we use i to denote the non-negative integer number of the Dirac comb's periods elapsed from the time of sending `request` to the time of receiving it at the master, and j to denote the non-negative integer number of the Dirac comb's periods elapsed from the time of sending `reply1` to the time of receiving it at the slave.

We denote τ_q and τ_p the actual times for transmitting the `request` and the `reply` messages, respectively. Thus,

$$\tau_q = \theta_q + i \cdot T - \epsilon, \qquad \tau_p = \theta_p + j \cdot T + \epsilon, \quad (3)$$

where ϵ is the time displacement between the slave's and master's Dirac combs. Here, we assume a constant ϵ to simplify the discussion. Therefore, the RTT computed by $\text{RTT} = (t_4 - t_1) - (t_3 - t_2)$ must satisfy

$$\text{RTT} = \tau_q + \tau_p = \theta_q + \theta_p + (i + j) \cdot T. \quad (4)$$

In Eq. (4), RTT, θ_q, and θ_p are measured in the synchronization session illustrated in Fig. 12; i and j are unknown non-negative integers. If the i and j can be determined, the estimated offset between the slave's and the master's clocks, denoted by δ, can be computed by either one of the following formulas:

$$\delta = t_1 - (t_2 - \tau_q) = t_1 - t_2 + \theta_q + i \cdot T - \epsilon, \quad (5)$$

$$\delta = t_4 - (t_3 + \tau_p) = t_4 - t_3 - \theta_p - j \cdot T - \epsilon. \quad (6)$$

It can be easily verified that the above two formulas give the same result. The analysis in the rest of this paper chooses to use Eq. (6). The ϵ is generally unknown. If we ignore it in Eq. (6) to compute δ, it becomes part of the clock offset estimation error.

Eq. (4) is an *integer-domain* underdetermined problem. Clearly, from Eq. (4), both i and j belong to the range $\left[0, \frac{\text{RTT} - \theta_q - \theta_p}{T}\right]$. Thus, Eq. (4) has a finite number of solutions for i and j. Note that, under the original NTP principle, we have a *real-domain* underdetermined problem of $\text{RTT} = \tau_q + \tau_p$ that has infinitely many solutions. NTP chooses a solution by assuming $\tau_q = \tau_p$, which does not hold in general. Thus, by introducing the Dirac combs, the *ambiguity* in determining τ_q, τ_p, and δ is substantially reduced from infinitely many possibilities to finite possibilities. Though we still have ambiguity in the integer domain, our analysis and extensive numeric results in Section 5.4 show that the ambiguity can be solved.

Note that, in [23], the periodic and synchronous power grid voltage signals collected directly from power outlets are used to synchronize two nodes that have high-speed wired network connections. The approach in [23] also uses the elapsed times from LIs (i.e., ϕ_1, ϕ_2, ϕ_3, ϕ_4) to deal with asymmetric communication delays and improve synchronization accuracy. However, due to the high-speed connectivity, it only considers the case where both i and j are zero. In contrast, with wireless connectivity, i and j are random and often non-zero due to the access time. Estimating i and j is challenging and it is the subject of Section 5.4.

5.4 Integer Ambiguity Solver (IAS)

Before we present the approach to solving integer ambiguity, we make the following two assumptions for simplicity of exposition. First, we assume that the ground-truth clock offset δ_{GT} is a constant during a synchronization process. From our performance evaluation in Section 8, TouchSync generally takes less than one second to achieve synchronization. Typical crystal oscillators found in MCUs and personal computers have drift rates of 30 to 50 ppm [26]. Thus, the maximum drift of the offset between two clocks during one second is $50\,\text{ppm} \times 1\,\text{s} \times 2 = 0.1\,\text{ms}$. This drift is smaller than the ms-level time displacement ϵ between two SEP signals, which dominates the synchronization error of TouchSync.

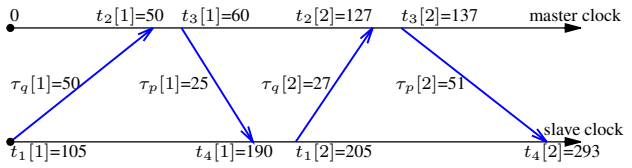


Fig. 13. An example of solving the integer ambiguity. The transmissions of the auxiliary `reply2` messages are omitted in the illustration.

Second, we assume $\epsilon = 0$. In Section 5.4.4, we will discuss how to deal with non-zero and time-varying ϵ.

We let $i_{\min}$ and $i_{\max}$ denote the minimum and maximum possible values for i; $j_{\min}$ and $j_{\max}$ denote the minimum and maximum possible values for j. For instance, from our one-way message transmission time measurements (cf. Appendix A.2), the BLE's slave-to-master transmission times are always greater than 30 ms. Thus, we may set $i_{\min} = 1$, since in our region T is 20 ms. When we have no prior knowledge about the ranges for i and j, we may simply set $i_{\min} = j_{\min} = 0$ and $i_{\max} = j_{\max} = \frac{\mathrm{RTT}-\theta_q-\theta_p}{T}$. Section 5.4.4 will discuss how the use of the prior knowledge impacts on the integer ambiguity solving.

TouchSync performs multiple synchronization sessions to solve the integer ambiguity problem. In this section, we use $x[k]$ to denote a quantity x in the k^{th} synchronization session. For instance, $\mathrm{RTT}[k]$ denotes the measured RTT in the k^{th} session. From Eqs. (4) and (6), for the k^{th} synchronization session, we have

$$\begin{cases} \mathrm{RTT}[k] = \theta_q[k] + \theta_p[k] + (i[k] + j[k]) \cdot T; \\ \delta = t_4[k] - t_3[k] - \theta_p[k] - j[k] \cdot T; \\ i_{\min} \le i[k] \le i_{\max}, \quad j_{\min} \le j[k] \le j_{\max}. \end{cases} \tag{7}$$

If TouchSync performs K synchronization sessions, we have an underdetermined system of $2K$ equations with $(2K+1)$ unknown variables (i.e., δ and $\{i[k], j[k] | k \in [1, K]\}$). In the integer domain, such an underdetermined system can have a unique solution.

5.4.1 An example of unique solution

We use an example in Fig. 13 to illustrate. The unit for time is ms, which is omitted in the following discussion for conciseness. In this example, $T = 20$, $i_{\min} = j_{\min} = 1$, $i_{\max} = j_{\max} = 4$, and the ground-truth clock offset $\delta_{GT} = 105$. Two synchronization sessions are performed. The timestamps and the actual message transmission delays are shown in Fig. 13. The ground-truth values for i and j in the two synchronization sessions are: $i[1] = 2$, $j[1] = 1$, $i[2] = 1$, and $j[2] = 2$. The RTTs can be computed as $\mathrm{RTT}[1] = 75$ and $\mathrm{RTT}[2] = 78$. With any synchronous Dirac combs, from Eqs. (1) and (2), the rounded phase differences computed by the two nodes must be $\theta_q[1] = 10$, $\theta_p[1] = 5$, $\theta_q[2] = 7$, and $\theta_p[2] = 11$. For the first synchronization session, Eq. (7) has two possible solutions only:

$$\{i[1]=1, j[1]=2, \delta=85\}, \{i[1]=2, j[1]=1, \delta=105\}. \tag{8}$$

For the second synchronization session, Eq. (7) has two possible solutions only as well:

$$\{i[2]=1, j[2]=2, \delta=105\}, \{i[2]=2, j[2]=1, \delta=125\}. \tag{9}$$

From Eqs. (8) and (9), $\delta = 105$ is the only common solution. Thus, we conclude that δ must be 105.

5.4.2 Program for solving integer ambiguity

From the above example, due to the diversity of the ground-truth values of i and j in multiple synchronization sessions, the intersection of the δ solution spaces of these synchronization sessions can be a single value. Thus, the integer ambiguity problem is solved. On the contrary, if the ground-truth i and j do not change over multiple synchronization sessions, the ambiguity remains. With application-layer timestamping, the message transmission times are highly dynamic due to the uncertain OS overhead and MAC. Such uncertainties and dynamics, which are undesirable in the original theme of NTP, interestingly, become desirable for solving the integer ambiguity in TouchSync.

From the above key observation, TouchSync performs the synchronization session illustrated in Fig. 12 *repeatedly* until the intersection among the δ solution spaces of all the synchronization sessions converges to a single value. The pseudocode of the algorithms running at the slave and the master can be found in Appendix B in the supplementary file of this paper.

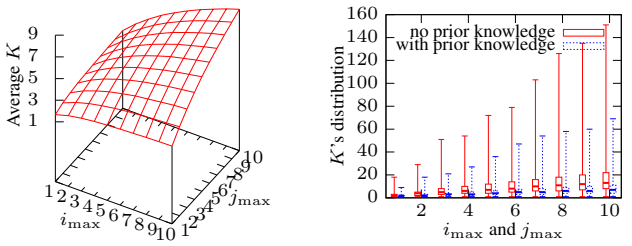
5.4.3 Convergence speed

We run a set of numeric experiments to understand the convergence speed of the IAS. We use the number of synchronization sessions until convergence to characterize the convergence speed, which is denoted by K in the rest of this paper. We fix $i_{\min}$ and $j_{\min}$ to be zero. For a certain setting $\langle i_{\max}, j_{\max} \rangle$, we conduct 100,000 synchronization processes to assess the distribution of K. For each synchronization session of a synchronization process, we randomly and uniformly generate the ground-truth i and j, as well as θ_q and θ_p within their respective ranges, i.e., $i \in [0, i_{\max}]$, $j \in [0, j_{\max}]$, and $\theta_q, \theta_p \in [0, T)$. Then, we simulate the integer ambiguity solving program presented in Section 5.4.2 to measure the K for each synchronization process. In practice, the i, j, θ_q and θ_p may not follow the uniform distributions. But the numeric results here help us understand the convergence speed. In Section 8.3, we will evaluate the convergence speed in real-world settings.

In Fig. 14(a), each grid point is the average of all K values in the 100,000 synchronization processes under a certain $\langle i_{\max}, j_{\max} \rangle$ setting. Fig. 14(b) shows the box plot for K under each setting where $i_{\max} = j_{\max}$. We note that all simulated synchronization processes converge. From the two figures, even if $i_{\max} = j_{\max} = 10$ (which means that the one-way communication delays are up to 200 ms for $T = 20$ ms), the average K is nine only. Although the K's distribution has a long tail as shown in Fig. 14(b), 75% of the K values are below 11. This result is consistent with our real experiment results in Table 1 of Section 8.3, where most K values are two only and the largest K is 12.

5.4.4 Discussions

First, we discuss how to address non-zero and time-varying ϵ. From the analysis in Eq. (7) that is based on $\epsilon = 0$, the difference between two δ solutions is multiple of T. This can also be seen from Eqs. (8) and (9). In practice, ϵ can be non-zero and time-varying. It will be a major part of the δ estimation error. From Fig. 6(b), the $|\epsilon|$ is at most 6 ms. Thus, the resulted variation to the δ solutions will be less



(a) Average K vs. $i_{\max}$ and $j_{\max}$.

(b) Box plot for K.

Fig. 14. Convergence speed of IAS.

than a half and one third of T, in the regions served by 60 Hz and 50 Hz power grids, respectively. Therefore, we can still correctly identify the correspondence among the δ elements in the set intersection operation. Specifically, if two δ elements have a difference smaller than $T/2$, they should be considered the same element in the set intersection operation; otherwise, they are different elements. For this correspondence identification to be correct, the ϵ needs to be smaller than $T/2$. After convergence, the final δ can be computed as the average of the δ elements that are considered the same. We have incorporated this in our implementation of TouchSync.

Second, we discuss how the use of the prior knowledge (i.e., $i_{\min}$, $i_{\max}$, $j_{\min}$, and $j_{\max}$) impacts on the integer ambiguity solving. With the prior knowledge, we may shrink the search range for i and j to speed up the convergence of the IAS. The prior knowledge can be based on the statistical information obtained in offline experiments. For instance, a group of the box plots are the results for the IAS with the prior knowledge of $i_{\max}$ and $j_{\max}$. The IAS can search the i and j within the ranges of $[0, i_{\max}]$ and $[0, j_{\max}]$, respectively. The other group of results are for the IAS without the prior knowledge. Thus, the IAS has to search within the range of $\left[0, \frac{\mathrm{RTT}-\theta_q-\theta_p}{T}\right]$ for both i and j. We can see that, if no prior knowledge is used, the K increases. But the IAS still always converges. Once the IAS converges, the synchronization error of TouchSync mainly depends on the time displacement ϵ.

6 TouchSync with Internal Periodic Signal

Our design of TouchSync in Section 5 is based on the periodic and synchronous SEP signals available to the slave and master nodes. However, in certain scenarios where the wearables are extremely far away from the powerlines or in a Faraday cage (e.g., an elevator cabin), the wearables can hardly sense EMR or SEP. This section extends the design of TouchSync to deal with such situations by letting the wearables generate internal periodic signals (IPSes) with the same period by themselves. Such IPSes are used to drive the clock synchronization. Section 6.1 presents our extended design. As the IPSes generated by any two wearables may have a random and bounded time displacement, it is interesting to investigate how this time displacement affects the performance of TouchSync, which is the subject of Section 6.2.

6.1 Extended Design

The key extensions made to the design presented in Section 5 include the coordination between the slave and the master on using the same IPS time period and the generation of their IPSes. Based on the generated IPS, each node follows exactly the procedures described in Section 5 to estimate the clock offset. In what follows, Sections 6.1.1 and 6.1.2 present the extensions to the TouchSync protocol and an Adaptive Period Mechanism that assists the convergence of TouchSync, respectively.

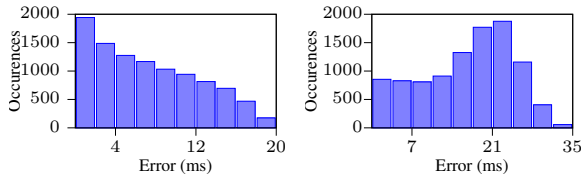
6.1.1 Protocol for IPS-based TouchSync

We extend the TouchSync protocol in Section 5.3.1 to use IPSes. When the slave is to estimate the offset between its clock and master's, it transmits an initial packet to the master, indicating the beginning of the synchronization process. The packet includes the time period T for generating the IPS. The two devices will record their time instants on transmitting and receiving the initial packet as t_0^{slave} and t_0^{master}. In our approach, these two time instants are used as the first ZCs of the two devices' IPSes. Then, same as the design in Section 6.1.1, one `request` and two `reply` packets will be exchanged for each synchronization session. Meanwhile, the time instants, i.e., t_1, t_2, t_3, and t_4 as illustrated in Fig. 10, are recorded by the two nodes. For each recorded time instant, the elapsed time from the corresponding LI is computed. For instance, the slave node computes ϕ_1 as $\phi_1 = (t_1 - t_0^{slave}) \bmod T$; the master computes ϕ_2 as $\phi_2 = (t_2 - t_0^{master}) \bmod T$. Then, the slave finds all the possible estimated clock offsets based on the analysis in Section 5.3.2. TouchSync repeatedly runs synchronization sessions until the ambiguity is solved using the IAS presented in Section 5.4.

In this extended design, the absolute time displacement between the two nodes is $|t_0^{slave} - t_0^{master}| \bmod T$. The above protocol transmits an initial packet to establish the initial ZCs at t_0^{slave} and t_0^{master}. As the transmission time of the initial packet is random, the resulting time displacement is also random. An alternative approach is to use NTP to establish the initial ZCs. However, as NTP is susceptible to link asymmetry that is generally true under stochastic wireless link quality, this alternative approach may not give smaller time displacements.

6.1.2 Adaptive Period Mechanism (APM)

When the IPS time period is larger, the number of IPS time periods elapsed during a synchronization session (i.e., $i + j$) is likely less. Hence, a large IPS period setting can generally reduce the TouchSync session's ambiguity and speed up the convergence. As an extreme example, when the IPS time period is very large such that $i + j = 0$ (i.e., no ambiguity) and TouchSync converges after one session. However, a larger IPS time period setting will lead to larger time displacements and therefore larger clock offset estimation errors. Thus, there exist a trade-off between the convergence speed and clock offset estimation accuracy. Based on this observation, we propose APM to dynamically increase IPS period to make sure that the system can alway converge. Specifically, the IPS-based TouchSync starts with a small IPS time period setting. Whenever the system cannot



(a) IPS-based TouchSync. (b) Ordinary NTP.

Fig. 15. Distribution of absolute clock offset estimation errors.

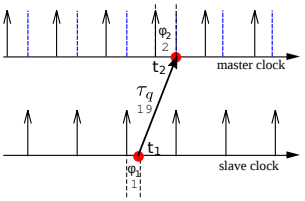


Fig. 16. An emample of IPS-based TouchSync. (unit: ms)

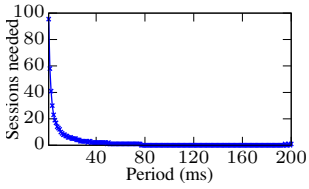


Fig. 17. Convergence speed under various settings of IPS period.

converge within a predefined number of synchronization sessions, it increases the IPS time period and restarts IAS. In other words, APM keeps increasing the IPS period until the ambiguity is solved. APM speeds up the convergence at the cost of larger clock offset estimation error bounds. The preset maximum number of synchronization sessions allowed for each IPS time period setting is the knob provided to the system designer for choosing a satisfactory trade-off between the convergence speed and error bound. We will evaluate this trade-off via numeric experiments in Section 6.2.2.

6.2 Numeric Experiments

We conduct numeric experiments to evaluate the performance of the IPS-based TouchSync in terms of clock offset estimation error and convergence speed, when the time displacement between the two self-generated IPSes varies. Specifically, we develop a simulator that can simulate the packet transmissions between the slave and the master. The ground-truth clock offset between the slave and the master is an arbitrary value. The first ZCs of the IPSes at the slave and the master are randomly selected. The transmission delays of the `request` and `reply` packets are uniformly and independently drawn from the range of $[0, 100\,\text{ms}]$. TouchSync repeatedly runs synchronization sessions and terminates when either the integer ambiguity is solved or the maximum session number limit of 200 is reached. The numeric experiment results are presented below.

6.2.1 Distribution of clock offset estimation errors

We run 10,000 synchronization processes with the IPS period fixed to 20ms. Fig. 15(a) shows the distribution of the clock offset estimation errors. We can see that the errors are bounded by $20\,\text{ms}$, i.e., the IPS period. Now, we use an example in Fig. 16 to explain this error bound. For this example, we assume that the integer ambiguity has been solved and only focus on the packet transmission from the slave to the master that is used to estimate the clock offset between the two nodes. Suppose the one-way transmission delay τ_q is $19\,\text{ms}$ and the solved value for i is 0. In Fig. 16, the two trains of arrows represent the Dirac combs based on the two nodes' IPSes that are not synchronous. To facilitate illustration, we add dashed lines on the master's timeline that are synchronous with the slave's Dirac combs. As illustrated in the figure, $\phi_1 = 1\,\text{ms}$ and $\phi_2 = 2\,\text{ms}$. From Eq. (1), we have $\theta_q = 1\,\text{ms}$. As the two nodes independently generate IPSes, they do not know the value of the time displacement between their IPSes. Thus, from Eq. (5), the slave node will compute the clock offset as $\delta = t_1 - t_2 + \theta_q$. If the clocks of the two nodes are actually synchronized such that $t_1 - t_2 = -19$, the computed $|\delta|$ of $18\,\text{ms}$ is the absolute clock offset estimation error. From this example, if $\phi_1 \to 0$, $\phi_2 \to 0$, and $\tau_q \to 20\,\text{ms}$, the absolute clock offset estimation will approach to $20\,\text{ms}$. This explains the error bound observed in Fig. 15(a).

IPS-based TouchSync does not rely on any external signal, which is the same as the NTP. For comparison, we evaluate side-by-side the clock offset estimation errors of the NTP. Fig. 15(b) shows the results. The absolute clock offset estimation errors are up to $35\,\text{ms}$. Note that, NTP's synchronization error is $(\tau_2 - \tau_1)/2$, where τ_1 and τ_2 are the one-way transmission delays of the `request` and `reply` packets (cf. Appendix A.1). Hence, unlike IPS-based TouchSync that has an error bound of the IPS time period, NTP's error bound depends on the two one-way packet transmission delays. Over a highly asymmetric link, NTP's error can be much higher than the IPS period.

6.2.2 Convergence speed

As discussed in Section 6.1, if the system uses a larger IPS period setting, it will need fewer sessions to converge. To demonstrate this, we simulate the IPS-based TouchSync system under different period settings and record the number of sessions needed for solving the integer ambiguity. Fig. 17 shows the number of sessions needed for convergence versus the IPS period. The result clearly shows the trade-off discussed in Section 6.1.2. From the result, when the IPS period is $10\,\text{ms}$ to ensure a $10\,\text{ms}$ error bound, a total of 13 synchronization sessions are performed before convergence. If we loosen the error bound to $60\,\text{ms}$, the system needs 2 sessions only to converge. We note that the convergence speed is also affected by the two one-way packet transmission delays, since a long transmission delay will lead to larger values of $i + j$ and thus increase the ambiguity. In Section 8, we will evaluate TouchSync's convergence speed under real-world settings.

6.3 Impact of Clock Skews

To simplify discussions, our analysis and the numeric experiments in Section 6.1 and Section 6.2 assume that the master and the slave have no clock skews. In practice, small clock skews will result in additional clock synchronization errors for IPS-based TouchSync. According to our experiments in Section 6.2, when the IPS' period is 40 ms, the IPS-based TouchSync takes less than one second before the system converges. The clock drift over a second time duration is often little. For instance, the average clock drift rate of MSP430's clocks is 44.2 ppm [29], which will introduce a drift of $44\ \mu s$ per second. Thus, the time displacement $|t_0^{slave} - t_0^{master}|$ that ranges from 0 to the IPS period dominates the synchronization error of IPS-based TouchSync.

The validness of the domination of the time displacement over clock skew depends on the convergence time. However, the convergence time can be monitored and managed. For instance, if we desire to maintain the synchronization error caused by clock skew below 100 μs, we can enforce a convergence time upper bound of 2 s given a drift rate of 50 ppm. If IPS-based TouchSync does not converge within 2s, the Adaptive Period Mechanism presented in Section 6.1.2 should be used to increase the IPS period.

In this paper, our analysis ignores clock skews and focuses on estimating the clock offset between the master and slave. Existing clock skew compensation approaches (e.g., [13], [14], [26], [27]) can be integrated with TouchSync to address clock skew and further improve the clock synchronization performance.

7 Implementation of TouchSync

Designed as an application-layer clock synchronization approach, TouchSync can be implemented as an app or part of an app, purely based on the standard wearable OS calls to sample the SEP signal, exchange network messages, and timestamp them in the application layer. To simplify the adoption of TouchSync by application developers, we have implemented TouchSync's platform-independent tasks in ANSI C and provide them in a `touchsync.h` header file [15]. These tasks include buffer management, SEP signal processing, and IAS. Other tasks of TouchSync, i.e., sensor sampling, synchronization message exchange and timestamping, are platform dependent. We leave them for the application developer to implement. As these tasks are basics for embedded programming, by using the platform-independent algorithms provided by `touchsync.h`, application developers without much knowledge in signal processing can readily implement TouchSync on different platforms. Our own Arduino and TinyOS programs that implement TouchSync's workflow have about 50 and 150 lines of code only, respectively. The computation overhead of our implementation on two platforms is omitted here and can be found in Appendix C.

We use a Monsoon power monitor to measure the energy consumption of TouchSync on a Flora node. With TouchSync running, the node consumes 57.16 mW on average. Without TouchSync, the node's average idle power consumption is 54.99 mW when the node is not in the sleep mode. Thus, TouchSync consumes 2.17 mW or 0.658 mA (with 3.3 V battery) on average. Assume we would like to run TouchSync on two wearable devices, each with a 150 mAh battery and a 50 ppm crystal, which can continuously run without sleep for a whole day after fully charged. In order to keep the clock offset within 7 ms, TouchSync should be activated every 140 seconds. Under such settings, the battery time of the wearable will be reduced by 39 seconds per day due to the running of TouchSync. In other words, the execution of TouchSync reduces the battery time by 0.045% only. Note that when the wearable has a sleeping schedule to prolong the battery time (e.g., around 10 days by a duty cycle of 10%), TouchSync can be executed when the wearable is not in the sleep mode.

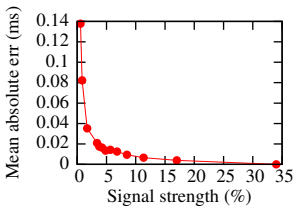


Fig. 18. Impact of signal strength.

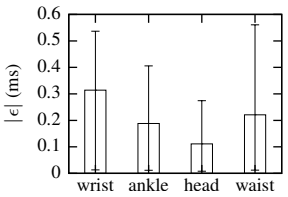


Fig. 19. Wearing position.

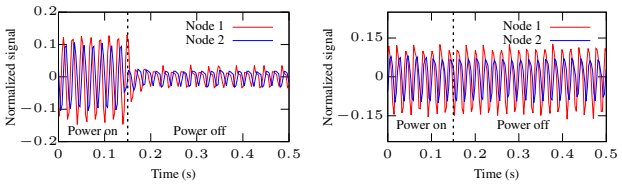
8 Performance Evaluation

We conduct extensive experiments to evaluate the performance of TouchSync in various real environments. Each experiment uses two Flora nodes, which act as the TouchSync slave and master, respectively. As Flora does not support BLE master mode, the two Floras cannot communicate directly. Thus, we use a RPi that operates as a BLE master to relay the data packets between the two Floras. This setting is consistent with most body-area networks with a smartphone as the hub. If the hub can also sample powerline radiation or SEP , each wearable can also synchronize with the hub directly using TouchSync. We use the approach discussed in Section 4.1 to obtain the ground truth clock of each Flora. The details and the results of our experiments are presented below.

8.1 Signal Strength and Wearing Position

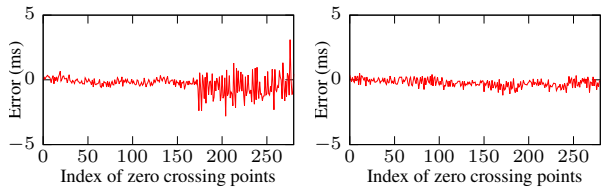
As the intensity of powerline radiation attenuates with distance, SEPs will have varying signal strength. Thus, we evaluate the impact of the SEP signal strength on the performance of the signal processing pipeline in Section 5.1. We measure the signal strength as follows. For a full-scale sinusoid signal with a peak-to-peak amplitude of one (normalized using ADC's reference voltage), its standard deviation is $0.5/\sqrt{2} = 0.354$. The signal strength of a normalized sinusoid with a standard deviation of σ is defined as $\sigma/0.354$. Thus, a 100% signal strength suggests a full-scale signal for the ADC. For this experiment, we use a Flora to record a SEP signal. The strength of this signal is 34%. We feed the signal processing pipeline with this signal to generate a series of *baseline* ZCs. Then, we scale down the amplitude of this signal, re-quantize it, and process it using the pipeline to generate another series of ZCs. We use the mean absolute error (MAE) of these ZCs with respect to the baseline ZCs as the error metric. Fig. 18 shows the MAE versus the strength of the scaled down signal. When we scale down the signal by 60 times, yielding a signal strength of 0.6%, the ZCs' MAE is 0.14 ms only. This suggests that TouchSync can still detect the ZCs accurately even when the SEP signal is rather weak.

We evaluate the impact of the wearing position on the synchrony of SEP signals. A researcher wears a Flora on his left wrist. Then, he conducts four tests by fixing the second Flora to his right wrist, right ankle, forehead, and waist, respectively. Each test lasts for two minutes. Fig. 19 shows the error bars (5%-95% confidence interval) for the absolute time displacement $|\epsilon|$ between the two Floras in these four tests. The average $|\epsilon|$ values in the four tests are close. This suggests that the wearing positions have little impact on the



(a) Electric oven (b) Microwave oven

Fig. 20. EMR signals near an electric oven and a microwave oven.



(a) Electric oven (b) Microwave oven

Fig. 21. Errors introduced into time displacements of the EMR signals.

synchrony of SEP signals and the synchronization accuracy of TouchSync.

8.2 Impact of High-Power Appliances on TouchSync

Since the periodic SEP signal is induced by the EMR field, EMR interferences may distort the SEP signals and result in large and dynamic time displacement between two SEP signals. This may consequently cause clock synchronization errors. High-power electrical appliances may generate time-varying EMR interferences. In this section, we investigate the impact of these appliances on the EMR waveforms and the time displacement between the EMR signals collected from two nearby Floras. We conduct experiments with an electric oven and a microwave oven as the electrical appliances, two representative high-power appliances found in home and office environments. Their rated powers are 800 W and 1050 W, respectively.

In the experiment for each appliance, we place the two Floras close to the tested appliance. The electric oven is placed on a table in a kitchen with no other appliances nearby, whereas the microwave is in an office pantry with other appliances running nearby including a fridge and a water heater. In each experiment, we switch on the tested appliance and switch it off by unplugging it from the power outlet. By investigating the data traces collected by the two Floras when the appliance is on and off, we can understand the impact of the appliance on the EMR and SEP signals.

Fig. 20 shows the raw EMR signals, collected from the electric oven and the microwave oven. From Fig. 20(a), the EMR amplitude decreases significantly when the electric oven is switched off. Note that the electric oven is a resistive load. Its change of operating status will lead to significant change of electric current through the appliance and the related powerlines. Thus, the operating status of the electric oven significantly affects the nearby EMR. In contrast, from Fig. 20(b), the EMR amplitude does not change when the microwave is switched off. A potential cause is that, the EMR sensed by the Flora is dominated by several other nearby appliances including a fridge and an automatic water heater. Microwave ovens generate EMR at 2.45 GHz.

TABLE 1
Signal strength and TouchSync accuracy.

	Test point	Without skin contact			With skin contact		
		Signal strength	K	error (ms)	Signal strength	K	error (ms)
Laboratory	L1	2.6%	3	-0.2	84.7%	2	-0.7
	L2	3.2%	2	0.3	31.5%	3	-0.7
	L3	2.3%	2	-2.5	26.1%	2	0.5
	L4	4.0%	1	-0.6	33.7%	2	0.0
	L5	0.8%	15	1.1	3.3%	2	-0.2
	L6	5.7%	10	-0.4	39.5%	10	-0.0
	L7	2.3%	n.a.	n.a.	3.0%	2	-0.9
	L8	4.6%	2	-1.5	8.3%	2	0.6
	L9	2.6%	1	-1.2	67.4%	2	-0.9
Home	H1	4.2%	2	-1.1	8.9%	2	-0.8
	H2	3.4%	1	-0.9	14.5%	2	-1.0
	H3	4.6%	1	-1.3	44.9%	2	0.2
	H4	7.8%	n.a.	n.a.	39.2%	2	0.3
	H5	3.8%	1	-1.6	3.9%	1	-2.8
	H6	3.9%	4	-4.4	9.9%	2	-2.3
	H7	5.0%	2	-1.9	6.8%	1	-2.9
	H8	8.2%	1	-11.5	54.7%	4	-1.3
	H9	2.9%	1	-2.4	9.1%	1	-1.3
Office	O1	4.0%	n.a.	n.a.	3.3%	4	0.4
	O2	5.6%	1	-7.9	2.9%	2	-1.6
	O3	1.7%	1	-0.4	3.9%	2	-0.3
	O4	5.4%	3	-2.5	5.8%	2	-0.8
	O5	4.8%	6	-6.2	5.6%	12	-0.2
Corridor	C1	3.6%	12	0.1	4.4%	11	0.7
	C2	6.2%	2	0.6	44.2%	2	-1.0
	C3	5.8%	1	-7.6	4.4%	1	-1.1
	C4	1.9%	1	-6.0	2.2%	1	-2.8
	C5	1.9%	2	-3.7	5.8%	1	-1.0

* n.a. means that TouchSync cannot converge due to large ϵ.

However, such high-frequency EMR cannot be effectively received by the Floras. From Fig. 20(b), the EMR signal received by the Flora is primarily at the 50 Hz. In other words, the 2.45 GHz EMR emitted by the microwave, if present, does not disrupt the 50 Hz EMR.

We also evaluate the impact of the electric and microwave ovens on the synchrony of the EMR signals collected from the two Floras. Fig 21 shows the time displacement over the courses in Fig. 20. From Fig. 21(a), after the electric oven is turned off, the intensity of time displacement fluctuation becomes larger. This is because, when the oven is switched off, the EMR strength and its signal-to-noise ratio decreases, resulting in more fluctuating time displacement. This suggests that, TouchSync's clock synchronization accuracy will be better when a nearby high-power resistive load is operating. But the mean value of the time displacement is still around zero. From Fig. 21(b), the time displacement keeps stable. From the results in Fig. 21, we can see that the nearby high-power appliances introduce little impact on the time displacement between two EMR signals and thus the performance of TouchSync.

8.3 Evaluation in Various Environments

We evaluate the SEP signal strength and the accuracy of TouchSync in various indoor environments.

Lab: We conduct experiments in a computer science laboratory with about 100 seats and various office facilities (lights, computers, printers, projectors, meeting rooms, etc). A researcher carries the Floras to each test point and conducts two experiments. In the first experiment, the Floras have no physical contact with human body; in the second

experiment, the researcher wears the two Floras. Thus, the experiment evaluates the same-wearer scenario. The example applications mentioned in Section 1, i.e., motion analysis, and muscle activation monitoring, belong to this scenario. Each synchronization session takes about 150 ms. A synchronization process completes once the IAS converges.

The first part of Table 1 shows the SEP's signal strength, the number of synchronization sessions until convergence (K), and the clock offset estimation error at each test point. Without skin contact, the signal strength is a few percent only. But TouchSync can still achieve a 3 ms accuracy. At L7, TouchSync cannot converge because of large and varying ϵ. The skin contact significantly increases the signal strength. Moreover, TouchSync converges after two synchronization sessions in most cases. For $K = 2$, a synchronization process takes less than one second. The absolute clock offset estimation errors are below 1 ms, lower than those without skin contact. However, without skin contact, the accuracy does not substantially degrade. This suggests that, TouchSync is resilient to the loss of skin contact due to say loose wearing.

Home: We conduct experiments in a 104 m^2 three-bedroom home with typical home furniture and appliances. A floor plan of the home is omitted here due to space constraints and can be found in Appendix D. We arbitrarily select nine test points. The second part of Table 1 shows the results. Without skin contact, the signal strength results are similar to those obtained in the laboratory. With skin contact, the signal strength increases and the absolute clock offset estimation errors are below 3 ms.

Office: The third part of Table 1 shows the results obtained at five test points in a 15 m^2 office. At test points O1 and O2, the signal strength with skin contact is slightly lower than that without skin contact. This is possible as the two tests were conducted during different time periods and the powerline radiation may vary over time due to changed electric currents. With skin contact, TouchSync gives sub-ms accuracy except at O2.

Corridor: We select five test points with equal spacing in a 200 m corridor of a campus building. The fourth part of Table 1 gives the results. With skin contact, TouchSync yields absolute clock offset estimation errors of about 1 ms except at C4. The errors with skin contact are lower than those without skin contact.

In summary, with skin contact, TouchSync gives sub-ms clock offset estimation errors at 20 test points out of totally 28 test points in Table 1. All errors are below 3 ms.

From Table 1, at 3 out of 28 test points, the test without skin contact gives higher signal strength than that with skin contact. This is because the two tests are conducted sequentially and the EMR may change over time. Moreover, the signal strength exhibits significant variation at different locations. This is because the EMR decays with the distance from the powerline. Nevertheless, the above results show the pervasive availability of SEP in indoor environments.

8.4 TouchSync-over-Internet

Tightly synchronizing wearables over long physical distances is often desirable. For instance, in distributed virtual reality applications, tight clock synchronization among participating sensing and rendering devices that may be geographically distributed is essential. Although the synchronization can be performed in a hop-by-hop manner (e.g., wearables ↔ smartphone ↔ cloud), errors accumulate over hops. In particular, tightly synchronizing a smartphone to global time has been a real and challenging problem – tests showed that the synchronization through LTE and Wi-Fi experiences hundreds of ms jitters [25]. In contrast, TouchSync can perform end-to-end synchronizations for wearables distributed in a geographic region served by the same power grid. The basis is that, the 50/60 Hz power grid voltage, which generates the powerline radiation, is highly synchronous across the whole power grid [22]. TouchSync can also synchronize wearables directly with a cloud server in the same region. The smartphone merely relays the messages exchanged among the wearables and the cloud server if the wearables cannot directly access Internet. The cloud server can use a sensor directly plugged in to a power outlet to capture the power grid voltage. Owing to the Internet connectivity, the end-to-end synchronization scheme greatly simplifies the system design and implementation.

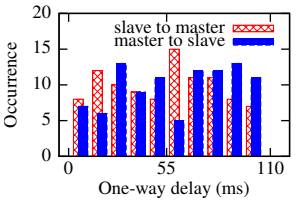


Fig. 22. One-way delays over a ngrok tunnel.

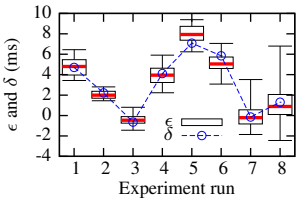


Fig. 23. Accuracy of TouchSync-over-Internet.

We conduct a proof-of-concept experiment of TouchSync-over-Internet as follows. Two researchers carry a Flora-RPi setup each to two buildings that are about 10 km apart. The RPi is attached with an Adafruit GPS receiver to obtain ground-truth coordinated universal time (UTC) with μs accuracy. The two nodes, one as TouchSync master and the other as TouchSync slave, communicate through a tunnel established by ngrok 1.7, an open-source reverse proxy often adopted for creating IoT networks. Fig. 22 shows the distributions of the two one-way delays over the ngrok tunnel. We can see that the ngrok exhibits significant dynamics. We evaluate TouchSync-over-Internet for eight times during a day. Fig. 23 shows the box plots of the time displacements (i.e., ϵ) between the SEPs captured by the two nodes. We can see that ϵ varies from -2 ms to 9 ms during the day. From the building managements, the two rooms where the master and slave nodes are located draw electricity from the R and Y phases of the power grid, respectively. There is a phase difference of $20/3 = 6.7$ ms between these two phases. Moreover, from power engineering, the difference between the power grid voltage phases at different geographic locations is non-zero and time-varying. These factors lead to the non-zero and time-varying ϵ in Fig. 23. The dotted line in Fig. 23 shows the synchronization errors of TouchSync-over-Internet (i.e., δ) in various experiment runs. They are within the range of ϵ, since ϵ is the major source of TouchSync's synchronization error. The largest δ is 7 ms. The integer ambiguity solver converges within 4 to 13 synchronization sessions.

9 Conclusion

TouchSync synchronizes the clocks of wearables by exploiting the wearers' skin electric potentials induced by power-line radiation. Different from existing WSN clock synchronization approaches that find difficulties in being applied on diverse IoT platforms due to their need of hardware-level packet timestamping or non-trivial extra hardware, TouchSync can be readily implemented as an app based on standard wearable OS calls. Extensive evaluation shows TouchSync's synchronization errors of below 3 ms and 7 ms on the same wearer and between two wearers 10 km apart, respectively.

References

[1] "Gartner says worldwide wearable device sales to grow 17 percent in 2017," 2017, http://gtnr.it/2AAHfgG.
[2] M. Chan, D. Estève, J.-Y. Fourniols, C. Escriba, and E. Campo, "Smart wearable systems: Current status and future challenges," *Artificial intelligence in medicine*, vol. 56, no. 3, pp. 137–156, 2012.
[3] F. Mokaya, R. Lucas, H. Noh, and P. Zhang, "Burnout: a wearable system for unobtrusive skeletal muscle fatigue estimation," in *IPSN*, 2016.
[4] K. Lorincz, B.-r. Chen, G. Challen, A. Chowdhury, S. Patel, P. Bonato, and M. Welsh, "Mercury: a wearable sensor network platform for high-fidelity motion analysis," in *SenSys*, 2009.
[5] F. Mokaya, R. Lucas, H. Y. Noh, and P. Zhang, "Myovibe: Vibration based wearable muscle activation detection in high mobility exercises," in *UbiComp*, 2015.
[6] Apple Inc., "Device synchronization over bluetooth," 2015, https://www.google.com/patents/US20150092642.
[7] "NTP: The network time protocol," 2017, http://www.ntp.org/.
[8] IEEE, "Ieee standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008*, pp. 1–269, 2008.
[9] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *OSDI*, 2002.
[10] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys*, 2003.
[11] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *SenSys*, 2004.
[12] R. Casas, H. J. Gracia, A. Marco, and J. L. Falco, "Synchronization in wireless sensor networks using bluetooth," in *WISES*, 2005.
[13] A. Rowe, V. Gupta, and R. R. Rajkumar, "Low-power clock synchronization using electromagnetic energy radiating from ac power lines," in *SenSys*, 2009.
[14] L. Li, G. Xing, L. Sun, W. Huangfu, R. Zhou, and H. Zhu, "Exploiting FM radio data system for adaptive clock calibration in sensor networks," in *MobiSys*, 2011.
[15] Z. Yan, Y. Li, R. Tan, and J. Huang, "Touchsync implementation," 2017, https://github.com/yanmarvin/touchsync.
[16] "Symmetricom annouces general availability of industry's first commercially-available chip scale atomtic clock," 2011, http://bit.ly/2ctnwjB.
[17] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *IPSN*, 2011.
[18] C. Lenzen, P. Sommer, and R. Wattenhofer, "Optimal clock synchronization in networks," in *SenSys*, 2009.
[19] F. Terraneo, A. Leva, S. Seva, M. Maggio, and A. V. Papadopoulos, "Reverse flooding: Exploiting radio interference for efficient propagation delay compensation in wsn clock synchronization," in *RTSS*, 2015.
[20] R. Lim, B. Maag, and L. Thiele, "Time-of-flight aware time synchronization for wireless embedded systems," in *EWSN*, 2016.
[21] Y. Chen, Q. Wang, M. Chang, and A. Terzis, "Ultra-low power time synchronization using passive radio receivers," in *IPSN*, 2011.
[22] S. Viswanathan, R. Tan, and D. Yau, "Exploiting power grid for accurate and secure clock synchronization in industrial IoT," in *RTSS*, 2016.
[23] D. Rabadi, R. Tan, D. Yau, and S. Viswanathan, "Taming asymmetric network delays for clock synchronization using power grid voltage," in *AsiaCCS*, 2017.
[24] Y. Li, R. Tan, and D. Yau, "Natural timestamping using powerline electromagnetic radiation," in *IPSN*, 2017.
[25] P. Lazik, N. Rajagopal, B. Sinopoli, and A. Rowe, "Ultrasonic time synchronization and ranging on smartphones," in *RTAS*, 2015.
[26] T. Hao, R. Zhou, G. Xing, and M. Mutka, "WizSync: Exploiting Wi-Fi infrastructure for clock synchronization in wireless sensor networks," in *RTSS*, 2011.
[27] Z. Li, W. Chen, C. Li, M. Li, X.-Y. Li, and Y. Liu, "FLIGHT: Clock calibration using fluorescent lighting," in *MobiCom*, 2012.
[28] J. He, X. Duan, P. Cheng, L. Shi, and L. Cai, "Distributed time synchronization under bounded noise in wireless sensor networks," in *CDC*, 2014.
[29] "Msp430 lfxt1 oscillator accuracy," http://www.ti.com/lit/an/slaa225/slaa225.pdf, 2004.

Zhenyu Yan is a Ph.D. candidate at School of Computer Science and Engineering, Nanyang Technological University, Singapore. He received his bachelor degree in Electronic Engineering and Finance from University of Electronic Science and Technology of China in 2016. His research interests include Internet-of-Things, mobile computing and sensor networks.

Rui Tan (M'08-SM'18) is an Assistant Professor at School of Computer Science and Engineering, Nanyang Technological University, Singapore. Previously, he was a Research Scientist (2012-2015) and a Senior Research Scientist (2015) at Advanced Digital Sciences Center, a Singapore-based research center of University of Illinois at Urbana-Champaign, and a postdoctoral Research Associate (2010-2012) at Michigan State University. He received the Ph.D. (2010) degree in computer science from City University of Hong Kong, the B.S. (2004) and M.S. (2007) degrees from Shanghai Jiao Tong University. His research interests include cyber-physical systems, sensor networks, and pervasive computing systems.

Yang Li is currently an assistant professor at Shenzhen University, China. Before that, he was a postdoctoral researcher in ADSC of UIUC, Singapore. He obtained his PhD degree in Energy and System from INSA Lyon, France in 2014. He earned his BSc and MSc degrees in Mechatronics Engineering from Northwestern Polytechnical University, China in 2008 and 2010. His research interests including smart manufacturing, energy harvesting and smart sensors. He is the recipient of IPSN2017 Best Paper award.

Jun Huang received the BS and MS degrees in computer science from Beihang University, China, and the PhD degree in computer science and engineering from Michigan State University. He is currently an assistant professor at the School of EECS, Peking University. He received the Best Paper Awards at the 18th IEEE International Conference on Network Protocols (ICNP) in 2010. His research interests include wireless and mobile systems.