FLOPSYNC-QACS: Quantization-Aware Clock Synchronization for Wireless Sensor Networks

Federico Terraneo*, Alessandro Vittorio Papadopoulos†, Alberto Leva*, Maria Prandini*
*Politecnico di Milano, Milano, Italy
†Mälardalen University, Västerås, Sweden

***Abstract*—The development of distributed real-time systems often relies on clock synchronization. However, achieving precise synchronization in the field of Wireless Sensor Networks (WSNs) is hampered by competing design challenges, such as energy consumption and cost constraints, e.g., in Internet of Things applications. For these reasons many WSN hardware platforms rely on a low frequency clock crystal to provide the local timebase. Although this solution is inexpensive and allows for a remarkably low energy consumption, it limits the resolution at which time can be measured. The FLOPSYNC synchronization scheme provides low-energy synchronization that takes into account the quartz crystal imperfections. The main limitation of the approach are the effects of quantization.**

In this paper we propose a clock synchronization scheme that explicitly takes into account quantization effects caused by low frequency clock crystal, thus addressing the clock synchronization issue in cost-sensitive WSN node platforms. The solution adopts switched control for minimizing the effect of quantization, with minimal overhead. We provide experimental evidence that the approach manages to reach a synchronization error of at most 1 clock tick in a real WSN.

I. Introduction

During the past decade, the Internet of Things (IoT) has gained significant attention both in academia and industry. The basic idea of IoT is to connect objects to the internet and make them communicate with each other. According to a recent study by Gartner [4], the current count of IoT devices is around 6.4 billion devices (not including smartphones, tablets, and computers), and it is expected to grow up to 21 billion by 2020. In such a scenario, wireless communication technologies will play a major role, and in particular, Wireless Sensor Networks (WSNs) will proliferate in many application domains. Indeed, the small, inexpensive and low power WSN nodes can be an enabling technology for IoT.

Among the various challenges, time synchronization is one of the most important for the correct operation of WSNs [21, 25]. In particular, it allows for successful communication between the different nodes, but also for location and proximity estimation [12], energy efficiency [10, 19], and mobility [20]. In general, WSN nodes coordination is crucial, especially when real-time operations must be executed. Since many IoT devices are battery powered, and time synchronization is continuously executed, energy efficient architectures and protocols are necessary [1].

Even more, since the cost of the nodes is relevant and a high-precision time synchronization may not be needed in some IoT applications, the use of low resolution clock oscillators is often a viable choice, see e.g. [15, 16]. Apparently, the clock resolution affects the minimum achievable synchronization error, and protocols able to push the performance to the limits are needed. In this paper, we propose a synchronization mechanism based on a switched control policy aimed to minimize the effect of quantization on the synchronization error, with a minimal overhead. The following work is cast in the framework of *multi-hop master-slave clock synchronization*, i.e., when the network is composed by a master, and of a number of slaves which synchronize their clock to the masters' clock.

The rest of the paper is organized as follows. Section II presents a technological view on the clock synchronization problem. Section III describes the proposed switched control scheme, while Section IV presents simulation and experimental results. Section V concludes the paper.

II. The synchronization problem

A master-slave clock synchronization scheme works by disseminating the timing information using packets transmitted by the radio transceiver of the WSN nodes. In simple topologies, such as star networks, synchronization packets can be transmitted by the master node. In multi-hop networks, flooding schemes [3, 13] allow for the dissemination of synchronization packets also to nodes that are not directly in the radio range of the master node. Packets may contain a timestamp with the time of the master node, or this information can be implicit if packets are transmitted periodically over a contention-delay-free MAC [22]. A clock skew compensation [9, 17, 26] scheme is often used to minimize the divergence of the nodes' local clock in between synchronizations. Drift compensation is also possible, accounting for common error causes such as temperature variations [18, 22]. Finally, propagation delay estimation and compensation [11, 23] can be employed for ultra high precision synchronization.

This work was partially supported by the European Commission under the project UnCoVerCPS with grant number 643921, and by the Swedish Foundation for Strategic Research under the project "Future factories in the cloud (FiC)" with grant number GMT14-0032.

Despite the many different features a master-slave clock synchronization scheme can be composed of, all of them have in common the need for timestamping incoming packets using the local clock. Timekeeping in WSN nodes is performed by dedicating a hardware timer/counter unit of the node microcontroller that is read to know the time, and allows to set interrupts for generating events. One way to perform incoming packet timestamping is reading the hardware counter in the packet reception interrupt handler [13]. Alternatively, one may use of a hardware input capture module [3, 22], a common feature of modern timer peripherals that takes a snapshot of the counter upon receiving an event. In all cases, the finite frequency of the local hardware counter introduces a quantization effect in packet timestamping, of increasing magnitude as the counter frequency is decreased.

Energy consumption constraints limit the frequency at which a WSN node hardware counter can be operated. It is well known that the power consumption is proportional to the operating frequency, load capacitance and the square of the operating voltage. The dependency of power, and thus energy consumption on the clock frequency is a relevant concern in the hardware counter, that has to remain active also in sleep states to not lose the notion of time. For this reason, it is common in WSN nodes to have the microcontroller CPU clocked at several megahertz, while the timer used for timekeeping at only 32768Hz [16]. This choice is based on the availability of inexpensive and ultra low power clock crystals for this frequency.

A breakthrough solution is the Virtual High-resolution Time (VHT) algorithm [19]. It resynchronizes a high frequency timer, which is turned off in deep sleep, while a low frequency timer is always active. Although this solution has been used to achieve high synchronization precision and ultra-low energy consumtion [22], its implementation requires hardware support that is uncommon in most WSN nodes. The main requirements for implementing VHT are two: a high frequency timer clocked with a stable (e.g. clock crystal) oscillator, and hardware support for timestamping an edge of the low frequency clock with the high frequency timer. In [19] this problem was solved through hardware modifications to an existing node and also proposed a hardware implementation in VHDL. Although specialized nodes appeared having this support [24], widely deployed nodes [6], such as the TelosB [16], are not capable enough to implement VHT. This is not only a legacy issue, as cost sensitive IoT applications may not have such strict timing requirements to justify the additional cost of VHT support. Addressing the clock synchronization problem using low frequency clock timers is therefore a relevant research topic that has the potential to enable low cost yet real-time capable WSN node platforms.

To further evidence how clock synchronization with a high and low frequency clock are two completely different

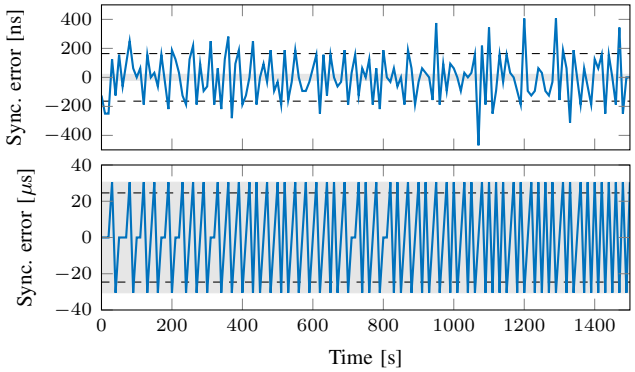

Figure 1: Experimental result showing the synchronization error using the high frequency timer (top graph) and low frequency timer (bottom graph) on the WandStem node. Note the different time units on the vertical axis.

scenarios, we made a test synchronizing two nodes using the FLOPSYNC-2 [22] scheme on the WandStem [24] WSN node platform. This platform has support for the VHT, so it was possible to compare the synchronization quality in both cases, on the same hardware. The tests were performed using a 10s synchronization period, and the reported error samples are taken at the end of each synchronization period, immediately before the next synchronization. The top graph in Figure 1 shows the results with a high frequency timer, while the bottom graph shows the low frequency timer case. Although the high frequency timer has a resolution of 20.8ns (indicated with the gray area), the standard deviation of the clock synchronization error in the reported test is 164ns (dashed lines in the figures), a significantly higher value. The reason for this is that at high frequencies, jitter caused by the oscillator phase noise, and the packet transmission jitter at the radio transceiver are greater than the quantization-induced error [22]. On the contrary, when the low frequency timer is used, the standard deviation of the error is 24.6μs (indicated with the dashed lines), which is lower than the 30.5μs timer resolution (indicated with the gray area). More significantly, the error shows a regular pattern composed of only three values: 0 and ± 1 timer tick, highlighting that the quantization induced error is greater in magnitude than the noise sources.

The clock synchronization algorithm proposed in this paper includes a switching control scheme that can minimize the effect of quantization on the clock synchronization error. The proposed solution is applicable in all cases where quantization is the major source of synchronization error.

III. FLOPSYNC-QACS

After detailing the sources of quantization in clock synchronization, we here formalize the problem and introduce FLOPSYNC-QACS, our proposed switched control scheme that minimizes the effect of quantization.

A. Sources of quantization

When we analyze the quantization problem from an hardware perspective, it is clear that the hardware counter is incremented on the active edge of the clock signal, and asynchronous events – such as the packet reception one – can occur at any time between two active edges. In this case the reported timestamp will naturally be the value of the counter incremented by the previous edge. Thus, the hardware timestamping operation works like the mathematical floor operation. A second quantization occurs when the output of the clock correction algorithm, which is computed using floating point or fixed point numbers, has to be converted back to the tick resolution. In this case, since the conversion is done by a software routine, it is possible to choose the quantization function, for example using the mathematical rounding operator.

From a mathematical perspective, a quantizer maps a real-valued function into a piecewise constant function taking values in a discrete set. Here quantizers are either rounding operators (for the corrective actions) or floor operators (for the measured synchronization error). Given a real number z, we denote with $\operatorname{sign}(z)$ the sign function, with $\lfloor z \rfloor$ the floor operator, and with $\rho(z)$ the rounding operator, with $\rho(0.5) = 1$, and $\rho(-0.5) = -1$. We also define the *rounding error* of a real number z as $\Delta_z := z - \rho(z)$. Notice that the rounding error of a real number z is always bounded as $|\Delta_z| \leq \frac{1}{2}$. Finally, note that given two real numbers $a \in \mathbb{R}$, and $b \in \mathbb{R}$, we have that $\rho(\rho(a) + b) = \rho(a) + \rho(b)$.

B. Problem formalization

The problem of time synchronisation in a distributed system is a well known and studied problem in computer science [2, 5, 8, 14], and has recently gained a lot of attention in control theory as well [7, 10]. We here consider the setting of a WSN where a the *master* node sends a synchronization signal with a fixed period T to all the *slave* nodes in the network. Then, synchronization is achieved by controllers of each slave node that communicates with the master and receives the synchronization packets every T seconds.
If a flooding scheme like glossy [3] is used for the transmission, it can be assumed that the medium access contention does not introduce uncertainty in the transmission time.

The synchronization error at time kT, $k \in \mathbb{N}$, can be defined as:

$$e(k) := t(k) - \hat{t}(k),$$

where $t(k)$ denotes the master node clock at the k-th synchronization, and $\hat{t}(k)$ is the slave estimate of the master node's clock. Since the error accumulates over time, during each time interval $[kT, (k+1)T]$ the synchronization error dynamics can be described as:

$$e(k+1) = e(k) + d(k), \tag{1}$$

where $d(k)$ is a disturbance that accounts for different factors influencing the synchronization error. Notice that the time scale of the considered dynamics is defined by the synchronization period T. In general, the disturbance over a synchronization period can be characterized as:

$$d(k) = -\int_{kT}^{(k+1)T} \frac{\delta_f(\tau)}{f_o} d\tau, \tag{2}$$

where f_o is the nominal frequency of the slave clock oscillator, and $\delta_f(t)$ the (continuous-time) variation of that frequency caused by manufacturing tolerances, aging, thermal stress, and short-term jitter. The minus sign in (2) is because a positive δ_f makes the local clock advance, while for convenience (1) contains $d(k)$ with the plus sign.

Notice that all the uncertainty is confined in the disturbance term $d(k)$, and based on model (1), a controller can be design to reject $d(k)$. Whence $d(k)$ is characterized, a simple control policy can be implemented with a very little computational overhead, like, e.g., [9], in contrast to other classical alternatives. The various sources of disturbances can be counteracted by considering the different time scale of their contributions:

- Tolerances due to *imperfections in the manufacturing process* of the quartz crystals result in a *constant frequency error* $\overline{\delta_f}$.
- *Aging* acts on a time scale of days while reasonable values for the synchronization period T are seconds or minutes, hence it can be thought of as a *constant disturbance* contribution, and eliminated at steady state by integral control.
- The temperature dependence of clock crystal oscillators is one of the most common source of variable disturbance [19]. In a wide variety of operating conditions a WSN undergoes either abrupt but sporadic thermal stress episodes, or environmental variations that are slow compared to the thermal dynamics of typical nodes. The proposed controller can be extended to compensate for abrupt thermal variation episodes [22]. In between such events, however, this disturbance contribution can be considered *constant* as well.
- *Short-term jitter* acts on the time scale of electronic noise, hence being too fast to compensate, and therefore providing the ultimate bound for the achievable synchronization quality. However, here we are addressing the case where quantization resolution a greater source of error than jitter, as shown in Section II.

The above characterization of the disturbance contributions allows us to focus on optimizing the controller for the case when d is constant, although the proposed controller will obviously still be able to cope with variable disturbances.

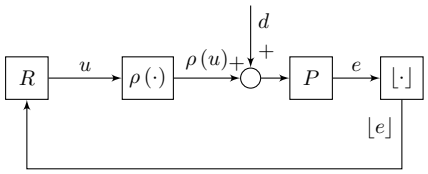

Figure 2: FLOPSYNC control scheme with quantizers.

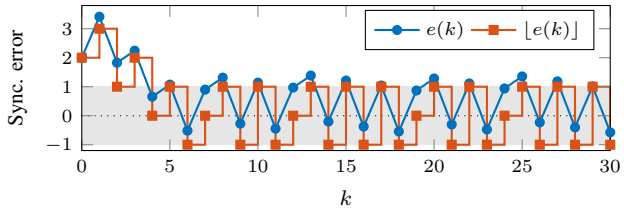

Figure 3: The effect of quantization in the FLOPSYNC synchronization scheme.

C. The FLOPSYNC control scheme

In [9], FLOPSYNC was proposed as a viable solution for clock synchronization in WSNs. FLOPSYNC introduces as control variable a corrective action u on the synchronization error, which is then quantized. Hence, (1) becomes:

$$e(k+1) = e(k) + \rho\left(u(k)\right) + d(k). \tag{3}$$

In the FLOPSYNC scheme, u is determined from the quantized measurements of the synchronization error $\lfloor e \rfloor$, by a discrete-time Proportional Integral (PI) controller as:

$$u(k+1) = u(k) + \lfloor e(k) \rfloor - \alpha \lfloor e(k+1) \rfloor \tag{4}$$

where α is the only design parameter. Figure 2 shows the resulting control scheme, where P is the dynamic process (3), and R is the controller (4). Plugging (4) into (3), we get that:

$$\begin{aligned} e(k+1) = e(k) + d(k) \\ + \rho\left(u(k-1) + \lfloor e(k-1) \rfloor - \alpha \lfloor e(k) \rfloor\right) \end{aligned}$$

In [9] the FLOPSYNC control scheme was designed by ignoring the quantizers. More specifically, when no quantization were in place, and inverting (3), leads to:

$$\begin{aligned} e(k+1) &= (1-\alpha)e(k) + e(k-1) + u(k-1) + d(k) \\ &= (2-\alpha)e(k) + d(k) - d(k-1) \end{aligned}$$

which corresponds to an asymptotically stable linear system if $1 < \alpha < 3$. For a constant disturbance, i.e., $d(k) = d(k-1) = \overline{d}$, the scheme guarantees the convergence of the synchronization error to zero, with a rate of convergence that depends on the parameter α.

When quantizers are in place, on the other hand, the synchronization error still converges towards zero, but, intuitively, it is not possible to distinguish errors below the clock resolution. Moreover, the disturbance d is integrated over time according to (3). The integrated residual disturbance is not detectable on the quantized output $\lfloor e \rfloor$ as long as it is smaller than the clock resolution due. This causes the controller to react whenever the quantization of the integrated residual disturbance switches to 1 or -1, steering the controlled system to a limit cycle of amplitude 2. An example of this effect is shown in Figure 3, with $\alpha = 1.5$, $d(k) = \overline{d} = \sqrt{2}$, and the control system initialized as $e(0) = 2$, $u(0) = 0$.

This paper proposes a switched control scheme that reduces the effect of quantization, steering the system to a limit cycle of an amplitude that is half of the one obtained with the FLOPSYNC control scheme in [9]. The proposed solution has the additional advantage of sticking to simple controllers that lead to an easily implemented system in an embedded device, with very low overhead.

D. The proposed switched control scheme

In this section, we propose a switched variant of the classical FLOPSYNC controller which improves its performance in the presence of quantization effects, and it is called FLOPSYNC-QACS.

The controller is composed by a linear part:

$$\tilde{u}(k+1) = \lfloor e(k) \rfloor - \alpha \lfloor e(k+1) \rfloor \tag{5}$$

and of a switched part where the control action $\tilde{u}$ is set as the input to the following modified discrete-time integrator:

$$\begin{cases} u(k+1) = u(k) + \tilde{u}(k+1), & \text{if } \lfloor e(k+1) \rfloor \neq 0 \\ u(k+1) = \rho\left(u(k)\right) + \tilde{u}(k+1), & \text{if } \lfloor e(k+1) \rfloor = 0 \end{cases}$$

that finally computes the actual corrective action u, based on the quantized synchronization error measurement $\lfloor e \rfloor$.

Figure 4 shows the resulting switched control scheme with $\tilde{R}$ being (5), P being the synchronization error dynamics (3), and z^{-1} being a unit delay operator in discrete time.

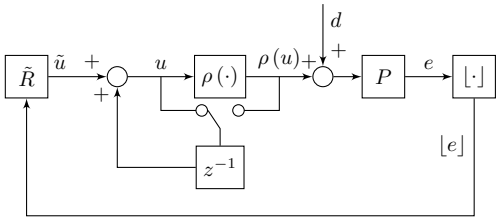

Figure 4: Control scheme of FLOPSYNC-QACS.

The switched control system dynamics is characterized by e as per (3), and by u, that can is computed as:

- if $\lfloor e(k+1) \rfloor = \lfloor e(k) + \rho\left(u(k)\right) + d(k) \rfloor = 0$, then:

$$u(k+1) = \rho\left(u(k)\right) + \lfloor e(k) \rfloor \tag{6}$$

- if $\lfloor e(k+1) \rfloor = \lfloor e(k) + \rho\left(u(k)\right) + d(k) \rfloor \neq 0$, then:

$$\begin{aligned} u(k+1) = u(k) + \lfloor e(k) \rfloor \\ - \alpha \lfloor e(k) + \rho\left(u(k)\right) + d(k) \rfloor \end{aligned} \tag{7}$$

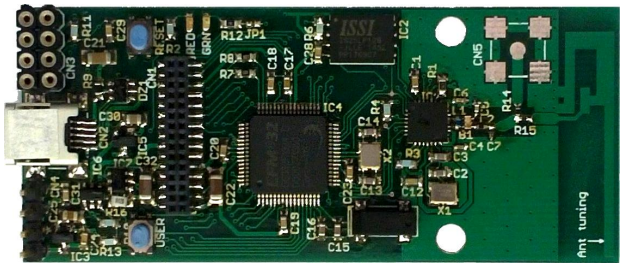

Figure 5: One of the WandStem WSN nodes that was used to test the FLOPSYNC-QACS scheme.

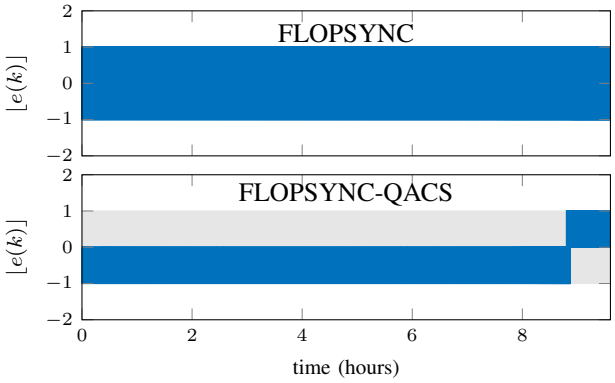

Figure 6: Experimental results comparing FLOPSYNC and FLOPSYNC-QACS.

Apparently, the computational complexity of the proposed solution is limited to the simple measurement of $\lfloor e(k) + \rho\left(u(k)\right) + d(k) \rfloor = \lfloor e(k+1) \rfloor$, and, based on the measured value, compute the corrective action as (6) or (7). As the following experiments will show, the proposed solution is able to reduce the amplitude of the limit cycle from 2 to 1, independently of the value of α.

IV. Experimental and simulation results

We implemented the FLOPSYNC and FLOPSYNC-QACS scheme on WSN nodes composed of WandStem [24] nodes, that employ ARM Cortex-M3 microcontrollers running at 48MHz, and CC2520 radio transceivers operating in the 2.4GHz band. One of the used nodes is shown in Figure 5. The control algorithms are implemented in C++ as an application for the Miosix[1] microcontroller operating system. For our implementation, we decided to use $\alpha = 11/8$, since it preserves stability of the closed-loop linear dynamics, and it can be easily implemented in an embedded device with fixed-point arithmetic.

A. An experimental test

We assessed the performance of the proposed solution in a real-world setting, where the actual disturbance acting on the system is not known, but it can be considered practically constant, according to the considerations in Section III-B. Real disturbances are actually varying, but it is safely assumed that such variability occurs at a time scale much longer than the synchronization period, and therefore the control scheme is able to reject them. The performed experimental test provide experimental evidence that this is the case.

In the test, three nodes are used. One plays the role of the master, broadcasting synchronization packets. Out of the other two, one runs the bare FLOPSYNC scheme, and the other FLOPSYNC switched variant of the scheme. The nodes are placed in an office environment, therefore exposed to radio interference from local wireless networks, and to temperature variations like those encountered in a typical indoor setting with standard climatization.

[1]Sources available at http://miosix.org

The synchronization period T is 10 seconds. The nodes' hardware timers have a measurement and actuation resolution – the *tick* – of 30.5μs, which is the source of quantization, and it is normalized to 1. The control parameter α was set to $11/8$. In order to show the long-term behavior of the system in the face of slowly varying disturbances, the experiment was set to last 10 hours.

Figure 6 shows both the quantized synchronization error in ticks for FLOPSYNC (left column) and FLOPSYNC-QACS (right column). The horizontal axes report the experiment time in hours. The $[-1, 1]$ synchronization error range is highlighted in both top plots with a gray area.

Notice that in the case of FLOPSYNC, the gray area is practically covered by the quantized synchronization error trajectory. This is because the quantized synchronization error oscillates between $\{-1, 0, 1\}$ with an excursion of amplitude 2.

In the case of FLOPSYNC-QACS, the quantized synchronization error first switches between $\{-1, 0\}$, then, after a brief transient, it switches between $\{0, 1\}$. For practically the whole experiment, the quantized synchronization error has an excursion of amplitude 1. More in details, the error lies in the $\{-1, 0\}$ or $\{0, 1\}$ range for 99.3% of the time.

We compare the two results by computing the Root Mean Square (RMS) performance index of the quantized synchronization error. The RMS computed in the case of FLOPSYNC is 0.878, while the RMS computed for FLOPSYNC-QACS is 0.499, i.e., about half than with bare FLOPSYNC. We can conclude that the proposed control scheme results in a lower RMS error magnitude also in a practical setting, where the disturbance is not rigorously constant.

Analyzing a bit more in detail the obtained result of FLOPSYNC-QACS, after about 8 hours and 30 minutes, the quantized synchronization error starts switching between the values $\{-1, 0, 1\}$, and then settles to a new regime, switching only between 0 and 1. This type of behavior is

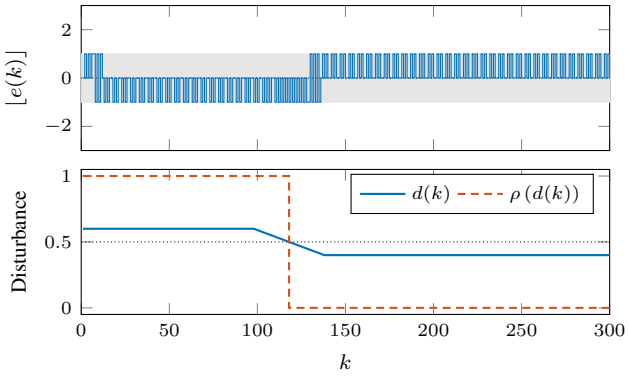

Figure 7: Replication of the experimental results with simulated dynamics.

probably explained considering that the disturbance is not exactly constant, but it might have a slowly varying behavior.

In order to better investigate what caused the transition, we performed a simulation study trying to replicate the same behavior with a slowly changing disturbance. Figure 7 shows the quantized synchronization error, and the disturbance acting on the system with its quantization. Remember that the corrective action is rounded, and it needs to compensate the quantized disturbance, i.e., in principle a perfect compensation is $u(k) = -\rho\,(d(k))$. Therefore, the transition between the two regimes happen exactly at the when the disturbance crosses the 0.5 threshold of the rounding. More specifically, we selected a disturbance that starts as a constant $d = \overline{d}_1 = 0.6$, i.e., $\Delta_d = -0.4 < 0$. Then from time $k = 100$ the disturbance slowly decreases linearly up to the value $d = \overline{d}_2 = 0.4$, i.e., $\Delta_d = 0.4 > 0$. Finally, the disturbance keeps constant and equal to $\overline{d}_2$. The simulation produces exactly the same behavior that can be observed in the experimental data of Figure 6. We can thus conclude that the behavior that appeared in the experimental results may have been caused by a disturbance similar to the one presented in the bottom graph of Figure 7.

V. Conclusions and Future Work

A control-based time synchronization mechanism for WSNs, called FLOPSYNC-QACS, was proposed for reducing the degradation effect due to quantization of both corrective actions and synchronization error. FLOPSYNC-QACS was implemented in a real WSN, and experimental results back up the proposed solution.

As a future work, we plan to provide formal guarantees on the obtainable performance of the proposed approach, like, for example, convergence time of the algorithm. We also plan to include a more thorough discussion and experimental analysis of the power consumption obtained with the proposed methodology, compared to state-of-the-art approaches.

References

[1] N. Aakvaag et al. “Timing and Power Issues in Wireless Sensor Networks: An Industrial Test Case”. In: *ICPP*. 2005, pp. 419–426.

[2] F. Cristian. “Probabilistic clock synchronization”. In: *Distributed Computing* 3.3 (1989), pp. 146–158.

[3] F. Ferrari et al. “Efficient network flooding and time synchronization with Glossy”. In: *IPSN*. 2011, pp. 73–84.

[4] Gartner. http://www.gartner.com/newsroom/id/3165317. 2015.

[5] R. Gusella and S. Zatti. “The accuracy of the clock synchronization achieved by TEMPO in Berkeley UNIX 4.3BSD”. In: *IEEE Trans. Soft. Eng.* 15.7 (1989), pp. 847–853.

[6] V. Handziski et al. “TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks”. In: *REALMAN*. 2006, pp. 63–70.

[7] J. He et al. “Time Synchronization in WSNs: A Maximum-Value-Based Consensus Approach”. In: *IEEE Trans. Aut. Cont.* 59.3 (2014), pp. 660–675.

[8] L. Lamport. “Time, Clocks, and the Ordering of Events in a Distributed System”. In: *Commun. ACM* 21.7 (July 1978), pp. 558–565.

[9] A. Leva and F. Terraneo. “Low power synchronisation in wireless sensor networks via simple feedback controllers: the FLOPSYNC scheme”. In: *ACC*. 2013, pp. 5017–5022.

[10] A. Leva et al. “High-Precision Low-Power Wireless Nodes’ Synchronization via Decentralized Control”. In: *IEEE Trans. Cont. Syst. Tech.* 24.4 (2016), pp. 1279–1293.

[11] R. Lim et al. “Time-of-Flight Aware Time Synchronization for Wireless Embedded Systems”. In: *EWSN*. 2016, pp. 149–158.

[12] G. Mao et al. “Wireless sensor network localization techniques”. In: *Comp. Netw.* 51.10 (2007), pp. 2529–2553.

[13] M. Maróti et al. “The Flooding Time Synchronization Protocol”. In: *SenSys*. 2004, pp. 39–49.

[14] D. L. Mills. *Network Time Protocol (NTP)*. Tech. rep. RFC-958. Network Working Group Report, 1985.

[15] S. Ping. *Delay Measurement Time Synchronization for Wireless Sensor Networks*. Tech. rep. IRB-TR-03-013. Intel Research Berkeley Lab, 2003.

[16] J. Polastre et al. “Telos: enabling ultra-low power wireless research”. In: *IPSN*. 2005, pp. 364–369.

[17] F. Ren et al. “Self-correcting time synchronization using reference broadcast in wireless sensor network”. In: *IEEE Wireless Comm.* (2008).

[18] T. Schmid et al. “Temperature compensated time synchronization”. In: *IEEE Embedded Systems Letters* 1.2 (2009), pp. 37–41.

[19] T. Schmid et al. “High-resolution, Low-power Time Synchronization an Oxymoron No More”. In: *IPSN*. 2010, pp. 151–161.

[20] R. Silva et al. “Mobility in wireless sensor networks – Survey and proposal”. In: *Comp. Comm.* 52 (2014), pp. 1–20.

[21] F. Sivrikaya and B. Yener. “Time synchronization in sensor networks: a survey”. In: *IEEE Network* 18.4 (2004), pp. 45–50.

[22] F. Terraneo et al. “FLOPSYNC-2: Efficient Monotonic Clock Synchronisation”. In: *RTSS*. 2014, pp. 11–20.

[23] F. Terraneo et al. “Reverse Flooding: Exploiting Radio Interference for Efficient Propagation Delay Compensation in WSN Clock Synchronization”. In: *RTSS*. 2015, pp. 175–184.

[24] F. Terraneo et al. “Demo: A High-Performance, Energy-Efficient Node for a Wide Range of WSN Applications”. In: *EWSN*. 2016, pp. 241–242.

[25] Y. C. Wu et al. “Clock Synchronization of Wireless Sensor Networks”. In: *IEEE Sign. Proc. Mag.* 28.1 (2011), pp. 124–138.

[26] S. Yoon et al. “Tiny-sync: Tight time synchronization for wireless sensor networks”. In: *ACM Trans. Sens. Netw.* (2007).