Synchronizing Tiny Sensors with SISP: a Convergence Study

Oana Andreea Hotescu, Katia Jaffrès-Runser, Adrien van den Bossche, Thierry Val

Synchronizing Tiny Sensors with SISP: A Convergence Study

Oana Hotescu, Katia Jaffrès-Runser
Institut de Recherche en Informatique de Toulouse
Université de Toulouse, INPT
2 rue Charles Camichel
Toulouse, France 31300
{oana.hotescu,kjr}@enseeiht.fr

Adrien Van Den Bossche, Thierry Val
Institut de Recherche en Informatique de Toulouse
Université de Toulouse, UT2J
BP60073
Blagnac, France 31703
{adrien.van-den-bossche,thierry.val}@irit.fr

ABSTRACT

The SImple Synchronization Protocol (SISP) has been designed for tiny sensors to offer a wireless synchronization service to the network. SISP is completely distributed with a flat architecture. Nodes broadcast a SYNC message periodically that contains the value of their view of a shared clock counter. Every time a SYNC message is received, nodes update their shared clock by averaging it with the clock value embedded in the message. This protocol converges in practice very well, and requires a small amount of energy as SYNC messages can be sent every second only. Moreover, computations are basic, perfectly fitting the tiny sensor platforms needed for the Internet of Things. Its distributed operations enable the network to adjust seamlessly to the appearance or disappearance of other nodes. This paper concentrates on the convergence analysis of this promising protocol. Convergence time and synchronization accuracy are determined analytically, by simulations and by experimenting a real sensor platform. All results show that this protocol offers an accuracy in the order of a few tens of microseconds. Moreover, our analytical derivations capture very well an upper bound on the synchronization accuracy.

CCS CONCEPTS

• **Networks** → **Protocol testing and verification**; **Sensor networks**; Network performance analysis;

KEYWORDS

Wireless Sensor Network; Synchronization; Convergence analysis

1 INTRODUCTION

The Internet of Things (IoT) is foreseen to be central to future communication technologies. In the next decade, not only humans will exchange information, but machines and objects of different sorts as well. Objects of all kinds, connected wirelessly using various technologies (WiFi, 5G, Ultra Wide Band, LoRa, etc.) will carry data over short or long distances. Different services are envisioned where this data plays a very important role, which may strongly impact the performance of a much larger cyber-physical system. For instance, industry will evolve to a new generation of factories where all processes and humans are monitored to measure the production efficiency, but as well to extract information and metrics that can be leveraged to further improve operations in real-time. Such data can as well warrant the quality of the products by completely tracing the history of all goods and processes.

In such a context, objects (sensors, actuators, robots, smartphones, ...) will coexist in the factory and mostly communicate wirelessly. The density of these systems calls for advanced communication techniques that seamlessly scale. Objects enter or leave the system in an autonomous fashion in this case. As such, network operations have to be designed in a completely distributed manner.

For these objects to efficiently communicate in such a dense context, wireless channel has to be divided among communicating nodes using time or frequency division medium access (or both). Current industry-oriented wireless solutions such as WirelessHart [1] or the TSCH mode of IEEE803.15.4e [2] define an FTDMA medium access layer. This layer offers 10 millisecond time slots to emit a 127 byte frame at 250 kbits/s and receive its acknowledgement. It has been highlighted in [3] that this slot duration is too large and detrimental to overall communication performance. Reducing this duration is only possible if a scalable synchronization protocol can be leveraged to finely synchronize these resource-limited objects.

Synchronizing nodes over a network is not a new problem and lots of solutions exist in the literature. The closest works to this study are related to the synchronization of wireless sensor networks. Representative protocols are RBS [4], TPSN [5], FTSP [6], PulseSync [7], SHARP [8] and SISP [9]. Our aim is to leverage a protocol that provides a seamless synchronization over tiny low-complexity platforms. RBS and TPSN have been created to synchronize a relatively small set of nodes and thus they don't scale well. FTSP and PulseSync, on the contrary, have been designed to flood beacons to rapidly transfer time information to remote sensors. Both algorithms achieve a synchronization accuracy of a few tens of microseconds. FTSP offers a maximum accuracy of 80μs, and PulseSync a maximum accuracy of 38μs for a line topology of 20 nodes. Both approaches synchronize the network to the time of a selected root node. If this root node dies, another node has to be elected and its time spread in the network, creating a possible cut of synchronization service.

This paper focuses on a different solution that floods beacons as well in the network. These beacons are not forwarded in a multi-hop fashion, but are beamed and exploited by the nodes to agree on a common notion of time called the shared clock. Since this shared clock results from the common decision of all nodes, the departure of one node has little impact on the shared clock value. There is no

https://doi.org/10.1145/3127540.3127564

central entity. This solution is called SISP which stands for *SImple Synchronization Protocol* [9].

SISP is completely distributed with a flat architecture. Nodes broadcast a SYNC message periodically that contains the value of their view of a shared clock counter. Every time a SYNC message is received, nodes update their shared clock by averaging it with the clock value embedded in the message. This protocol converges in practice very well, and requires a small amount of energy as SYNC messages can be sent every second only. Moreover, computations are basic, perfectly fitting the tiny sensor platforms needed for the Internet of Things. Its distributed agreement on a common clock enables the network to adjust seamlessly to the appearance or disappearance of other nodes.

SISP has been evaluated experimentally in the past [9], [8]. It has been shown in [8] that it clearly outperforms RBS on an Arduino Fio embedded platform running a 8MHz micro-controller. RBS reaches only an accuracy of 1509 μs in average while SISP offers a 112μs accuracy for a 2-node topology (beacons are emitted every 500ms in both protocols). So far, there is no thorough quantification of the SISP convergence properties. The goal of this paper is to fill this gap and exhibit the short convergence of this promising protocol to a stable synchronization with low error. Convergence time and synchronization accuracy are determined analytically, by simulations and though experimentations. All results show that this protocol offers a maximum accuracy in the order of a few tens of microseconds on the investigated hardware. Moreover, our analytical derivations capture very well an upper bound on the synchronization accuracy.

The paper is organized as follows. Section 2 introduces SISP and the model we will leverage for its analysis. The following three sections analyze its convergence properties using three different means: simulations in Section 3, theoretic analysis in Section 4 and experiments in Section 5. Finally, Section 6 concludes this work and draws the main lines of future works.

2 PROTOCOL AND MODEL

2.1 SISP protocol

SISP is a SImple Synchronization Protocol designed for lightweight wireless sensors [9]. SISP is completely distributed, and there is no hierarchy among nodes. Through the exchange of periodic SYNC messages, all nodes adopt a global time reference with a precision of a few tens of microseconds. This global time reference is not the absolute universal time but a time that all nodes agree to follow together. Algorithm 1 presents the actions performed by the `sisp()` function that is called periodically by each node of the network.

SISP defines two integer counters per node: the local clock LCLK and the shared clock SCLK. At onset, both counters are set to zero. Both clocks are incremented by one unit every time the `sisp()` procedure is called. Every P procedure call, a node broadcasts a SYNC message that conveys the SCKL value the sender node measures at emission time. For all other calls, she listens to SYNC messages sent by neighbours. If a SYNC message is received, she reads the embedded clock value, RCLK, and updates her shared clock following:

$$SCLK = \left\lfloor \frac{RCLK + SCLK}{2} \right\rfloor \quad (1)$$

This operation averages both clock values and rounds up the result to the closest inferior integer with $\lfloor\ \rfloor$ operators. SISP is a perfect fit for tiny microprocessors as it requires only one addition plus one division by 2 (i.e. 1 bit right shift).

SYNC frames are not acknowledged, only nodes receiving the frame update their SCLK counter. SISP can be implemented over any type of medium access protocol that offers broadcast communication service. However, its accuracy and convergence duration will be impacted by the number of SYNC messages that collide. Future works will study the impact of SYNC message losses on SISP accuracy and convergence duration. In the following, all derivations, simulations and experiments are made for the ideal case where no SYNC frame collision occurs.

SISP is illustrated in Figure 2 with a sample execution carried out over a 2-node topology. Node $N0$ begins her emissions before node $N1$ by sending SYNC frames every 100 `sisp()` procedure calls. The first three SYNC emissions have no effect on the network since there is no active node. Once $N1$ is turned on, she initialises her SCLK counter to 0. After 20 calls, $N1$ receives the SYNC frame of $N0$ and offsets its SCLK according to Eq. (1). When the LCLK counter of $N1$ reaches 100, $N1$ sends a new SYNC message that causes a change in the SCLK value of $N0$. The execution continues this way, which leads to the gradual convergence of the shared clocks of both nodes after a couple of SYNC broadcasts. The remainder of this paper concentrates on the convergence study of this protocol for simple topologies.

2.2 SISP model

In this paper, we adopt the following model to capture the core elements of SISP for a network composed of N nodes. Each node Ni, $i \in \{0, .., N\}$, has a local hardware clock denoted $LCLK_{Ni}$ and a view of the shared clock denoted $SCLK_{Ni}$. Throughout this study we adopt the notations of Table 1.

The reference clock for all derivations is chosen as the one of node $N0$, naming $LCLK_{N0}$. The local clocks of all other nodes are modeled as linear functions of this reference clock:

$$LCLK_{Ni} = \alpha_{Ni} * LCLK_{N0} + \beta_{Ni} \quad (2)$$

where α_{Ni} and β_{Ni} are respectively the drift and offset of the local clock of Ni with respect to the one of $N0$. Offset and drift can take

```
sisp(){
    lclk = lclk + 1
    sclk = sclk + 1
    if(lclk mod P != 0){
        if msg_received(rclk){
            sclk = (sclk + rclk) / 2
            //if rclk=sclk then nodes share the same SCLK
        }
        else
        {
            broadcast(sclk)
        }
    }
}
```

Figure 1: SISP procedure: LCLK, SCLK and P are global integer variables.

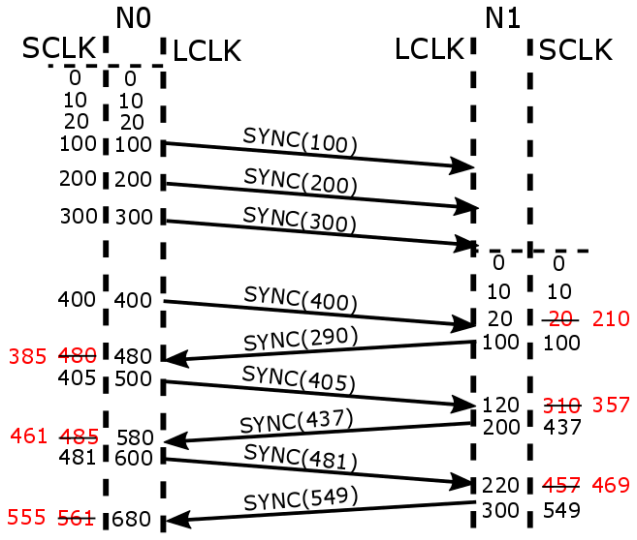

Figure 2: SISP sequence diagram

Table 1: Table of notations

Notation	Definition
α_{Ni}	Node N_i clock drift
$LCLK_{Ni}$	Node N_i local clock
$SCLK_{Ni}$	Node N_i shared clock
t_k	A discrete clock that moves one step forward every time a SYNC message is emitted by a node of the network. Integer k counts the number of SYNC messages emitted since the beginning of the synchronization.
D_k^{Ni}	Duration in seconds that separates t_k and t_{k-1} on node N_i
$d_{NiNj}(t_k)$	Difference between the shared clocks of node N_i and node N_j at t_k
P	SYNC message period in clock tops
a	Synchronization accuracy
t_a	Convergence duration

positive or negative values. A positive (resp. negative) drift implies that Ni runs faster (resp. slower) than $N0$. If α_{Ni} equals one, both clocks run at the same speed. In this case, synchronization is simpler as only offset has to be compensated for by the protocol. Of course, this model is a simplification of reality: drift can change over time because of the variations of the oscillators frequencies of $N0$ and $N1$ with heat, load, etc. The impact of real variations on SISP synchronization will be observed in our experimental study. The linear model will be leveraged in the simulation and analytic studies.

2.3 SISP convergence

This paper aims at measuring and calculating a bound on the convergence time and synchronization accuracy of SISP.

Convergence time. In this protocol, the first SYNC messages mostly compensate for the clock offsets of all nodes. The shared clocks of all nodes converge to a value close to the average of their offsets $\beta_{avg} = 1/N \sum_{i=0}^{i=N-1} \beta_{Ni}$. The time needed for the shared clocks to reach this average offset β_{avg} is defined here as the *convergence time* of SISP. It is denoted by t_a and expressed in seconds.

Synchronization accuracy. Once the shared clocks of all nodes have reached this average offset, all subsequent SYNC emissions permit to combat the drifts by resetting the shared clock periodically. The difference between the shared clocks of two nodes depends on their relative drift and the synchronization period P. For a given P value, the larger the drift between two nodes, the faster their shared clocks diverge before a new SYNC resets their values. The accuracy a of SISP for a set of N nodes is defined as the *maximum shared clock deviation* existing in the network once convergence has occurred. Each pair of node may experience a different shared clock difference. As such, the accuracy a can be formulated as the maximum shared clock difference over all pairs of nodes:

$$a = \max_{(i,j)} d_{NiNj} \tag{3}$$

Convergence time and accuracy will be investigated using (i) simulations, (ii) analysis and (iii) measurements. For the analysis, we will look for an upper bound on accuracy.

2.4 Investigated topologies

Three elementary topologies are investigated in this work that are representative of small deployments:

- the *2-node* topology, where two nodes are in direct view.
- the *3-node ring* topology, where each sensor can communicate with the two other ones. They create a clique.
- the *3-node in line* topology, where nodes are aligned but only the central node can communicate with the two other nodes. Border nodes can not communicate with each other.

All studies have been made by accounting for the technical features of a real sensor platform: the DecaWino sensor [10]. It is an open-source hardware design equipped with the transceiver DWM1000 and an Arduino board, the Teensy 3.2 with ARM Cortex M4 32-bit MCU rated at 64GHz, 64kB RAM and 256kB program memory. The hardware clock of this microprocessor drifts of around 20 ppm (parts per millions). It means that after one million seconds, the oscillator exhibits a deviation of +/- 20 seconds. Our measurements have been made with this sensor platform.

Next and unless specified otherwise, a SYNC message is emitted by a node every second. The `sisp()` procedure is called every microsecond. This setting, for a clock drift of 20 ppm, should limit the time deviation to +/- 20 microseconds.

3 SIMULATION STUDY

Convergence and synchronization accuracy have been investigated first by simulation. An in-house simulator has been designed in Java for this purpose following the SISP and network model introduced earlier. Results for the three topologies of interest are given next. For each of them, convergence and accuracy are given when the drift of all nodes equals 0 ppm or 20 ppm. For 0 ppm, we set all nodes to $\alpha_{Ni} = 1$ and for 20 ppm, we set them to $\alpha_{Ni} = 0,100020$.

3.1 2-node topology

Figure 3 plots the difference of the SCLK counters of both nodes over time. If this difference is zero, both nodes have adopted the same shared clock and are thus perfectly synchronized. The x-axis represents the local clock of $N0$, which represents our reference clock.

Figure 3-(top) gives the shared clock difference when there is no clock drift. Before 10 seconds, the difference of shared clocks is decreasing to compensate for the offset difference of both nodes. After 10 seconds, since no drift exists, clocks stay in tune. For the 20ppm case, represented on Figure 3-(bottom), a similar decay of the SCKL difference is observed, but after 9 seconds, the difference oscillates around a value of 10μs. The accuracy is here of 11μs, given by the maximum shared clock difference. This observation is in line with a SYNC period of 1s. and a 20ppm oscillator.

On Figure 4, we investigate the influence of P on the shared clock difference observed at convergence for different values of clock drift. This figure has been obtained for an oscillator frequency

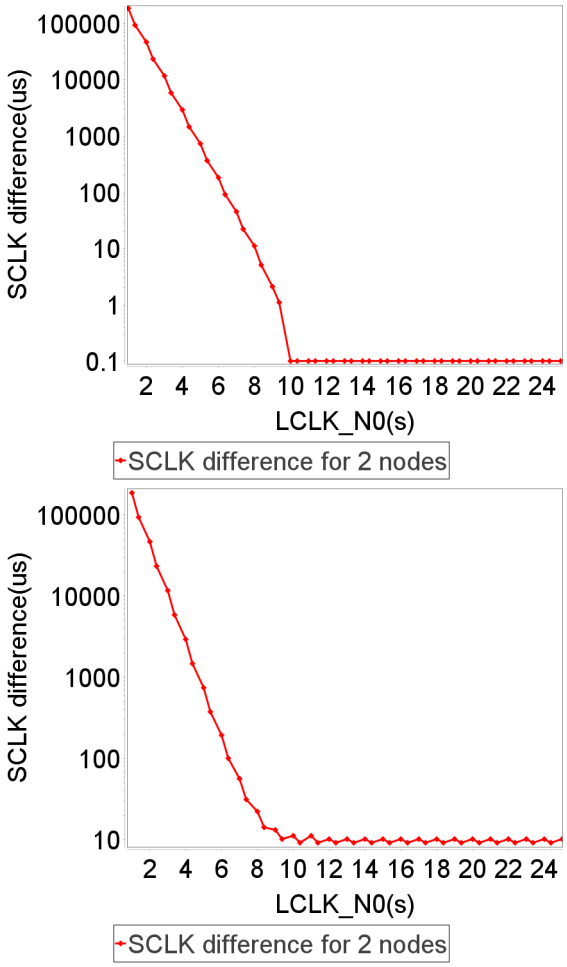

Figure 3: Difference of the shared clocks (in log scale) over time for the 2-node topology, with a 0ppm drift (top) and 20ppm (bottom)

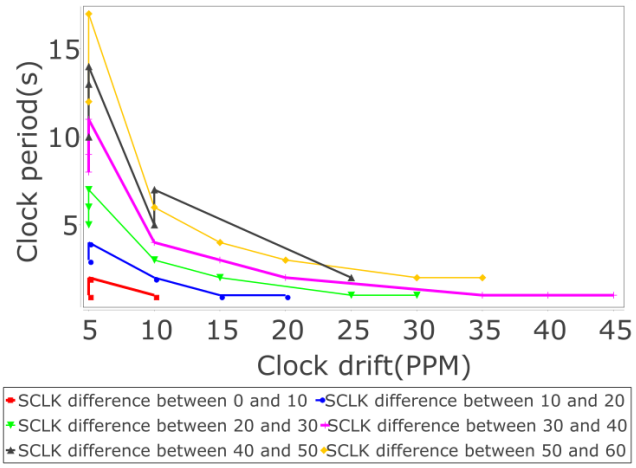

Figure 4: Impact of P on the accuracy for different drifts.

of 1MHz. The x-axis represents the clock drift varying from 0 to 50 ppm while the y-axis represents the period P of SYNC messages in seconds. For instance, for a drift of 20 ppm, the lowest shared clock difference that can be seen here is between 10 and 20μs. This is possible for a SYNC period of 1s. A SYNC period of 3s. offers an accuracy between 30 and 40μs. The general message is that to obtain a better accuracy, shorter SYNC periods have to be chosen.

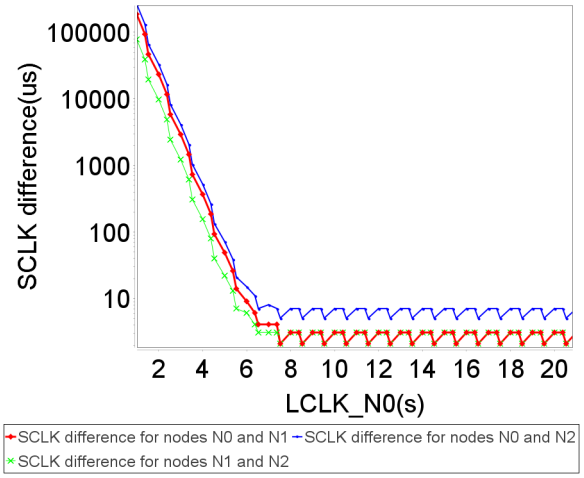

Figure 5: Difference of the shared clocks (in log scale) for a 3-node ring topology. The drift of node $N1$ is 10ppm and the drift of $N2$ is 20ppm.

3.2 3-node topologies

When the drift equals 0ppm for all three nodes, all pairs converge to a difference of shared clocks equal to zero. This result isn't plotted as it will be demonstrated analytically in Section 4. Simulation results for the case where the drifts are non-null are given next.

Figure 5 represents the shared clock difference for a 3-node ring topology, where the local clock of node $N0$ is given as the reference

clock. The drift of $N1$ is set to 10ppm and the one of $N2$ to 20ppm. On Figure 5, the three curves represent the shared clock difference observed for the three pairs of nodes. At convergence, the largest difference is observed between nodes $N0$ and $N2$. The shared clock difference is the same between the other 2 pair of nodes, which is correct as they have both a relative drift of 10ppm. As such, the overall synchronization accuracy is given by the pair of nodes experiencing the largest relative drift.

In Figure 6, the pairwise shared clock differences for the 3-node line topology is given. As for the 3-node ring topology, the local clock of $N0$ is the reference clock and the drifts are equal to 10ppm for $N1$ and 20ppm for $N2$. The three differences are periodic as observed for the ring topology, but the convergence duration is twice as long as for the ring topology. This can be easily understood as direct communications between border nodes are impossible: clock updates need two beacons to be accounted for by all nodes. Again, the pair of nodes that experiences the largest relative drift (naming $N2$ and $N0$) is the one showing the largest shared clock difference.

3.3 Simulation results summary

Table 2 shows for each topology the synchronization accuracy and the convergence time observed by simulations. Not surprisingly, it is the 3-node ring topology that offers the shortest convergence time, as 2 nodes can update their shared clock at each SYNC message emission. The longest convergence is observed for the 3-node line topology, as updates have to be 'relayed' by the central node. For the no-drift cases, perfect synchronization is achieved and accuracy is 0. For the 20 ppm drift, an oscillatory behavior is observed after convergence whose maximum value, given by the accuracy, is bounded. Bounds are in the order of a few tens of microseconds, which is in-line with the setting of a one second SYNC emission. In the following, convergence time and synchronization accuracy are investigated analytically.

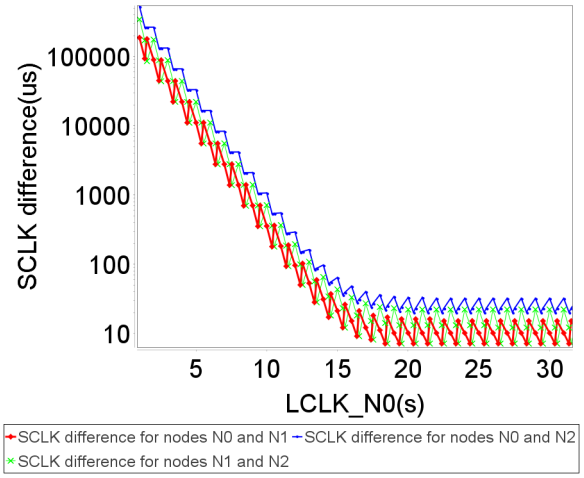

Figure 6: Difference of the shared clocks (in log scale) for a 3-node line topology. The drift of node $N1$ is 10ppm and the drift of $N2$ is 20ppm.

Table 2: Simulation study: synchronization accuracy and convergence time for all topologies.

Topology	Synchronization accuracy (in μs.)	Convergence time (in s.)
2-node ; 0 ppm	0	10
2-node ; 20 ppm	11	9
3-node ring ; 0 ppm	0	8
3-node ring ; 10/20 ppm	7	7.5
3-node line ; 0 ppm	0	21
3-node line ; 10/20 ppm	32	19.5

4 ANALYTIC STUDY

In this section, we conduct an analysis of the convergence of SISP synchronization protocol and we provide a bound on the synchronization accuracy and the minimal number of messages needed to achieve the synchronization state. We begin by describing the methods used to prove the protocol convergence, then we summarize the main results obtained for the considered topologies. A detailed proof for the 2-node topology is given, together with the main proof elements of the 3-node topologies.

4.1 Methodology

Expressions for the convergence time and an upper bound on the synchronization accuracy are derived here for each topology. We study the case where clocks don't drift away (i.e. $\alpha_{Ni} = 1$) and the one where a known drift of $\alpha_{Ni} \neq 1$ is experienced by the nodes. Derivations are made in the latter case for non-homogeneous offsets and drifts as given in the model of Eq.(2). Moreover, offsets and drifts are not a function of time, and thus are assumed to be constant. Propagation delays are neglegted.

Synchronization accuracy is proved by deriving the difference of shared clock values, denoted $d_{NiNj}(t_k)$ and by expressing it as a sequence of real values indexed by t_k. We prove that this difference is a decreasing sequence and that its limit is its infimum using the following Lemma:

Lemma 4.1. *If a sequence of real numbers is decreasing and bounded below, then its infimum is the limit.*

4.2 Results summary

Bounds obtained analytically are summarized in Table 3. We define $\bar{D} = \frac{P}{1+\alpha_{N1}}$ as the average value of D and $\sigma = \sqrt{\frac{1}{N} * \sum_{k=1}^{N} (D_k^{N0} - \bar{D})^2}$ its standard deviation. Function $LCM(x, y)$ extracts the least common multiple of x and y, with $x > 0$ and $y > 0$.

Synchronization accuracy is proved to be 0 if $\alpha_i = 0, \forall i \in \{0, .., N-1\}$. For the 2-node $\alpha_{Ni} \neq 1$ scenario, we provide an upper bound on the accuracy that is tight. For the 3-node ring and line topologies, we haven't succeeded in deriving analytic bounds for the $\alpha_{Ni} \neq 1$ case so far. However, for the 3-node topologies SISP converges to a perfect synchronization if $\alpha_i = 0$ for all nodes. Thus, we argue that if it is possible to learn and compensate for these drifts over time with an improved version of SISP, we can prove the convergence of this new protocol as it becomes equivalent to the no-drift case. Future works will concentrate on the design of this drift-aware

Table 3: Synchronization accuracy and convergence time expressions. Synchronization accuracy for the 2-node $\alpha_{Ni} \neq 1$ case is an upper bound.

Topology	Synchronization accuracy a	Convergence time t_a
2-node $\alpha_{Ni} = 1$	0	$\log_2 \lvert\beta_{N1}\rvert - 1$
2-node $\alpha_{Ni} \neq 1$	$(1 - \alpha_{N1}) * (\bar{D} + \sigma) + \frac{4}{3}$	$\max(T_{N0N1},\ \log_2 \lvert\beta_{N1}\rvert - 1)$ with $T_{N0N1} = LCM(P, P * \alpha_{N1})/\alpha_{N1}$
3-node ring $\alpha_{Ni} = 1$	0	$\log_2(\max_i \lvert\beta_{Ni}\rvert) - 1$
3-node ring $\alpha_{Ni} \neq 1$	-	$\max\left[LCM(T_{N0N1}, T_{N0N2}),\ \log_2(\max_i \lvert\beta_{Ni}\rvert) - 1\right]$
3-node line $\alpha_{Ni} = 1$	0	$\log_2(\max_i \lvert\beta_{Ni}\rvert) - 1$
3-node line $\alpha_{Ni} \neq 1$	-	$\max\left[LCM(T_{N0N1}, T_{N0N2}),\ \log_2(\max_i \lvert\beta_{Ni}\rvert) - 1\right]$

SISP protocol. Next, a detailed proof for the 2-node topology convergence time and accuracy is given.

4.3 2-node topology

We illustrate this proof with the sequence of SISP protocol messages given in Figure 7. Assuming the nodes begin sending messages at time t_0, the shared clock values at t_0 for the two nodes can be calculated for the reference clock $LCLK_{N0}$. By recurrence, the shared clock expression at time t_k can be deducted from the shared clock expression at t_{k-1}. Following, the SCLK difference is extracted and interpreted as a sequence. We prove by recurrence that this SCLK difference is a decreasing sequence and deduce its limit. Lemma 4.1 is then leveraged to prove protocol convergence.

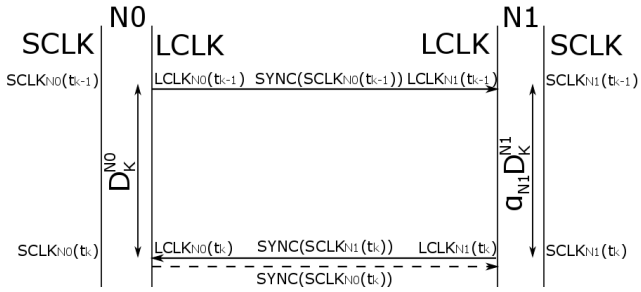

Figure 7: SISP protocol execution for 2-node scenario

4.3.1 *No drift case ($\alpha_{Ni} = 1$).*

Local clocks of the two nodes run at the same speed. Thus, the duration D_k^{N1} elapsed between the two shared clock updates on node $N1$ is the same as D_k^{N0} on node $N0$.

Synchronization accuracy. The shared clocks values of both nodes are given in Eq. (4) if the first SYNC message is sent by $N0$ and in Eq. (5) if it is sent by $N1$:

$$\begin{aligned} SCLK_{N0}(t_k) &= SCLK_{N0}(t_{k-1}) + D_k^{N0} \\ SCLK_{N1}(t_k) &= \frac{1}{2} * \left[SCLK_{N0}(t_{k-1}) + SCLK_{N1}(t_{k-1}) + 2 * D_k^{N0}\right] \end{aligned} \tag{4}$$

$$\begin{aligned} SCLK_{N1}(t_k) &= SCLK_{N1}(t_{k-1}) + D_k^{N1} \\ SCLK_{N0}(t_k) &= \frac{1}{2} * \left[SCLK_{N0}(t_{k-1}) + SCLK_{N1}(t_{k-1}) + 2 * D_k^{N0}\right] \end{aligned} \tag{5}$$

We are interested in identifying the time t_k where the shared clock values of the two nodes are equal; in this case, the difference between the two shared clocks is 0. We denote this difference $d_{N0N1}(t_k) = SCLK_{N0}(t_k) - SCLK_{N1}(t_k)$. Since only two nodes are considered in this topology, we will drop the index $N0N1$ to simplify notations in all derivations related to the 2-nodes scenario. We substitute in this formulæ the equations of shared clocks given in Eq. (4) or in Eq. (5). For both cases, we obtain after substitution:

$$d(t_k) = \frac{1}{2} * (SCLK_{N0}(t_{k-1}) - SCLK_{N1}(t_{k-1})) = \frac{1}{2} * d(t_{k-1})$$

As such, the difference $d(t_k)$ can be expressed according to the difference at t_0:

$$d(t_k) = \frac{1}{2^k} * d(t_0)$$

The limit of this sequence is 0. It is straightforward to prove by induction that this sequence is decreasing for increasing k. From Lemma 4.1, we can state that the shared clock difference converges to 0. Thus, shared clock values converge to the same value and perfect synchronization is thus achieved.

Convergence time. Convergence time t_a is deduced from the minimum number of messages needed to achieve convergence. It is deduced from setting $\frac{1}{2^k} * d(t_0)$ to 1 and solving it for k:

$$t_a = k = \log_2 d(t_0) = \log_2 \lvert\beta_{N1}\rvert - 1 \tag{6}$$

as $d(t_0) = \beta_{N1}/2$.

4.3.2 *Drift case ($\alpha_{N1} \neq 1$).*

Clock $LCLK_{N0}$ is still the reference clock, and the local clock of $N1$ drifts away from $LCLK_{N0}$ with rate $\alpha_{N1} \neq 1$. As such, $D_k^{N1} = \alpha_{N1} * D_k^{N0}$.

Convergence time. The simulation results of Section 3 show that the shared clock difference repeats itself periodically after all offsets have been compensated for. The period of the shared clock difference d is given by the least common multiple (LCM) of the SYNC message periods of both nodes: $LCM(P, P * \alpha_{N1})$ (its the hyper-period of the two periodic flows emitted by $N0$ and $N1$). This value has to be converted to the time reference of node $N0$. The period of d is denoted T_{N0N1} and given by:

$$T_{N0N1} = \frac{1}{\alpha_{N1}} * LCM(P, P * \alpha_{N1}) \tag{7}$$

Convergence occurs if all offsets have been compensated for and one hyper-period T_{N0N1} has at least elapsed:

$$t_a = \max(T_{N0N1}, \log_2 |\beta_{N1}| - 1) \tag{8}$$

Synchronization accuracy. As before, we aim at calculating the shared clock difference $d(t_k)$ to prove its convergence and calculate its accuracy. The shared clocks values of both nodes are given in Eq. (9) if the first SYNC message is sent by $N0$ and in Eq. (10) if it is sent by $N1$:

$$\begin{aligned} SCLK_{N0}(t_k) &= SCLK_{N0}(t_{k-1}) + D_k^{N0} \\ SCLK_{N1}(t_k) &= \left\lfloor \frac{1}{2} * [SCLK_{N0}(t_{k-1}) + SCLK_{N1}(t_{k-1}) \right. \\ &\qquad \left. + (\alpha_{N1} + 1) * D_k^{N0}] \right\rfloor \end{aligned} \tag{9}$$

$$\begin{aligned} SCLK_{N1}(t_k) &= SCLK_{N1}(t_{k-1}) + \alpha_{N1} * D_k^{N0} \\ SCLK_{N0}(t_k) &= \left\lfloor \frac{1}{2} * [SCLK_{N0}(t_{k-1}) + SCLK_{N1}(t_{k-1}) \right. \\ &\qquad \left. + (\alpha_{N1} + 1) * D_k^{N0}] \right\rfloor \end{aligned} \tag{10}$$

The difference $d(t_k) = SCLK_{N0}(t_k) - SCLK_{N1}(t_k)$ can be obtained by substituting $SCLK_{N0}(t_k)$ and $SCLK_{N1}(t_k)$ using (9) and (10), respectively. Expressions for the shared clock difference are given in (11) if the first SYNC message is sent by $N0$ and (12) otherwise.

$$d(t_k) = D_k^{N0} - \left\lfloor \frac{1}{2} * \left[-d(t_{k-1}) + (\alpha_{N1} + 1) * D_k^{N0} \right] \right\rfloor \tag{11}$$

$$d(t_k) = \left\lfloor \frac{1}{2} * \left[d(t_{k-1}) + (\alpha_{N1} + 1) * D_k^{N0} \right] \right\rfloor - \alpha_{N1} * D_k^{N0} \tag{12}$$

Using the properties of the integer part it is possible to bound the shared clock difference as shown in inequalities (13) and (14).

$$\begin{aligned} \frac{1}{2} * \left[d(t_{k-1}) + (1 - \alpha_{N1}) * D_k^{N0} \right] &< d(t_k) \\ &< \frac{1}{2} * \left[d(t_{k-1}) + (1 - \alpha_{N1}) * D_k^{N0} \right] + 1 \end{aligned} \tag{13}$$

$$\begin{aligned} \frac{1}{2} * \left[d(t_{k-1}) + (1 - \alpha_{N1}) * D_k^{N0} \right] - 1 &< d(t_k) \\ &< \frac{1}{2} * \left[d(t_{k-1}) + (1 - \alpha_{N1}) * D_k^{N0} \right] \end{aligned} \tag{14}$$

By recurrence, the difference at t_k can be expressed according to the difference at t_0 as in the inequalities (15) and (16).

$$A - \frac{2}{3} < d(t_k) < A + \frac{4}{3} \tag{15}$$

$$A - \frac{4}{3} < d(t_k) < A + \frac{2}{3} \tag{16}$$

where :

$$A = \frac{1}{2^k} * \left[d(t_0) + (1 - \alpha_{N1}) * (D_1^{N0} + 2 * D_2^{N0} + \ldots + 2^{k-1} * D_k^{N0}) \right].$$

Accuracy upper bound. We recall that D_k^{N0} represents the difference between emission and reception dates of a SYNC message at $N0$. We can assume that in average, it is close to $P/2$, with P the SYNC emission period of SISP. To obtain an estimate of A, and thus an estimate of the bounds on $d(t_k)$, we assume D_k^{N0} is normally distributed: 50% of its values are less than $P/2$ and 50% of the values are greater than $P/2$. The mean of this distribution is equal to $\bar{D} = P/(1 + \alpha_{N1})$ and the standard deviation to $\sigma = \sqrt{\frac{1}{k} * \sum_{i=1}^{k} (D_i^{N0} - \bar{D})^2}$.

With this assumption, we can substitute the durations D_k^{N0} by the mean duration $\bar{D}$ and the standard deviation σ in A. As such, we obtain an approximated value of A :

$$\tilde{A} = \frac{1}{2^k} * \left[d(t_0) + (1 - \alpha_{N1})(2^k - 1)(\bar{D} + \sigma) \right]$$

Calculating the limit of $\tilde{A}$ as k grows to infinity, we get:

$$\tilde{A}_\infty = \lim_{k \to \infty} \tilde{A} = (1 - \alpha_{N1})(\bar{D} + \sigma)$$

Since we may not know which node has started to emit SYNC messages first, we extract from Eq. (15) and (16) an upper bound on $d(t_k)$ which is given by

$$d(t_k) < \tilde{A}_\infty + 4/3 \tag{17}$$

The average value of $d(t_k)$ can as well be computed by setting σ to zero in $\tilde{A}_\infty$. This average value derivation only depends from system parameters and can be computed without any precise information on the protocol execution. To get a more precise estimation of the accuracy, a few values of D_k^{N0} are required to calculate the standard deviation σ. These values can be either obtained by

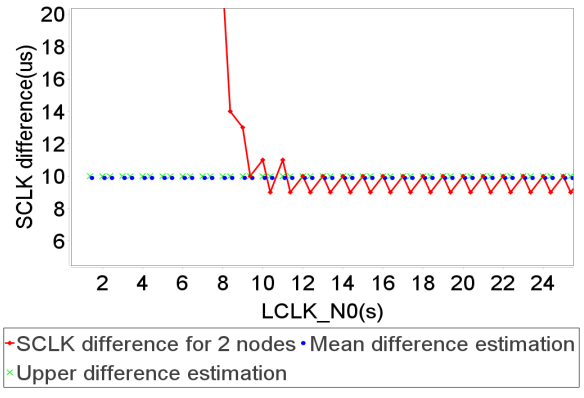

Figure 8: Mean and upper bound estimation of the shared clock difference for the 2-node topology for a 20ppm drift.

the first steps of a simulation or by measurements of the SYNC message emission and reception dates.

Figure 8 illustrates, on top of the simulated shared clock difference of Figure 3, our average and upper bound on $d(t_k)$. The average calculation already provides a meaningful order of magnitude of the accuracy. A more precise estimation, using the upper bound of Eq.(17), is plotted as well. In this example the upper bound on the synchronization accuracy is less than 10μs.

4.4 3-node topology

4.4.1 *No drift case ($\alpha_{Ni} = 1$).*

This section concentrates on the 3-node topology for the case where no drift exists between the nodes. Derivations are similar to the ones presented earlier for the 2-node topology. Thus, only basic explanations are provided in the following.

Synchronization accuracy. As for the 2-node topology, the difference of SCLK values is computed for each couple of nodes. The protocol is convergent if the three differences converge. By induction we can prove that the sequences of shared clock differences for each pairs are decreasing and that we can calculate their limit which is 0. This derivation holds whether nodes are in a ring setting or in a line setting.

Convergence time. The convergence time is calculated for each pair of nodes as done for the 2-node topology by counting the minimum number of messages to achieve a difference of 1. Convergence time for the whole network is given by the maximum convergence time for the all pairs of nodes. Formally, we have:

$$t_a = \log_2 \max_i(d_{N0Ni}(t_0)) = \log_2(\max_i |\beta_{Ni}|) - 1$$

4.4.2 *Drift case ($\alpha_{Ni} \neq 1$).*

Convergence time. As for the 2-node case, the simulation results of Section 3 show that the shared clock difference repeats itself periodically after all offsets have been compensated for. The period of the shared clock difference for the two pairs $(N0, N1)$ and $(N0, N2)$ are given by the least common multiple (LCM) of the SYNC message periods of both nodes:

$$T_{N0Ni} = \frac{1}{\alpha_{Ni}} * LCM(P, P * \alpha_{Ni}),\ i = 1, 2 \qquad (18)$$

A global period can be computed from T_{N0N1} and T_{N0N2} by computing $T = LCM(T_{N0N1}, T_{N0N2})$. Convergence occurs if all offsets have been compensated for and one global hyper-period T has at least elapsed:

$$t_a = \max(T,\ \log_2(\max_i |\beta_{Ni}|) - 1) \qquad (19)$$

Synchronization accuracy. As defined previously, it is given by the maximum shared clock difference observed after convergence over the three pairs of nodes. As observed in the simulations, the maximum shared clock difference is observed for the pair of nodes having the largest difference in drifts. Accounting for all drifts in an analytic derivation of the shared clock difference of all pairs is challenging for both ring and line topologies and have not been derived for this study. However, it is interesting to note that if it is possible to learn and compensate the drifts from the SYNC messages in SISP, calculating the accuracy resumes to the case where no drift exists. In this case, it is possible to prove the convergence and offer a theoretical accuracy of 0. Future works will concentrate on deriving a novel version of SISP that mitigates clock drifts by prediction as proposed in [7].

5 EMPIRICAL STUDY

In order to complete the theoretical and simulation study of the SISP protocol convergence, we have deployed SISP on the DecaWiNo nodes introduced earlier. The implemented protocol uses the same configuration as the one deployed in simulation. The SYNC message period is equal to 1s.

Figure 9 shows the shared clock difference $d(t_k)$ measured for a 2-node topology. The x-axis shows the reference clock $LCLK_{N0}$ and the y-axis the shared clock difference. From this curve, it can be seen that convergence occurs at 13 seconds and that the synchronization accuracy is then around 10 microseconds. The analytic upper bound on the accuracy is plotted in red as well. This bound accounts for the standard deviation derived from the D_k^{N0} values obtained by simulation. Our bound clearly fits very well the measured values.

On Figure 10, the shared clock difference for each couple of nodes has been represented for the 3-node ring topology. The three differences have a similar evolution. They are decreasing until 15s on the reference local clock and then they present regular variations. The synchronization accuracy is of about 8 μs. Similarly, Figure 11 shows the three shared clock differences for the 3-node line topology. Convergence time is clearly longer for the line topology than for the ring topology (26s. versus 15s.). The synchronization accuracy is larger as well (32μs. versus 8μs.). Measurements are close to the values observed by simulations for 2-node and 3-node ring scenarios. For the line scenario, the convergence time

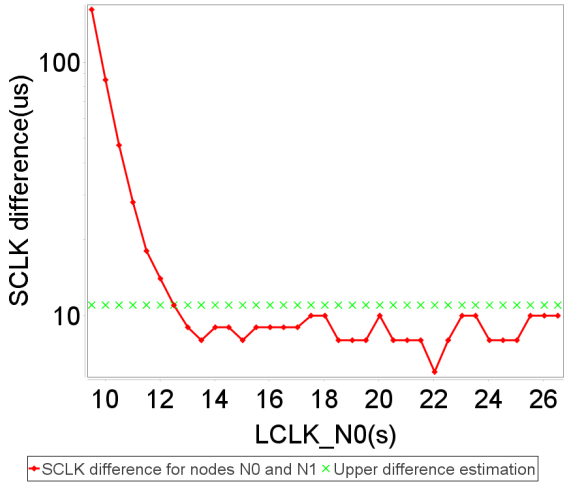

Figure 9: Measured difference of the shared clocks (in log scale) for a 2-node topology.

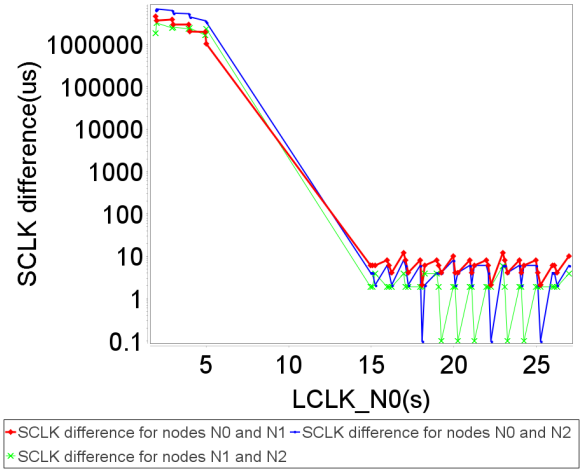

Figure 10: Measured difference of the shared clocks (in log scale) for a 3-node ring topology.

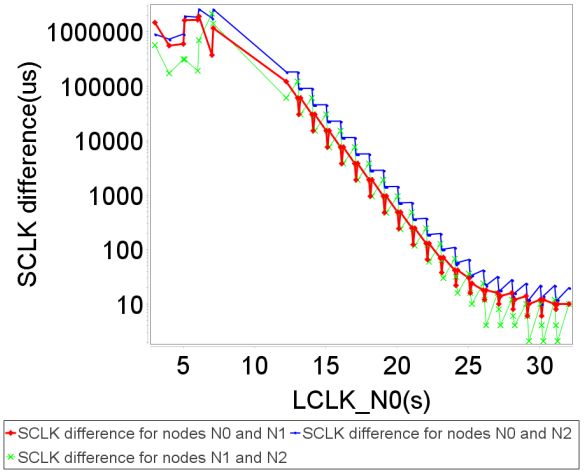

Figure 11: Measured difference of the shared clocks (in log scale) for a 3-node line topology.

Table 4: Measured synchronization accuracy and convergence time. Theoretical values are given in parenthesis if available.

Topology	Synchronization accuracy (μs)	Convergence time (s)
2-node	10 (10)	13 (16)
3-node ring	8	15 (19)
3-node line	32	26 (19)

calculation should be improved to account for the propagation of messages.

Table 4 summarized the synchronization accuracy and the convergence time observed by measurement. Theoretical values are included as well, if available. These results are similar to the simulation results of Table 2. Overall, SISP only needs a few seconds to achieve a stable state. This convergence time depends on the number of nodes and on the network topology. The synchronization accuracy is the order of a few tens of microseconds, which is in line with the protocol design goal.

6 CONCLUSIONS

In this paper we studied the convergence properties of SISP, a lightweight, totally distributed synchronization protocol for wireless sensor networks. Three different approaches have been presented to investigate the protocol behavior. First, simulations have helped us to establish that there is a convergence time before nodes agree on a shared clock. If the clock of nodes drifts apart, the protocol produces periodic variations of the shared clock after initial convergence. Second, we have derived analytic expressions to quantify both convergence time and synchronization accuracy. For the 2-node scenario, we even have been able to calculate a tight upper bound on the accuracy. Last, we present measurements on a real wireless sensor network which are totally in line with simulation and theoretical results.

Future works will now concentrate on extending these results to a larger number of nodes and to the case where SYNC messages collide. Another promising research is the definition of a new version of SISP that compensates for the drift of nodes by learning it over time. This will ensure proper convergence as in this case, the problem reduces to the case where no drifts exist (they are mitigated by the learning step). In other words, we will be able to easily calculate the accuracy and convergence time as the α_{Ni} values will be equal to one.

ACKNOWLEDGEMENTS

This work has been supported by Region Occitanie and European FEDER program under the MHTag GUINESS project.

REFERENCES

[1] *WirelessHART Specification 75, TDMA Data-Link Layer*, HART Communication Foundation Standard hCF SPEC-75, 2008.1.1
[2] *IEEE Standard for Local and Metropolitan Area Networks—Part 15.4 Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer.* IEEE Standard 802.15.4e-2012, Apr. 2012
[3] Q. Wang, K. Jaffrès-Runser, Y. Xu, J.-L. Scharbarg, Z. An and K. Fraboul *TDMA versus CSMA/CA for wireless multi-hop communications: a stochastic worst-case delay analysis*, in IEEE Transactions on Industrial Informatics, vol. 13, no. 2, pp. 877-887, April 2017.
[4] J. Elson, L. Girod, and D. Estrin. *Fine-grained Network Time Synchronization Using Reference Broadcasts.* OSDI '02, 2002. Boston, Massachusetts, USA, pp.147–163.
[5] S. Ganeriwal, R. Kumar, and M. Srivastava. *Timing-sync Protocol for Sensor Networks* SenSys'03, 2003, Los Angeles, California, USA, pp. 138–149,
[6] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. *The Flooding Time Synchronization Protocol* SenSys '04, 2004, Baltimore, MD, USA, pp. 39–49
[7] C. Lenzen, P. Sommer, and R. Wattenhofer. *PulseSync: an efficient and scalable clock synchronization protocol.* IEEE/ACM Trans. Netw. 23, 3 (June 2015), pp. 717-727.
[8] S. Gonzalez, T. Camp and K. Jaffrès-Runser. *The Sticking Heartbeat Aperture Resynchronization Protocol.* ICCCN'17, Vancouver, Canada, August 2017.
[9] A. van den Bossche, T. Val, and R. Dalce. *SISP: a lightweight Synchronization Protocol for Wireless Sensor Networks.* ETFA WiP, 2011. Toulouse, France, pp. 1–4.
[10] A. van den Bossche. *DecaWiNo ressources.* https://www.irit.fr/ Adrien.Van-Den-Bossche/decaduino/index.html, last accessed April 2017.