sensors

Article

Characterization, Statistical Analysis and Method Selection in the Two-Clocks Synchronization Problem for Pairwise Interconnected Sensors

Juan-Antonio Fernández-Madrigal [1,*], Angeles Navarro [2], Rafael Asenjo [2] and Ana Cruz-Martín [1]

1 Department of System Engineering and Automation, University of Málaga, 29016 Málaga, Spain; acm@uma.es
2 Department of Computer Architecture, University of Málaga, 29016 Málaga, Spain; angeles@ac.uma.es (A.N.); asenjo@uma.es (R.A.)
* Correspondence: jafernandez@uma.es

Received: 14 July 2020; Accepted: 14 August 2020; Published: 26 August 2020

Abstract: Time synchronization among sensor devices connected through non-deterministic media is a fundamental requirement for sensor fusion and other distributed tasks that need a common time reference. In many of the time synchronization methods existing in literature, the estimation of the relation between pairs of clocks is a core concept; moreover, in applications that do not have general connectivity among its devices but a simple pairwise topology, such as embedded systems, mobile robots or home automation, two-clock synchronization is actually the basic form of the time estimation problem. In these kinds of applications, especially for critical ones, not only the quality of the estimation of the relation between two clocks is important, but also the bounds the methods provide for the estimated values, and their computational effort (since many are small systems). In this paper, we characterize, with a thorough parameterization, the possible scenarios where two-clock synchronization is to be solved, and then conduct a rigorous statistical study of both scenarios and methods. The study is based on exhaustive simulations run in a super-computer. Our aim is to provide a sound basis to select the best clock synchronization algorithm depending on the application requirements and characteristics, and also to deduce which ones of these characteristics are most relevant, in general, when solving the problem. For our comparisons we have considered several representative methods for clock synchronization according to a novel taxonomy that we also propose in the paper, and in particular, a few geometrical ones that have special desirable characteristics for the two-clock problem. We illustrate the method selection procedure with practical use-cases of sensory systems where two-clock synchronization is essential.

Keywords: clock synchronization; networked sensor fusion and decisions; sensor applications

1. Introduction

In many sensorics applications, the data gathered by different devices must be fused in order to perform some task as a whole, e.g., localization [1], monitoring/surveillance [2], remote sensing [3], and many more (see for example surveys [4,5]). Time synchronization among these sensor devices, that are usually connected through non-deterministic media, is a fundamental requirement for this purpose: data from different sources should share a common notion of time, i.e., a unique and consistent reference clock, in order to be correctly merged.

There are numerous clock synchronization methods devised for general topologies of transmissions: they can be found almost half a century ago in the case of network computing [6], and decades ago in the case of wireless sensors [7,8]. All of them consist of exchanging messages among the devices until the estimates converge. The final accuracy of these estimates is subjected to the inherent uncertainty in communications [9–11]; better average accuracies can be achieved if probabilistic methods are used, but at the expense of not having hard guarantees for the worst case [12].

In the core of time synchronization methods there is still the problem of synchronizing two clocks [5]. Solving this problem involves exchanging messages between the devices in order to reach a shared meaning of time (This is internal synchronization [13], as opposed to external synchronization, where the clocks are synchronized w.r.t. some time external to the system.). Those exchanged messages should contain non-decreasing monotonic timestamps of the corresponding local clocks, to be processed in order to estimate their relation.

There exist several schemes for messaging, e.g., two-way message exchange, one-way message dissemination, or receiver-receiver synchronization [14]. Methods based on one-way messages only (Figure 1-bottom-left) are usually designed to take advantage of broadcasting [15–18], which is only available in general connected networks—sometimes broadcasting has to be implemented at the link level [12], something that requires significant support from the hardware as well. It is known that with a pure, non-reciprocal one-way message exchange scheme, it is impossible to estimate the clock offset and fixed delay precisely [19]. Therefore, the most relevant and meaningful scheme for two-clock synchronization becomes two-way exchange (Figure 1-bottom-right), that dates back at least to three decades ago [11]. Notice that in a sensory application it is not complicated to piggy-back sensory messages with the required timestamps, obtaining such a communication scheme without much modification to existing software; this fits particularly well in the case where a central device gathers sensory data from the rest periodically.

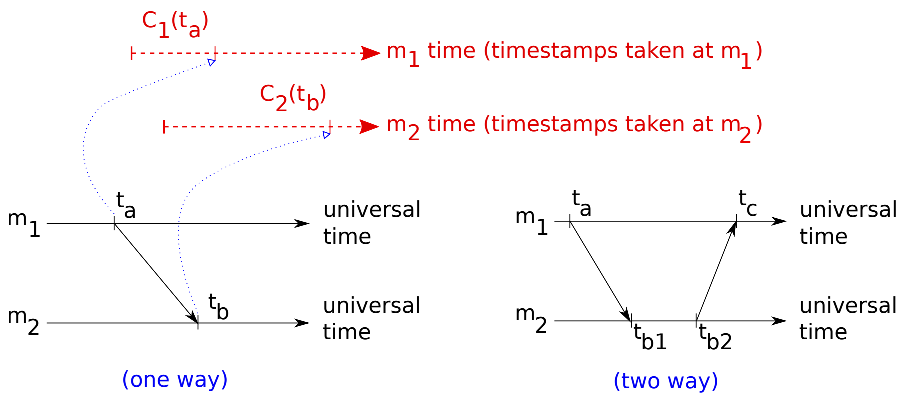


Figure 1. The most common message exchange schemes for clock synchronization: (**left**) one-way, (**right**) two-way. Note that the local time of each device (**top**) can start at any point in the universal time line and tick at any rate.

In the two-way scheme, one device (m_1) sends a message to the other (m_2) and waits for a response; both record and annotate in their messages the timestamps corresponding to the arrival and departure of measurements according to their local clocks: $C_1(t_a)$, $C_2(t_{b1})$, $C_2(t_{b2})$, and $C_1(t_c)$. After the exchange, m_1, that has gathered the four timestamps, uses them for improving its current estimation of the relation between both clocks.

That relation, although appearing in the literature under several forms and names, can be formalized with two quantities: the relative drift, that we denote as α_R, which relates the deviations of the frequencies of both clocks from their nominal values, and the relative offset, β_R, that reflects the difference between their starting times—their zero-time timestamps—conveniently adjusted by their drifts. Formally, the two-clock synchronization problem can be stated as,

$$\forall t, \quad C_2(t) = \alpha_R C_1(t) + \beta_R + \varphi_R(t), \tag{1}$$

where it can be shown that [20],

$$\alpha_R = (r_2\tau_1)/(r_1\tau_2) \tag{2}$$
$$\beta_R = (r_2/\tau_2)(t_{0,1} - t_{0,2}) \tag{3}$$
$$\varphi_R(t) \in (-r_2, (r_2\tau_1)/\tau_2 \tag{4}$$

being the following the basic parameters of the problem formulation:

- $r_1 > 0$, $r_2 > 0$ are the nominal clock periods of both devices;
- $\tau_1 > 0$, $\tau_2 > 0$ are the actual clock periods;
- $t_{0,1} \in \mathbb{R}$, $t_{0,2} \in \mathbb{R}$ are the universal time moments when both devices started their respective clocks;
- $\varphi_R(t)$ is the error produced by using discrete timestamps.

This formalization is valid as long as we assume that during the typically short time that passes between consecutive steps of any synchronization algorithm, i.e., between periodic requests from sensory data, the local clock in charge of timestamping at the i-th device is a linear function of the "universal time" $t \in \mathbb{R}$, i.e., an affine transformation $\Phi_i(t) = \frac{t-t_{0,i}}{\tau_i}$. The result of this function can be discretized to model the timestamp (also inducing $\varphi_R(t)$) in this way: $C_i(t) = r_i\lfloor\Phi(t)\rfloor, \forall t > t_{0,i}$.

Sufficient stability of τ_i in short periods of time has been recognized by several studies [21] (Typical crystal oscillators are accurate on the order of one part in 104 to 106 [12], although some authors have reported wider ranges, from −750 ppm to 750 ppm [21]. These frequency accuracies are known to be related to the possible bounds of the clock skew.), although there are algorithms that have explored more sophisticated models [18,22] and coped in a natural way with the variation of frequency (named skew) through signal processing [14]. However, these approaches do not provide hard guarantees due to their statistical nature. Note that, if required, variations in τ_i can be treated in deterministic methods without changing their fundamental design, for instance through moving windows on the sequence of two-way messages [12,23].

Far from being a simplistic view if considering generally connected systems, the two-clock synchronization problem explained so far, which is the focus of this paper, is of paramount importance in applications whose transmissions do not form arbitrary topologies but consist of a number of sensor devices transmitting data only in pairs—often wired—for instance, the sensory apparatus of a mobile robot, the set of sensors in a factory cell, remote surveillance applications, or home automation.

The primary interest when performing pairwise clock synchronization, especially for small and critical applications, lies in:

1. The quality of the estimation: accuracy and precision.
2. The existence of hard guarantees, i.e., strict bounds, in the resulting estimates, particularly important for critical systems.
3. The applicability of the methods to the special needs of these systems: limited computational cost, the shape of the transmission delays distributions, etc.
4. The number of estimated parameters that relate both clocks.

Providing hard guarantees on the estimates (point 2) and, simultaneously, a reasonable accuracy and precision in the results (point 1) is a complex matter, especially if we are also interested in the topics highlighted in point 3 above, as it happens in many pairwise applications. In this work, we aim to analyze all of these aspects from a practical perspective.

Due to the large number and diversity of existing clock synchronization methods, the first contribution of this paper is a concise taxonomy focused on the points of interest listed above, i.e., aimed at sensory applications where two-clock synchronization is sufficient. We have placed diversity of existing algorithms into that taxonomy, and, based on that, selected a number of representative ones for the subsequent study.

Due both to the diversity of algorithms and the difficulty of deducing analytically, in closed-form, their performance (some methods do offer analytical results for their bounds, but most do not have any analytical form for their average results, for instance), the second contribution of this paper is a statistical, yet rigorous, analysis of both methods and scenarios. We characterize pairwise synchronization scenarios with a number of parameters chosen to reflect a diversity of applications, ranging from clock frequencies to transmissions delays. For gathering data for the statistical study, we have carried out exhaustive simulations in a super-computer —they would have been difficult to run in a reasonable time and with the same thoroughness in conventional machines—, from which we have drawn a number of conclusions.

Our study of the simulation results includes a statistical comparison, based on ANOVA [24], of the performance of the selected methods in that variety of scenarios. We use a number of measures of performance for our comparisons, strongly related to the previously-listed aspects of the quality of the estimates, computational cost, etc. This statistical analysis has identified which method performs better in each aspect, but also which parameters of the scenario, i.e., the particular sensory system, have the strongest influence on these measures.

Finally, and using these results, we also contribute with some guidelines to select the best synchronization method for each situation, and illustrate them with a number of practical sensory applications where two-clock synchronization appears.

The main contributions of this paper are, in summary:

- A taxonomy of methods defined from the perspective of the essential aspects of two-clock synchronization applications listed above.
- A highly parameterized and detailed characterization of scenarios where two-clock synchronization can be carried out.
- A rigorous, thorough statistical comparison of a number of representative two-clock synchronization methods run in simulated environments synthetically generated from the previous characterization and evaluated with a set of performance measures that cover their applicability, quality of estimation and existence of hard guarantees.
- A study of which characteristics in the sensor scenario are statistically relevant for the two-clock synchronization problem and guidelines to select the best synchronization method for a given distributed sensor system, illustrated with some practical use cases.

The structure of the rest of the paper is as follows. Section 2 gives an overview of the two-clock synchronization problem, introduces a concise taxonomy of methods applicable to synchronize pairs of clocks—even when many were devised originally for general networks—that is focused on the most relevant aspects of pairwise systems, and justifies the choice of the representative algorithms used in further comparisons. Section 3 describes the simulation framework, the statistical results obtained with it and their use in practical cases. Finally, we highlight the main conclusions of this work and outline future lines of research.

2. Related Work and Synchronization Methods

In this section we propose a taxonomy of existing synchronization methods from the perspective of two-clock synchronization, also describing briefly those selected for the analysis of later sections.

In order to classify the existing methods that can be used for synchronizing two clocks according to the most relevant aspects for pairwise systems, explained in the introduction, we propose, in the following, a concise taxonomy (regarding the wide diversity of existing methods) formed by distinguishing whether the method:

1. produces some measure of uncertainty along with the estimates. Many methods do not produce any uncertainty measure and therefore the degree of belief in the estimate cannot be completely assessed [6,11,12,15,18,21,23,25,26]. In the case of those that use consensus in a

network, some measure of their convergence to a common clock may exist, but that does not indicate the precision of such a clock either [4,16,17].

2. provides hard guarantees on the estimates versus soft guarantees or none at all. The former are so-called deterministic; they focus on the worst-case performance, some of them giving those guarantees for both α_R and β_R [25,27,28], others only for β_R [4,11,15,23] or indirectly [13,29]. Notice that hard guarantees lead to a measure of uncertainty in the estimates: the bounds can be interpreted as the extremes of a uniform distribution. On the other hand, methods that provide soft guarantees are so-called probabilistic and focus on the average performance [22,26], having usually better expected estimates than deterministic methods (e.g., microseconds vs. milliseconds errors [28]). Finally, among the methods that do not include in their design any guarantee on the accuracy or precision of their estimates are [12,16–19,21].
3. does not impose any particular constraints in transmission delays or in other aspects of the system. For instance, some methods assume bounded delays [4,11,15], particular forms for their probability distributions [19], or knowledge about their moments [26]. Assuming also that a bounded clock drifts [6,11,15,26–28] and/or offsets [6], or very short nominal clock periods w.r.t. network delays [26], is common and reasonable. Sometimes, an useful assumption is symmetric communications [11,13,23], i.e., the same delays in both directions of transmissions.
4. provides estimates for both α_R and β_R versus only one. The methods providing both estimates can do it either directly [19,25,27,28] or indirectly [13,18]. The latter typically estimate β_R [11,15,23], and only in rare occasions restrict to α_R [21]. Notice that, if the synchronization process is repeated frequently, estimating only β_R can be enough; α_R becomes useful if synchronization is not so frequent, since it permits to predict the evolution of other devices clocks without communicating with them [27].

Items 1 and 2 are related to the quality of the estimates, while items 3 and 4 define the kind of situations where the synchronization can be used. To this concise taxonomy we could add the computational cost axis, but that can be roughly simplified by the fact that deterministic methods tend to be more computationally demanding than probabilistic ones, mainly because the latter are usually devised for networks with not very powerful computing devices. Notice, though, that the availability of hard guarantees, only available in the former, is relevant for many pairwise systems.

Since comparing and analyzing the performance of every method referenced above is out of our possibilities, we have selected for the study in this paper a reduced number of algorithms that cover well the variety defined by the proposed taxonomy. We briefly comment on them in the following.

The well known NTP [23] is relevant for its pervasivity in computers connected to the Internet, for its low-cost implementation, and for representing many approaches designed in the mid-nineties for wired networks. It consists of diverse modules, among which there is a very efficient algorithm to estimate the relative offset of two clocks, i.e., β_R. NTP does not estimate α_R, but disciplines clocks for achieving $\alpha_R \simeq 1$. The offset estimation algorithm works with a moving window for better adapting to changing drifts. It does not provide any measure of uncertainty in the estimate, but it usually achieves millisecond precision [22]. Its estimations assume symmetric communications. An evolution of NTP is PTP ("Precision Clock Synchronization Protocol", IEEE standard 1588 [30]); although this protocol can provide much better synchronization than NTP (below microseconds [22]) under the same symmetry assumption, it is used in special-purpose industrial automation and measurement networks since it requires specific hardware (e.g., high-speed Ethernet LAN) with automatic timestamping, which is prohibitive for most pairwise systems.

Berthaud's algorithm [27] is an excellent representative of the so-called "geometrical" approaches [19,25,27,28], built upon earlier frameworks (The use of convex polygons and estimation of clock functions as tangents to them was first proposed by [31], according to [28].) like [32], who exploit an intuitive spatial interpretation of the mathematical synchronization problem. Such a geometrical perspective is attractive because a linear bi-parametrical setting based on unknowns α_R and β_R, along with linear constraints existing between time measurements gathered in successive message exchanges, can serve to define polygons in a mathematical plane that provide the hard guarantees: the true values of the unknown parameters, i.e., a point, must lie within them. That point varies as successive messages shrink the polygons, ensuring convergence to the final estimate. Berthaud's algorithm is "indirect" since it defines the mentioned polygons in the plane of measured local times instead of the α_R/β_R plane. An issue with this is that once a suitable bounding region is defined on the former, the estimates for the parameters α_R and β_R must be deduced through additional mathematical transformations and therefore computational cost. Another one is that even when hard bounds are provided for the estimates, they can usually only be transformed into rectangular intervals in the α_R/β_R plane, ending in very conservative guarantees (larger uncertainties). Moreover, Berthaud's method assumes a maximum value for the relative drift and of constant drifts during estimation—the latter is easily relaxed if the method uses a moving window of messages. Its computational cost is generally high.

The geometrical algorithm presented in [20], that can be abbreviated as DGP for Direct Geometrical Pairwise synchronization, shows similar features in the estimation of α_R and β_R as that of Berthaud's, but it runs a more computationally efficient procedure due to its direct work on the α_R/β_R plane. One of the first "direct" geometrical approaches was [19], that reported a nice statistical solution in the form of maximum likelihood estimator, however without hard guarantees and under the assumption of a very particular form for the probability distribution of the communication delays (exponential), specifically aimed at wireless sensor networks. In general, communication delays, that are the composition of diverse terms [12], are likely to be produced by heavy-(right)tailed distributions such as the lognormal [33,34] or the log–logistic [35]. Thus, a less restricted approach is needed which is applicable to pairwise systems. The DGP method that we include in this study does not require any particular form for the delays distribution and includes hard guarantees along with the estimates [20].

We also include in our comparisons a variant of DGP called DGP-α1: it assumes, as an additional restriction, that $\alpha_R = 1$, i.e., that both clocks have equal drifts, and thus it only estimates β_R. This can be a reasonable assumption in systems where the pairs of devices are placed within a relatively small physical area (embedded systems) and thus share similar environmental conditions, and/or when they are mounted with similar quality components. Systems that use some clock disciplining (whose result will be the regulation of α_R around a desired value of 1), for instance, those that include NTP, can be a proper base for this variant. In all those situations, the results of this method are even more computationally efficient than DGP while providing comparable quality—achieving a performance that is quite close to the fastest, non-polygonal synchronization algorithms—while maintaining the quality of the estimates and the hard guarantees of geometrical approaches.

Finally, linear regression is a very simple and efficient approach to estimate the relation between two clocks, and it has been used by several time synchronization methods (e.g., FTSP [36]), especially in wireless sensor networks, where the computational power is limited [12]. Due to its nature, it cannot provide any guarantees or measures of uncertainty in the estimates, though it can serve for estimating both α_R and β_R and it does not impose special constraints on the system (again, it can be adapted to working with a moving window of messages to increase its robustness against changes in drifts). Among its disadvantages, it is recognized in the literature that regression alone is not enough for performing a good estimation [21]. We include it here as a simplistic reference for several measures of performance.

3. Statistical Analysis of Methods and Systems

This section is structured as follows. In Section 3.1 we describe both the simulation and the statistical frameworks devised for analyzing methods and systems in the pairwise synchronization problem statistically. Section 3.2 is devoted to the study of how the pairwise system characteristics influence the synchronization results, particularly the average performance of synchronization, and Section 3.3 to the comparison of the methods. Finally, in Section 3.4 we provide guidelines based on these results that serve to choose the best clock synchronization method for a given application, and illustrate that selection with practical examples.

3.1. Overview of the Simulation and Statistical Frameworks

We have designed a simulation framework both to have a reference true value of (α_R,β_R) with respect to which we can measure the quality of the estimates, and to easily set up a wide diversity of pairwise sensor systems (scenarios) where the methods described in Section 2 are run a large number of times for their performance to be statistically compared.

Our framework is based on a thorough parameterization of scenarios: it allows us to set a number of configuration axes at different independent positions in order to define the particular scenario to test (see Table 1). The first five axes configure the true clock relation existing between the two devices, m_1 and m_2, i.e., they completely set α_R and β_R, while the rest configure the transmission and execution characteristics of the pairwise application. For simulating round-trip times realistically we use a three-parametric log–logistic distribution, that has shown better modeling capabilities of actual round-trip times than other marginal probabilistic models in a variety of real networked systems [35–37]—it is more flexible than the log-normal, and can approach heavy-tailed, exponential-like and even Gaussian-like distributions very closely.

Table 1. Configuration axes for defining a pairwise scenario in simulation. r_i is the nominal clock period and τ_i the actual one.

Axis	Name	Explanation	Extremes
1	τ_1	True value of the period of m_1.	2:100 (ns)
2	τ_2	True value of the period of m_2.	2:100 (ns)
3	α_1	True value of m_1 clock drift, i.e., r_1/τ_1.	0.5:2
4	α_2	True value of m_2 clock drift, i.e., r_2/τ_2.	0.5:2
5	β_R	True value of β_R in seconds.	−600:600
6	EM_{rt}	Expected and minimum values for the round-trip time (total time in a two-way message exchange).	0.001:1 (s)
7	PE_{net/m_2}	Proportion of the expected and minimum transmission time within a round trip (the execution time in m_2 accounts for the rest).	0.01:0.99
8	*Symmetry*	Proportion of $m_1 \rightarrow m_2$ expected and minimum times within the transmission time ($m_2 \rightarrow m_1$ accounts for the rest).	0.5:1
9	V_{rt}	Variance in the round trip times.	4:40 (ms^2)
10	PV_{net/m_2}	Proportion of the variance of transmissions w.r.t. the variance of m_2 execution times.	0.01:0.99
11	*Gap*	Delay between consecutive two-way message exchanges.	0.001:0.1 (s)

We can independently define any continuous position in each axis within two given extremes. We have chosen for our experiments extreme values that cover a wide variety of pairwise systems (last column of Table 1). For instance, most microcontrollers for embedded systems in the market use clock frequencies in the range of tenths-hundreds of MHz, thus we have set extreme values in the axes to explore scenarios that have clocks ranging from 10 MHz to 500 MHz. Since clock drifts can vary a lot—although they are normally kept in low values—we have considered nominal period deviations from those clocks (α_i) going from a half to double. Starting times of clocks can occur very diversely; we have considered an accordingly wide range of offsets (β_R): within ± 10 min. The delays inserted between consecutive two-way message exchanges, something that the programmer may choose in wide ranges too, have been set from 1 ms to 0.1 s. The proportions defined by axes PE_{net/m_2} and PV_{net/m_2} have been set slightly away from both 0 and 1 just to avoid degenerate cases of instantaneous times, but otherwise, they cover all the possibilities. Notice that the $Symmetry$ axis goes from 0.5 to 1 instead of 0 to 1 because for the formal specification of the clock synchronization problem the asymmetry existing between communications in the $m_1 \rightarrow m_2$ and $m_2 \rightarrow m_1$ directions is not distinguishable, i.e., it does not matter where the longer delay occurs; again, the chosen range covers all the possible situations.

We also impose two additional, minor constraints on the scenario defined by the position of the configuration axes: r_i must be multiple of 1 ns (we work with timestamps that have 32 bits seconds and nanoseconds parts, which is common in general-purpose operating systems), and the minimum one-way time and the minimum execution time in m_2 are both lower-bounded by 1 μs (Although modern computers reach nanoseconds in effective—average—computation time and define data structures with such resolution for storing timestamps, that is not achieved reliably due to the high dispersion caused by inner CPU modules such as caches, branch predictors, reorder buffers, and because of the bottlenecks of I/O hardware and OS software.).

Once the configuration axes are assigned concrete values, i.e., a particular scenario is completely defined, the framework can simulate a number of two-way message exchanges between m_1 and m_2 (the first t_a is taken as $max(t_{0,1}, t_{0,2}) + Gap$), launch the synchronization algorithms on these exchange messages, and calculate a number of performance measures. A sequence of two-way message exchanges in a given scenario is there alled an experiment. We ran experiments of 1000 two-way exchange messages, which was long enough for all algorithms to settle their estimates except for rare cases of divergence.

The performance measures we use on each experiment are defined in Table 2 according to the four points of interest listed in the introduction section: $\mathcal{M}_1$ and $\mathcal{M}_2$ are related to the speed of convergence of the algorithms; $\mathcal{M}_3$ and $\mathcal{M}_4$ give the quality of their estimations; $\mathcal{M}_5$ through $\mathcal{M}_8$ measure the computational cost in different forms; finally, $\mathcal{M}_9$ through $\mathcal{M}_{12}$ measure the amount of uncertainty associated to the estimates (some algorithms do not provide such information).

For $\mathcal{M}_1$ and $\mathcal{M}_2$ we use a procedure for detecting steady-state entry that is based on a smoothed derivative of the sequence of estimates: once that the derivative (slope) falls in absolute value below a certain threshold, i.e., the estimate becomes flat enough, and never grows above the threshold again, the algorithm is considered to be steady for that estimate. The rest of the measures are calculated with the last S iterations of the algorithms within the experiment, i.e., when it is reasonable to assume that it has reached steady-state; we have set $S = 10\%$ of the total number of iterations within the experiment.

For the statistical analyses of both Sections 3.2 and 3.3 we have simulated all the scenarios resulting from possible combinations of the two extreme positions of the axes of Table 1, doing a large number of experiments (2000) in each scenario. Then, we conducted an ANOVA on the collected measures (transformed by a log operation for improving the normality of the data [24] and analyzing each measure $\mathcal{M}_i$ separately). Doing 2000 experiments per scenario allows us to obtain both high statistical power and a truthful model of the underlying probability distributions, and also to have enough data to form smaller subsets if ANOVA uses blocking, i.e., if it needs to split data. Since such a high power makes small differences in averages distinguishable for the ANOVA even when they are not actually that different, we also used $\hat{\omega}^2$ in our analysis, a measure of association strength, considering very relevant a strength $\hat{\omega}^2 > 0.1$—a threshold commonly used for representing medium/large strength in other contexts [38].

Table 2. Performance measures calculated in each experiment.

Label	Name	Explanation
$\mathcal{M}_1$	$\alpha_R{-}entry$	Number of two-way message exchanges needed to reach steady-state in the estimation of α_R.
$\mathcal{M}_2$	$\beta_R - entry$	The same in the estimation of β_R.
$\mathcal{M}_3$	$\alpha_R - error$	Average error in the estimation of α_R.
$\mathcal{M}_4$	$\beta_R - error$	Average error in the estimation of β_R.
$\mathcal{M}_5$	$vertex - mode$	Mode of the number of vertices (only in polygonal algorithms).
$\mathcal{M}_6$	$vertex - max$	Maximum number of vertices (idem).
$\mathcal{M}_7$	$comp$	Average execution time of one iteration of the algorithm (secs).
$\mathcal{M}_8$	$std - comp$	Standard deviation of the execution time of one iteration of the algorithm.
$\mathcal{M}_9$	$area$	Average steady-state area of the polygons (average area of the rectangles formed by the hard guarantees in α_R and β_R for the case of Berthaud's).
$\mathcal{M}_{10}$	$diag$	The same for the average diagonal length of polygons/rectangles.
$\mathcal{M}_{11}$	$\alpha_R - uncert$	Average steady-state uncertainty in α_R, i.e., distance between its bounds in hard guaranteed algorithms.
$\mathcal{M}_{12}$	$\beta_R - uncert$	The same for β_R.

Although so many configuration axes give us great flexibility and fine control when simulating a diversity of scenarios, this becomes an issue regarding the time for running our exhaustive simulations. Even considering only the two extreme positions for each axis, as we have explained above, we got $2^{11} = 2048$ possible combinations (scenarios); simulating one scenario out of the 2048, i.e., running 2000 experiments in it, having each experiment 1000 two-way message exchanges and launching the clock synchronization algorithms after each message exchange, takes around 1.2 h in a PC with an Xeon Intel processor @2.6GHz. This amounts to almost three and a half months of expected full-time computation for all the 2048 scenarios (We have implemented the code in Matlab and then generated self-contained executables with `mcc`. Coding directly in C or C++ would reduce the execution time by a small constant, which would not lead to a manageable total time either).

Our solution to this issue has been to work with the Picasso super-computer of the University of Málaga, that has completed all experiments for the 2048 scenarios in about 5 and a half hours, so we got a speed-up of almost 87x w.r.t. the sequential execution (This factor actually depends on the workload of the super-computer when the experiments are launched. These savings represent the case when all the experiments are run once, but we had to repeat the execution several times while debugging the code. All in all, we can estimate an approximate total reduction from more than a year of sequential work to less than one day). At the time these experiments were launched, Picasso delivered 74 TFLOPS and featured 4016 CPU cores (Intel E5-2670 processors at 2.60 GHz), 22.4 TB of RAM and 750 TB of disk. Its interconnection network is based on Infiniband QDR/FDR, and its work queue system, which allows us to submit several jobs at the same time, is based on Slurm [39]. Actually, we rely on Slurm array jobs in order to automatically enqueue all the 2048 scenarios with a single Slurm batch script, that contains:

```
#SBATCH --array=1-2048
```

The 2048 jobs are enqueued into one of the Picasso batch queues. Each job had access to its `job_id` via the `SLURM_ARRAY_TASK_ID` environment variable and therefore can simulate a specific scenario defined by that *id*.

3.2. Results (I): Influence of the Scenario on Clock Synchronization

We have first analyzed the data produced by the statistical simulations described in Section 3.1 to study which configuration axes of Table 1, i.e., which characteristics of the pairwise system scenario have a relevant influence on the measures of Table 2, unregarding the synchronization algorithm. Since some algorithms work with both $\alpha_R = 1$ and $\alpha_R \neq 1$ but others only with the former, we perform separate analyses in the cases where the sets of applicable algorithms differ.

For this study, we have designed a between-within balanced ANOVA [24] for each configuration axis of Table 1, in which all experiments are divided into two blocks (the between factor) corresponding to each of the extreme values for that axis. Within each block, we merge the results of all the applicable algorithms. Our goal is to detect whether there are statistically significant differences in the average values of the measures $\mathcal{M}_i$ of Table 2 when a fundamental characteristic of the application scenario—axis—changes from its minimum to its maximum.

The results are collected in Tables 3–5, where performance measures (first column) and configuration axes (gray shaded columns) are shown, except for those producing very insignificant results ($\hat{\omega}^2 < 0.01$). Each entry $(\mathcal{M}_i, axis_j)$ shows, separated by ':'and rounded to the closest integer in its range whenever possible, the average value of the i-th measure obtained with the j-th axis at its minimum and at its maximum extremes respectively. We have highlighted in bold those that are above the association strength threshold $\hat{\omega}^2 = 0.1$, which are considered greatly relevant, while the ones with a modest relevance, i.e., $\hat{\omega}^2 \in [0.01, 0.1)$, have been left in regular face.

Table 3. Effects of the scenario on clock synchronization—clocks. Bold values are very relevant, as explained in the main text.

$\mathcal{M}_i$	α_R	α_1	α_2	β_R
$\alpha_R - entry$ (# msgs)		225:120	120:225	
$\beta_R - entry$ (# msgs)	1	32:39	32:39	**67:9**
	$\neq 1$			**135:21**
$\alpha_R - error$ (millionths)		6:2	2:6	
$\beta_R - error$ (thousandths)	1	29:116	29:116	73:72
	$\neq 1$	117:30	30:117	74:73
comp (μs)		240.4:240.3	242:239	
area (thousandths)		1.9:0.5	0.2:2.1	
diag (thousandths)			287:647	**633:301**
$\alpha_R - uncert$ (millionths)		1386:346	346:1386	
$\beta_R - uncert$ (thousandths)	1	183:531	183:531	423:290
	$\neq 1$	647:287	287:647	**633:301**

Table 4. Effects of the scenario on clock synchronization—delays.

$\mathcal{M}_i$	α_R	EM_{rt}	$PE_{net/m2}$	$Symmetry$
$\alpha_R - entry$ (# msgs)		**128:209**	171:241	
$\beta_R - entry$ (# msgs)	1	**18:55**		
	$\neq 1$	**36:111**		
$\alpha_R - error$ (millionths)		3.8:4.4	**0.4:7.8**	
$\beta_R - error$ (thousandths)	1	**1:143**	**1:142**	**2:141**
	$\neq 1$	**1:147**	**1:146**	**2:146**
$vertex - mode$ (# vertices)		15:18	15:17	**14:18**
$vertex - max$ (# vertices)		18:21	19:21	**17:22**
$comp$ (μs)			238:243	232:249
$area$ (thousandths)		**0.02:2.29**	**0.0001:2**	
$diag$ (thousandths)		**49:884**	**8:926**	
$\alpha_R - uncert$ (millionths)		**254:1478**	**15:1717**	
$\beta_R - uncert$ (thousandths)	1	**20:693**	**6:708**	
	$\neq 1$	**49:884**	**8:926**	

Table 5. Effects of the scenario on clock synchronization—variances.

$\mathcal{M}_i$	α_R	V_{rt}	Gap
$\alpha_R - entry$ (# msgs)			189:142
$\beta_R - entry$ (# msgs)	1		
	$\neq 1$		
$\alpha_R - error$ (millionths)		2:6	**6:2**
$vertex - mode$ (# vertices)		17:15	
$vertex - max$ (# vertices)		21:19	
$area$ (thousandths)			1.2:1.1
$diag$ (thousandths)			497:436
$\alpha_R - uncert$ (millionths)			1024:708
$\beta_R - uncert$ (thousandths)	1		
	$\neq 1$		497:436

Our main findings can be explained from the content of those tables as follows:

- Estimating the relative drift of both clocks (α_R) is faster (converges after fewer message exchanges) when round-trip times are short, i.e., when both communications and execution times in m_2 are short.
- Estimating the relative offset of the clocks (β_R) is faster if $\beta_R > 0$, i.e., when m_1 starts its clock after m_2 (or, equivalently, if $t_{0,2} < t_{0,1}$), and also when round-trip times are short.
- The error estimating the relative drift (α_R) is smaller in scenarios where the expected delay in communications is shorter than the expected execution time in m_2 (i.e., when PE_{net/m_2} is small), and also when the gap between consecutive message exchanges is longer. Notice that the second

condition, in principle, can be manipulated in any algorithm, since the gap is defined by the program in charge of message exchanges, although at the cost of needing a longer time to converge when it is increased.

- The error in estimating the relative offset (β_R) is smaller in three situations: when the round-trip times are shorter, when the communications delays are shorter than the execution times in m_2, and when the communications are symmetric in both $m_1 \rightarrow m_2$ and $m_2 \rightarrow m_1$ directions. Devices connected point-to-point with a high-speed full-duplex link, for instance, are likely to satisfy all of these.
- Differences detected in execution times of the algorithms when exposed to different scenarios are not highly significant, but in the case of geometrical methods, they are slightly more efficient in the number of polygon vertices in symmetric communication scenarios. Another result is that the number of vertices is kept around tenths, corroborating a hypothesis stated in [20].
- The total amount of uncertainty in geometrical algorithms, either measured as polygon areas or diagonals ($\mathcal{M}_9, area$ and $\mathcal{M}_{10}, diag$), and also the isolated amount of uncertainty in α_R and β_R (which is not only produced by the geometrical algorithms), or, in other words, the inaccuracy of the algorithms, is smaller in scenarios with shorter round-trip times and with faster communications compared to the execution times in m_2.
- The clock drifts in m_1 and m_2, and the amount of uncertainty in the round-trip times, only produce modest effects in the synchronization problem: nominal periods in m_1 longer than the actual ones are of benefit for the convergence speed in the estimate of α_R, for the error in that estimate, for the error in the estimate of β_R in the case $\alpha_R \neq 1$, and for the amount of uncertainty in the estimation of both α_R and β_R if $\alpha_R \neq 1$; on the contrary, nominal periods of m_2 shorter than the actual ones are good for obtaining faster convergence in the estimate of α_R, for the error in that estimate, for the error in the estimate of β_R, and for the amount of uncertainty in both estimates; last but not least, deterministic scenarios, where variance in the round-trip times is smaller (including communications and execution times), produce smaller errors in the estimate of α_R.
- None of the explored scenarios produce a detectable difference in the dispersion of execution times of the algorithms ($\mathcal{M}_8$, $std - comp$), thus this is not shown in the tables. That jitter is not expected to be significantly influenced by the scenario, only by the CPU and OS executing the algorithms.
- Finally, τ_1, τ_2 (true periods of clocks) and $PV_{net/m2}$ (proportion of variance in transmission delays w.r.t. the one in execution times in m_2) have no detectable influence in the process of clock synchronization.

3.3. Results (II): Comparative Performance of Synchronization Methods

In this section, we show the results obtained in the simulated experiments described in Section 3.1 when comparing the methods for clock synchronization to each other. We are interested first in ordering the methods according to their average performance measures.

For that purpose, we have made a between-within balanced ANOVA design for each measure defined in Table 2. This time we have used the clock synchronization methods as treatments (within) and split the experiments data into blocks (between) aimed at optimising the unimodality of the data and therefore the reliability of the ANOVA. Doing this clustering, i.e., finding the blocks that exhibit optimal unimodality, involves two problems: potentially searching among all possible subsets of scenarios and calculating in each one some measure of unimodality.

Regarding the latter, we have used the Hartigan's DIP statistic and its significance for the empirical pdf [40], for which public implementations exist [41]; the significance level of that test has been set to 0.001 to minimize the number of false-positive (apparent but not actual unimodal blocks).

The former subproblem (searching among all possible subsets of scenarios) is harder: the number of subsets of a given set is exponential in the cardinal of the latter, and, in this case, if we use as a basis for blocking the axes positions in the experiments, intractable (2^{2048}). For addressing that issue, we have implemented a heuristic search for optimizing the unimodality of the blocks that works like this: start with blocks produced by combinations of only the configuration axes that have some significant effect in the given performance measure (these axes have been identified in Section 3.2) and then perform hill-climbing by adding new varying axes, therefore increasing the number of blocks until no better unimodality is obtained for all of them. In practice, this method has worked very well except in a few cases where it stuck in local minima with poor unimodality; there we have mixed the automatic heuristic with an interactive, manually guided search.

The between-within designs formed in that way have passed two different statistical analyses to assess an ordering among the clock synchronization methods. The first one is a Tukey's comparison [42], that, provided that an ANOVA test obtains strong enough results (high $\hat{\omega}^2$), deduces a statistically significant pairwise ordering. The second one consists of comparing each method results in each individual experiment to the results of the other methods, and tallying the proportion of times that the former is better (this has been called Common Language Effect Size(CLES) [24]). The CLES ordering can be interesting for particular applications, and it provides an alternative when the strength of the ANOVA results is not enough.

Notice that the number of possible orderings that can arise when comparing n synchronization methods to each other ranges from 2 (if $n = 2$) to 120 ($n = 5$), which is too much information that should be condensed to be properly understood. For simplifying those results, Table 6 shows only the best method in the ANOVA ordering, and Table 7 the best one in the CLES ordering. Algorithms are (gray shaded columns): DGP, Berthaud's (B's), DGP-α1, NTP, and regression (Reg). Each entry $(\mathcal{M}_i, alg_j)$ in the tables shows the percentage of blocks in the design where the j-th algorithm ranks the absolute best in the i-th measure. Notice that a method not ranking the best does not mean that it cannot be the best for particular scenarios, as we will see in more detail in the next section.

In Table 6 all the results except the $\alpha_R - entry$ measure for the B's method have high significance with association strength greater than or equal to the 0.1 threshold. In both tables we use a light orange background for those algorithms that are the best in more than the 50% of the blocks; notice that the highlighted cells are mostly the same in both tables. Algorithms that cannot run or are not meaningful for a measure or α_R class have been left empty.

Table 6. Orderings of the algorithms (ANOVA).

$\mathcal{M}_i$	α_R	DGP	B's	DGP-α1	NTP	Reg
$\alpha_R - entry$		37.5	0			62.5
$\beta_R - entry$	1	18.8	0	12.5	62.5	6.3
	$\neq 1$	75	0			25
$\alpha_R - error$		55.5	0.8			43.8
$\beta_R - error$	1	6.3	2.3	79	0	12.5
	$\neq 1$	62.5	12.1			25.4
$vertex - mode$		100	0			
$vertex - max$		100	0			
$comp$		0	0	0	100	0
$std - comp$		0	0	0	100	0
$area$		100	0			
$diag$		100	0			
$\alpha_R - uncert$		100	0			
$\beta_R - uncert$	1	0	0	93.8	6.3	
	$\neq 1$	100	0			

Table 7. Orderings of the algorithms (CLES).

$\mathcal{M}_i$	α_R	DGP	B's	DGP-α1	NTP	Reg
$\alpha_R - entry$		25	0			75
$\beta_R - entry$	1	65.6	0	12.5	21.9	0
	$\neq 1$	59.4	0			40.62
$\alpha_R - error$		55.5	0.8			43.8
$\beta_R - error$	1	6.3	10.9	68	0	14.8
	$\neq 1$	57.4	16.4			26.2
$vertex - mode$		100	0			
$vertex - max$		100	0			
$comp$		0	0	100	0	0
$std - comp$		0	0	0	100	0
$area$		100	0			
$diag$		100	0			
$\alpha_R - uncert$		100	0			
$\beta_R - uncert$	1	8.1	0	91.9	0	
	$\neq 1$	100	0			

The main results of these tables can be explained as follows:

- In general, methods DGP and DGP-α1 ranked first in a large number of scenarios and measures. Particularly, they are the absolute best (the best in all scenarios, with strong statistical significance) concerning the amount of uncertainty in the estimates, regardless of how we measure that uncertainty. They are also the best considering the estimate error for the majority of scenarios. In the estimate error of β_R when $\alpha_R = 1$, all the algorithms find special difficulties to outperform the others.
- B's method did not ranked first in any measure, although, as we will see in the next section, it is the best in particular kinds of scenarios. Concretely, DGP outranks it with strong statistical significance when measuring the complexity of the constructed polygons in all orderings and cases (scenarios), i.e., in the computational effort of the geometrical paradigm.
- NTP ranks first, with strong statistical significance and in a majority of scenarios, in the speed of convergence (# of message exchanges to reach steady state) in the estimate of β_R, a measure where, if $\alpha_R = 1$, all algorithms find difficulties to outperform the rest. In those cases where it cannot be used (i.e., when $\alpha_R \neq 1$), it is the DGP method the one of choice. The smallest expected execution time ($\mathcal{M}_7$, i.e., $comp$) corresponds to NTP with strong statistical significance in all scenarios, although when comparing each scenario individually (CLES) the best is the DGP-α1 variation. NTP always has the smallest jitter in execution times ($std - comp$).
- Regression only stands out in the speed of convergence to the estimate of α_R.

3.4. Results (III): Guidelines for Selecting Methods

Tables 6 and 7 provide a simplified view of the results concerning method comparisons. It would also be interesting to know, for particular scenarios (i.e., for practical applications of the two-clock synchronization problem), which method is best suited, and also to grasp some general idea of the magnitudes of the performance measures that the methods produce (The average performance values are only of secondary importance in this paper, though, since they come from the particular extreme values in configuration axes of Table 1, that have been selected for covering a wide variety of scenarios and not for reflecting what happens in any particular one).

Unfortunately, in our experiments we have 2048 different scenarios, because we have simulated all the possible combinations of extreme positions in the configuration axes of Table 1, and listing all the performance measures of each method in each scenario would be impractical; a more concise account of the results is required for reaching meaningful, generalizable conclusions. Therefore, instead of studying particular scenarios, we have partitioned all of them into clusters (in a reduced number, ranging from 5 to 8) for each performance measure $\mathcal{M}_i$. Each cluster contains scenarios for which the synchronization methods produce a distinctive pattern in the magnitudes of $\mathcal{M}_i$, what we have called a behaviour. After clustering, it is much easier and useful to draw meaningful conclusions for the behaviour observed in each cluster than considering individual scenarios.

In order to facilitate the identification of the scenarios contained in a given cluster, and thus to identify the pertinence of that behaviour for a given practical application where two-clock synchronization is to be solved, clusters have been labelled as described in the following. Each particular scenario in our experiments corresponds to a combination of extreme positions of the configuration axes, that can be either at their minimum values ('$-$' or '0') or at their maximum ('+' or '1'), therefore it can be denoted as a unique binary number of 11 bits, one-bit position per configuration axis for an easier interpretation of the most important axes extreme positions. The reader can refer to Table 8 (where the columns labelled with 'UC*', meaning Use Case, will be explained in Section 3.4). A given cluster of scenarios can be denoted as a number using three digits: '+' if all the scenarios in the cluster have that axis at the maximum value, '-' if all have it at the minimum value, and '*', if the axis has different values in different scenarios of the cluster. (Obtaining this three-digit number for a cluster of scenarios denoted as binary numbers can be done automatically if we cast it as the problem of finding the minimal simplification of a propositional logic formula. We have used for that purpose the exact Quine–McCluskey algorithm [43], particularly its efficient implementation in [44]). Such specification is simple enough for identifying the scenarios contained into the cluster concisely.

Table 8. Interpretation of axes and use cases of method selection. $\neq$ stands for $\alpha_R \neq 1$, and $=$ for $\alpha_R = 1$.

Axis	Interpretation of the Axis Being '+' ('-' Would Be the Opposite)	UC1	UC2	UC3	UC4
3 (4)	Clock freq. in m_1 (m_2) higher than nominal.	$\neq$	$\neq$	$\neq$	$=$
5	m_2 clock started before m_1 clock.		+		+
6	Long round trips (transmissions + m_2 exec.).	-	-	-	+
7	Transmissions slower than m_2 exec.	-	-	+	+
8	High asimmetry in transmission times.	-	-	-	+
9	Large uncert. (variance) in round trips.	-	+	-	+
10	Uncert. in transmissions larger than in m_2 exec.		-	-	+
11	Long gaps between consecutive exchanges.	*	+	-	+

The partitioning of the 2048 scenarios into clusters for a given performance measure $\mathcal{M}_i$ is hard, since the total number of possible clusterings in a set of 2048 elements is too large (Given by the Bell number [45]). Therefore, finding clusters that contain scenarios exhibiting a distinctive behaviour in the performance $\mathcal{M}_i$ can only be done approximately. In our case, we have used an incremental procedure, starting by considering as a preliminary cluster each of the blocks of scenarios already found when optimizing unimodality in the analysis of the previous section; the rationale for this is that unimodality is clearly a sign of a compact, unique behaviour in a performance measure. Then, we have merged those preliminary clusters to each other as long as the result continue exhibiting, through visual inspection, similar behaviours in the magnitudes of the measure, until the smallest possible amount of clusters are left, reaching the number of 5–8 clusters commented before (Notice that merging blocks used in the results of Tables 6 and 7 may hide orderings that were shown there. Concretely, this happens with the methods and measures collected in cells $(\mathcal{M}_2, DGP)$, $(\mathcal{M}_4, B's)$, $(\mathcal{M}_{12}, DGP)$ and $(\mathcal{M}_{12}, NTP)$).

Our procedure has produced the clusterings shown in the figures reported in Appendix A (Figures A1–A10): each figure corresponds to the clustering obtained on a given performance measure $\mathcal{M}_i$, and shows the patterns that we have found, arranged in rows; each pattern is a given cluster or behaviour exhibited by all the synchronization methods in $\mathcal{M}_i$. In Figure 2 we replicate the first of those Figure A1, corresponding to the $\mathcal{M}_1$ ($\alpha_R - entry$) measure, for the reader's convenience in order to illustrate their content.

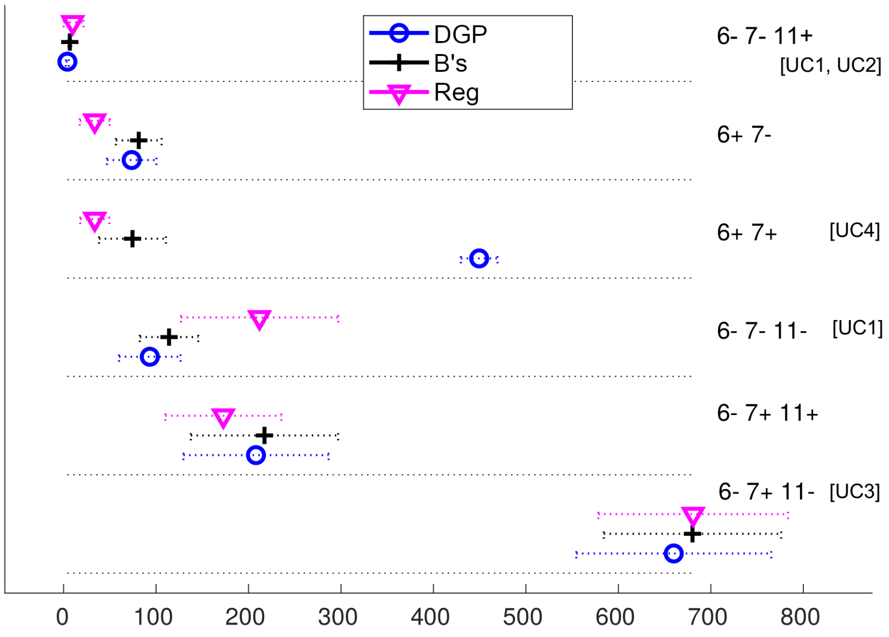


Figure 2. On the right side, from top to bottom, the six distinctive behaviours—clusters of scenarios—found for the $\mathcal{M}_1$ ($\alpha_R - entry$) measure, arranged in six rows. The abscissa is the magnitude of that measure (number of iterations of the methods before entering steady-state). Each method within a behaviour has a point marker with its average value in the measure and $\pm 1\sigma$ intervals around it (only illustrative of the dispersion, due to skewness). The lower the values, the better. The behaviours are labelled with the sets of scenarios that produce them (axes at their minimum extreme ($-$) or at their maximum (+) in all scenarios of the cluster). In square brackets, we also indicate the practical use cases that match the behaviours, from the ones described in Section 3.4.

These figures can be used as a guide to select the best two-clock synchronization algorithm for a given practical application in a quite direct manner: if the application configuration axes are at positions that match any of the behaviours shown in the figures for the performance measures of interest, the method of choice should be similar to the one with the best performance in those figures.

Use Cases

As practical examples of the use of Figures A1–A10 to select a synchronization method for a given application, we describe here a few common use cases (UC). The last columns of Table 8 summarizes them by indicating the position of the configuration axes that a scenario should have; in that table '$\neq$' abbreviates '$3 \neq 4$' (i.e., $\alpha_r \neq 1$) and '$=$' abbreviates '$3 = 4$' (i.e., $\alpha_R = 1$); blank entries are not at any extreme in the corresponding UC; '*' means that the extreme can be freely set at least in some particular instantiation of the UC.

Notice that the use cases we have considered here are very generic ones for the sole purpose of illustrating the utility of our statistical results (although their text can be translated directly to precise positions in the configuration axes of our framework); any particular application can vary in the indicated axes extremes. For completing the possibilities of use of our study in real cases, we finish this section with a particular use case in robotics.

- **[UC1] Embedded application**. This case of use represents a central embedded m_1 device (e.g., an SBC with a microprocessor) communicating with a number of slower microcontrollers that manage the rest of sensors (several m_2), all connected through a PCB short-length bus (e.g., I2C) and being powered up nearly simultaneously, with no provision for clock disciplining and not much computer power either. The gap between consecutive two-way message exchanges can be varied depending on the particular use of the system.
- **[UC2] Mobile robot**. A mobile robot is usually equipped with an on-board laptop with not-too-high computational power that gathers information from several sensors through USB connections (no network in between), being that all of these sensors are controlled by CPUs much slower than the one in the laptop. The sensoric hardware is usually powered up before the laptop is. Transmission times have a larger variance than in UC1 when we consider the non-real-time software in the laptop. The period of sensor/actuation sampling in a service mobile robot is usually of tenths of seconds, which limits the possibility of reducing the gap between consecutive two-way message exchanges beyond that. Since an important concern in mobile robots is the duration of their battery, computational savings are usually desirable.
- **[UC3] Domotic application**. Home automation hardware and software are particularly diverse since the market is still very fragmented [46]. Here we consider a very generic case where relatively fast and deterministic transmissions exist between the central hub/gateway, that we consider having a relatively high computational power, and the sensor devices, but those devices are faster in computing than transmitting. We also consider a short sampling period of the sensors and therefore a short gap between consecutive two-way message exchanges.
- **[UC4] Remote surveillance**. In remote surveillance a number of sensory devices are placed at a considerable distance from the central station, a modern computer capable of handling graphical interfaces, that samples them periodically. We assume an Ethernet connection that uses a general network to access the devices, i.e., routing with possible different paths in both directions, which is likely to produce long transmission delays with large variance and asymmetry. It is reasonable to consider the sensors to be powered up before the central station starts to work, and, since the transmissions are relatively slow, the gap between consecutive two-way message exchanges cannot be short. Unlike the previous use cases, all devices being a part of a network provides them with the possibility of disciplining their clocks, i.e., having $\alpha_R \simeq 1$.

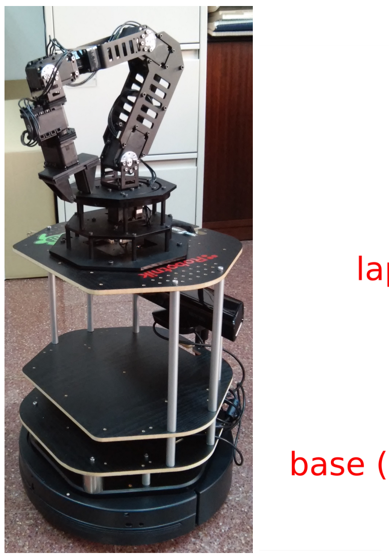
After consulting Figures A1–A10 looking for behaviours that match the UC configurations of axes shown in Table 8, the best method depending on which measure $\mathcal{M}_i$ we consider more important is shown in Table 9. In that table we have indicated the recommended method on average and, in parentheses, which alternative should be selected, if available, that has a better average computational cost according to Figure A7. We have also added a variant of UC2 in that table (UC2') that assumes some clock discipline is performed onboard the robot, i.e., $\alpha_R \simeq 1$. In that case, both the DGP-α1 and the NTP methods produce the same $\beta_R - entry$ average results, with NTP having a slightly better average execution time.

As a more detailed example of the use of our statistical analyses for selecting a two-clock synchronization method, let us consider a particular case in robotics taken from the general paradigm of UC2: the CRUMB robot, a mobile system we employ in our research based on a Kobuki platform augmented with a manipulator and other sensors (Figure 3). This system exhibits the star-like transmission topology we have focused on in this paper, being that the laptop is a central device in charge of gathering data from multiple sensors and making decisions (navigation, mapping, pick-and-delivery tasks, ...).

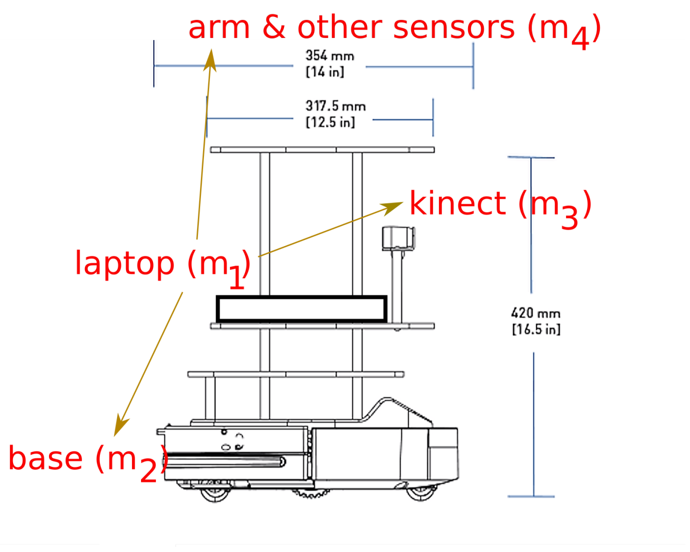
Table 9. Recommended synchronization methods for the use cases in Section 3.4.

UC	α_R *Entry*	β_R *Entry*	α_R *Error*	β_R *Error*	α_R *Uncert.*	β_R *Uncert.*
1	D (R)	D (R)	B (D)	D (R)	D	D
2	D (R)	D (R)	B (D)	D (R)	D	D
2′	D (R)	D1(N)/N	B (D)	D1 (N)	D	D1 (N)
3	D (R)	D (R)	B (D) R (D)	D (R)	D	D
4	R	R (N)	R (D)	D (D1) D1 (N)	D	D1 (N)

(D)—DGP; (D1)—DGP-α1; (B)—Berthaud's; (N)—NTP; (R)—Regression.



(a)



(b)

Figure 3. (**a**) CRUMB: a Turtlebot-2 robot composed of a Kobuki mobile base, a manipulator mounted on top and room for placing a laptop and other possible sensors [47]. (**b**) Star-like topology of the data transmissions in this robot, where the laptop acts as the central device.

For illustrating the selection of a method for this system, we can consider the laptop (m_1) and the base (m_2), since the latter contains multiple sensory devices itself and the former can provide the data from the others. In that configuration, m_1 is an Intel Celeron @2.16GHz, which would yield a nominal clock resolution of 0.43 ns; however, since the OS (Linux) only provides nanosecond resolution, we take $r_1 = 1$ ns instead. That laptop is usually connected to the Internet and runs a clock disciplining software (embedded in the NTP protocol), thus we can assume $\alpha_1 = r_1/\tau_1 \simeq 1 \rightarrow \tau_1 = 1$ ns. On the other side, m_2 is an embedded computer that the manufacturer assures it does sensor sampling at a rate of 50 Hz; in particular, it is an STM32 256K high-performance microcontroller of a model undisclosed by the manufacturer but that corresponds to a clock of @72MHz in the information publicly available for that minimal specifications and family, which means that $r_2 = 14$ ns. Since this microcontroller has no clock disciplining, we can assume some drift for the clock resolution, e.g., $\alpha_2 \simeq 0.9 \rightarrow \tau_2 = 16$ ns.

As for the transmissions, m_1 and m_2 are connected through USB 2.0, that provides 60 MB/s. The data transmitted in the $m_2 \rightarrow m_1$ direction consists mainly of the information read from a bumper, a cliff detector, a wheel drop detector, a gyro sensor and the battery level; all in all, this amounts to about 20 bytes, that could be physically transmitted in 318 ns; adding a few bytes in the other direction and a minimal amount of m_1 software processing of the data, it would be typical an expected round trip time (E_{rt}) of a few hundreds of microseconds plus the time spent in m_2 for gathering the sensory data. Due to the processing in m_1, there would be some variance in the round trip time (V_{rt}; on the other hand, the microcontroller can be assumed to be practically deterministic); taking into account the tenths of millisecond resolution of task preemption in a standard Linux OS, we can consider that variance to be similar to the one of a uniform distribution for the $[0, 10]$ ms interval, i.e., $V_{rt} \simeq 8$ μs^2.

Transmissions between this pair of devices are quite symmetrical if we consider the few information going in any of the directions. The proportion of time spent in the transmissions plus the m_1 processing time must also be quite short compared to the time spent in the base microcontroller to gather the sensory data ($1/50s = 20$ ms); that proportion (P_{net/m_2}) would then be around 0.01. Finally, the *gap* between consecutive exchange messages can be established as for any basic mobile robot operations, where it is typical to process sensory information every 100–200 ms, and we can also assume that the time between the base being started up ($t_{0.2} = 0$ s) and the laptop finishing its loading of the OS ($t_{0.1}$) will be a few seconds, e.g., 10 s, which means that $\beta_R = 0.9 \cdot 10 = -9$. Notice that if we would like to consider each of the sensors of the base individually (as a separate m_i), but using the same microcontroller, the only variation in all these specifications would be $t_{0.2}$ to represent the time offset between the sequential sensor data gathering.

In summary, our particular robotic system can be mapped into the positions of the axes shown in Table 10. The last two columns in Table 10 indicate, in a linear scale bounded to 0 to 1, the normalized position of the system in each axis if we take into consideration the bounds for the extremes given in Table 1, and the closest extreme to that position ('-' meaning '0', '+' meaning '1'), respectively.

The data collected in the last column of Table 10 can now be directly used to select the best two-clock synchronization method for the system, by looking for those closest extremes in Figures A1–A10. As explained in the introduction, in a pairwise system we are interested in:

Table 10. Configuration axes positions of the CRUMB robot example.

Axis	Name	Value	Position [0...1]	Closest Extreme
1	τ_1	1 ns	0	1-
2	τ_2	16 ns	0.16	2-
3	α_1	$\simeq$1	0.33	3-
4	α_2	0.9	0.27	4-
5	β_R	-9 s	0.49	5-/+
6	EM_{rt}	20,200 μs	0.0192	6-
7	PE_{net/m_2}	0.01	0	7-
8	*Symmetry*	1	1	8+
9	V_{rt}	8 μs^2	0	9-
10	PV_{net/m_2}	0.99	1	10+
11	*Gap*	200 ms	1	11+

(i) The quality of the estimation: convergence (i.e., $\mathcal{M}_{1-2}$), accuracy (i.e., $\mathcal{M}_{3-4}$), and precision (mostly $\mathcal{M}_{11-12}$).
(ii) The existence of hard guarantees: only DGP, DGP-α1 and B's provide those.
(iii) The constraints existing in the system: computational cost, mainly $\mathcal{M}_7$ and $\mathcal{M}_8$.
(iv) The number of estimated parameters: both α_R and β_R, or only β_R as it happens with NTP and DGP-α1.

We do not use here the rest of performance measures for their utility is related to more specific technical aspects, as explained in Appendix A.

In the particular case of this robotic system, we should use non-hard guaranteed methods only if they would provide better quality. The computational cost of the method will be supported by the laptop onboard the robot, thus it is important to reduce it as much as possible in order to reduce laptop battery consumption and thus to extend operational autonomy.

Table 11 shows the behaviours identified in Figures A1–A10 for the axes extremes listed in the last column of Table 10, and also which method performs better in each measure of interest. It is easy to see that the most distinctive measures are accuracy and computational cost; if we are concerned by computational cost—battery life—, and need a good quality in the estimate plus an estimation of both α_R and $beta_R$, the choice should be DGP. B's would be the method of choice only if the accuracy of the estimation of $beta_R$ is crucial (at the cost of much more computational cost and slightly worse precision). Convergence to stable estimates is very similar for all methods in the case of α_R if the axis 5 is in its greatest extreme, i.e., if the mobile base is started up much longer before the laptop is, but clearly better in the case of DGP for more normal situations.

Table 11. Selection of methods for the CRUMB robot example.

Measure	Name	Behaviour	Method Ordering
$\mathcal{M}_1$	$\alpha_R - entry$	1st	$DGP < B's < Reg$, all similar
$\mathcal{M}_2$	$\beta_R - entry$	1st if 5+, or 5th if 5-	$DGP < B's < Reg$, all similar if 5+
$\mathcal{M}_3$	$\alpha_R - error$	1st	$B's < DGP < Reg$
$\mathcal{M}_4$	$\beta_R - error$	1st	$DGP < B's < Reg$
$\mathcal{M}_7$	$comp$	n/a	$NTP < DGP\text{-}\alpha1 < Reg < DGP < B's$
$\mathcal{M}_8$	$std - comp$	n/a	$NTP < DGP\text{-}\alpha1 < Reg < DGP < B's$
$\mathcal{M}_{11}$	$\alpha_R - uncert$	1st	$DGP < B's$, all similar
$\mathcal{M}_{12}$	$\beta_R - uncert$	1st	$DGP < B's$, all similar

4. Conclusions

In this work, we have studied the problem of two-clock synchronization under the perspective of its application to sets of devices that communicate pairwise. There are relevant sensor applications where this form of synchronization is a core issue: embedded systems, robotics, domotics, etc. In such systems, clock synchronization is reduced to estimate the two main parameters that relate the clocks: α_R and β_R. It is of special importance to consider the uncertainty that the method has on the estimates (especially for embedded and critical systems), the rigidity of those estimates (hard vs. soft or stochastic), and its computational cost. Thus a particularized analysis of these systems is of special interest.

In the paper, we present a through statistical study that reveal some conclusions difficult or impossible to obtain from analytical approaches. We have selected, from the wide diversity that exists in the literature, a number of representative clock synchronization methods that can be applied to pairwise systems (some of them are commonly used as part of general network synchronization). Our selection of methods has been justified on a taxonomy that reflects the most relevant aspects of the problem in pairwise applications.

For carrying out a rigorous statistical analysis, we have implemented a rich simulation framework based on two-way message exchanges that allows us to represent most pairwise systems. It can be configured with a diversity of "axes", simulates realistic transmissions between the devices, collects abundant data from the simulation in the form of performance measures, and conducts statistical tests, particularly ANOVA, to make comparisons.

Exhaustive simulated experiments have produced a number of interesting results that we claim can be useful both to detect which characteristic of the application are prone to impair/benefit any method, and which is the best synchronization approach for it.

Our analyses indicate that geometrical approaches produce the best results in many use-case applications of two-clock synchronization, and in important performance measures (being DGP-α1 preferrable if computational cost must be reduced). The well-known NTP method, provided that both clocks are disciplined, performs well if there are long transmission times with high uncertainty, as, for instance, in remote surveillance applications. More detailed results and implications have been commented in the previous sections.

In the future we plan to study the dynamics of pairwise clock synchronization, i.e., the case where the parameters to estimate vary over time, and also the effects of other dynamics of the system in the problem (e.g., abrupt changes of regimes in transmission delays). This will likely lead to new variants of the geometrical methods. Applications such as robotics have a special interest for us and therefore we will dive into their particularities and the role of clock synchronization in their architectures, a problem that it is often oversighted.

Author Contributions: Conceptualization, J.-A.F.-M.; methodology, J.-A.F.-M. and A.N.; software, J.-A.F.-M. and R.A.; validation, J.-A.F.-M. and A.C.-M.; resources, A.N. and R.A.; data curation, J.-A.F.-M.; writing—review and editing, A.C.-M.; funding acquisition, all authors. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been funded by Junta de Andalucía research projects P08-TIC-04282, UMA18-FEDERJA-108 and UMA18-FEDERJA-113, and by the "Plan Propio de Investigación de la Universidad de Málaga".

Acknowledgments: The authors would like to thank Pablo Guerrero, Santiago Cárdenas, Conchi Carretero and Antonio Barrera, all with the Applied Mathematics Dpt. of the University of Málaga (UMA), for the deep discussions about the mathematical formalization of the process of estimating α_R and β_R. Jesús Briales also helped in searching for the optimization of those estimates on complex, noisy surfaces. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Malaga.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Distinctive Performance Behaviours of the Synchronization Methods

This appendix lists all figures corresponding to the clusterings (=*behaviours*) obtained on the performance measures $\mathcal{M}_i$ when comparing synchronization methods, showing the patterns that we have found in the methods arranged in rows. We do not include in those figures the clusterings for $\mathcal{M}_5$ $(vertex - mode)$, $\mathcal{M}_6$ $(vertex - max)$, $\mathcal{M}_9$ $(area)$ and $\mathcal{M}_9$ $(diag)$ because they only concern the comparison between the geometrical approaches (*DGP* vs. *Berthaud's*), already covered in Tables 3–5, and in all observed behaviours the former method always beats the latter.

Please refer to Section 3.4 for the details of how these figures have been obtained and of their use for practical applications.

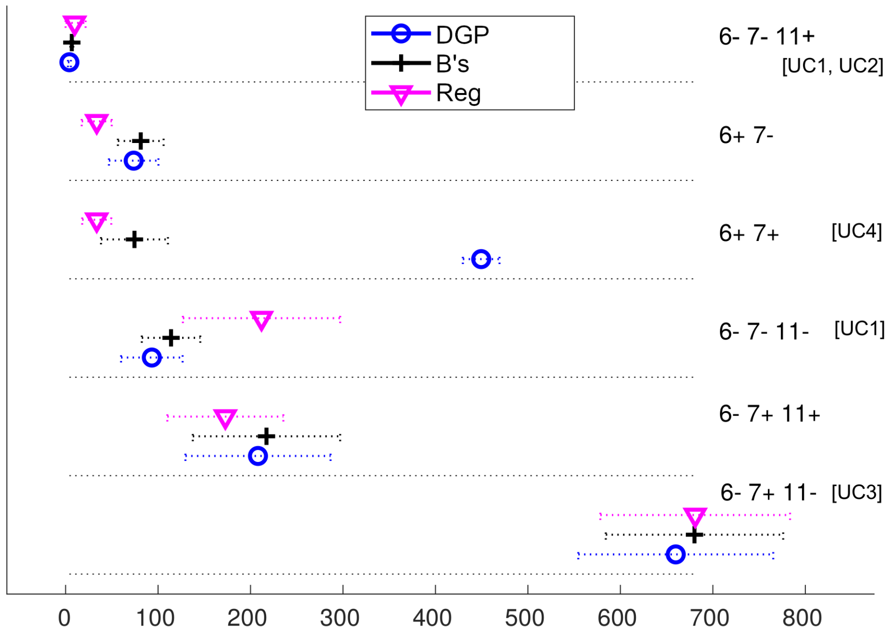


Figure A1. On the right side, from top to bottom, the six distinctive behaviours—clusters of scenarios—found for the $\mathcal{M}_1$ ($\alpha_R - entry$) measure, arranged in six rows. The abscissa is the magnitude of that measure (number of iterations of the methods before entering steady-state). Each method within a behaviour has a point marker with its average value in the measure and $\pm 1\sigma$ intervals around it (only illustrative of the dispersion, due to skewness). The lower values, the better. The behaviours are labelled with the sets of scenarios that produce them (axes at their minimum extreme (−) or at their maximum (+) in all scenarios of the cluster). In square brackets, we also indicate the practical use cases that match the behaviours, from the ones described in Section 3.4.

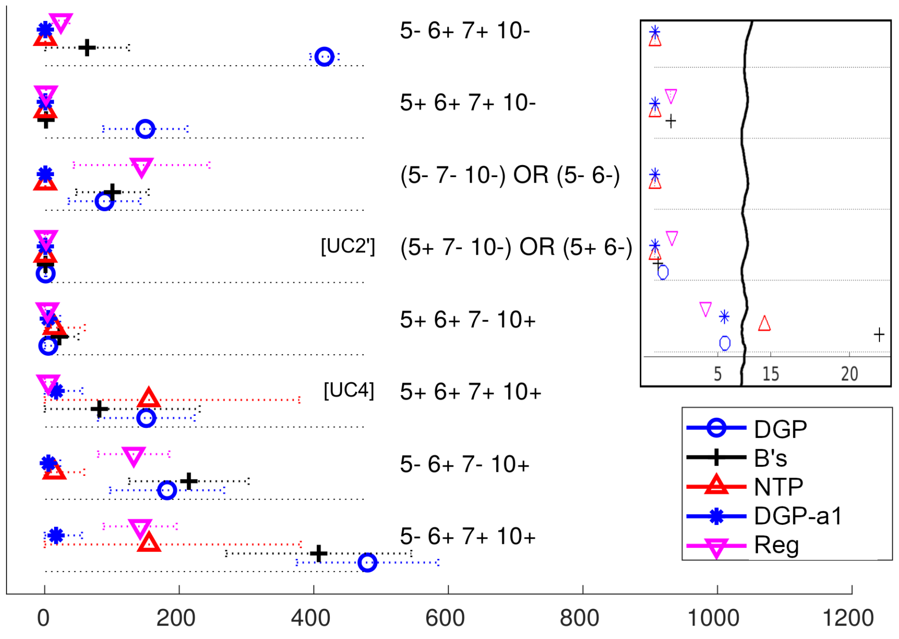


Figure A2. Eight distinctive behaviours have been found in the scenarios for $\mathcal{M}_2$ ($\beta_R - entry$) when $\alpha_R = 1$. Axes 3 and 4 must be at the same extreme position to assure $\alpha_R = 1$. Top-right) Zoom in on the average values at the left-most part of the first five behaviours.

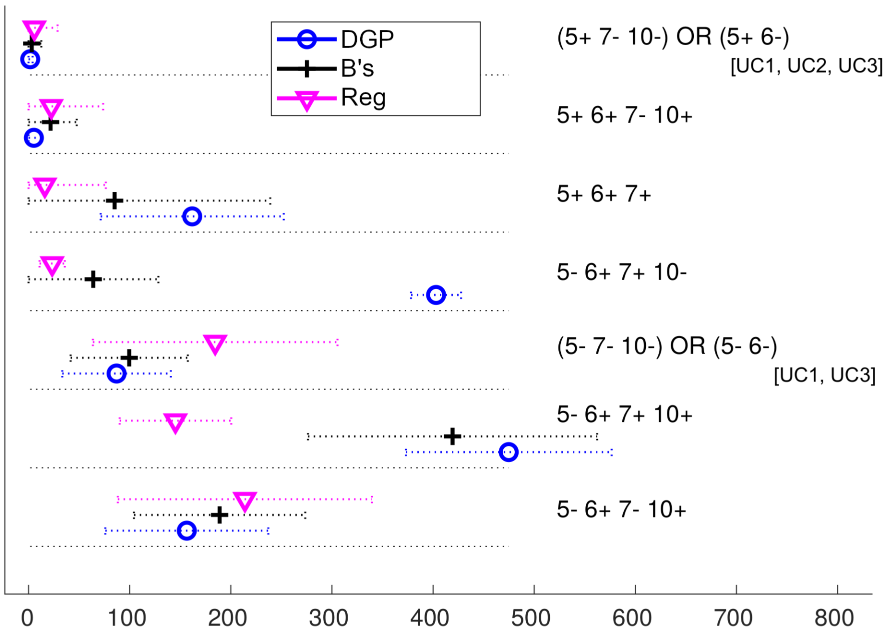


Figure A3. The seven distinctive behaviours found in the scenarios for $\mathcal{M}_2$ ($\beta_R - entry$) when $\alpha_R \neq 1$ (consequently, axes 3 and 4 must be at different extreme positions).

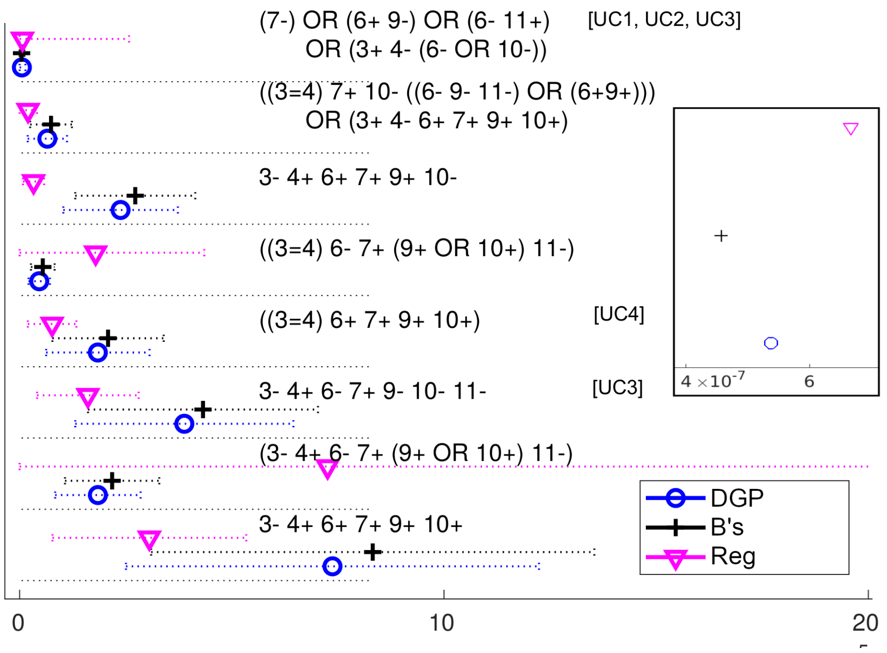


Figure A4. The eight distinctive behaviours found in the scenarios for $\mathcal{M}_3$ ($\alpha_R - error$; its magnitude is in the abscissa). Right Zoom in on the first behaviour.

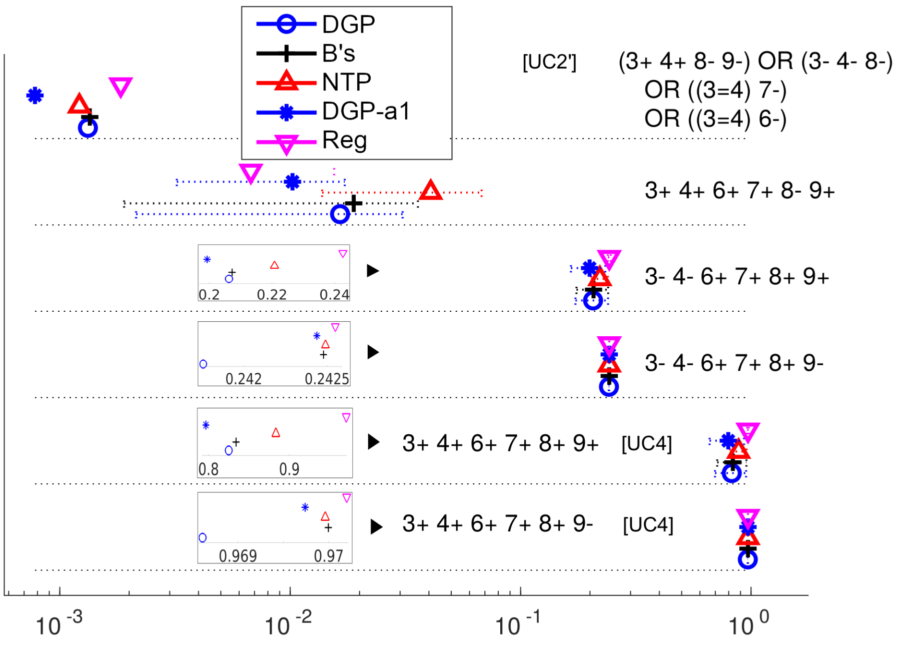


Figure A5. The six distinctive behaviours found in the scenarios for $\mathcal{M}_4$ ($\beta_R - error$; its magnitude, in seconds, in the abscissa) when $\alpha_R = 1$. The x-axis has been set to logarithmic scale for better observation of the small differences among algorithms in each behaviour. For the last four behaviours, additional zooms have been included.

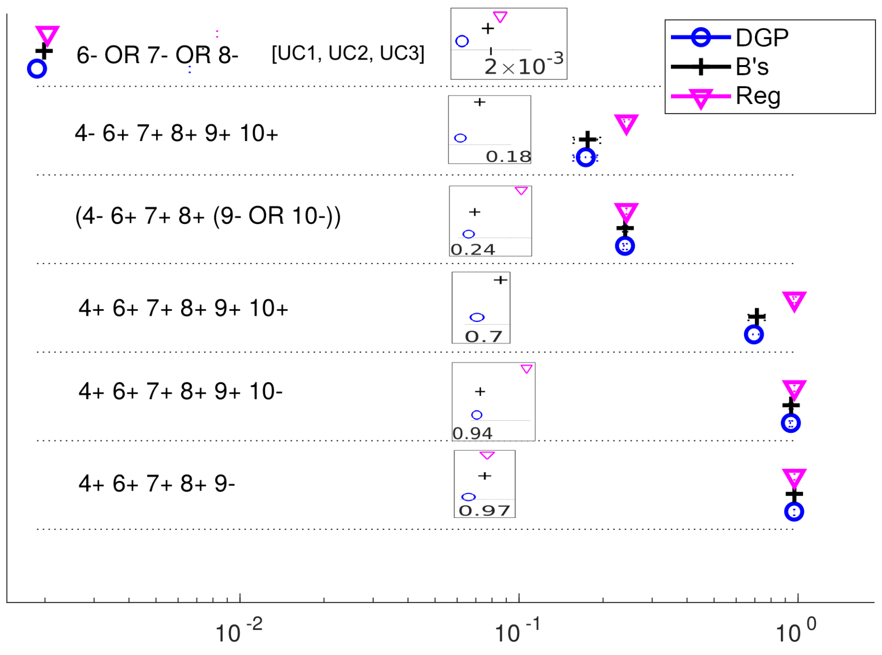
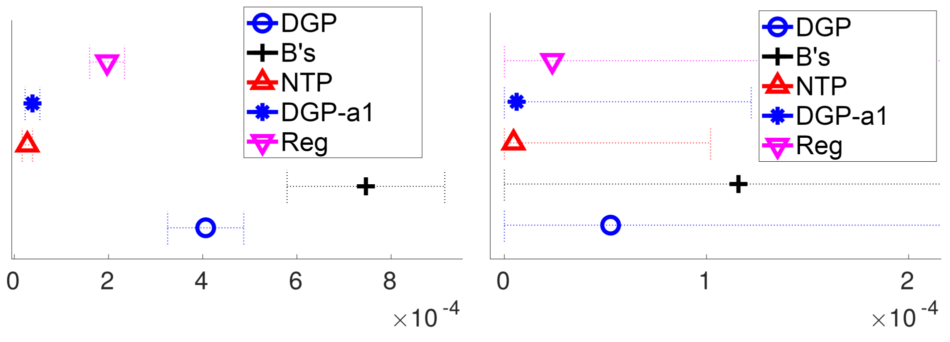


Figure A6. The six distinctive behaviours found for $\mathcal{M}_4$ ($\beta_R - error$; its magnitude, in seconds, in the abscissa) when $\alpha_R \neq 1$ (axes 3 and 4 are at different extreme positions here). We have used a logarithmic x axis for better distinction between methods in each behaviour, and added some zoom.



(a) (b)

Figure A7. All scenarios produce a single behaviour for all methods both in (**a**) $\mathcal{M}_7$ (*comp*) and (**b**) $\mathcal{M}_8$ (*std* − *comp*). The magnitude of the measure is in the abscissa, in seconds.

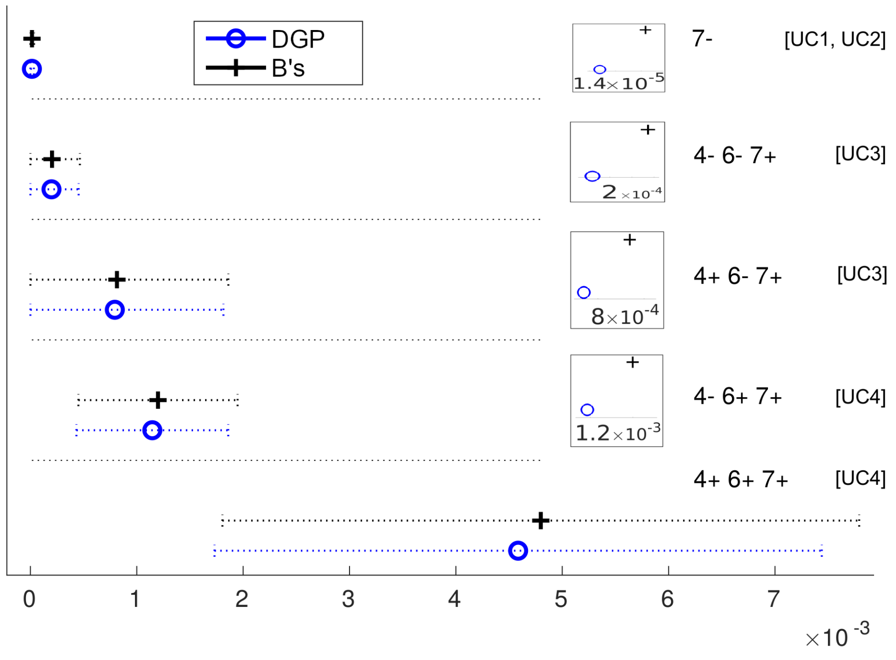


Figure A8. The five distinctive behaviours found in all scenarios for $\mathcal{M}_{11}$ (α_R − *uncert*), with zoom in those with more similar measure for both methods.

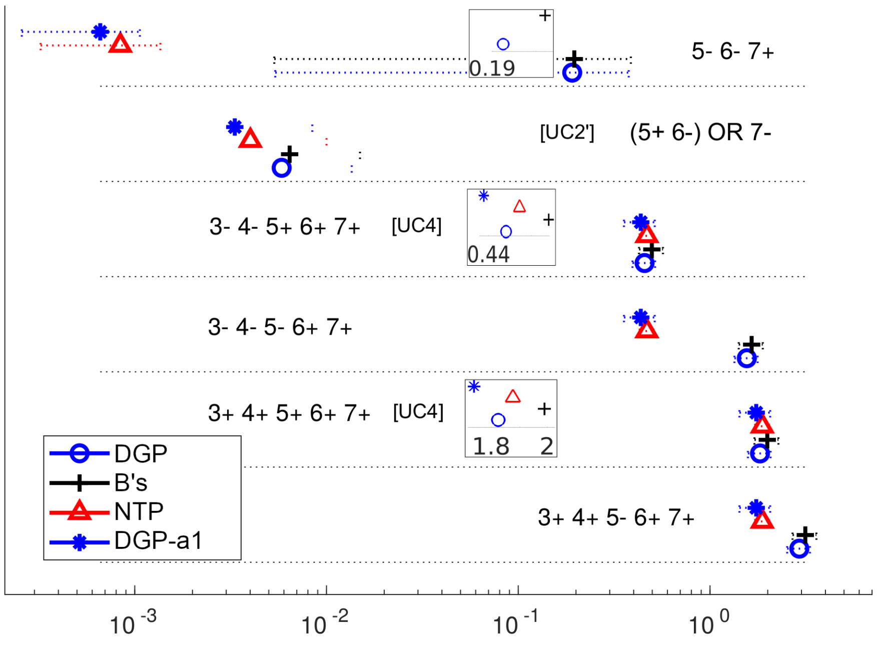


Figure A9. The six distinctive behaviours found in all scenarios for $\mathcal{M}_{12}$ ($\beta_R - uncert$) when $\alpha_R = 1$, with logarithmic x axis and zoom in some behaviours for better showing the methods differences. Axes 3 and 4 are always in the same extreme position.

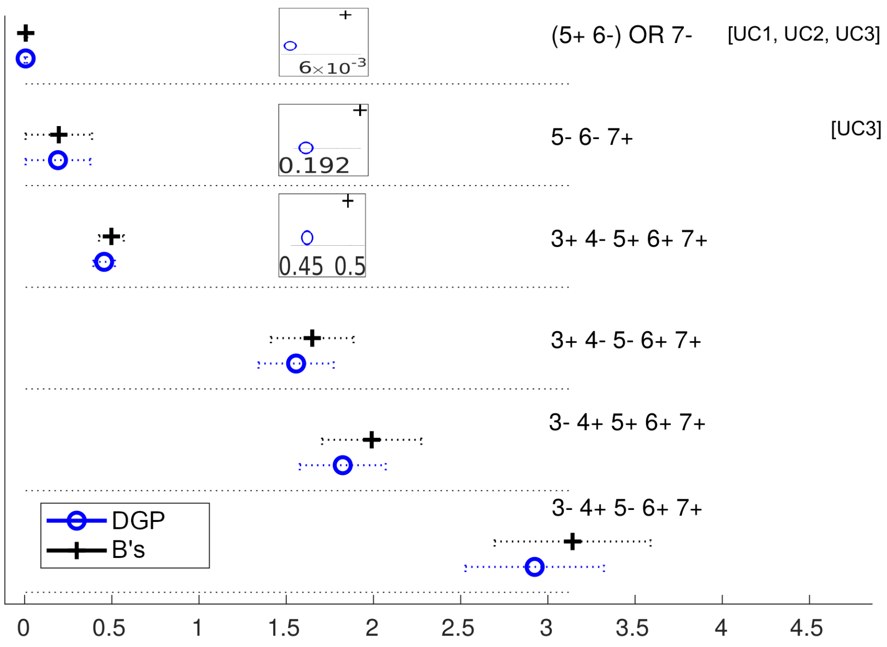


Figure A10. The six distinctive behaviours found in all scenarios for $\mathcal{M}_{12}$ ($\beta_R - uncert$) when $\alpha_R \neq 1$ (axes 3 and 4 must therefore be at different extreme positions), with zoom in those with more similar measure for both methods.

References

1. Liu, Y.; Shen, Y.; Guo, D.; Win, M. Network Localization and Synchronization Using Full-Duplex Radios. *IEEE Trans. Signal Process.* **2018**, *66*, 714–728. [CrossRef]
2. Fernández, J.; Hernando, A.; Ortega, J.; Santidrian, L. Synchronization in a Wireless Sensor Network Designed for Surveillance Applications. In Proceedings of the 2009 Fifth International Conference on Wireless and Mobile Communications, Beijing, China, 24–26 September 2009; pp. 369–372.

3. Li, J.; Jia, L.; Liu, G. Multisensor Time Synchronization Error Modeling and Compensation Method for Distributed POS. *IEEE Trans. Instrum. Meas.* **2016**, *65*, 2637–2645. [CrossRef]
4. Garone, E.; Gasparri, A.; Lamonaca, F. Clock synchronization protocol for wireless sensor networks with bounded communication delays. *Automatica* **2015**, *59*, 60–72. [CrossRef]
5. Sundararaman, B.; Buy, U.; Kshemkalyani, A. Clock synchronization for wireless sensor networks: A Survey. *Ad Hoc Netw.* **2005**, *3*, 281–323. [CrossRef]
6. Lamport, L. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* **1978**, *21*, 558–565. [CrossRef]
7. Rhee, I.K.; Lee, J.; Kim, J.; Serpedin, E.; Wu, Y.C. Clock Synchronization in Wireless Sensor Networks: An Overview. *Sensors* **2009**, *9*, 56–85. [CrossRef]
8. Sarvghadi, M.; Wan, T. Overview of time synchronization protocols in wireless sensor networks. In Proceedings of the 2nd International Conference on Electronic Design (ICED), Penang, Malaysia, 19–21 August 2014; pp. 204–209.
9. Lundelius, J.; Lynch, N. An Upper and Lower Bound for Clock Synchronization. *Inf. Control* **1984**, *62*, 190–204. [CrossRef]
10. Reginald, F.; Welch, J. Brief Announcement: A Tight Lower Bound for Clock Synchronization in Odd-Ary M-Toroids. In Proceedings of the 32nd International Symposium on Distributed Computing (DISC 2018), New Orleans, LA, USA, 15–19 October 2018.
11. Gusella, R.; Zatti, S. The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD. *IEEE Trans. Softw. Eng.* **1989**, *15*, 847–853. [CrossRef]
12. Elson, J.; Girod, L.; Estrin, D. Fine-grained network time synchronization using reference broadcasts. In Proceedings of Fifth Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, USA, 9–11 December 2002; Volume 36, pp. 147–163.
13. Christian, F. Probabilistic clock synchronization. *Distrib. Comput.* **1989**, *3*, 146–158. [CrossRef]
14. Wu, Y.; Chaudhari, Q.; Serpedin, E. Clock Synchronization of Wireless Sensor Networks. *IEEE Signal Process. Mag.* **2011**, *28*, 124–138. [CrossRef]
15. Lundelius, J.; Lynch, N. A New Fault-Tolerant Algorithm for Clock Synchronization. In Proceedings of the 3rd annual ACM Symposium on Principles of Distributed Computing (PODC), Vancouver, BC, Canada, 27–29 August 1984.
16. Stanković, M.; Stanković, S.; Johansson, K. Distributed time synchronization for networks with random delays and measurement noise. *Automatica* **2018**, *93*, 126–137. [CrossRef]
17. Xie, K.; Cai, Q.; Fu, M. A fast clock synchronization algorithm for wireless sensor networks. *Automatica* **2018**, *92*, 133–142. [CrossRef]
18. Bolognani, S.; Carli, R.; Lovisari, E.; Zampieri, S. A Randomized Linear Algorithm for Clock Synchronization in Multi-Agent Systems. *IEEE Trans. Autom. Control* **2016**, *61*, 1711–1726. [CrossRef]
19. Leng, M.; Wu, Y.C. Low-Complexity Maximum-Likelihood Estimator for Clock Synchronization of Wireless Sensor Nodes Under Exponential Delays. *IEEE Trans. Signal Process.* **2011**, *59*, 4860–4870. [CrossRef]
20. Fernández-Madrigal, J.; Martínez-Tenor, A. Two-Clocks Synchronization for Networked Sensors. In Proceedings of the IEEE Sensors, Valencia, Spain, 2–5 November 2014.
21. Saputra, O.; Wei-Chung, T.; Tsung-Han, C. Hough Transform-Based Clock Skew Measurement Over Network. *IEEE Trans. Instrum. Meas.* **2015**, *64*, 3209–3216. [CrossRef]
22. Giorgi, G. An Event-Based Kalman Filter for Clock Synchronization. *IEEE Trans. Instrum. Meas.* **2015**, *64*, 449–457. [CrossRef]
23. Mills, D. Internet Time Synchronization. The Network Time Protocol. *IEEE Trans. Commun.* **1991**, *39*, 1482–1493. [CrossRef]
24. Maxwell, S.; Delaney, H. *Designing Experiments and Analyzing Data. A Model Comparison Perspective*, 2nd ed.; Lawrence Erlbaum Assoc.: London, UK, 2004; ISBN 0-8058-3718-3.
25. Sichitiu, M.; Veerarittiphan, C. Simple, Accurate Time Synchronization for Wireless Sensor Networks. In Proceedings of the IEEE Conference on Wireless Communications and Networking (WCNC), New Orleans, LA, USA, 16–20 March 2003; Volume 2, pp. 1266–1273.
26. Arvind, K. Probabilistic clock synchronization in distributed systems. *IEEE Trans. Parallel Distrib. Syst.* **1994**, *5*, 474–487. [CrossRef]

27. Berthaud, J. Time Synchronization Over Networks Using Convex Closures. *IEEE/ACM Trans. Netw.* **2000**, *8*, 265–277. [CrossRef]
28. Sugihara, R.; Gupta, R. Clock Synchronization with Deterministic Accuracy Guarantee. In Proceedings of the EWSN 2011: Wireless Sensor Networks, Bonn, Germany, 23–25 February 2011; Lecture Notes in Computer Science; Volume 6567.
29. Lemmon, M.; Ganguly, J.; Xia, L. Model-based Clock Synchronization in Networks with Drifting Clocks. In Proceedings of the IEEE Pacific Rim International Symposium on Dependable Computing (PRDC), Los Angeles, CA, USA, 20 December 2000.
30. Institute of Electrical and Electronics Engineers, *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*; IEEE: Palo Alto, CA, USA, 2002.
31. Duda, A.; Harrus, G.; Haddad, Y.; Bernard, G. Estimating global time in distributed systems. In Proceedings of the 7th ICDCS, Berlin, Germany, September 1987.
32. Ashton, P. *Algorithms for Off-Line Clock Synchronization*; Technical report; Department of Computer Science, University of Canterbury: Christchurch, New Zealand, 1995.
33. Cao, J.; Cleveland, W.S.; Lin, D.; Sun, D.X. Internet Traffic Tends Toward Poisson and Independent as the Load Increases. In *NonLinear Estimation and Classification*; Springer: New York, NY, USA, 2002.
34. Gago-Benítez, A.; Fernández-Madrigal, J.; Cruz-Martín, A. A computationally efficient algorithm for modeling multi-regime delays in the sensory flow of networked telerobots. In Proceedings of the International Conference on Control, Robotics and Cybernetics (ICCRC), Cape Town, South Africa, 18–19 August 2012.
35. Gago-Benitez, A.; Fernández-Madrigal, J.; Cruz-Martín, A. Log-logistic modelling of sensory flow delays in networked telerobots. *IEEE Sens.* **2013**, *13*, 2944–2953. [CrossRef]
36. Maróti, M.; Kusy, B.; Simon, G.; Ĺdeczi, A. The flooding time synchronization protocol. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, MD, USA, 3–5 November 2004.
37. Gago-Benítez, A.; Fernández-Madrigal, J.; Cruz-Martín, A. Marginal Probabilistic Modeling of the Delays in the Sensory Data Transmission of Networked Telerobots. *Sensors* **2014**, *14*, 2305–2349. [CrossRef] [PubMed]
38. Murphy, K.; Myors, B. *Statistical Power Analysis. A Simple and General Model for Traditional and Modern Hypothesis Tests*, 2nd ed.; Lawrence Erlbaum Assoc.: London, UK, 2004; ISBN 0-8058-4525-9.
39. Jette, M.; Yoo, A.; Grondona, M. SLURM: Simple Linux Utility for Resource Management. In Proceedings of the Job Scheduling Strategies for Parallel Processing (JSSPP), Seattle, WA, USA, 24 June 2003; Lecture Notes in Computer Science.
40. Hartigan, J.; Hartigan, P. The Dip Test of Unimodality. *Ann. Stat.* **1985**, *13*, 70–84. [CrossRef]
41. Price, N. Implementation of the Dip Test in Matlab. Available online: http://www.nicprice.net/diptest/ (accessed on 14 March 2019).
42. Tukey, J. Comparing individual means in the analysis of variance. *Biometrics* **1949**, *5*, 99–114. [CrossRef] [PubMed]
43. Curtis, H. McCluskey's Method. In *A New Approach to the Design of Switching Circuits*; The Bell Laboratories Series; D. van Nostrand Company, Inc.: Princeton, NJ, USA, 1962; Chapter 2.3, pp. 90–160.
44. Käström, P. Implementation of the Quine-McCluskey Algorithm, Optimized for Speed. Available online: https://es.mathworks.com/matlabcentral/fileexchange/37118-mintruthtable-tt-flags (accessed on 16 April 2019).
45. Bell, E. The iterated exponential integers. *Ann. Math.* **1938**, *39*, 539–557. [CrossRef]
46. Adroit Market Research Expects the Home Automation Market to be Worth US$46.22 Billion by 2025. Available online: http://knxtoday.com/2019/07/13879/adroit-market-research-expects-the-home-automation-market-to-be-worth-us46-22-billion-by-2025.html (accessed on 14 February 2020).
47. Fernández-Madrigal, J.; Cruz-Martín, A. The CRUMB Mobile Robot. Available online: https://babel.isa.uma.es/crumb/index.php/sample-page/ (accessed on 31 July 2020).