A Precise and Hardware-Efficient Time Synchronization Method for Wearable Wired Networks

Fardin Derogarian*, João Canas Ferreira, Vítor M. Grade Tavares

INESC TEC, Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, 4200 - 465, Porto, Portugal

***Abstract*—This paper presents and evaluates a high-precision, one-way, master-to-slave time synchronization protocol to minimize the clock time skew in low-power wearable sensor networks. The protocol is implemented in the MAC layer, and is based on directly eliminating deterministic delays during transmission from source to destination node, at hardware level. The proposed protocol keeps the one-hop average synchronization error close to the signal propagation delay and the one-hop peak-to-peak jitter equal to the period of each node's system clock period. Both values grow linearly as the hop count increases. The protocol can achieve synchronization in the range of a few nanoseconds, enough to satisfy the requirements of many applications for wearable networks, with one-way messages. Both theoretical analysis and experimental results, in wired wearable networks, show that the proposed protocol has a better performance than PTP, a standard timing protocol for both single and multi-hop situations. The proposed approach is simpler, requires no calculations, and exchanges fewer messages. Experimental results obtained with an implementation of the protocol in 0.35 µm CMOS technology show that this approach keeps the one-hop average clock skew around 4.6 ns and peak-to-peak skew around 50 ns for a system clock frequency of 20 MHz.**

***Index Terms*—Time synchronization protocol, wearable wired sensor network, hardware implementation.**

I. Introduction

Synchronization, in general, and time synchronization, in particular, are important features of many networks and distributed systems. In sensor networks, like in many other data acquisition systems, it is often necessary to record timing information with the data. For these systems, maintaining performance and precise synchronization between the nodes, using only limited resources, can be a significant challenge. Monitoring of the environment, localization, security, TDMA-based protocols, coordinated sleep wake-up scheduling mechanisms and data fusion are some of the applications based on time synchronization [20]. For instance, in gait analysis for evaluation and diagnosis of mobility impairments it is important to know the relative timing of several different surface electromyography signals and their time relation to the inertial information about the movements of the lower limbs.

In a sensor network, each individual Sensor Node (SN) is usually equipped with an independent clock generator. The use of a crystal quartz oscillator ensures good local clock accuracy. Although this kind of oscillators have good stability (with variation around a few parts per million (ppm)) that is not enough to keep synchronization during the whole time of activity; since the clock generators are working independently, small drifts in clock generation lead to synchronization loss. One way to maintain precise synchronization is to use a GPS receiver, so that SNs can achieve clock synchronization in the nanosecond range. However, sensors usually do not need such a high accuracy. In addition, the use of GPS receivers in resource limited systems is not always practical or possible (for indoor use, for instance). An alternative approach is to use a synchronization mechanism that overcomes and compensates the clock drift in each node [12], [21], [23], [27].

The time synchronization problem, which has been studied extensively [6], [20], [23], [25], is impacted by many factors and may require the consideration of issues at several protocol layers. A significant issue is the variable amount of time required to send a single message. This time is usually composed of four components [17]: 1) the time required

*Correspondence to: Fardin Derogarian, INESC TEC, Faculdade de Engenharia da Universidade do Porto (FEUP), Rua Dr. Roberto Frias, 4200 - 465, Porto, Portugal. Email: mpt09020@fe.up.pt, Tel: +351 968 726 403, Fax: +351 222 094 050

for assembling a message and handing it over to the MAC layer; 2) the time required for accessing the channel; 3) the propagation delay between the communicating nodes; 4) the time required for acquiring and processing the message at the receiver. Most of these factors are non-deterministic in general. The exception is the propagation time, which depends on the communication environment and may be considered to be constant for stationary nodes. In order to reduce the effects of delay on synchronization, some protocols, such as Reference-Broadcast Synchronization (RBS), sample and inject the time information into the message while transmitting [7], thereby reducing the variability associated with the first two factors mentioned above.

Many synchronization protocols use two-way message exchanges between pairs of nodes to estimate the clock time skew and offset [8], [14], [19]; the nodes exchange messages with timing information in order to estimate the round trip delay, which is then used to adjust the local timing information of the client node. Depending on the protocol, the exchange may be started by the clock source node or by the client node (i.e., the node whose clock is to be adjusted). A second approach uses only one-way communication [7], [15], [17], [26]. In this case, the clock source sends timing information to client nodes, which process the information to estimate the local time adjustment. This approach requires fewer messages, but usually their accuracy is lower than the two-way methods.

This paper presents a one-way method for synchronization at the MAC layer of nodes in a wearable, wired sensor network intended for clinical applications. The proposed approach minimizes the average clock time skew up-to the propagation delay. In particular, this paper addresses the need for good time synchronization in simultaneous acquisition of surface electromyographic signals coming from different muscles. In our main application case, the electrodes are embedded in the clothes of the patient and connected to SNs equipped with A/D converters. The SNs are connected together in a network using conducting yarns embedded in the clothes.

In the context of such wearable sensor network, the main contributions of this work are: (1) analytic characterization of the proposed protocol, including the determination of the average local and global clock skew, as well as an analyzis of the probability of synchronization in the presence of message failures; (2) extensive experimental evaluation of the protocol using a hardware implementation (CMOS integrated circuit – IC) reported in [5]. The proposed protocol is shown to keep the one-hop average synchronization error close to the signal propagation delay and the one-hop peak-to-peak jitter equal to the period of each node's system clock period.

The rest of the paper is organized as follows. Section II provides the background and describes related work. Section III presents the motivation for the proposed protocol. Section IV gives an overview of the protocol and its implementation. The presentation and discussion of the experimental results is done in Section VI and the main conclusions are summed up in Section VII.

II. Related Work

Synchronization protocols can be classified according to the relationship between the clock reference and client nodes [25]. A peer-to-peer protocol allows each node to get timing information directly from any other suitable node. The advantages of this kind of protocols are flexibility and resilience against node failures, but their control is complex. In many cases, it is not possible for two nodes to communicate directly because communication occurs over multi-hop connections, particularly when the number of nodes is large. The alternative is to use a master-slave approach, where the master node is the time reference and synchronization performed by either sender-to-receiver or receiver-to-receiver policies. In the first case, the receiver synchronizes its clock with the sender and then sends the updated timing data to the next node. In the second case, the sender broadcasts timing information and then the receivers exchange messages among themselves instead of interacting with the sender.

The well-known Network Time Protocol (NTP) is based on a two-way message exchange [19]. This effective and robust protocol finds wide use on the Internet. NTP clients synchronize their local clocks to NTP time servers by statistical analysis of the round trip time. The fact that NTP imposes significant implementation complexity restricts its adoption in sensor networks.

The RBS protocol [7] is a one-way protocol for Wireless Sensor Networks (WSNs), in which a clock reference node broadcasts its time, thereby allowing all adjacent nodes to receive the same timing message with very little variability. The nodes use a sequence of broadcast synchronization messages to estimate both offset and skew of the local clocks. The implementation of the RBS protocol on IEEE 802.11 wireless networks described in [7] can achieve a one-hop accuracy of $1.85\,\mu s \pm 1.28\,\mu s$.

The Timing-sync Protocol for Sensor Networks (TPSN) is a sender-receiver, two-way synchronization protocol for multi-hop networks [8]. TPSN performs synchronization in two phases: level discovery phase and synchronization phase. In the first phase, a root node chosen as clock reference elects and builds a spanning tree of the network. In the second phase, nodes synchronize to their parent in the tree by exchanging messages at the initiative of the child node. When the root node or the network topology change, TPSN has to start again with the discovery phase, resulting in an increased message traffic and network energy consumption.

The Flooding Time Synchronization Protocol (FTSP) is another one-way protocol for multi-hop ad-hoc networks [18]. A root node periodically broadcasts a message with global time stamping information. Each receiver node estimates its clock offset and skew by combining both global and local time information. The root node is elected dynamically and periodically based on the smallest node identifier and is responsible for keeping the global time of the network. FTSP is an effective synchronization mechanism for multi-hop ad-hoc networks, but in [15] it is shown that the clock skew increases exponentially with increasing network size.

An hardware-based synchronized clock circuit for use in sensor networks is presented in [22]. Synchronization is based on using the ambient magnetic field emitted by 60 Hz power lines. Another clock synchronization circuit, but this time based on the ambient electric field, is described in [1]. The first approach is able to achieve an average synchronization of less than 1 ms between all nodes in a multi-hop network, while the second approach is able to keep the clock drift within $0.864\,\mu s$ after running for 24 hours. The main limitation of these approaches is the necessity for having an additional module on each node; in addition, power-line fields are not stable and available everywhere.

IEEE standard 1588 [10] defines the Precision Time Protocol (PTP), a hierarchical, master-slave protocol for clock distribution with very precise synchronization [14]. Under this protocol, the clock server periodically sends synchronization messages. A client replies with a message including the reception time of the first message from the server and the reply time according to his own clock. Then the server estimates the client's delay and time offset and sends this information to the client, which adjusts his local time. The clients must periodically update their time to mitigate the effects of local clock drift. PTP can ensure that the one-hop clock skew stays in the sub-microsecond range [2]. The issues affecting the use of PTP for timing recovery in packet-based networks, particularly those aimed at telecommunication networks, are discussed in [24].

The protocols mentioned in this section are widely used, but their generality implies that they are not hardware-aware protocols. Therefore, their performance depends on the implementation characteristics of each particular system. Since one of the challenges in synchronization is the estimation of the non-deterministic delay factors due to the properties of the hardware, reducing them may lead to a simpler and more resource-efficient synchronization method for specific applications.

III. Motivation

The synchronization protocol proposed in this work was motivated by the requirements of a wearable system for data capture of human locomotion, including kinematic parameters of the lower limbs and myoelectric signals of the surface muscles [28]. The acquired data is intended for use in the detection and identification of mobility impairments, selection of orthotics or prosthetics devices, and evaluation of the effectiveness of rehabilitation treatments. However, the proposed approach should be easily extended to other similar biomedical applications.

One of the main requirements of the project is that data needs to be acquired in a practical and non-invasive way. Previous research work has shown that the most comfortable and easiest way to monitor physiologic signals is to use garments [9], [16]. The final system includes sensor nodes embedded in textile fabrics, connected to each other with conductive yarns in a mesh topology. The use of fabrics with embedded conductors makes the product more user-friendly and comfortable. Each sensor node is capable of acquiring sEMG (surface electromyography) signals from electrode pairs and kinematic data from 3D inertial sensors, but must meet severe restrictions on physical size, so as not to impair comfort and wearability.

Since the relative timing of the acquired data is relevant for subsequent processing, a synchronization mechanism for data time-stamping is required. Data rates in BAN applications are significantly higher than in other sensor networks (e.g., 32 kbps to 64 kbps for a 16-bit EMG sensor) [13]. Therefore, the synchronization protocol has to ensure timing accuracy in the range of a few microseconds. The main aspects that have been considered in the design of the protocol are:

Energy efficiency: Due to the portability requirements of the aforementioned system, its sensor nodes use a small battery for power supply. To ensure an adequate operation time with such a limited energy source, the efficient usage of energy in all parts of the system has to be enforced. The communication infrastructure, including transmitter and receiver circuits, consumes a significant amount of energy when they are in active mode. Therefore, the number of control messages and transmitted data packets has to be reduced as much as possible. In the proposed protocol, synchronization is based on one-way message exchange, which consumes less energy than two-way methods. The use of a one-way method might increase the processing cost or the time to achieve synchronization, but, as evidenced by the experimental results discussed in Section VI, the performance of the proposed protocol is not affected by these drawbacks.

Accuracy: Unknown delay and latency of messages generally have an effect on time synchronization. Messages experience different delays while passing between network layers. Increasing the number of involved layers produces a cumulative increase of end-to-end delay. This applies specially to the network layer, because the buffering of messages and the variable service time due to changing network traffic make it hard to estimate message delay and reduce the accuracy. Therefore, many synchronization protocols have been implemented in the MAC layer, so as to achieve high accuracy and fast synchronization. The current protocol has also been designed for use in the MAC layer.

Simplicity: The limited computational resources available in sensor nodes always constitute a key factor that has to be considered in the design of both hardware and software. Keeping the communication and data processing delays within a fixed range reduces message processing and the size of the hardware needed to implement the protocol in the MAC layer. In comparison to two-way protocols, the proposed method requires less calculations, because it minimizes the variability factors associated with message sending.

A one-way, master-slave, sender-to-receiver mechanism that satisfies these criteria and enables a resource-efficient hardware implementation is described in the next section.

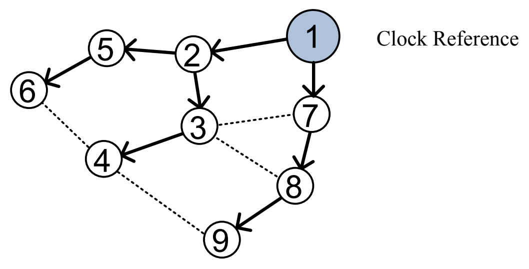
IV. Description of the Synchronization Protocol

This section describes the proposed time synchronization protocol and its intended deployment. Figure 1 shows an example of a mesh network of SNs connected to each other by wires (conductive yarns in the intended application). All SNs are equipped with a multi-port network module. Such a network module includes synchronization and packet routing sub-modules working independently of each other. Each device can perform time synchronization with the sender node over one port, while participating in packet routing with the other free ports. Node 1 is the Base Station (BS) responsible for network management, master clock time reference and data collection. A routing protocol is needed to route packets from SNs to BS. Here the energy-efficient routing protocol – Source Routing for Minimum Cost Forwarding (SRMCF) – from [3] is used, which is based on the combination of source routing and minimum cost forwarding.

Before starting normal activity, the system must go through a setup phase. In this step, the routing protocol finds all minimum-cost paths between BS and SNs, as shown by bold lines in Figure 1. The dashed lines are redundancy links that increase the robustness of the system against line breaks caused by wear. The SRMCF protocol requires each node to determine its neighboring node that lies on the path with minimum cost to the BS (called the *near-node*). The path information is then sent to the BS, who keeps a database of available SNs and the corresponding paths. Synchronization information is transmitted as control messages over the minimum-cost spanning-tree built at this phase.

After the networking setup, the BS, acting as a clock reference, synchronizes the network by sending timing information whenever a neighbour SN sends a request. Each SN synchronizes with its *near-node* and rejects timing messages from the other nodes. For instance, node 3 will synchronize with node 2 and reject messages from nodes 7 and 8. Because messages received over different paths experience different delay variations (due to different number of hops, network traffic and packet loss), accepting timing information from various sources would cause message jitter.

End-to-end communication time is usually not deterministic and suffers specially from buffering in the network layer. This protocol is designed for implementation in the MAC layer in order to reduce the effects of delay and the clock time skew. A dedicated hardware transmitter module on each node



..

Fig. 1. A network of sensors with the BS node acting as clock time reference. The bold lines indicate the tree used for synchronization messages.

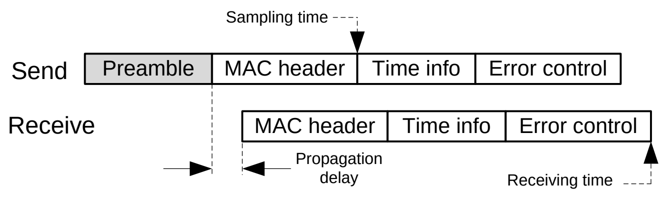


Fig. 2. Format of the timing information message (MAC layer) and received message at the receiver.

is responsible for the generation of the timing messages. To reduce energy consumption, the communication modules are in sleep mode when there is no data to be transfered.

Figure 2 shows a data frame with a timing message, both as constructed by the sender and as processed by the receiver. The preamble of a timing message is a string of 1s, which is used to wake up the receiver before starting the actual data transfer. The MAC header enables the receiver to recognize an incoming timing message. Then the receiver activates a hardware module designed to process the timing information directly without the involvement of other parts of the receiver. Immediately after sending the MAC header, the master samples the current time, formats the timing information, and sends it. Sampling the time at the sender in this way reduces the delay error due to the message sending process. Error control information is put at the end of the message, so that its integrity can be checked by the receiver.

Delay due to the propagation of signals over the link between the nodes is typically very small and negligible (for instance, a delay of about 3.3 ns per meter was measured in our wearable network). The main delay is due to the difference between the instant of time sampled at the sender and that taken when the message is received. For the data frame shown in Figure 2, the transfer takes a fixed number of system clock cycles, because the number of bits inserted in the message after time sampling is fixed. Therefore, the receiver always gets messages after a fixed delay (ignoring clock drift and propagation time). If the transmitter adds this fixed number of clocks to the sampled time, then the receiver has all the necessary information to determine exactly the instant when the incoming packet arrived (as measured by the clock of the sender). If the receiver validates the new timing information, then it can set its own time to the received time without further processing.

After synchronizing its time with the sender node, each SN sends a timing message to its successors on the clock time tree. For example in Fig. 1, node 2 sends timing messages to the nodes 3 and 5. After adjusting their time, nodes 5 will send timing messages to nodes 6, and node 3 will send a message to node 4.

We assume that each SN uses the time information to manage a globally synchronized clock signal *Clk-sync*, whose frequency is smaller than the system frequency. A simple way to generate such a clock signal is to use an auto-reload down-counter: when the value of the counter reaches zero, an active transition of *Clk-sync* is generated and the counter is reloaded with a predefined value (equal for all nodes). In this case, synchronizing *Clk-sync* signals is equivalent to keeping the values of all counters synchronized. The counter in each node is driven by the local system clock, which exhibits some clock skew relatively to the system clock of other nodes. To limit the skew between nodes, each SN periodically updates its counter with the value specified in the timing message coming from its reference node.

A SN may also manage a time stamp to annotate any acquired data. It is assumed that the current time stamp at each SN is determined by an up-counter driven by the globally synchronized *Clk-Sync* signal. The time resolution depends on the frequency of *Clk-Sync*, which is application dependent and must be selected according to characteristics of the sensed phenomenon. For example, the sampling rate of electromyography signals is 1 kHz to 2 kHz, which is much smaller than the system clock frequency (usually above 1 MHz).

The use of *Clk-sync* ensures that the time stamp counter operates at the same frequency in all nodes. However, it is still necessary to ensure that its contents are the same by using a dedicated message. After starting up, each SN first synchronizes the *Clk-sync* down counter; then, the time stamp counter contents is synchronized once. Afterwards, the *Clock-sync* counter must be synchronized periodically to compensate for the drift of the local system clock.

Under the proposed protocol, the processing of timing messages at the transmitter side is simply limited to the addition of a fixed number to the sampled time, and the receiver side does not perform any kind of processing to estimate clock time skew. Such minimal processing is important, because the protocol is intended for hardware implementation in an energy-constrained environment. Eliminating the effects of sending and receiving delays on timing messages, as described above, minimizes the clock time skew between SNs and leads to a high precision synchronization mechanism for wired wearable networks with a simple hardware implementation.

V. Analytic Characterization of the Protocol

This section provides an analytic characterization of the proposed protocol's behavior in wired wearable sensor networks. A comparison with Precision Time Protocol (PTP) is done when appropriate. The aspects addressed are: delay and offset bounds, average local and global time skew, and probability of synchronization as a function of packet loss.

A. Instantaneous Delay and Skew

We assume that all nodes have the same nominal system clock period τ. However, the actual clock period of node i will exhibit a small difference due to random local clock drift p_i, so that in general the clock period τ_i at node i is given by:

$$\tau_i = \frac{1}{f_i} = \frac{\tau}{1+p_i} \tag{1}$$

The value of p_i depends on the oscillator, but should remain in the range of few ppm for a stabilized oscillator with crystal quartz.

Consider the multi-hop network of Figure 1 with minimum cost paths forming a spanning tree, and assume that node 1, the BS, is the time reference, which periodically broadcasts its time to other nodes. Figure 3 a) shows the synchronization timing diagram for PTP. The time reference node (node 1) starts by sending a message to the client node (node 2) at time $T_M(t_1)$ (master clock time). Node 2 receives it at $T_S(t_2)$ (slave clock time) and replies at $T_S(t_3)$ with a message that is received by node 1 at $T_M(t_4)$. Node 1 calculates the delay and offset according to Equations (2) and (3) [11], [14]:

$$\text{delay} = \frac{(T_S(t_2) - T_M(t_1)) + (T_M(t_4) - T_S(t_3))}{2} \tag{2}$$

$$\text{offset} = \frac{(T_S(t_2) - T_M(t1)) - (T_M(t_4) - T_S(t_3))}{2} \tag{3}$$

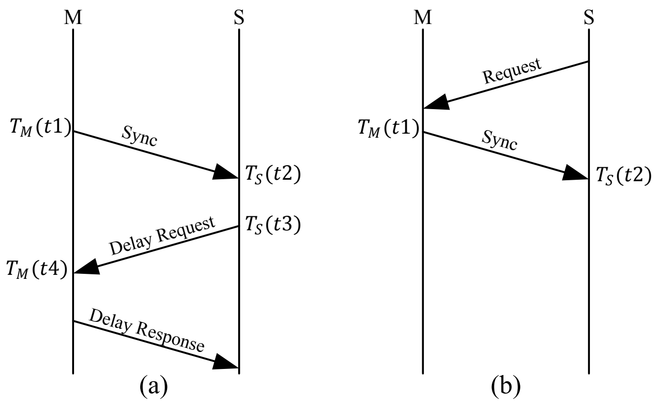


Fig. 3. Timing diagram: a) PTP protocol, b) proposed protocol

If the time value included in the *Sync* request is $T_M(t_1)$, then $T_S(t_2)$ is given by

$$T_S(t_2) = T_M(t_1) + s\,\tau_1 + c\,\tau_d + d_\ell + r\,\tau_2 + S_r(t), \tag{4}$$

where s is the number of clock cycles used in internal processing after $T_M(t_1)$, c is the number of bits transmitted at data-rate period τ_d, d_ℓ is the propagation delay between nodes 1 and 2 (the sum of delays of line and I/O ports), r is the number of clock cycles for message reception at the receiver (with clock period τ_2), and $S_r(t)$ determines the sampling range and is the time difference between signal appearance at the input of the sampling module of receiver and the actual sampling instant. The data rate is not necessarily equal to the system clock frequency of the transmitter, so that in general we have

$$\tau_d = m\,\tau_1, \qquad m \in \mathbb{N}^+. \tag{5}$$

The propagation delay $d_{\ell_{1,2}}$ is usually very small in comparison with the clock speed. We will also assume that $d_{\ell_{1,2}} = d_{\ell_{2,1}} = d_\ell$. Parameters c and r are usually fixed, depending on the packet size. Because we assume that the implementation is in the MAC layer, the receiver node processes the message almost immediately after receiving it, i.e., without putting it in a queue, which decreases the uncertainty delay at the receiver. Under these conditions the delay becomes:

$$\begin{aligned}\text{delay} &= \frac{((s+mc)\tau_1 + d_\ell + r\tau_2 + S_r(t))}{2} + \\ &\quad \frac{((s+mc)\tau_2 + d_\ell + r\tau_1 + S_r(t))}{2} \\ &= d_\ell + S_r(t) + \\ &\quad \frac{(s+mc+r)\tau}{2}\left(\frac{2+p_1+p_2}{(1+p_1)(1+p_2)}\right).\end{aligned} \tag{6}$$

Under the same assumptions, the offset is given by:

$$\text{offset} = (s + mc + r)\tau \left(\frac{p_2 - p_1}{(1+p_1)(1+p_2)} \right). \tag{7}$$

Since p_1 and p_2 are very small values, Equation (6) simplifies to:

$$\text{delay} \approx d_\ell + S_r(t) + (s + mc + r)\tau. \tag{8}$$

Since PTP measures time only in multiple units of τ and, for this application, $d_\ell < \tau$ and $S_r(t) < \tau$, calculation of the delay with a round trip message exchange would always produce a constant value — $(s + mc + r)\tau$. If the transmission delay is kept fixed, then there is no need for multiple calculations to find its value as time progresses. On the other hand, the offset found from Equation (7) is very small (the drift is in the ppm range as stated above), and will be undetectable by a processor or microcontroller because the system clock does not define sufficient time-resolution. In other words, there is no need to have a round trip message when the delay is fixed. In this case, a single message (as in Figure 3(b)) is enough to send timing information, while achieving the same precision. Therefore, a simpler protocol with fewer message exchanges can be used. For this reason, the remainder of this subsection will be concentrated on the one-way method as a proposed approach to achieve minimum clock time skew. Here the clock time skew refers to the total time deviation of each node from the time reference.

Although SNs communicate in asynchronous mode, they are implemented as synchronous logical circuits, where nodes process data at the edge of the clock signal. Since the hardware clock generators at each node are independent, two nodes never present exactly the same clock signal. The situation is illustrated by the timing diagram of Figure 4, which shows the events at two communicating nodes. The transmitter sends data at the positive edge of the clock and the receiver detects incoming data at the negative edge. In this example, bit b_{n-1} is sent at time t_1, and the receiver will acquire it after $d_\ell + S_r$. The ideal value of $S_r(t)$ is half of the data rate period, so that sampling occurs exactly in the middle of the incoming signal. In practice, due to the receiver clock variation and jitter, the sampling point changes around the middle of the incoming signal, as Figure 4 shows highlighted in gray.

Equation (4) still applies for a single message (Figure 3 b). The proposed approach determines that the transmitter needs to add n clock cycles to T_M in order to minimize the time skew or error. Assuming $\tau_1 \approx \tau_2$, we have

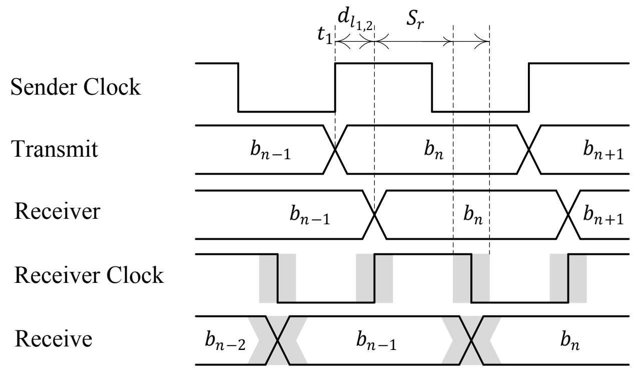


Fig. 4. Clocks and signals in sender and receiver nodes

$$T_S(t_2) \approx T_M(t_1) + n\tau_1, \quad n \in \mathbb{N} \tag{9}$$

The integer value n calculated from Equations (4) and (9) is

$$\begin{aligned} n &= \left\lceil \frac{(s+mc)\tau_1 + d_\ell + r\tau_2 + S_r(t)}{\tau_1} \right\rceil \\ &\approx (s + mc + r) + \left\lceil \frac{d_\ell + S_r(t)}{\tau_1} \right\rceil . \end{aligned} \tag{10}$$

By adding n to the time included in the packet, the receiver sets its own time to be approximately equal to the time of the reference node:

$$T_S(t_2) \approx T_M(t_2). \tag{11}$$

Since t_1 and t_2 are arbitrary times, this approach ensures that Equation (11) is always valid.

Using Equations (4) and (9), the instantaneous time skew between transmitter and receiver is found to be:

$$\begin{aligned} &C_S(t) = T_M(t) - T_S(t) \\ &= (s+mc)\tau_1 + d_\ell + r\tau_2 + S_r(t) - n\tau_1 \\ &= r(\tau_2 - \tau_1) + d_\ell + S_r(t) - \tau_1 \left\lceil \frac{d_\ell + S_r(t)}{\tau_1} \right\rceil . \end{aligned} \tag{12}$$

In comparison with other terms, the value of $(\tau_2 - \tau_1)$ is very small (e.g. 30 ps at 20 MHz with a crystal quartz oscillator) and may be ignored. Therefore, time skew mainly depends on propagation time and sampling range $S_r(t)$.

This previous analysis applies to all nodes that are one hop away from the reference node but it should be noted that regardless of the master clock variation or position, the slave node always follow the master clock. So, the above calculation can be applied for all the nodes with one hop distance from each other. In this way, the time skew accumulates as the hop count increases. In other words, for a node which is h hops away from the time reference node, time skew under the proposed approach is given by:

$$C_S^h(t) = h\left(d_\ell + S_r(t) - \tau\left[\frac{d_\ell + S_r(t)}{\tau}\right]\right). \tag{13}$$

where the upper index of C_S^h refers to the number of the hops.

The variation of the time skew can be estimated from Equation (13) as:

$$\frac{\mathrm{d}C_S^h}{\mathrm{d}t} = h\left(\frac{\mathrm{d}d_\ell}{\mathrm{d}t} + \frac{\mathrm{d}S_r}{\mathrm{d}t}\right). \tag{14}$$

Signal propagation is almost constant in a wired wearable network, so the previous expression simplifies to:

$$\frac{\mathrm{d}C_S^h}{\mathrm{d}t} \approx h\frac{\mathrm{d}S_r}{\mathrm{d}t}. \tag{15}$$

Consequently, the time deviation of time skew is a linear function of $S_r(t)$. So, it is expected that the any variation on the recovered clock at the receiver materializes linearly on the synchronized time. Here no restriction is made on $S_r(t)$ (besides being limited in range as discussed below), meaning that in general any arbitrary synchronization method can be used under considered assumptions.

B. Average Time Skew

The value of n given by Equation (10) depends mostly on constant parameters, but varies with the instantaneous value of $S_r(t)$. The use of a fixed value of n implies that $S_r(t)$ has to be constrained to a certain range. For baseband communications the optimal average value of $S_r(t)$ is the middle of the incoming data bit. This condition can be used to calculate a fixed value of n for use in the proposed approach. From Equation (5) we have

$$\langle S_r(t)\rangle = \frac{\tau_d}{2} = \frac{m}{2}\tau, \tag{16}$$

where $\langle S_r\rangle$ is the average of $S_r(t)$. Using $\langle S_r\rangle$ in Equation (10) results in

$$n = (s + mc + r) + \left[\frac{d_\ell + \frac{m}{2}\tau}{\tau}\right]. \tag{17}$$

Since $d_\ell \ll \tau$ in wired wearable sensor networks Equation (17) simplifies to

$$n = (s + mc + r) + \left\lceil\frac{m}{2}\right\rceil \tag{18}$$

and the average skew is

$$\langle C_S^h\rangle = h\left(d_\ell + \frac{m}{2}\tau - \left\lceil\frac{m}{2}\right\rceil\tau\right). \tag{19}$$

This equation gives different results for even and odd values of m:

$$\langle C_S^h\rangle = \begin{cases} h\left(d_\ell + \frac{1}{2}\tau\right), & m \text{ odd} \\ hd_\ell, & m \text{ even}\end{cases} \tag{20}$$

For even m, $\langle C_S^h\rangle$ grows as the number of hops and only depends on the signal propagation delay between the nodes; when m is odd, $\langle C_S^h\rangle$ also depends on the system clock period.

The behavior of $\langle C_S^h\rangle$ for odd values of m can be controlled by a modification of the adding strategy. For such a system, the average time skew at nodes one hop away from the reference node is:

$$\langle C_S^1\rangle = d_\ell + \frac{1}{2}\tau. \tag{21}$$

Now for a second hop, add the value $n-1$ instead of n to the reference time in the message. The average time skew for nodes that are two hops away becomes

$$\langle C_S^2\rangle = d_\ell + \frac{1}{2}\tau + d_\ell - \frac{1}{2}\tau = 2d_\ell. \tag{22}$$

As a consequence, the average time skew decreases and the value is the same as that for even m. By continuing this procedure of adding n for nodes with odd hop counts and $n-1$ for nodes with even hop counts, $\langle C_S^h\rangle$ for nodes h hops away from the the reference results in:

$$\langle C_S^h\rangle = d_\ell\sum_{i=1}^{h} i + \sum_{i=1}^{h}\frac{1}{2}\tau(-1)^{1+i}, \quad m \text{ odd}. \tag{23}$$

In general, for any $m \in \mathbb{N}^+$, Equations (20) and (23) give:

$$\langle C_S^h\rangle = d_\ell\sum_{i=1}^{h} i + (m \bmod 2)\sum_{i=1}^{h}\frac{1}{2}\tau\,(-1)^{1+i}. \tag{24}$$

It can be conclude that by alternating the additive value, the average time skew for networks with odd m decreases significantly. It should be noted that in any case and independently of m, the instant time skew of Equation (15) is valid. Figure 5 depicts the $\langle C_S^h\rangle$ values calculated from Equations (20) and (23) assuming that $d_\ell = 0.1\,\tau$. Using the adjustment included in Equation (23), the value of $\langle C_S^h\rangle$ for odd m is near to the values for even m and is much smaller than the unadjusted value obtained from Equation (20).

C. Impact of Clock Drift and Update Interval

The validity of the calculated bounds for the clock time skew between nodes requires a process of periodically updating and synchronizing of the nodes, since the accumulation of skew produces a clock offset that needs to be accounted for. For that, the appropriate updating moment has to be

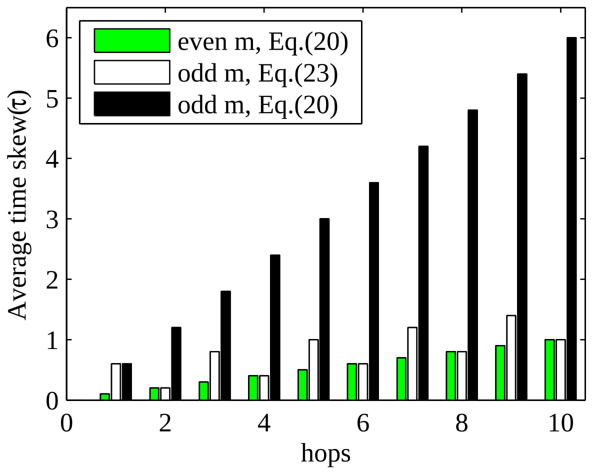


Fig. 5. Average skew $\langle C_S^h \rangle$ as a function of hop count h.

determined, as the time between updates depends on the expected clock drift.

Immediately after a successful synchronization, Equations (15) and (24) are valid. To estimate the effects of clock drift, suppose that the local system clocks have a normally distributed frequency with mean value $f_c = \frac{1}{\tau}$, and that nodes 1 and 2 initially have the same clock offset $T(0)$. The time difference due to clock drift after k clock cycles is

$$\Delta t_{2,1} = k\left(\tau_2 - \tau_1\right), \tag{25}$$

where $\tau_1 = 1/f_1$ and $\tau_2 = 1/f_2$ are the system clock periods of nodes 1 and 2, respectively. Adjusting Equation (24) to account for $\Delta t_{2,1}$ results in

$$\begin{aligned} \langle C_S^h \rangle = d_\ell \sum_{i=1}^{h} i + (m \bmod 2) \sum_{i=1}^{h} \tfrac{1}{2}\tau\left(-1\right)^{1+i} \\ + \langle \sum_{i=1}^{h} \Delta t_{i+1,i} \rangle \end{aligned} \tag{26}$$

Minimizing the effect of Δt requires reducing the value of k. The best case occurs when the *Sync* message arrives at the receiver just before an active edge of *Clk-sync* is generated.

The necessary synchronization accuracy depends on the application. For a signal sampling rate of 1 kHz, a synchronization accuracy around 100 µs (10 % of the sampling rate) may be chosen. In many cases, it will not be necessary to update the time in each period of $Clk-sync$. In general, the update interval can be determined by considering the required accuracy, Equation (26) and the clock drift of the oscillators.

D. Probability of Synchronization

The entire process of synchronization is based on the successful reception of the timing messages. Consider the network of Figure 1, where SN2 receives its synchronization messages from SN1. The probability of having both nodes synchronized is

$$p_{2,1} = P(t_2 = t_1) = p_{request} \times p_{sync}, \tag{27}$$

where $p_{2,1}$ is the probability of successful message exchange, which is the product of the probabilities for successful delivery of *Request* and *Sync* messages. The probability of message delivery depends on network traffic, topology and node or link failures.

For successful synchronization of a given node, all previous nodes in the path to the reference node must be synchronized. Equation (27) can then be extended to a node located h hops away from the reference node:

$$p_{h,1} = \prod_{i=2}^{h} p_{i,i-1} \tag{28}$$

If the probabilities $p_{request} = p_{sync} = p$ are equal for all links, Equation (28) simplifies to:

$$p_{h,1} = p^{2(h-1)}. \tag{29}$$

Now consider the situation when PTP is used. This protocol uses 3 packets to complete the synchronization. So, Equation (30) for SN2 will be:

$$p_{2,1} = p_{sync} \times p_{request} \times p_{response}, \tag{30}$$

where $p_{response}$ denotes the probability of a successful $Response$ packet message transmission. Assuming that all messages have the same probability p of being successfully transmitted, Equation (29) can be rewritten for PTP as

$$p_{h,1} = p^{3(h-1)}. \tag{31}$$

A comparison of Equations (29) and (31) shows that, for the application in wearable networks, PTP is more prone to failure in the synchronization process than the protocol proposed in this work, because PTP requires more messages to be transmitted.

VI. Experimental Results

This protocol has been implemented as part of a communication prototype made as an integrated circuit (IC) shown in Figure 6. The area labeled TS in the figure is the synchronization circuit. The circuit has been fabricated in a 0.35 µm CMOS process; more details about its architecture and performance can be found in [5]. The ASIC was designed

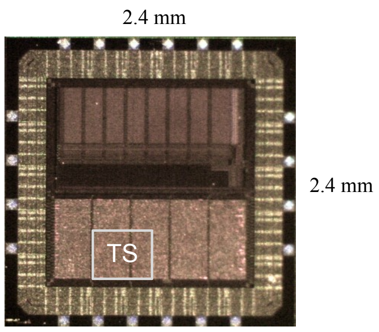


Fig. 6. Microphotograph of the CMOS communication ASIC showing the synchronization circuit (TS).

for a wearable sensor network that is dedicated to acquiring data from inertial sensors and electromyography electrodes [4].

All experimental results have been obtained with the prototype IC operating at 20 MHz, with a data rate of 10 MHz ($m = 2$). The measurements were performed on a linear arrangement of SNs with eight nodes (BS, SN2, SN3,... SN8) as shown in Figure 7. This arrangement represents the worst-case situation for time synchronization. The value used for compensating the constant delay was experimentally determined to be $n = 002\text{F}_{16}$. Clock signals have been observed and measured with a digital oscilloscope.

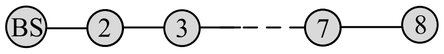


Fig. 7. The linear arrangement of 8 SNs used for experimental evaluation.

A. One-hop clock skew

Figure 8 shows the one-hop clock skew between SN2 and BS as clock reference. The oscilloscope images of the rising edge of clock *Clk-sync* were generated using the infinite time persist display mode.

According to Equation (15), clock skew is proportional to the variation of the recovered clock signal. For the IC implementation used here, the maximum value of $\mathrm{d}S_r/\mathrm{d}t$ is one system clock cycle ($\pm$ 25 ns), a fact confirmed by the measured total clock skew variation range (clock jitter) of 50 ns. The measured average one-hop clock skew is 4.6 ns, which is the signal propagation delay between BS and SN2.

B. Multi-hop clock skew

According to Equation (24), clock skew increases linearly with the number of hops. Figure 9 compares the measured

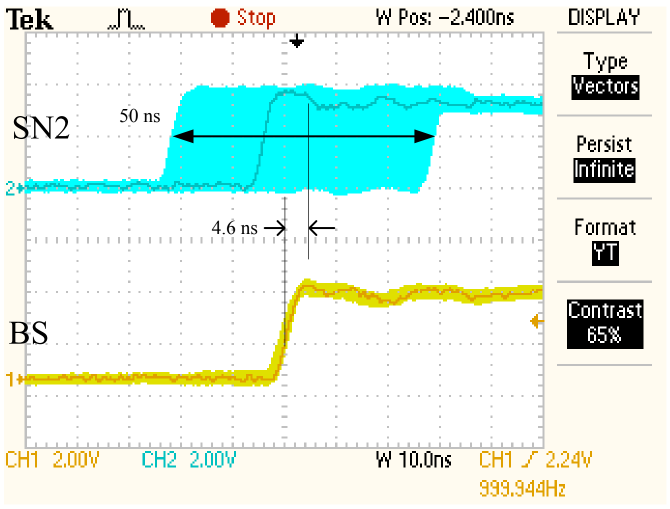


Fig. 8. Measured one-hop clock skew ($n = 002\text{F}_{16}$) between BS and SN2.

skew at each node in the sensor network with the calculated values. As expected, the average clock skew increases by almost 4.6 ns per hop. The measured clock skew variation range also increases in agreement with Equation (24): 50 ns at node SN2, 100 ns at node SN3, and reaches 350 ns at SN8.

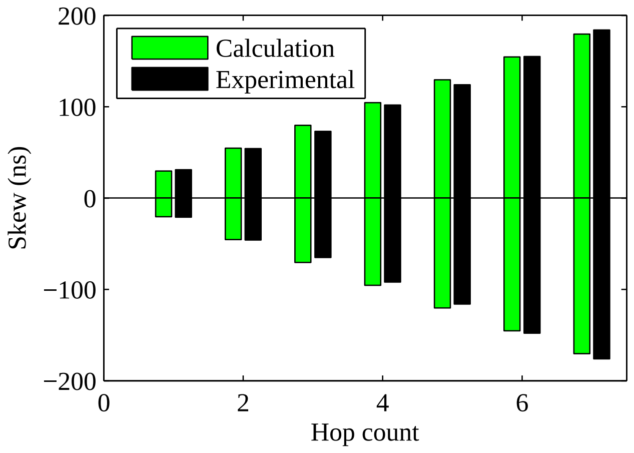


Fig. 9. Measured and calculated multi-hop clock skew.

Figure 10 shows the measured node-to-node and global node-to-BS average clock skew. Since $m = 2$, the average clock skew depends on the propagation delay between the nodes (4.6 ns) and accumulates as the number of hop increases, reaching 30 ns at SN8. As expected, the average clock skew is much lower than the total clock skew, which lies in the range of $(-176\,\text{ns}, 184\,\text{ns})$, as shown in Figure 9. As assumed in Section V, the delay due to signal propagation is much smaller than the clock signal variation.

In wearable systems, the sampling frequency is usually in the range of a few kHz. Therefore, it can be safely said that,

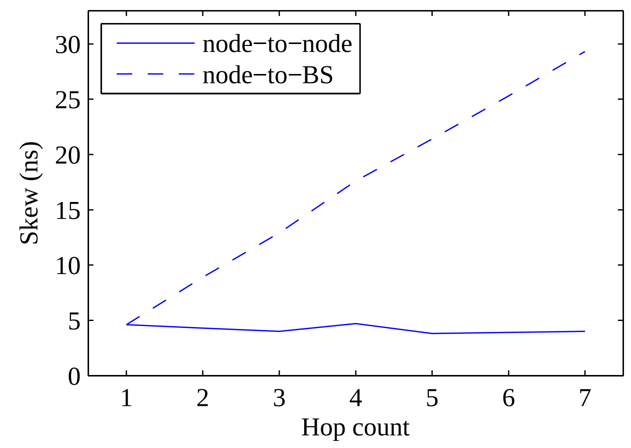


Fig. 10. Average clock skew.

according to the measured values of clock skew, the proposed method is applicable to most wired wearable sensor networks.

C. Effects of Timing Message Interval and Failure on Synchronization

In many applications, message failure is inevitable and it may happen for several reasons such as collisions and high traffic load. To evaluate the effect of message failure on synchronization, the clock skew between BS and SN2 has been measured as a function of the number of consecutive missing messages. The results are shown in Figure 11 for two different *Clk-sync* frequencies. In both cases, the timing update messages are sent just before the rising edge of *Clk-sync*. Therefore, the interval between updates used for the 2 kHz clock signal is half the interval used for the 1 kHz case. Therefore, the clock skew for the latter case is larger than the skew for the 2 kHz case.

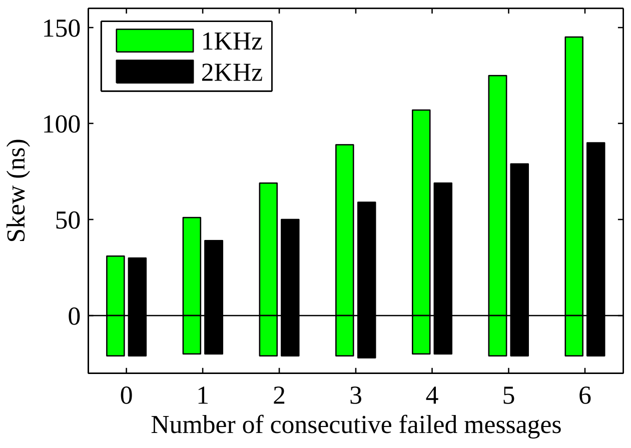


Fig. 11. One-hop skew after consecutive timing message failures.

According to Equation (25), the linear growth of clock skew depends on the clock frequency difference between the nodes. In the absence of failures, the synchronization accuracy is in the range [−21 ns; 29 ns], as expected. The observed increase of the clock skew in the presence failures means that, in this case, $\Delta t > 0 \Rightarrow \tau_S > \tau_M$. This means that the upper (positive) bound of the skew grows with the number of missing messages, while the lower (negative) bound stays the same. For situations where $\Delta t < 0$, it is the lower bound that becomes increasingly more negative. Regardless of the clock skew, a successful synchronization message puts the skew of the receiver back in the range [−21 ns; 29 ns].

If a connection line is broken between two nodes, we expect the skew of the disconnected segment to grow without limit, but the synchronization between nodes in each segment to be maintained. To confirm this behavior, nodes SN2 and SN3 were disconnected at $t = 37$ s in order to obtain the results for average clock skew shown in Figure 12. Until the disconnection occurred all nodes stayed synchronized. At $t = 37$ s SN3 starts running free and its skew starts to increase as expected. The nodes on the disconnected segment get their time reference (directly or indirectly) from SN3 and stay synchronized with it. The clock skew after the disconnection depends on the clock difference between BS and SN3. In the present case, the positive slope of the skew curve is due to the fact that the clock of SN3 has a somewhat higher clock frequency that the clock of BS. If the clock of the disconnected node shows a smaller frequency, the slope of the skew curve will become negative.

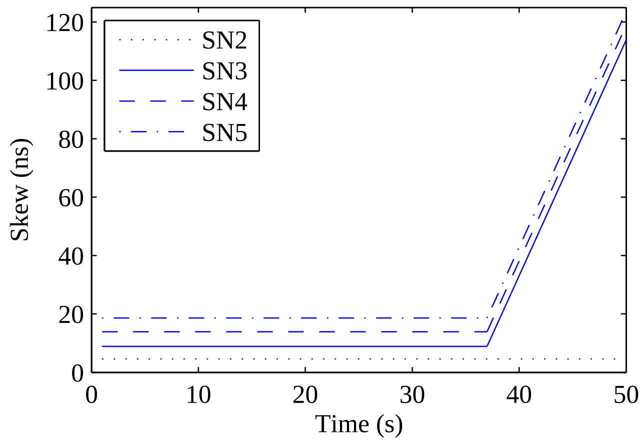


Fig. 12. Average clock skew after disconnection of the line between SN2 and SN3.

The periodic exchange of timing messages generates additional traffic. Measurements show that with a 1 ms interval between synchronization messages, 0.52 % of the channel bandwidth is used for synchronization, which is almost negligible. This allows the system to achieve high precision with an average clock skew below 5 ns. Some Body Area Networks (BAN) applications may not need such a high precision. If precisions in the range of a few microseconds is enough, the time interval between messages can be made larger (in the range of a second), consuming even less energy and channel bandwidth.

Figure 13 shows the measured skew between BS and SN as a function of time for two update intervals: 1.5 s and 5 s. The skew depends on the clock frequency difference between the nodes, which has been measured to be 3.7 ppm. Since the clock frequency of each node stays constant, the accumulation of the small difference grows linearly, reaching 18.5 µs (for a 5 s interval) and 5.58 µs (for a 1.5 s interval). In both cases, the absence of synchronization message for a long time results in significantly higher skew. Nevertheless, a message synchronizes *Clk-sync* immediately.

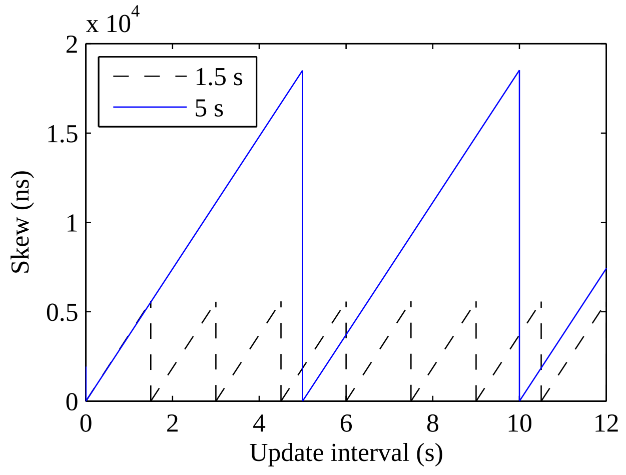


Fig. 13. Clock skew for update interval of 1.5 s and 5 s, *Clk-sync* = 1 kHz.

Overall, the experimental results confirm the synchronization behavior of the protocol as explained and analyzed in the previous sections.

VII. Conclusion

The time synchronization protocol described in this paper is intended for hardware implemented at the MAC layer. It is based on one-way master-to-slave message exchange. For wearable sensors, in which sensors are connected to each other with conductive yarns, the transmission delay is kept in a fixed range by using dedicated transmitters and receivers together with the proposed protocol. This leads to high precision time synchronization by minimizing the clock skew. Theoretical analysis and experimental results indicate that in the proposed protocol, the average of one-hop time skew is equal to the signal propagation from sender to receiver, which is in the range of a few nano seconds for the case of wearable systems, and is the minimum value possible when synchronization is performed at the hardware level. In a multi-hop network, global average-time skew grows linearly with hop count. As the experimental results indicate, the average global skew after 8 hops is less than 30 ns. The value of 8 hops in the clock tree is bigger than the number of almost all existing wearable systems. This means that the aforementioned method is scalable for wearable networks.

In the context of wearable systems, the proposed protocol achieves better performance than PTP, better energy efficiency and less complexity, while keeping a level of precision in synchronization that is sufficient to satisfy many body-area networks applications. The main characteristics responsible for such improvements are reduced message communication and fewer calculations required for synchronization.

References

[1] M. Buevich, N. Rajagopal, and A. Rowe. Hardware assisted clock synchronization for real-time sensor networks. In *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, pages 268–277, Dec 2013. 3

[2] T. Cooklev, J. Eidson, and A. Pakdaman. An implementation of IEEE 1588 over IEEE 802.11b for synchronization of wireless local area network nodes. *IEEE Transactions on Instrumentation and Measurement*, 56(5):1632–1639, Oct 2007. 3

[3] F. Derogarian, J. C. Ferreira, and V. M. G. Tavares. A routing protocol for WSN based on the implemention of source routing for minumum cost forwarding method. In *Proc. 5th Intl. Conf. on Sensor Tech. Appl. (SENSORCOMM 2011)*, pages 85–90, Aug. 2010. 4

[4] F. Derogarian, J. C. Ferreira, and V. M. G. Tavars. Design and implementation of hybrid circuit/packet switching for wearable systems. In *23rd Intl. Symposium on Industrial Electronics, ISIE2014*, June 2014. 10

[5] F. Derogarian, J. C. Ferreira, and V. M. G. Tavars. A time synchronization circuit with an average 4.6 ns one-hop skew for wired wearable networks. In *17th Euromicro Conf. on Digital Systems Design, DSD2014*, Aug. 2014. 2, 9

[6] C. D. Dominicis, P. Pivato, P. Ferrari, D. Macii, E. Sisinni, and A. Flammini. Timestamping of IEEE 802.15.4a CSS signals for wireless ranging and time synchronization. *IEEE Transactions on Instrumentation and Measurement*, 62(8):2286–2296, Aug 2013. 1

[7] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proc. 5th symposium on Operating systems design and implementation, SIGOPS*, pages 147–163, 2002. 2, 3

[8] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proc. 1st Intl. Conf. on Embedded Networked Sensor Systems*, SenSys ’03, pages 138–149, 2003. 2, 3

[9] L. Gatzoulis and I. Iakovidis. Wearable and portable ehealth systems. *IEEE Engineering in Medicine and Biology Magazine*, 26(5):51–56, 2007. 3

[10] IEEE 1588-2008 standard for a precision clock synchronization protocol for networked measurement and control systems, 2008. 3

[11] IEEE. *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. IEEE, July 2008. 6

[12] S. Lasassmeh and J. Conrad. Time synchronization in wireless sensor networks: A survey. In *Proc. IEEE SoutheastCon 2010 (SoutheastCon)*, pages 242–245, 2010. 1

[13] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester. A survey on wireless body area networks. *J. Wireless Networks*, 17(1):1–18, 2011. 4

[14] K. Lee and J. Eidson. IEEE-1588 standard for a precision clock synchronization protocol for networked measurement and control systems. In *34th Annual Precise Time and Time Interval(PTTI) Meeting*, pages 98–105, 2002. 2, 3, 6

[15] C. Lenzen, P. Sommer, and R. Wattenhofer. Optimal clock synchronization in networks. In *Proc. 7th ACM Conf. on Embedded Networked Sensor Systems*, SenSys ’09, pages 225–238, 2009. 2, 3

[16] A. Lymberis and A. Dittmar. Advanced wearable health systems and applications - research and development efforts in the european union. *IEEE Magazine, Engineering in Medicine and Biology*, 26(3):29–33, 2007. 3

[17] M. Maggs, S. O’Keefe, and D. Thiel. Consensus clock synchronization for wireless sensor networks. *IEEE Journal of Sensors*, 12(6):2269–2277, June 2012. 1, 2

[18] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proc. 2nd Intl. Conf. on Embedded Networked Sensor Systems*, pages 39–49, 2004. 3

[19] D. Mills. Internet time synchronization: the network time protocol. *IEEE J. Communications*, 39(10):1482–1493, 1991. 2

[20] P. Ranganathan and K. Nygard. Time synchronization in wireless sensor networks: A survey. *Intl. J. UbiComp*, 1(2):92–102, 2010. 1

[21] I.-K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y.-C. Wu. Clock synchronization in wireless sensor networks: An overview. *J. Sensors*, 9(1):56–85, 2009. 1

[22] A. Rowe, V. Gupta, and R. R. Rajkumar. Low-power clock synchronization using electromagnetic energy radiating from ac power lines. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys ’09, pages 211–224, New York, NY, USA, 2009. ACM. 3

[23] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE J. Network*, 18(4):45–50, 2004. 1

[24] R. Subrahmanyan. Timing recovery for IEEE 1588 applications in telecommunications. *IEEE Transactions on Instrumentation and Measurement*, 58(6):1858–1868, June 2009. 3

[25] B. Sundararaman, U. Buy, and A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: A survey. *Elsevier J. Ad Hoc Networks*, 3:281–323, 2005. 1, 2

[26] J. Wu, L. Zhang, Y. Bai, and Y. Sun. Cluster-based consensus time synchronization for wireless sensor networks. *IEEE Journal of Sensors*, 15(3):1404–1413, March 2015. 2

[27] Y.-C. Wu, Q. Chaudhari, and E. Serpedin. Clock synchronization of wireless sensor networks. *IEEE Signal Processing Magazine*, 28(1):124–138, 2011. 1

[28] A. Zambrano, F. Derogarian, R. Dias, M. Abreu, A. Catarino, A. Rocha, J. da Silva, J. Ferreira, V. Tavares, and M. Correia. A wearable sensor network for human locomotion data capture. In *9th Intl. Conf. on Wearable micro and nano technologies for personalized health; pHealth2012*, pages 216–223, 2012. 3