Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Consensus based synchronization of clocks to diminish the effect of clock drifts in microgrids

Miguel Parada Contzen [*]

[*] *Electric and Electronics Engineering Department, Universidad del Bío-Bío, Chile (e-mail: mparada@ubiobio.cl).*

Abstract: Inverter based microgrids implemented with more than one Grid Forming source can suffer of, so called, clock drifts when inaccurate time measurement is present. Considering that the main cause of this problem is the, slightly, different value of the time register at each clock, the synchronization approach seems like a natural choice even when additional hardware for the implementation of the sources might be required. We explore several alternatives for improving synchronization, understood as the clocks sharing the same value for their respective time register. This leads to propose a consensus based algorithm for synchronization with relatively low communication and computation requirements. The analysis is complemented with simulation examples to show the main characteristic of the different algorithms.

Keywords: Clock synchronization, Clock drifts, Synchronization, Micro-grids, Consensus

1. INTRODUCTION

For the control of microgrids, typically a three level hierarchy is proposed *e.g.* Bidram and Davoudi (2012); Guerrero et al. (2013); Palizban and Kauhaniemi (2015). At the secondary level, global control objectives are studied and the proposed solutions relay greatly on the distributed implementation of the *Grid Forming* (Rocabert et al. (2012)) power sources at the primary level. However, it has been reported that the differences between the speeds at which the local clocks are ticking impacts on the performance of the entire system, for example (Zambroni de Souza and Castilla, 2019, Ch. 6.6.3) or Kolluri et al. (2017); Castilla et al. (2018).

In this scenario, some publication propose secondary controllers focusing on diminishing the effect of the so called *clock drifts* on the control objectives. For example, in Schiffer et al. (2015, 2016) is observed, and experimentally validated, that the traditional droop controllers used to achieve active power sharing are robust to clock drifts. More recently, Kolluri et al. (2018); Martí et al. (2018); Castilla et al. (2019) also use secondary control mechanisms to overcome the impact of clock drifts.

Other references address the issue through time synchronization protocols based on, for example, Global Positioning System (GPS) signals Golsorkhi et al. (2017); Youssef et al. (2018) or Network Time Protocol (NTP) Lu et al. (2017). Accepting that accurate time measurement requires specialized hardware, particularly a dedicated additional processor which would probably increase the implementation costs, the strategy to synchronize the clocks of Grid Forming sources can be further exploited.

The history of time synchronization is lengthly, see for example Gersho and Karafin (1966). The application of leader-follower ideas for synchronization in wireless sensors networks (WSN), a very similar problem to that of inverter based microgrids, is well documented. A good summary of technologies can be found in Maróti et al. (2004). Furthermore, consensus based protocols for this kind of applications and other wireless based hardware are also available in Rentel and Kunz (2005, 2008); Maggs et al. (2012). Similarly, the papers Schenato and Gamba (2007); Schenato and Florentin (2009) propose a distributed consensus synchronization protocol in wireless sensor network that relates greatly to the cases studied here.

In this paper we explore different strategies in order to improve the synchronization of distributed clocks, to advance towards algorithms that can be applied to improve the performance of microgrids based on grid forming sources. In the following Section we introduce formally the problem of time synchronization in distributed clocks. Section 3 explains how clocks can be independently calibrated to improve their performance. This idea is then exploited in Section 4 to define a dynamical calibration protocol based on external signals as would be, for example, a GPS synchronization pulse. The following Section modifies this algorithm in order to synchronize the clocks not to an external signal, but with a "leader" clock within the network. Finally, this idea is generalized to propose a consensus-like synchronization algorithm that does not need external signals, is robust against communication failures, and can be distributively implemented.

2. DIGITAL CLOCKS IMPLEMENTATION - PROBLEM STATEMENT

We are interested in a set of N independent processors with different clock implementations in a set $\mathcal{V} = \{1, 2, \dots N\}$. Each digital clock is based on an interruption triggered operative systems (OS) and any integrated circuit with the ability of interrupting the main processor unit at every clock tick. For simplicity we assume that all the clocks in $\mathcal{V}$ work at the same nominal frequency $f > 0$, typically in $[MHz]$, with a period $T = 1/f$ in $[\mu s]$. This describes a discrete time process, that we assume occurs on a regular basis over a real or reference time measurement that we will use to compare all other clocks to.

In the i-th processor, ideally the k_i-th tick would be followed T time units later by the (k_i+1)-th tick. However, because of hardware limitations or other external causes, this is not precisely the case. Indeed the interval between ticks can be described by approximately the nominal period, but considering an error $\epsilon_i \in \mathbb{R}$ such that $|\epsilon_i| \ll 1$. This parameter depends on several factors and we assume that its exact value cannot be found. In this way, $(1+\epsilon_i)T$ time units pass before the next tick, even though the clock only acknowledge T time units. This problem becomes important as different clocks may be characterized by different $\epsilon_i \neq \epsilon_j$, and therefore the time measurement carried out by each clock differs, not only, from real or reference time, but also from the measurement of other neighbor clocks.

The physical ticks of the i-th clock can be denoted by a discrete variable $k_i \in \mathbb{N}_0$. If the register of time is recorded in a variable $\tau_i(k_i) \geq 0$, we can write for every $i \in \mathcal{V}$ a software realization of the clock as

$$\tau_i(k_i) = \tau_i(k_i - 1) + T. \tag{1}$$

Real or reference time observed at each tick is denoted by

$$t(k_i) = t(k_i - 1) + (1 + \epsilon_i)T. \tag{2}$$

Note that $\tau_i(k_i)$ is a discrete time variable that takes a new value at each tick and remain constant in between ticks. Variable $t(k_i)$, on the contrary, is a continuous variable that is observed at each tick, but its independent of them.

From here, because (1) and (2) are algebraic progressions, it is easy to verify that,

$$\tau_i(k_i) = Tk_i + \tau_{i,0} \quad \text{and} \quad t(k_i) = (1 + \epsilon_i)Tk_i + t_{i,0}.$$

The free term $\tau_{i,0} = \tau_i(0)$ corresponds to the initial value of the measured time register. We can safely assume that this term is zero. On the other hand, $t_{i,0} = t(0)$ corresponds to the instant in which the i-th clock starts running in the real time frame. If it is positive, then the measurement process starts after real time.

Solving the second equation for k_i and replacing in the first one, we obtain a linear relationship between measured time $\tau_i(k_i)$ and sampled real time $t(k_i)$:

$$\tau_i(k_i) = \frac{1}{(1 + \epsilon_i)} t(k_i) + \hat{\tau}_{i,0} \tag{3}$$

where $\hat{\tau}_{i,0} := \tau_{i,0} - \frac{1}{1+\epsilon_i} t_{i,0}$ is an unknown offset and $m_i := 1/(1 + \epsilon_i)$ the clock's skew. In the ideal case, when $\epsilon_i = 0$, the skew of (3) is one and the offset depends on the initial value of the clock and the instant in which the measurement process starts. Note that

$$m_i = 1 + \frac{1}{1 + \epsilon_i} - 1 = 1 + \frac{-\epsilon_i}{1 + \epsilon_i} \approx 1.$$

Therefore the quantity $\Delta m_i := m_i - 1 = -\epsilon_i/(1 + \epsilon_i)$ describes the skew deviation of the corresponding clock with respect to the ideal value. If $\Delta m_i > 0$, then the i-th measurement is faster than real time.

Example 1. *Consider $N = 10$ clocks working at a nominal frequency of $f = 1.7[MHz]$. For simulations purposes we assume that the parameters of these clocks are exactly known and they are such that $\epsilon_i \in [-0.1778,\ 0.2222]$. For the first $50[\mu s]$ since the reference time started, Figure 1 a) shows the evolution of the measured time by each clock. Note that the deviation of the measurements is evident, although the initial value of the clocks was relatively close to zero.* ■

3. STATIC A PRIORI CALIBRATION

As with any measurement instrument, a calibration process can be carried out on the clocks to obtain more precise values. If we have the ability to compare the measured values with respect to a reference instrument, we can estimate the value of the clocks' skews in order to have a more accurate update process.

Indeed consider that we can measure the skews of the different clocks with respect to the reference and characterize them through scalars $\hat{\epsilon}_i \approx \epsilon_i$. Instead of using the algorithm described in equation (1), we can update the register of time through the following software realization:

$$\tau_i(k_i) = \tau_i(k_i - 1) + (1 + \hat{\epsilon}_i)T. \tag{4}$$

From here, we have that $\tau_i(k_i) = (1 + \hat{\epsilon}_i)Tk_i + \tau_{i,0}$. As the physical clock is not modified, only its software realization, the ticks of the clock occur at the same rate. Therefore, expression (2) for real time sampled at every tick remains the same. Combining both we obtain,

$$\tau_i(k_i) = \frac{1 + \hat{\epsilon}_i}{1 + \epsilon_i} t(k_i) + (1 + \hat{\epsilon}_i)\hat{\tau}_{i,0}.$$

The skew of this clock with respect to real time differs from the uncontrolled case through the factor $1 + \hat{\epsilon}_i$:

$$\hat{m}_i := \frac{1 + \hat{\epsilon}_i}{1 + \epsilon_i} = 1 + \frac{\hat{\epsilon}_i - \epsilon_i}{1 + \epsilon_i} = 1 + \Delta\hat{m}_i.$$

If $|\hat{\epsilon}_i - \epsilon_i|/|\epsilon_i| < 1$, then we have that the skew deviation $|\Delta\hat{m}_i| = |\hat{\epsilon}_i - \epsilon_i|/|1 + \epsilon_i| < |\epsilon_i|/|1 + \epsilon_i| = |\Delta m_i|$ and this realization of the clock is more accurate with respect to real time than the uncontrolled case in the previous section.

Of course, as quantity ϵ_i is unknown, we cannot verify this relationship explicitly. However, when $\epsilon_i = \hat{\epsilon}_i + \delta_i$, where $\delta_i \approx 0$ is a measurement error given by the resolution of the reference instrument, we have that

$$\lim_{\delta_i \to 0} \frac{|\hat{\epsilon}_i - \epsilon_i|}{|\epsilon_i|} = \lim_{\delta_i \to 0} \frac{|\delta_i|}{|\hat{\epsilon}_i + \delta_i|} = 0 < 1.$$

Implying that a good approximation $\hat{\epsilon}_i \approx \epsilon_i$ will indeed result on a more precise clock. Note however, that the effect of the initial conditions of the clocks are not taking into account by this procedure, resulting in systematic measurements errors with respect to real time.

Example 2. *For the same clocks as in the Example 1, an estimation of the ticks period error is considered. A simulation in the exact same initial conditions is shown in Fig. 1 b). Note that the skew of the different clocks is clearly closer to real time, although they are different as longer running time would show. The different values of the clocks are then mainly explained by their different initial conditions, which imply different offsets of the virtual clocks with respect to reference time.* ■

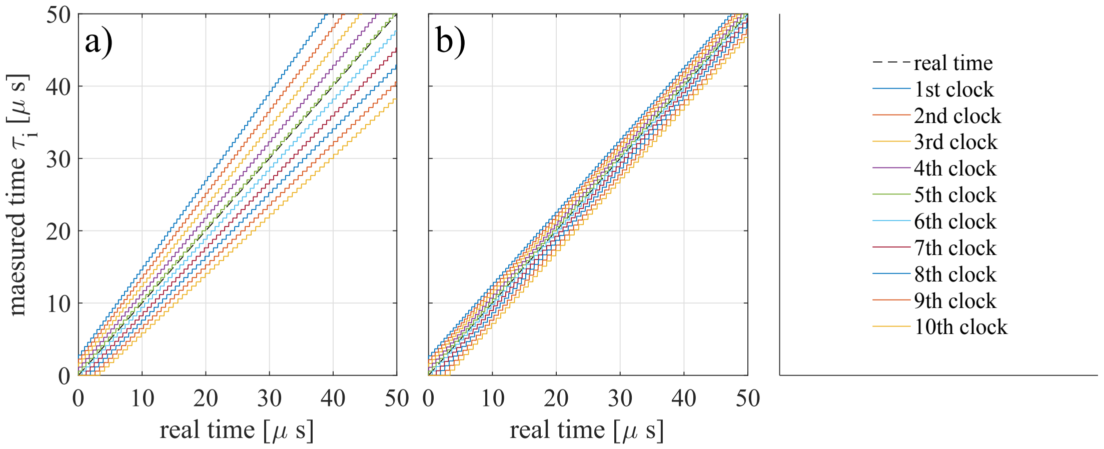

Fig. 1. a) Uncontrolled digital clocks in Example 1. b) Calibrated digital clocks in Example 2.

4. EXTERNAL CLOCK DYNAMIC CALIBRATION

Other strategy to synchronize different clocks is to rely on an external signal that periodically sends a reference time update to the different clocks. Synchronization though Global Positioning System (GPS) is perhaps the easiest possibility, *e.g.* Golsorkhi et al. (2017); Youssef et al. (2018).

In this case, we assume that the external calibration signal $\tau_g \in \mathbb{R}^+$ might be slightly behind real time, but it has the same skew. That is,

$$\tau_g = t - \Delta\tau_g,$$

with $\Delta\tau_g > 0$. Every T_g real time units, an external signal might be available for the clock at its l_i-th tick with a probability $P_i\{\hat{\tau}_g(l_i)\} > 0$. For example, a GPS synchronization signal occurs every 30$[s]$, the time delay is around $\Delta\tau_g \approx 100[ns]$, and the probability of catching the synchronization signal depends on the hardware implementation of the GPS communication device.

We denote as $\hat{\tau}_g(l_i)$ the value of the external signal available at the synchronization tick $l_i \in \mathbb{N}$, which is not necessarily exactly the same as the external signal sampled at that tick $\tau_g(l_i)$ due to quantization errors. Indeed, the value $\hat{\tau}_g(l_i) \neq \tau_g(l_i)$ was received at some point between the tick $l_i - 1$ and tick l_i and therefore it differs from the value of the external signal in a maximum of one physical clock period of $(1+\epsilon_i)T$ time units. We can then write that $\tau_g(l_i) = \hat{\tau}_g(l_i) + \delta_g(l_i)$, where the quantization error $\delta_g(l_i)$ is such that $0 \le \delta_g(l_i) \le (1+\epsilon_i)T$.

At every instant in which a new synchronization signal value was received, the clock can reset its time register to the synchronization value reported by the external signal. That is, for every tick $l_i \in \mathbb{N}$ such that a new $\hat{\tau}_g(l_i)$ is available, the clock resets to $\tau_i(l_i) = \hat{\tau}_g(l_i)$, correcting the measurement to obtain a more accurate value at least at these synchronization ticks.

Furthermore, one can also calculate the skew of the clock with respect to the previous received synchronization signal and use this calculation to obtain a more precise measurement. Assume that the synchronization signal value is available at $l_i \in \mathbb{N}$ and that the previous synchronization value was available at $h_i < l_i$. Note that $\hat{\tau}_g(l_i) - \hat{\tau}_g(h_i)$ is not necessarily equal to the external signal actualization period T_g because of the quantization errors and because some actualizations might be skipped as a result of the stochastic nature of the receiving process, which is described by the probability $P_i\{\hat{\tau}_g(l_i)\} > 0$.

For every $k_i \in \mathbb{N}$ such that $h_i < k_i < l_i$, when no new synchronization signal is available, the clock can be updated by

$$\tau_i(k_i) = \tau_i(k_i - 1) + (1 + g_i(h_i))T, \qquad (5)$$

where $g_i(h_i) \approx \epsilon_i$ is an estimation of the physical clock's deviation. Solving (5) we obtain that $\tau_i(k_i) = (1 + g_i(h_i))T(k_i - h_i) + \hat{\tau}_g(h_i)$. Similarly, real time sampled at the ticks evolves as $t(k_i) = (1+\epsilon_i)T(k_i - h_i) + t(h_i)$.

Note that because of the update policy described above, the value of the clock at the synchronization tick l_i is the one reported by the synchronization signal, which is not necessarily the same as the value predicted by (5). That is, $\tau_i(l_i) = \hat{\tau}_g(l_i) \neq \hat{\tau}_i(l_i) := (1+g_i(h_i))T(l_i - h_i) + \hat{\tau}_g(h_i)$.

With this definitions we can write that,

$$\begin{aligned}
\hat{\tau}_i(l_i) - \hat{\tau}_g(h_i) &= (1 + g_i(h_i))T(l_i - h_i) \\
&= (1 + g_i(h_i))\frac{t(l_i) - t(h_i)}{1+\epsilon_i} \\
&= \frac{1 + g_i(h_i)}{1+\epsilon_i}\left(\tau_g(l_i) - \tau_g(h_i)\right) \\
&= \frac{1 + g_i(h_i)}{1+\epsilon_i} \cdot \\
&\quad \cdot \left(\hat{\tau}_g(l_i) - \hat{\tau}_g(h_i) + \delta_g(l_i) - \delta_g(h_i)\right)
\end{aligned}$$

then, if $\delta_g(l_i) - \delta_g(h_i) \approx 0$, we can obtain

$$\epsilon_i \approx (1 + g_i(h_i))\,\frac{\hat{\tau}_i(l_i) - \hat{\tau}_g(h_i)}{\hat{\tau}_g(l_i) - \hat{\tau}_g(h_i)} - 1.$$

If the value $g_i(l_i)$ is updated to match the previous expression, a better estimation of the physical clock's deviation can be obtained. In this way, the following algorithm can be defined $\forall i \in \mathcal{V}$ at every tick $k_i \in \mathbb{N}$:

```
i) Update the time register:
       $\tau_i(k_i) = \tau_i(k_i - 1) + (1 + g_i(k_i))T$;
ii) If a new external signal, $\hat{\tau}_g(k_i)$, was
    received:
```

```
    a) Update virtual's clock skew:
       $g_i(k_i) = (1+g_i(k_i-1))\frac{\hat{\tau}_g(k_i)-\tau_{old}(k_i)}{\tau_i(k_i)-\tau_{old}(k_i)} - 1;$
    b) Reset time register:
       $\tau_i(k_i) = \hat{\tau}_g(k_i);$
    c) Overwrite last synchronization value:
       $\tau_{old}(k_i) = \hat{\tau}_g(k_i);$
```

Because it is not necessary that all clocks receive the external signal at every moment, this algorithm can be asynchronously and distributively implemented at each clock without need of communication between the clocks.

Example 3. *Consider the same clocks as in Example 1. With an external signal characterized by a synchronization period of $T_g = 5[\mu s]$ time units and a relatively high delay with respect to real time of $\Delta\tau_g = 4[\mu s]$ time units, with relatively low catching probabilities of $P_i\{\hat{\tau}_g(l_i)\} = 0.50$, $\forall i \in \mathcal{V}$. These values do not characterize any particular technology, but are chosen arbitrarily to accentuate their graphical effect on the behavior of the measurements. With $g_i(0) = \hat{\epsilon}_i$ and the same initial conditions as before, the measurements might produce the signals drawn in Fig. 2 a) depending on the stochastic process that characterizes the external signal catching process. After an initial period of adjusting, the measurements converge to the external signal time, even though it is evident that some synchronization signals are missed by some of the clocks.* ■

5. LEADER-FOLLOWER DYNAMIC CALIBRATION

In some occasions, it is not desire that the synchronization mechanism relays on external information. If the different clocks have the ability to communicate the value of their time register, a leader-follower based algorithm can be thought in order to achieve synchronization. Several technologies, *e.g.* Network Time Protocol (NTP), Timing-sync Protocol for Sensor Networks (TPSN), Reference Broadcast Synchronization (RBS), Flooding Tie Sync Protocol (FTSP) Maróti et al. (2004), etc, use similar reference-follower ideas. The NTP synchronizes each agent with a time server through statistical analysis of round-trip time. The RBS, TPSN, and FTSP are adapted from NTP to wireless sensors networks (WSN) with increasingly better capabilities.

In our case, we consider a very simple case based on the external synchronization idea. For this algorithm, we choose one of the clocks in $\mathcal{V}$ to be the "leader" who would serve as reference for every other clock in the network. That is, we consider a root clock $r \in \mathcal{V}$, that runs at its particular speed with respect to real time, and who every T_r time units broadcasts its register value, $\tau_r(k_r)$, to the other clocks which will receive this information with a probability $P_i\{\hat{\tau}_r(k_i)\}$. These clocks reset their own register to the received value $\hat{\tau}_r(k_i)$ and adjust their own skews as if the root clock was the external signal in the previous section.

The suggested algorithm can be written as follow $\forall i \in \mathcal{V}$ at every tick $k_i \in \mathbb{N}$:

```
i) Update the time register:
   $\tau_i(k_i) = \tau_i(k_i-1) + (1+g_i(k_i))T;$
ii) If the clock is not the root, $i \neq r$:
    a) If a new external signal, $\hat{\tau}_r(k_i)$, was received:
       1) Update virtual's clock skew:
          $g_i(k_i) = g_{new} := (1+g_i(k_i-1))\frac{\hat{\tau}_r(k_i)-\tau_{old}(k_i)}{\tau_i(k_i)-\tau_{old}(k_i)} - 1;$
       2) Overwrite time register:
          $\tau_i(k_i) = \hat{\tau}_r(k_i);$
       3) Overwrite last synchronization value:
          $\tau_{old}(k_i) = \hat{\tau}_r(k_i);$
iii) Else, if the clock is the root, $i = r$:
    a) If ticks counter is larger than accepted value, $c_i \geq \Delta k_r$ :
       1) Send register value to others.
       2) Reset counter:
          $c_i = 0;$
    b) Else, update counter:
       $c_i = c_i + 1;$
```

Example 4. *With the same clocks as in Example 1, we choose the less precise clock as the root. The current root register value $\tau_r(k_r)$ will be sent every $\Delta k_r = 15$ ticks, and the probability that the rest of the clocks catch this information will be assumed constant and identical as that in the Example 3, i.e. $P_i\{\hat{\tau}_r(k_i)\} = 0.50$. Fig. 2 b) shows one possible trajectory of the time registers under identical initial conditions as before. Note that after an initial period of adjustment, the clocks tend to synchronize around the value of the root clock, even though in the periods between the synchronization signals, the clocks diverge slightly from another. Observe further that at certain instances, the registers decrease their value to adjust with the upcoming data from the root. This gives the impression that during these ticks time went backwards.* ■

It is easy to think several simple modifications of the previous algorithm. For example, instead of defining a unique root node, what could be described through a directed star tree-graph $\mathcal{T}_r = (\mathcal{V}, \mathcal{E}_r)$, one could define clusters of clocks that try to synchronize with a local root, while all these roots try to synchronize with another master clock, defining a synchronization hierarchy as in TPSN. In fact, any connected directed tree graph between the N nodes can be used to describe a different synchronization protocol with distinct dynamic behaviors. Other possibilities include switching between different graphs in a round robin fashion or stochastically.

Furthermore, some minor algorithmic modifications can also be considered to avoid undesired behaviors. Particularly, instead of updating the relative error approximation $g_i(k_i)$ only with the last available information, one could define a weight $p_i \in [0,1]$, such that

$$g_i(k_i) = p_i \cdot g_i(k_i-1) + (1-p_i) \cdot g_{new}. \qquad (6)$$

In this way, the current value of the error approximation also depends on the past values, what would avoid abrupt changes between $g_i(k_i)$ and $g_i(k_i-1)$ that might result of communication problems, as quantization errors and delays, or resolution issues. High and low saturation values of the change rate of the error approximation can also be considered for the same purposes.

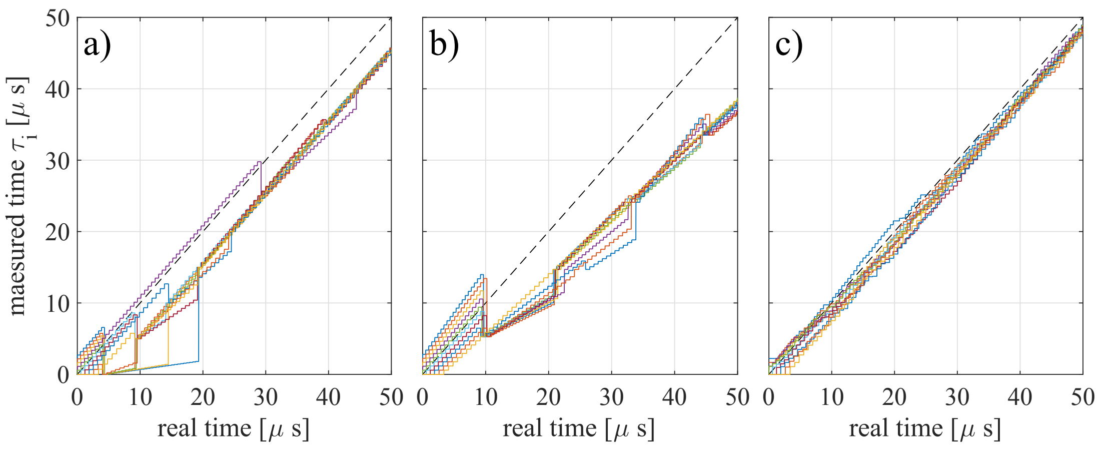

Fig. 2. a) Externally and dynamically calibrated digital clocks in Example 3. b) Leader-follower calibrated digital clocks in Example 4. c) Consensus calibrated digital clocks in Example 5.

6. CONSENSUS BASED DYNAMIC CALIBRATION

The leader-follower algorithm described in the previous section can be further generalized to define a consensus based synchronization algorithm. Consensus algorithms are well studied and are characterized by an averaging mechanism that weights the information of several nodes in order to report a common time. Some examples are ATS Schenato and Gamba (2007); Schenato and Florentin (2009), CCS Maggs et al. (2012), or CSMNS Rentel and Kunz (2005, 2008).

Consider that, besides its own register $\tau_{ii}(k_i)$, each processor $i \in \mathcal{V}$ implements also registers $\tau_{ij}(k_i)$ that are synchronized with the information that comes from the j-th clock, with $j \in \mathcal{V}\backslash\{i\}$. To synchronize these registers, we can use the same algorithm described in the previous section considering that all clocks are simultaneously the root of a different tree, and that all synchronization protocols are running in parallel. That is, the tree whose root is the i-th node serves its own register $\tau_{ii}(k_i) = \tau_{ii}(k_i - 1) + (1 + g_{ii}(k_i))T$ as the synchronization signal for all registers $\tau_{ji}(k_j)$. The probability that the i-th clock catches the signal broadcast by the j-th agent will be denoted $P_{ij}\{\hat{\tau}_{jj}(k_i)\} \geq 0$. In this way, every clock has access to a proxy of the value of each other clock, and can report time as an average of all these values.

That is, each clock $i \in \mathcal{V}$ can define an average time indicator in the following way:

$$\tau_i(k_i) = \sum_{j\in\mathcal{V}} w_{ij}\tau_{ij}(k_i),$$

where $\sum_{j\in\mathcal{V}} w_{ij} = 1$ and $w_{ij} \in [0,1]$ are scalars that weight the influence of each clock on the acknowledged time register $\tau_i(k_i)$.

Furthermore, as can be seen in Example 4, because g_{ij} are only approximations of the relative error between clocks, the synchronization of the registers with a root clock can imply that, at certain instants, the corresponding register is updated with a value that is smaller than the previous value. *i.e.* $\tau_{ij}(k_i) = \hat{\tau}_{jj}(k_i) < \tau_{ij}(k_i - 1)$. Exception rules can be considered to avoid that this is reflected in the average register $\tau_i(k_i)$.

The algorithmic modifications described in the previous section can also be considered. In particular, for $i \neq j$, we can impose that the change of the error estimation is bounded. That is,

$$|g_{ij}(k_i) - g_{ij}(k_i - 1)| \leq \delta_{ij}.$$

Defining weights $p_{ij} \in [0,1]$ we can also update the value of the error estimation considering the information of the past in a similar way as in equation (6).

With this, the following algorithm can be proposed $\forall i \in \mathcal{V}$ at every tick $k_i \in \mathbb{N}$:

```
i) For each $j \in \mathcal{V}$:
   a) Update register:
        $\tau_{ij}(k_i) = \tau_{ij}(k_i - 1) + (1 + g_{ij}(k_i))T;$
   b) If $i == j$:
      1) If ticks counter is larger than accept-
         ed value, $c_i \geq \Delta k_i$ :
           • Send register value to others.
           • Reset counter:
               $c_i = 0;$
      2) Else, update counter:
               $c_i = c_i + 1;$
   c) If $i \neq j$ and if a new external signal,
      $\hat{\tau}_{jj}(k_i)$, was received:
      1) Update virtual's clock skew:
           $g_{new} = (1 + g_{ij}(k_i - 1))\frac{\hat{\tau}_{jj}(k_i) - \tau_{ij,old}(k_i)}{\tau_{ij}(k_i) - \tau_{ij,old}(k_i)} - 1;$
           $g_{ij}(k_i) = p_{ij} \cdot g_{ij}(k_i - 1) + (1 - p_{ij}) \cdot g_{new};$
      2) Saturate virtual's clock skew:
           $g_{ij}(k_i) = \min\{g_{ij}(k_i - 1) + \delta_{ij}, g_{ij}(k_i)\};$
           $g_{ij}(k_i) = \max\{g_{ij}(k_i - 1) - \delta_{ij}, g_{ij}(k_i)\};$
      3) Overwrite time register:
           $\tau_{ij}(k_i) = \hat{\tau}_{jj}(k_i);$
      4) Overwrite last synchronization value:
           $\tau_{ij,old}(k_i) = \tau_{ij}(k_i);$
ii) Update average register:
      $\tau_i(k_i) = \max\left\{\tau_i(k_i - 1), \sum_{j\in\mathcal{V}} w_{ij}\tau_{ij}(k_i)\right\};$
```

The weights $w_{ij} \in [0,1]$ and $p_{ij} \in [0,1]$ can be chosen by the users and therefore it is possible to tune this algorithm considering particular implementation issues. For example, to weight less the proxies of clocks that are known to be less accurate or where communication problems make the received data unreliable. Furthermore, there are also several possibilities to modify the algorithm to consider, for example, clusters of clocks that synchronize with each other or external synchronization signals available at some or all nodes.

Example 5. *Consider again the same clocks as in the previous examples. Each clock will broadcast its synchronization signal every* $\Delta k_i = 15$ *ticks. For every pair* $(i,j) \in \mathcal{V} \times \mathcal{V}$*, the catching probabilities will be considered equal* $P_{ij}\{\hat{\tau}_{jj}(k_i)\} = 0.50$*, which is the same as in Example 4. The tune parameters of the algorithm will be equal for every pair with* $w_{ij} = 1/N$ *and* $p_{ij} = 0.1$*. With this, the clocks can behave like in Fig. 2 c) for identical initial conditions as before. Note that the skews are closer to one than in Example 4.* ■

7. CONCLUSION

In this paper we have developed a consensus based algorithm thought to synchronize different hardware clocks in microgrids. We begun from the premise that additional time-measuring hardware is considered for the implementation of Grid Forming power sources, and that this hardware is capable of communicate with other clocks in order share time stamps information and asynchronously update their time measurement. The resulting protocol is then robust to information loss and communication delays. Future work includes the mathematical validation of this statements and implementation in standard hardware.

REFERENCES

Bidram, A. and Davoudi, A. (2012). Hierarchical structure of microgrids control system. *IEEE Transactions on Smart Grid*, 3, 1963– 1976.

Castilla, M., Camacho, A., Martí, P., Velasco, M., and Moradi Ghahderijani, M. (2018). Impact of clock drifts on communication-free secondary control schemes for inverter-based islanded microgrids. *IEEE Transactions on Industrial Electronics*, 65, 4739– 4749.

Castilla, M., Camacho, A., Miret, J., Velasco, M., and Martí, P. (2019). Local secondary control for inverter-based islanded microgrids with accurate active power sharing under high load conditions. *IEEE Transactions on Industrial Electronics*, 66, 2529– 2539.

Gersho, A. and Karafin, J. (1966). Mutual synchronization of geographically separated oscillators. *The Bell System Technical Journal*, 45, 1689– 1704.

Golsorkhi, M., Lu, D., and Guerrero Zapata, J.M. (2017). A GPS-based decentralized control method for islanded microgrids. *IEEE Transactions on power electronics*, 32, 1615– 1625.

Guerrero, J.M., Chandorkar, M., Lee, T.L., and Loh, P.C. (2013). Advanced control architectures for intelligent microgrids, part I. *IEEE Transactions on Industrial Electronics*, 60, 1254–1262.

Kolluri, R.R., Mareels, I., Alpcan, T., Brazil, M., de Hoog, J., and Thomas, D.A. (2017). Power sharing in angle droop controlled microgrids. *IEEE Transactions on Power Systems.*

Kolluri, R.R., Mareels, I., Teixeira, C., Tong, S., Alpcan, T., Brazil, M., de Hoog, J., and Thomas, D.A. (2018). Overcoming the impact of clock drifts on power sharing for microgrids. In *IEEE Power & Energy Society General Meeting (PESGM).*

Lu, H., Zhan, L., Liu, Y., and Gao, W. (2017). A microgrid monitoring system over mobile platforms. *IEEE Transactions on Smart Grid*, 8, 749– 758.

Maggs, M.K., O'Keefe, S.G., and Thiel, D.V. (2012). Consensus clock synchronization for wireless sensor networks. *IEEE Sensors Journal*, 12, 2269– 2277.

Maróti, M., Kusy, B., Simon, G., and Lédczi, Á. (2004). The flooding time synchronization protocol. In *Conference: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004.*

Martí, P., Torres-Martínez, J., Rosero, C.X., Velasco, M., Miret, J., and Castilla, M. (2018). Analysis of the effect of clock drifts on frequency regulation and power sharing in inverter-based islanded microgrids. *IEEE Transactions on power electronics*, 33, 10363– 10379.

Palizban, O. and Kauhaniemi, K. (2015). Hierarchical control structure in microgrids with distributed generation: Island and grid-connected mode. *Renewable and Sustainable Energy Reviews*, 44, 797– 813.

Rentel, C.H. and Kunz, T. (2005). A clock-sampling mutual network time-synchronization algorithm for wireless ad hoc networks. In *Wireless Communications and Networking Conference*, 638– 644.

Rentel, C.H. and Kunz, T. (2008). A mutual network synchronization method for wireless ad hoc and sensor networks. *IEEE Transactions on Mobile Computing*, 7, 633– 646.

Rocabert, J., Luna, Á., Blaabjerg, F., and Rodríguez, P. (2012). Control of power converters in AC microgrids. *IEEE Transactions on Power Electronics*, 27.

Schenato, L. and Florentin, F. (2009). Average TimeSync: a consensus-based protocol for time synchronization in wireless sensor networks. In *Proceedings of the First IFAC Workshop on Estimation and Control of Networked Systems*, 30– 35.

Schenato, L. and Gamba, G. (2007). A distributed consensus protocol for clock synchronization in wireless sensor network. In *Proceedings of the 46 th IEEE Conference on Decision and Control*, 2289– 2294.

Schiffer, J., Hans, C.A., Kral, T., Ortega, R., and Raisch, J. (2016). Modelling, analysis and experimental validation of clock drift effects in low-inertial power systems. *IEEE Transactions on Industrial Electronics*, PP.

Schiffer, J., Ortega, R., Hans, C.A., and Raisch, J. (2015). Droop-controlled inverter-based microgrids are robust to clock drifts. In *American Control Conference*, 2341– 2346.

Youssef, T.A., Salem, A., Elsied, M., Mabwe, A.M., Abido, M.A.Y., and Mohammed, O.A. (2018). GPS synchronization of smart distributed converters for microgrid applications. *Energies*, 11, 695.

Zambroni de Souza, A.C. and Castilla, M. (eds.) (2019). *Microgrids. Design and Implementation.* Springer Nature Switzerland AG, 1 edition.