2023 31st Mediterranean Conference on Control and Automation (MED)
June 26 - 29, 2023. Limassol, Cyprus

2023 31st Mediterranean Conference on Control and Automation (MED) | 979-8-3503-1543-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/MED59994.2023.10185676

Wireless synchronisation as a control problem embedded in new-generation networked automation systems

Alberto Leva, Federico Terraneo and William Fornaciari

***Abstract*— In the present and rapidly evolving industrial *scenario*, wireless networked controls are gaining importance. This brings about new problems concerning the use of the radio channel, as well as the energy efficiency of the involved devices (often running on battery). We argue that such problems, among which a fundamental one is clock synchronisation, should be addressed by dedicated control structures embedded in the used hardware/software architecture, and that the construction of the said controls should follow strictly system-theoretical principles to the maximum extent. In this paper, building on previous experience, we present such a synchronisation solution together with a formal model for its operation, also accounting for non-idealities in the reference time base. Some experimental results are reported to support the statements made.**

I. Introduction, state of the art, contribution

This paper belongs to a long-term research aimed at making time synchronisation an embedded integral part of industrial wireless controls, instead of just a source of problems that resides in the host hardware/software architecture. In other words, our position is to not take the said architecture as an immutable entity that causes latencies and delays versus which controllers needs making resilient [9] and that in the opinion of some authors [22] can even question the applicability of well assessed control structures like the PID. On the contrary, we aim to embed into the above architecture suitable controls dedicated to synchronisation and timing, transparent to the developer of application software. Doing so allows to counteract timing-related issues at their origin, and make them practically negligible when designing the control system for the primary scope of the plant.

Numerous surveys [11], [4], [5] indicate that the interest in wireless systems is growing, both in the process control [21] and in manufacturing applications [14]. In the former case wireless architectures allow to reduce wiring that is often exposed to adverse environmental conditions, in the latter the absence of cables facilitates factory floor reconfigurations. In both cases one has to do with a shared communication medium (a limited set of radio channels where strict regulations exist about the allowable utilisation) and frequency with battery-operated devices. Hence, minimising the radio use and aiming for energy efficiency are enabling factors for the industrial acceptance of any solution. Also, tight synchronisation is vital for applications involving sampling rate optimisation [10] or event-based communication [13].

[1]A. leva, F. Terraneo and W. Fornaciari are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy, {alberto.leva, federico.terraneo, william.fornaciari}@polimi.it

As per the current technology, high-precision wireless synchronisation requires each node to (i) measure its clock error with respect to the designated reference, and (ii) correct its rate to keep the error low without requiring too frequent synchronisation actions: (i) requires to take into account non-idealities of the radio communication channel, while (ii) deals with non-idealities of the local node timebase.

For (i) feedback control solutions were proposed, but in general using established schemes instead of a tailor-made one. Examples include FBS [3] and FLOPSYNC [12], based on PI control, SCTS [17], that uses a PLL, RBS [6] and FTSP [16] that exploit linear regression to estimate the rate error — a problem on which several works set the focus [23], [2], [17]. Concerning time-dissemination, available solutions rely on either pairwise synchronisation or flooding. TPSN [8] is the most famous example the first class, nut needs a spanning tree of the network and entails a substantial transmission overhead. The second class, where FTSP [16] is the precursor and Glossy [7] the most famous scheme, strongly mitigate transmission ans thus energy overhead, but has the disadvantage that nodes do not know how the network is composed and thus – contrary to the first class, see e.g. TATS [15] – cannot compensate for propagation delays.

Our research, at present focusing on the so called "asymmetric master-slave" synchronisation problem (See Section II) but in principle more general, aims to provide an efficient solution to the issues just evidenced. In particular, in this paper we combine a rate correction [20] and a radio delay compensation scheme [19], coming to devise a way for nodes to maintain aboard them a *virtual clock* available to the running applications. Also, we investigate the effects of a non-ideal reference (master) clock, presenting a theoretical model that we assess via suitably designed experiments.

II. Problem and model

The asymmetric master-slave synchronisation problem is formulated as follows. One node is the (time) *master*, all the others are *slaves*. The master clock is the reference to which slaves must synchronise. Some slaves are in the master radio range (we name this set *hop 1*) and can receive direct communications from it. Other slaves can only hear from the master thanks to others that rebroadcast the necessary communications: depending on how many rebroadcasts are needed to reach it, such a slave will belong to hop 2, 3, and so on. Each slave takes action when the master initiates a network-wide synchronisation (sync) event. There is no slave to master communication, whence the name "asymmetric".

To obtain the model we need for control, let t_m and t_s be the master and slave time, and t indicate the "universal" or "exact" time (in fact unknown to any node, as no clock is perfect). Clocks are based on crystal oscillators built to resonate at a nominal frequency f_o at temperature T_o. However at T_o the frequency will in fact be $f_o + \delta f_o$, where δf_o is a production tolerance. Also, physiological ageing makes frequency change very slowly, typically by just a few parts per million per year [1]; we call this variation $\delta f_a(t)$. Third, as frequency depends on temperature, we add a term $\delta f_\theta(t)$, of time scale – given the size of typical devices – from tens to hundreds of seconds hence comparable to that of clock synchronisation. Finally, oscillators are subject to a short-term jitter term $\delta f_j(t)$, owing to electronic noise and component non-linearity. We thus have

$$\delta f(t) := \delta f_o + \delta f_a(t) + \delta f_\theta(t) + \delta f_j(t), \tag{1}$$

whence the $t_s \mapsto t_m$ relationship

$$t_s(t_m) = \int_0^{t_m} \frac{f(\tau)}{f_o} d\tau = \int_0^{t_m} \frac{f_o + \delta_f(\tau)}{f_o} d\tau. \tag{2}$$

According to the Network Time Protocol (NTP) notation, the error and error rate of t_s with respect to its reference t_m are respectively called *offset* and *skew*, and are defined as

$$o(t_m) := t_m - t_s(t_m), \quad s(t_m) := \frac{do(t_m)}{dt_m}, \tag{3}$$

which recalling (2) to express offset and skew as functions of t_m gives

$$t_s(t_m) = \int_0^{t_m} \big(1 - s(\tau)\big) d\tau. \tag{4}$$

For synchronisation we are interested in the value of the offset at the periodic sync events. Letting the index $k = 0, 1, \ldots$ count those events, we can define the sampled master and slave times, and the sampled offset, as

$$t_m(k) := kT, \quad t_s(k) := t_s(kT), \quad o(k) := o(kT), \tag{5}$$

and the period-cumulated skew as

$$s_T(k) := \int_{kT}^{(k+1)T} s(\tau) d\tau. \tag{6}$$

Given this, we have

$$\begin{aligned} o(k) = t_m(k) - t_s(k) = kT - \int_0^{kT} (1 - s(\tau))\, d\tau = \int_0^{kT} s(\tau) d\tau \\ = \sum_{\ell=0}^{k-1} \int_{\ell T}^{(\ell+1)T} s(\tau) d\tau = \sum_{\ell=0}^{k-1} s_T(\ell), \end{aligned} \tag{7}$$

and as a consequence of the introduced abstractions, the very simple model

$$o(k) = o(k-1) + s_T(k-1). \tag{8}$$

where the set point for $o(k)$ is clearly zero and the goal of the feedback control to design is to reject the disturbance $s_T(k)$ as effectively as possible.

III. Feedback control aboard each slave

We now describe the feedback loop that runs aboard each slave, organising the section into sensing synchronisation the error, rejecting the cumulated skew disturbance, and compensating for radio propagation delays.

A. Sensing the synchronisation error

The master sends out sync packets at a period T known network-wide (in this paper we do not discuss the network formation phase) and containing only a hop counter h, that the master sets to zero. Each slave timestamps the received packet in its own clock, and retransmits it after a small delay τ_r with h incremented by one. As τ_r is practically constant thanks to transceiver-level hardware support for timestamping and rebroadcast, all the nodes in the same hop will transmit simultaneously. This allows to exploit the constructive interference effect discussed in [7] as the basis of the Glossy flooding scheme, for disseminating the sync event through the network hops.

A peculiarity of our proposal is that the sync packet does not contain a timestamp of the master clock, because the τ_r-based flooding scheme is so deterministic that the time when the packet is sent is a reliable enough timing information. The only additional datum to compensate a slave clock is an initial offset, obtained from the master at network formation/join time via a request/response protocol.

Suppose now that a slave receives a sync packet sent at the master time kT with hop count h. The master time at the said packet arrival is

$$t_m^a(k) = kT + h\tau_r + \tau_p + j_f(k), \tag{9}$$

where τ_p is the total radio propagation time, and $j_f(k)$ the timing jitter introduced by the flooding scheme. Recalling (3) and (5), the slave time at the same arrival is conversely

$$t_s^a(k) = t_m^a(k) - o(k) = kT + h\tau_r + \tau_p + j_f(k) - o(k), \tag{10}$$

where "^{a}" denotes that this is the *actual* value, as opposite to the *expected* one $t_s^e(k)$ that the slave can compute as

$$t_s^e(k) = kT + h\tau_r. \tag{11}$$

thus measuring its synchronisation error as

$$e_\mu(k) = t_s^e(k) - t_s^a(k), \tag{12}$$

whence the sought model for a slave node clock completed with error measurement is

$$\left\{ \begin{array}{rcl} o(k) & = & o(k-1) + s_T(k-1) \\ e_\mu(k) & = & o(k) - \tau_p - j_f(k) \end{array} \right. \tag{13}$$

B. Rejecting the cumulated skew disturbance

Assume that the slave clock is running uncorrected, while feedback control acts by altering the expected arrival time for the $(k+1)$-th sync packet by a quantity $u(k)$. Clearly, if the said feedback steers the steady-state error to zero in the presence of constant skew, then $T + u$ in the slave time matches T in the master time, so that, by rejecting the disturbance, the skew (which will serve for the virtual clock

in Section III-F) is quantified as well. The above amounts to computing a corrected expected arrival time t_s^{ec} as

$$t_s^{ec}(k) = t_s^{ec}(k-1) + T + u(k-1), \quad t_s^{ec}(0) = h\tau_r, \qquad (14)$$

and yields the corrected measured error

$$\begin{aligned} e_\mu^c(k) &:= t_s^{ec}(k) - t_s^a(k) \\ &= kT + h\tau_r + \sum_{\ell=0}^{k-1} u(\ell) - \big(kT + h\tau_r + \tau_p + j_f(k) - o(k)\big) \\ &= o(k) - \tau_p - j_f(k) + \sum_{\ell=0}^{k-1} u(\ell). \end{aligned} \qquad (15)$$

Based on (15) and (7) we can thus write

$$e_\mu^c(k) = \sum_{\ell=0}^{k-1} (s_T(\ell) + u(\ell)) - \tau_p - j_f(k). \qquad (16)$$

finally obtain the model of the controlled system with sensing and actuation as

$$\begin{cases} \varepsilon_\mu^c(k) &= \varepsilon_\mu^c(k-1) + s_T(k-1) + u(k-1) \\ e_m(k) &= \varepsilon_\mu^c(k) - \tau_p - j_f(k) \end{cases} \qquad (17)$$

The model is very simple and contains no uncertainty, as this is confined in the way s_T is generated, as well as in the value of τ_p and $j_f(k)$. As such, the design of a feedback controller is quite straightforward. A possible control law comes from endowing the loop with a double integrator, since as discussed in [20] thermal stress results in ramp-like disturbances over the time scale of interest. This gives

$$\begin{aligned} u(k) \quad = \quad & 2u(k-1) - u(k-2) + 3(1-\alpha)e_m(k) \\ & -3(1-\alpha^2)e_m(k-1) + (1-\alpha^3)e_m(k-2), \end{aligned} \qquad (18)$$

where parameter $\alpha \in [0,1)$ is used to trade error convergence speed ($\alpha \to 0$) versus insensitivity to fast jitter ($\alpha \to 1$). Since the law (18) is not the contribution of this paper, we do not discuss it further and refer the reader to [20] for details.

C. Compensating for radio propagation delays

As our contribution is geared to industrial – and particularly, process and manufacturing – applications, we assume that nodes do not move (at least, not frequently with respect to the time scale of network formation). This allows us to define a *flooding graph* of the network based on the idea of "hop" as introduced above. Assuming also that radio ranges are symmetric – i.e., if node A hears B then B hears A – the result will be like the example in Figure 1.

For the purpose of this section, each node has to estimate the radio propagation delay τ_p from the master node to it. With flooding-based synchronisation, however, there is a problem. Nodes in a hop can receive packets from multiple nodes belonging to the previous one, as is the case for nodes 4 and 5 in Figure 1. Hence in general the flooding graph is not a tree, and each node has no knowledge of the structure of that graph. This prevents the use of techniques based on round-trip delay measurements, because these would require to have *one* predecessor node in the previous hop, not a *predecessor* set as in our setting (e.g., nodes 1 and 2 are the predecessor set for node 5 in Figure 1). We thus have to split the problem in two, as explained below.

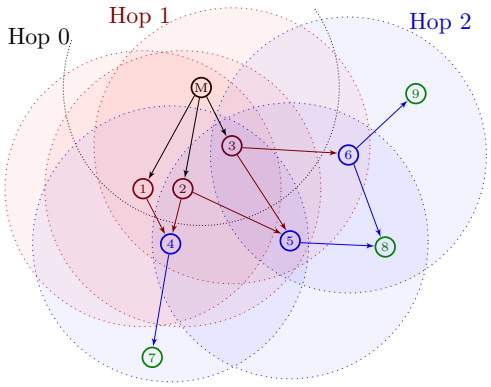

Fig. 1. Flooding graph example with node radio ranges: arrows indicate the flooding direction.

D. Last-hop delay

We first address how a node i can measure the propagation delay from its predecessor set $\mathscr{P}_i$. To this end, we exploit a side effect of the constructive interference idea: A node p in the previous hop with respect to i is either at a comparable distance with the closest ones, hence can interfere constructively with them, or not be at a comparable distance, and in this case its transmission will be shadowed by capture effects. But if a node is at a comparable distance with the closest ones, also its cumulative propagation delay from the master $\delta_{M\to p}$ will be comparable with theirs. For example, in Figure 1, $\delta_{M\to 1}$ and $\delta_{M\to 2}$ are comparable, while $\delta_{M\to 3}$ could not be, but its transmissions are not heard by node 4. This allows us to define the predecessor set $\mathscr{P}$ of node i as "the set of the closest nodes in the previous hop that are received with a comparable power (in Figure 1, $\mathscr{P}_4 = \{1,2\}$).

Once $\mathscr{P}_i$ is defined, node i can measure the last-hop propagation delay by replicating in a round-trip measurement the concurrent transmission conditions that occur during flooding. Referring again to Figure 1, node 4 can initiate the measurement by sending out a packet with its hop number minus one (i.e., 1). Under our assumptions, this packet can only be received by nodes belonging to the previous, same and subsequent hop. These check that the content matches their hop number, ignoring the packet if not. In the match case, they wait for a fixed time τ_w and then reply with another packet. In our example, the packet sent by node 4 is received by 1, 2, 3, 5 and 7; 5 and 7 ignore it, while after τ_w 1, 2 and 3 reply, but the response from 3 is shadowed by distance. Node 4 can thus measure the time difference between its packet transmission and the concurrently received responses from 1 and 2, and knowing τ_w, it can estimate the propagation delay from its predecessor set.

E. Cumulated delay

Once a node knows its last-hop delay, it needs an estimate of the sum of all such delays for the hops that separate it from

the master. We address the problem recursively, querying nodes in the previous hop for their cumulated delay from the master. However we have another problem. Since all transmissions are broadcast, we are in fact querying for its delay a *predecessore set*. Nodes in that set could have similar but not identical measurements of their delay from the master. As such, when queried, they would respond with numbers that are reasonably close but not equal.

In the absence of topology information – otherwise a node could query another specific one instead of broadcasting the query to a predecessor set – we found a solution that makes the radio channel itself fuse the received information. We do this with *ad hoc* encoding that we call *bar encoding*, yielding in intelligible packets even when some contained symbols interfere destructively. To explain, assume to have an 8-bytes packet payload, and that we encode a number between 0 and 16 as the number of consecutive 0xF (i.e., half-byte or *nibble*) blocks starting from the packet beginning, leaving the remaining blocks at 0x0. When sending two different numbers, for example 6 and 8, the two packets

```
ffffff000000000
ffffffff00000000
```

would be transmitted, and the generic received packet will look like

```
ffffffXX00000000,
```

where `X` is an unpredictable value due to the interference on the channel. The receiver can then detect that the values form the predecessor set lie in the 6–8 range, and take for example the average. We omit implementation details for space limitations.

Putting it all together and indicating with e^c_m the measured synchronisation error corrected with delay compensation, we can therefore contribute the anticipated complete model of the closed-loop system as

$$\left\{\begin{array}{rcl} u(k) &=& 2u(k-1)-u(k-2)+3(1-\alpha)\,e^c_m(k) \\ & & -3(1-\alpha^2)\,e^c_m(k-1)+(1-\alpha^3)\,e^c_m(k-2) \\ \varepsilon^c_\mu(k) &=& \varepsilon^c_\mu(k-1)+s_T(k-1)+u(k-1) \\ e_m(k) &=& \varepsilon^c_\mu(k)-\tau_p-j_f(k) \\ e^c_m(k) &=& e_m(k)+\widehat{\tau}_p \\ & =& \varepsilon^c_\mu(k)-\tau_p+\widehat{\tau}_p-j_f(k) \end{array}\right. \tag{19}$$

F. Virtual clock

To provide applications with timing information, we use a *virtual clock* approach. An algorithm is applied to the uncorrected hardware clock every time t_s needs converting into an estimate of t_m. At each sync event, the slave controller computes a new $u(k)$, which the virtual clock uses to compute the skew estimate for the period beginning at kT as

$$\widehat{s}(k) = -u(k)/T. \tag{20}$$

and the required master time estimate $\widehat{t}_m(t_s)$ is

$$\widehat{t}_m(t_s) = kT - \widehat{\tau}_p + \frac{1}{1-\widehat{s}(k)}\left[t_s - t^e_s(k)\right], \tag{21}$$

where kT is the last sync event time and $t^e_s(k)$ the slave time when the last sync packet was expected.

IV. Accounting for master clock non-idealities

As our second contribution, we now discuss what happens if the master clock does not match to the universal time t. In this case we have to admit that a sync event is triggered when the difference $\Delta c_m(k)$ between the master *clock counter* – c_m to name it – and the same counter at the last event reaches f_oT, where f_o is the nominal clock frequency. It is sensible to assume that T is an integer multiple of $1/f_o$, as c_m is an integer and any interval can only be appreciated as a multiple of the clock period. Hence the transmission condition can be written with the $=$ (not $\geq$) sign, reading

$$\Delta c_m(k) = f_oT, \tag{22}$$

whence

$$\int_{(k-1)T}^{(k-1)T+\Delta t_m(k)} \left(f_o+\delta_{fm}(\tau)\right) d\tau = f_oT. \tag{23}$$

where $\delta_{fm}(t)$ is the time-varying master clock frequency error. In (23) the unknown is $\Delta t_m(k)$, i.e., the actual interval (in the universal time t) between two sync events (we assuming f_o to be the same for all clocks, as not doing so would just uselessly complicate the computations).

Denoting by $\overline{\delta}_{fm}(k)$ the average error within the k-th integration interval in (23), we have

$$\left(f_o+\overline{\delta}_{fm}(k)\right)\Delta t_m(k) = f_oT \tag{24}$$

which yields an estimate of $\Delta t_m(k)$ as

$$\Delta t_m(k) = \frac{f_oT}{f_o+\overline{\delta}_{fm}(k)}. \tag{25}$$

Considering now a slave clock counter $c_s(k)$ of a slave, its variation $\Delta c_s(k)$ over one *actual* synchronisation period is

$$\Delta c_s(k) = c_s(k)-c_s(k-1) = \left(f_o+\overline{\delta}_{fs}(k)\right)\Delta t_m(k) \tag{26}$$

where $\overline{\delta}_{fs}(k)$ is the average slave frequency error over one period. Substituting (25) into (26), then, we get

$$\Delta c_s(k) = \frac{f_o+\overline{\delta}_{fs}(k)}{f_o+\overline{\delta}_{fm}(k)} f_oT. \tag{27}$$

As for the slave *synchronisation error*, we notice that $c_s(k)$ is the clock at to the *actual* arrival of the k-th sync packet. Since the expected time for the same packet in slave clock ticks is $c_s(k-1)+f_oT+f_ou(k-1)$, the *uncorrected* error (expected minus actual in the absence of control) – in the presence of master non-ideality, expressed in slave clock ticks, is

$$e_{ticks}(k) = c_s(k-1)+f_oT-c_s(k-1)-\Delta c_s(k) = f_oT-\Delta c_s(k), \tag{28}$$

from which we have

$$e_{ticks}(k) = \left(1-\frac{f_o+\overline{\delta}_{fs}(k)}{f_o+\overline{\delta}_{fm}(k)}\right) f_oT. \tag{29}$$

Finally, if both the master and the slave frequency error vanish, for the error in slave ticks and in time we respectively have

$$\delta_{fm} \to 0 \quad \Rightarrow \quad e_{ticks}(k) \to -\overline{\delta}_{fs}(k)T, \;\; e(k) \to -\frac{\overline{\delta}_{fs}(k)}{f_o}T,$$

$$\delta_{fs} \to 0 \quad \Rightarrow \quad e_{ticks}(k) \to \frac{f_o\overline{\delta}_{fm}(k)}{f_o+\overline{\delta}_{fm}(k)}T \approx \overline{\delta}_{fm}(k)T, \quad e(k) \to \frac{\overline{\delta}_{fm}(k)}{f_o}T \tag{30}$$

We can now draw some interesting conclusions. First, if the master clock is exact, the only errors are introduced by the slaves (which is obvious, but now emerges from equations) and increases with the sync period T. In the opposite case, fast flooding effectively prevents the master clock error from increasing while sync packets traverse the network hops. A first consequence of the above is the *necessity* of fast flooding. A second one is that, if all the slave clocks ave similar precision, any node can be the master — or elected as a new master if the previous one ceases to operate. The same would not hold true if clock precisions are different: in that case, the contributed model explicitly shows that only "precise enough" nodes would qualify.

V. Experimental results

Referring the reader to our previous papers for the operation of the FLOPSYNC synchronisation scheme – that is impossible to explain herein – as for cumulated skew rejection and delay compensation, we now present two experiments aimed at investigating the effects of a non ideal master clock, so as to verify the modelling hypotheses of Section IV and assess (i) the combination of the above components, (ii) their correct operation also in the presence of a non ideal master clock, and (iii) that the said "correctness" is adequately explained by the proposed model.

We operate in an indoor laboratory, in the presence of the usual radio traffic — hence we claim no complete coverage of the possible environment conditions, but at the same time that the addressed situation is not idealised at all. The experimental setup is made of three WandStem nodes [18] hereafter referred to as "master", "hop 1" and "hop 2"; a rubidium clock plays the role of the source for the universal time t, and the nodes synchronisation errors with respect to that time are measured by instructing all the nodes to periodically timestamp the rubidium source in their local clock. It is important to remark that the rubidium clock is a completely exogenous timebase, i.e., the master is not synchronised to it. We carried out the experiments in an indoor setting for convenience, but we did not take any action to isolate the setup from the disturbance conditions found in a laboratory — a situation that we deem a decent representative of the typical use of wireless control devices.

A. Experiment 1

In the first experimental test, the master node is subjected to thermal stress by means of an external heater. The expected result is that both the hop 1 and the hop 2 node exhibit a transient synchronisation error, that is eventually recovered by the feedback inherent to the FLOPSYNC scheme, and that since neither of those two nodes is heated, the errors have more or less the same aspect.

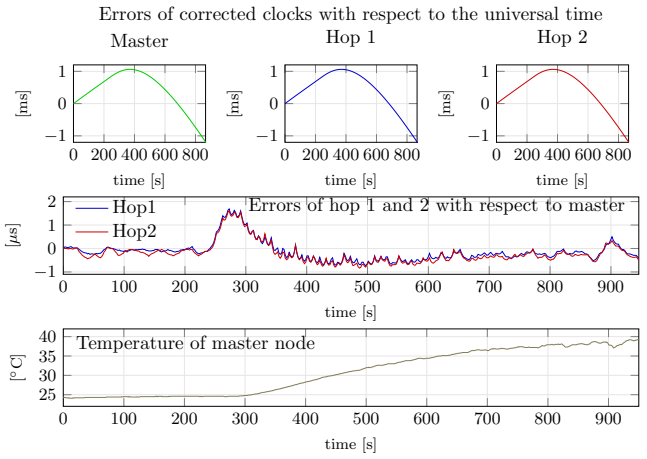

Fig. 2. Effect of thermal stress applied to the master node.

The above expectations are confirmed by the experiment outcome, reported in in Figure 2. The top row of plots contains the synchronisation errors of the three node clocks with respect to the universal time: the skew of the master node apparently changes owing to the applied thermal stress, and since the other nodes stay synchronised with the master, the said change is visible on their clocks as well. The centre plot shows the errors of hop 1 and 2 with respect to the master, that – as expected – are similar (note that the time scale, differently from the first row, is in μs). The bottom plot, for completeness, reports the temperature transient of the heated master node.

B. Experiment 2

The second experiment is similar to the first one, but the node subjected to thermal stress is the one at hop 1 instead of the master. The expected outcome is that hop 1 exhibits a transient sync error, also this time recovered by the feedback inherent to FLOPSYNC, but since this error does not prevent the flooding-based synchronisation to operate, hop 2 is practically not affected.

Figure 3, organised in the same way as Figure 2, shows the results. This time the master exhibits a quasi-constant skew with respect to the universal time, as its temperature does not vary; as a result, at the ms scale of the top row of plots, the synchronised nodes show the same behaviour. The μs scale of the centre plot, however, shows that – as expected – the heated node at hop 1 exhibits a well visible error, while the node at hop 2 is in fact not affected. Here too, for completeness, the bottom row reports the temperature of the heated node at hop 1.

Based on the shown experiments, plus several analogous others not reported here for brevity, we can conclude that the proposed modelling hypotheses are confirmed in practice.

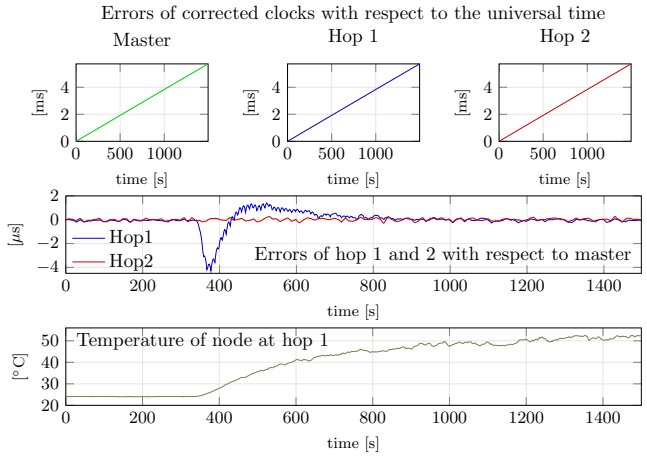

Fig. 3. Effect of thermal stress applied to the node at hop 1.

VI. Conclusions and Future Work

We presented a complete and strictly control-theoretical solution to clock synchronisation in multi-hop mesh wireless networks, combining previous partial results, and doing so we provided a comprehensive model suitable for feedback control design, and capable of formally accounting for non-idealities in the reference time base.

We believe that such results can pave the way toward embedding control-based high-precision timing primitives as integral parts of new-generation wireless control network stacks — a matter that is hard to address by just resorting to standard protocols, as the required accuracies call for a coordinated design and management of all the layers of the said stack. The only drawback – or price to pay – is a slightly increased computational complexity, well tolerable however by modern hardware.

Future research will be directed at a more comprehensive experimental campaign and more in general at pursuing the objective just stated, as well as continuing the development of the WandStem-based implementation of our ideas, both as a demonstration and as an open test platform available to the scientific and engineering community.

References

[1] *Plastic Tuning Fork Crystals [Online].* `http://www.cardinalxtal.com/uploads/files/cpfb.pdf`.

[2] M. Buevich, N. Rajagopal, and A. Rowe. Hardware assisted clock synchronization for real-time sensor networks. In *Proc. 34th IEEE Real-Time Systems Symposium*, pages 268–277, Vancouver, British Columbia, Canada, 2013.

[3] J. Chen, Q. Yu, Y. Zhang, H. Chen, and Y. Sun. Feedback-based clock synchronization in wireless sensor networks: A control theoretic approach. *IEEE Transactions on Vehicular Technology*, 59(6):2963–2973, 2010.

[4] Y. Cheng, D. Yang, H. Zhou, and H. Wang. Adopting IEEE 802.11 MAC for industrial delay-sensitive wireless control and monitoring applications: A survey. *Computer Networks*, 157:41–67, 2019.

[5] P.A.M. Devan, F.A. Hussin, R. Ibrahim, K. Bingi, and F.A. Khanday. A survey on the application of WirelessHART for industrial process monitoring and control. *Sensors*, 21(15):4951, 2021.

[6] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Operating Systems Reviews*, 36(SI):147–163, 2002.

[7] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with Glossy. In *Proc. 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 73–84, Chicago, IL, USA, 2011.

[8] S. Ganeriwal, R. Kumar, and M.B. Srivastava. Timing-sync protocol for sensor networks. In *Proc. 1st International Conference on Embedded Networked Sensor Systems*, pages 138–149, Los Angeles, CA, USA, 2003.

[9] S.M. Hassan, R. Ibrahim, N. Saad, V.S. Asirvadam, and T.D. Chung. Predictive PI controller for wireless control system with variable network delay and disturbance. In *Proc. 2nd IEEE International Symposium on Robotics and Manufacturing Automation*, pages 1–6, Ipoh, Malaysia, 2016.

[10] D. Kim, Y. Won, S. Kim, Y. Eun, K.J. Park, and K.H. Johansson. Sampling rate optimization for IEEE 802.11 wireless control systems. In *Proc. 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 87–96, Montreal, Quebec, Canada, 2019.

[11] L. Lanlan, W. Xianjv, C. Wenyan, and C.Z. Liew. Extensive survey on networked wireless control. In *Advances in Intelligent Systems and Interactive Applications: Proc. 2nd International Conference on Intelligent and Interactive Systems and Applications*, pages 634–639, Beijing, P.R. China, 2018.

[12] A. Leva and F. Terraneo. Low power synchronisation in wireless sensor networks via simple feedback controllers: The FLOPSYNC scheme. In *Proc. 2013 American Control Conference*, pages 5017–5022, Washington, DC, USA, 2013.

[13] A. Leva, F. Terraneo, and S. Seva. A multitransmission event-based architecture for energy-efficient autotuning wireless controls. *IEEE Transactions on Control Systems Technology*, 30(4):1510–1524, 2021.

[14] W. Liang, M. Zheng, J. Zhang, H. Shi, H. Yu, Y. Yang, S. Liu, W. Yang, and W. Zhao. WIA-FA and its applications to digital factory: A wireless network solution for factory automation. *Proceedings of the IEEE*, 107(6):1053–1073, 2019.

[15] R. Lim, B. Maag, and L. Thiele. Time-of-flight aware time synchronization for wireless embedded systems. In *Proc. 2016 International Conference on Embedded Wireless Systems and Networks*, pages 149–158, Graz, Austria, 2016.

[16] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi. The Flooding Time Synchronization Protocol. In *Proc. 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004.

[17] F. Ren, C. Lin, and F. Liu. Self-correcting time synchronization using reference broadcast in wireless sensor network. *IEEE Wireless Communications*, 15(4):79–85, 2008.

[18] F. Terraneo, A. Leva, and W. Fornaciari. A high-performance, energy-efficient node for a wide range of WSN applications. In *Proc. 2016 International Conference on Embedded Wireless Systems and Networks*, pages 241–242, Graz, Austria, 2016.

[19] F. Terraneo, A. Leva, S. Seva, M. Maggio, and A.V. Papadopoulos. Reverse flooding: Exploiting radio interference for efficient propagation delay compensation in WSN clock synchronization. In *Proc. IEEE Real-Time Systems Symposium*, pages 175–184, San Antonio, TX, USA, 2015.

[20] F. Terraneo, L. Rinaldi, M. Maggio, A.V. Papadopoulos, and A. Leva. FLOPSYNC-2: Efficient monotonic clock synchronisation. In *Proc. 35th IEEE Real-Time Systems Symposium*, pages 11–20, Rome, Italy, 2014.

[21] S. Thube and P. Syal. Dynamic PIDPlus controller for wireless closed loop control of lag and dead time dominant slower processes. In *Proc. 2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation*, pages 215–221, Goa, India, 2021.

[22] C.D. Tran, R. Ibrahim, V.S. Asirvadam, N. Saad, and H.S. Miya. Internal model control for industrial wireless plant using WirelessHART hardware-in-the-loop simulator. *ISA transactions*, 75:236–246, 2018.

[23] S. Yoon, C. Veerarittiphan, and M.L. Sichitiu. Tiny-sync: Tight time synchronization for wireless sensor networks. *ACM Transactions on Sensor Networks*, 3(2):8–40, 2007.