Safe Estimation of Time Uncertainty of Local Clocks

Andrea Bondavalli, Francesco Brancati, Andrea Ceccarelli
University of Florence, Viale Morgagni 65, I-50134, Firenze, Italy
{bondavalli, francesco.brancati, andrea. ceccarelli}@unifi.it

***Abstract*—The Reliable and Self-Aware Clock (R&SAClock) is a new software clock aimed at providing resilient time information. It uses and exploits the information collected from any chosen clock synchronization mechanism to provide both the current time and the synchronization uncertainty, intended as a conservative and self-adaptive estimation of the distance from an external global time. This paper describes an algorithm that uses statistical analysis to compute the synchronization uncertainty with a given coverage. Simulations are presented that show the behavior of the algorithm and its effectiveness.**

Keywords- Software clocks; synchronization uncertainty; power-law; statistics; simulation;

I. Introduction

Many pervasive and distributed systems require clocks to be synchronized to an external global timebase in order to correctly execute their services [1], [5]. Some of these services take advantage from the awareness of the actual distance from global time (the *offset* [4]), which, unfortunately, is typically a variable factor very hard to compute due to changes in system dynamics, faults, or environmental changes [1], [10], [9]. Synchronization mechanisms typically compute an *estimated offset*, without offering guarantees of closeness of this value to offset. Worst case bounds on offset are usually provided (*accuracy*, [5]), and being imposed to systems as requirements, but these bounds are usually far from typical execution scenarios and consequently are of little use in practice.

The notion of uncertainty of the time measurement as used in metrology [2] can provide an estimation of synchronization quality that in practice can result more useful than accuracy and offset. We introduce the notion of *synchronization uncertainty* [3] as an adaptive and conservative estimation of the distance of local clock from global time. The following two examples, reported from [3], aim to clarify the possible application contexts of synchronization uncertainty in real systems.

As a first example, we consider real-time and adaptive applications in Wireless Sensor Networks (WSNs [1]) as the sensing applications: these applications require a specific synchronization uncertainty in order to collect reliable data, but keeping clocks continuously synchronized brings relevant energy consumption: thus, adaptive requirements for synchronization uncertainty may be established, with the aim to achieve an optimal trade-off between the energy costs of transmitting synchronization beacons and the need to contain synchronization uncertainty within system requirements to collect reliable data [1].

As a second example we consider the testing and monitoring of distributed systems. When measuring time intervals in distributed systems, related to different nodes of the system (e.g., the one-way delay), it is important to understand, quantify and control the misalignment of distributed clocks i.e., the synchronization uncertainty, to avoid erroneous data.

A software clock component named Reliable and Self-Aware Clock (R&SAClock [3]) has recently been proposed and developed to equip systems with the ability to compute and use the synchronization uncertainty [3]. R&SAClock is not a synchronization mechanism, but it acts as a new software clock that exploits services and data provided by any chosen synchronization mechanism (for external clock synchronization) to provide its users both the current time and the current synchronization uncertainty.

In this paper we present an algorithm that, given a desired or required coverage as a set-up parameter, computes the synchronization uncertainty interval through time resorting to a statistical analysis of the behavior of the local oscillator and of the synchronization mechanism. Through a set of simulations, we show evidence of the behavior and effectiveness of the algorithm. A very basic 'feasibility' algorithm to compute synchronization uncertainty (called *UEA-basic,* basic Uncertainty Evaluation Algorithm) was proposed in [3]. In this paper we define an algorithm (called *UEA-SPS,* Statistical Predictor and Safety Margin UEA, *SPS* hereafter) which is completely different, both in its concepts and in its performance.

The rest of the paper is organized as follows. In Section II the basics of the R&SAClock are introduced, in Section III our algorithm is presented, and in Section IV simulation results of the algorithm are shown. Conclusions are in Section V.

II. The Reliable and Self-Aware Clock

A. Basic Notions of Time and Clocks

Let us consider a distributed system composed of a set of nodes. We define *global time* as the unique time view shared by the nodes of the system, *reference clock* as the clock that always holds the global time, and *reference node* as the node that owns the reference clock. Given a local clock *c* and any *time instant t*, we define *c(t)* as the *time value* read by local clock *c* at time *t*.

The behavior of a local clock *c* is characterized by the quantities *offset*, *accuracy* and *drift*. The offset $\Theta_c(t) = t - c(t)$ is the actual distance of local clock *c* of the node *n* from the global time at time *t* [4]. This distance may vary through time. Accuracy A_c is an upper bound of the offset [5]; accuracy is often adopted in the definition of system requirements and therefore targeted by clock synchronization mechanisms. *Drift* $\rho_c(t)$ describes the rate of deviation of a local clock c at time t from global time [5].

Despite their theoretical importance, accuracy and offset are usually of practical little use for systems, since accuracy is usually a high value, not representative of actual distance from global time, and offset is difficult to measure exactly at any time *t*. Synchronization mechanisms typically compute an *estimated offset* $\widetilde{\Theta}_c(t)$ (and an *estimated drift* $\tilde{\rho}_c(t)$), without offering guarantees and only at synchronization instants. We define the *synchronization uncertainty* (or simply *uncertainty*) $U_c(t)$ as an adaptive and conservative evaluation of offset $\Theta_c(t)$ at any time *t*; uncertainty is such that $A_c \geq U_c(t) \geq |\Theta_c(t)| \geq 0$ [3].

B. Basic Specifications of the R&SAClock

The R&SAClock is a software clock that provides to users (e.g., system processes) both the time value and the synchronization uncertainty associated to the time value. When a user asks the current time to R&SAClock (by invoking function *getTime*), R&SAClock provides an enriched time value [*likelyTime, minTime, maxTime, FLAG*]. *LikelyTime* is the time value computed reading the local clock (i.e., $c(t)$). *MinTime* and *maxTime* are based on the synchronization uncertainty provided by the internal mechanisms of R&SAClock. More specifically, for a clock c at any time instant t, we extend the notion of synchronization uncertainty $U_c(t)$ distinguishing between a right uncertainty (positive) $U_{cr}(t)$ and a left uncertainty (negative) $U_{cl}(t)$, such that $U_c(t) = \max[U_{cr}(t); -U_{cl}(t)]$. Values *minTime* and *maxTime* are respectively a left and a right bound of the reasonable values that can be attributed to the actual time: *minTime* is set to $c(t) + U_{cl}(t)$ and *maxTime* is set to $c(t) + U_{cr}(t)$.

The user that exploits the R&SAClock can impose an *accuracy requirement*, that is the largest synchronization uncertainty that the user can accept in order to work correctly. Correspondingly, R&SAClock can give value to its output *FLAG*, which is a Boolean value that indicates whether the current synchronization uncertainty is within the accuracy requirement or not. It is evident that the main core of R&SAClock is the *uncertainty evaluation* algorithm (*UEA*), that equips the R&SAClock with the ability to compute the uncertainty.

III. The Statistical Predictor and Safety Margin Uncertainty Evaluation Algorithm

In this Section we present the SPS version of UEA for a local software clock c that is disciplined by an external clock synchronization mechanism. To ease the readability of the notation, the subscript c is omitted from the expressions presented in the rest of this paper. In Table I the main quantities involved in the SPS are shown and explained.

A. Basic Assumptions

We consider a synchronization mechanism that, at synchronization instants, computes the estimated offset and the estimated drift (it performs a synchronization).

We assume t_0 the time in which the most recent synchronization is performed: at time t_0 the synchronization mechanism computes the estimated offset $\widetilde{\Theta}(t_0)$ and possibly the estimated drift $\tilde{\rho}(t_0)$ (if not provided by the mechanism, it can be easily computed by the R&SAClock itself).

B. SPS Overview

The SPS computes the uncertainty at a time t with a coverage, intended as the probability that $A_c \geq U(t) \geq |\Theta_c(t)| \geq 0$ holds. The computed uncertainty is composed by three quantities: i) the *estimated offset*, ii) the output of a *predictor* function and iii) the output of a *safety margin* function. The computation of synchronization uncertainty requires a right uncertainty $U_r(t)$ and a left uncertainty $U_l(t)$: consequently, we define a *right predictor* function and a *right safety margin* function for right uncertainty, and a *left predictor* function and a *left safety margin* function for left uncertainty. The output of the SPS at $t \geq t_0$ is constituted by the two values:

$$U_r(t)=\max(0, \widetilde{\Theta}(t_0))+P_r(t)+SM_r(t_0) \quad (1)$$

$$U_l(t)=\min(0, \widetilde{\Theta}(t_0))+P_l(t)+SM_l(t_0). \quad (2)$$

The *estimated offset* $\widetilde{\Theta}(t_0)$ is computed by the synchronization mechanism and can contain errors. If the estimated offset is positive, it influences the computation of an

TABLE I. Main Quantities and Parameters for the SPS

Symbol	Definition
t_0	time in which the most recent synchronization is performed
$\widetilde{\Theta}(t_0)$	estimated offset at time t_0
$\tilde{\rho}(t_0)$	estimated drift at time t_0
M	maximum number of samples of the estimated drift that the SPS collects ($M > 0$)
m	current number of samples of the estimated drift collected ($m \leq M$); these are the m most recent samples of the estimated drift
N	maximum number of samples of the estimated offset that the UEA collects ($N > 0$)
n	current number of samples of the estimated offset collected ($n \leq N$); these are the n most recent samples of the estimated offset
σ_d^2	population variance of the estimated drift
s_d^2	sample variance of the estimated drift
σ_{ds}^2	safe bound on the variance σ_d^2: with probability p_{ds} we have that $\sigma_d^2 \leq \sigma_{ds}^2$
σ_o^2	population variance of the estimated offset
σ_{os}^2	safe bound on the variance σ_o^2: with probability p_{os} we have that $\sigma_o^2 \leq \sigma_{os}^2$
p_{ds}	probability that $\sigma_d^2 \leq \sigma_{ds}^2$
p_{dv}	a safe bound of the drift variation since t_0 is computed with probability p_{dv}
$p_{ds} \circ p_{dv}$	a combination of these two values represents the coverage of the prediction function
p_{os}	probability that $\sigma_o^2 \leq \sigma_{os}^2$
p_{ov}	a safe bound of the offset at t_0 is computed with

Symbol	Definition
	probability p_{ov}
$p_{os} \circ p_{ov}$	a combination of these two values represents the coverage of the safety margin function

upper bound on the offset itself and consequently is considered in (1). If it is negative, it is ignored. A symmetric reasoning holds for (2).

The *predictor* functions (left and right) predict the behavior of the oscillator and continuously provide bounds (lower and upper) which constitute a safe (pessimistic) estimation of the oscillator drift and consequently a bound on the offset.

The *safety margin* functions (left and right) aim at compensating possible errors in the prediction and/or in the estimation of the offset. The *safety margin* values (negative and positive respectively) are computed at t_0 and are updated only at a new synchronization.

The set-up parameters used by SPS (detailed in Table I) are: the four probabilities $p_{ds}, p_{dv}, p_{os}, p_{ov}$ and the values *M* and *N*. The coverage and the performance (how much the synchronization uncertainty is effectively tight to the estimated offset) achieved by SPS depends on these parameters.

C. The Right Predictor Function

The behavior of the oscillator drift is modeled with the random walk frequency noise model, one of the five canonical models used to model oscillators (the power-law models [6]), that we considered as appropriate and used.

Obviously the parameters of this random walk are unknown and depend on the specific oscillator used. We compute them resorting to the observation of the last m samples of the drift. Let s_d^2 be the variance of such samples. From s_d^2 we compute a safe upper bound σ_{ds}^2 to the population variance σ_d^2 with probability p_{ds}. As explained in the left part of Fig. 1, s_d^2 is the $1 - p_{ds}$ percentile of the χ^2 distribution for the variance of the *m* samples [7].

We use then σ_{ds}^2 instead of σ_d^2 to plot a Gaussian distribution with mean in 0 and variance σ_{ds}^2 (right part of Fig. 1) and to compute the diffusion coefficient D describing the random walk of the oscillator at hand [8]. D and the inverse of the Gauss error function $erf^{-1}(p_{dv})$ [7] (that indicates the interval underlying a cumulative probability p_{dv} - right part of Fig. 1) are finally used to compute an upper bound to the variations of the drift $VD_r(t)$ in the time interval $t - t_0$.

$$VD_r(t) = erf^{-1}(p_{dv}) \cdot \sqrt{D \cdot (t - t_0)}. \qquad (3)$$

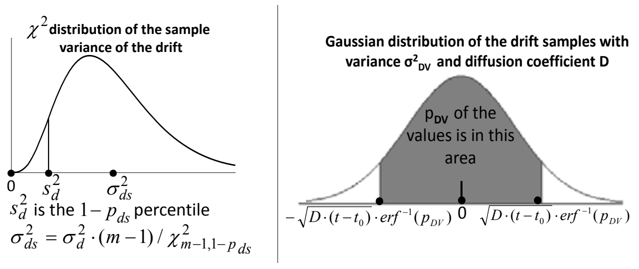


Figure 1. χ^2distribution (left) and Gaussian distribution (right) of the *m* sample of the estimated drift.

While $VD_r(t)$ represents the variations of the drift, $\tilde{\rho}(t_0)$ is the drift at time t_0. This is supposed to be compensated by the external synchronization algorithm (or by other components of the R&SAClock) so it is expected to be 0. However this compensation may not happen for whatever reason and the initial drift may remain during the interval $(t - t_0)$. In Section IV we will explore the impact of this choice by computing the expected offset accounting for the initial drift (Fig. 2 and Fig-3) and without it (Fig. 4 to Fig. 8). To consider it, since we are dealing with the right uncertainty, $\max(0, \tilde{\rho}(t_0))$ needs to be accounted for. $P_r(t)$ the bound on the variations of the offset thus is:

$$P_r(t) = \int_{t_0}^{t} \left(VD_r(x) + \max(0, \tilde{\rho}(t_0))\right) dx =$$

$$= \mathrm{erf}^{-1}(p_{dv}) \sqrt{D} \frac{2}{3} (\mathrm{t} - \mathrm{t_0})^{\frac{3}{2}} + \max(0, \tilde{\rho}(t_0)) (\mathrm{t} - \mathrm{t_0}) \quad (4)$$

D. The Left Predictor Function

The computation of the drift variations performed in the left predictor function is totally symmetrical to the right one. So if one decides to neglect the initial drift the function returns the opposite value of the right predictor function. If the initial drift needs to be considered then we need to consider $min(0, \tilde{\rho}(t_0))$:

$$VD_l(t) = -erf^{-1}(p_{dv}) \cdot \sqrt{D \cdot (t - t_0)} \qquad (5)$$

$$P_l(t) = \int_{t_0}^{t} \left(VD_l(x) + \min(0, \tilde{\rho}(t_0))\right) dx =$$

$$= -\mathrm{erf}^{-1}(p_{dv}) \sqrt{D} \frac{2}{3} (\mathrm{t} - \mathrm{t_0})^{\frac{3}{2}} + \min(0, \tilde{\rho}(t_0)) (\mathrm{t} - \mathrm{t_0}) \, (6)$$

E. The Right Safety Margin Function

The safety margin function is computed starting from the collection of n samples of the estimated offset. A safe upper bound σ_{os}^2 to the population variance σ_o^2 of the estimated offset is computed with probability p_{os}. Similarly to (3), we compute $SM_r(t_0)$ with a probability p_{ov} as follows:

$$SM_r(t_0) = \sqrt{2} \cdot \mathrm{erf}^{-1}(p_{ov}) \cdot \sigma_{os}. \qquad (7)$$

F. The Left Safety Margin Function

Also in the case of the safety margin function the computation is symmetrical and therefore the opposite value is returned by the left function:

$$SM_l(t_0) = -SM_r(t_0) \qquad (8)$$

IV. Simulation Results

In this Section we show the results of the simulations of the SPS, performed in order to analyze its effectiveness and potential. Each simulation is performed by tracing the execution of a real clock disciplined by NTP (Network Time Protocol [4]), and collecting (exploiting the NTP services):

i) the estimated offsets,

ii) the estimated drifts and

iii) the local time in which such values have been collected.

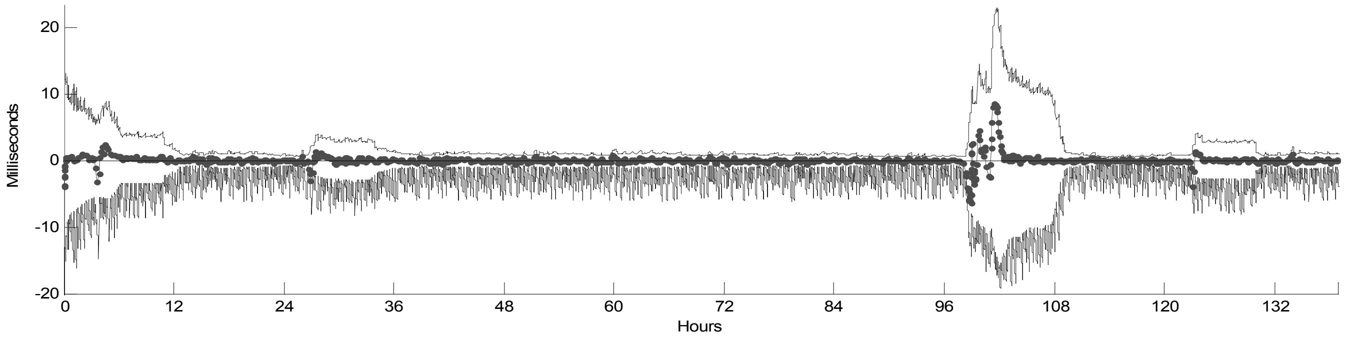


Figure 2. The evolution of the local clock, of the offset estimated by NTP, and of the synchronization uncertainty in a trace of 132 hours.

This data allows to trace the local clock with respect to the NTP reference. In a post-processing phase, we apply the SPS on the data collected and we compute the left and right synchronization uncertainty.

These simulations do not allow to compare the synchronization uncertainty with respect to the global time: they are not intended to validate SPS, but only to explain and analyze its behavior.

We traced the clocks of five common PCs (from our lab) of different factories and quality, executing different Linux versions and different NTP versions (4.0 or newer). Four of the five PC had a negative drift varying from $-70PPM$ to $-40PPM$ and only one (which was in a different building) had a positive drift about $65PPM$. However, despite their values, all data had the same trend: after an initial transient phase it remains within a limited interval. We applied the SPS to the traces collected, varying the configuration parameters $p_{dv}, p_{ds}, p_{ov}, p_{os}$, M and N. All simulations confirmed our expectations and provided quite consistent indications.

The first scenario we consider is formed by one sample trace of 132 hours, with the NTP synchronization interval set to 16 seconds (this means that NTP attempts a synchronization, not that a synchronization really happens), and the NTP server reachable through the internet. NTP is started at time 0 (the beginning of the data collection). Fig. 2 shows three curves over the 132 hours; the x-axis represents the length of the trace (in hours), and the y-axis is the distance from the local time (in milliseconds). The dot-marked curve represents the estimated offset computed by NTP and the two external ones represent the right uncertainty (upper) and the left uncertainty (lower), computed with the default values: $p_{dv}, p_{ds}, p_{ov}, p_{os}$ are set to 0,99 and M and N are set to 50. In this case the drift was negative and the left uncertainty was computed assuming the drift may not be corrected.

The estimated offset is always between the left and right synchronization uncertainty. During the initial transient phase (from the start of the simulation to about hour 6), synchronization uncertainty is quite large (several milliseconds). At steady state, the estimated offset is about 1 or 2 milliseconds. At hour 98, the estimated offset suddenly increases (up to 7 milliseconds) and at hour 102 it is back to 1 or 2 milliseconds.

A. Results varing the parameters of the predictor function

Fig. 3, using a subset of the data used in the previous setting, shows the synchronization uncertainty assuming that from time t=72.5 hours, no further synchronizations happen. The figure reports the synchronization uncertainty values using two different coverage values for the predictor function.

The dashed line represents the synchronization uncertainty computed setting p_{dv}, p_{ds} to 0,999999, while the continuous one is computed leaving p_{dv}, p_{ds} set to 0,99 (the other parameters are set to their default values).

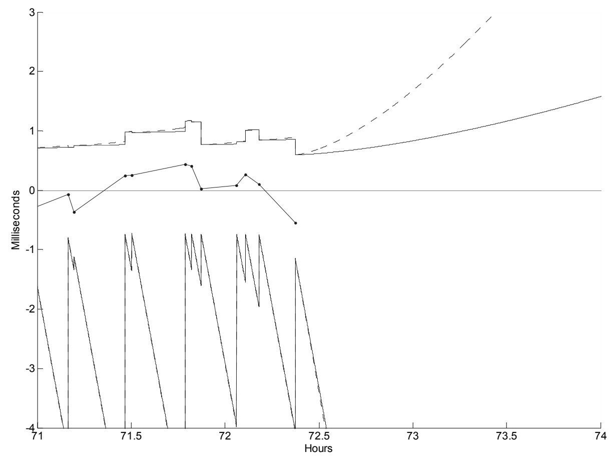


Figure 3. Synchronization uncertainty varying the prediction function parameter.

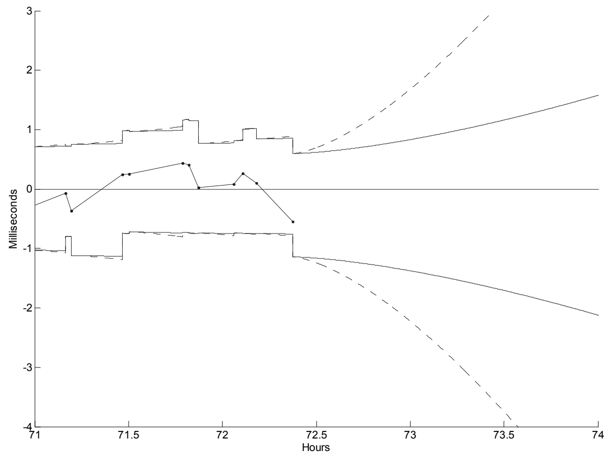
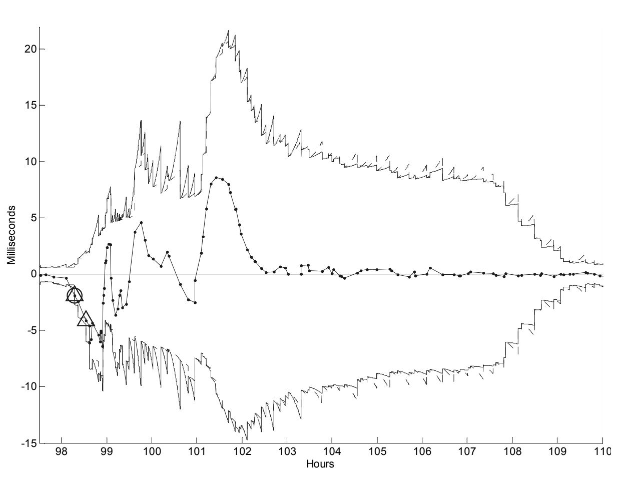
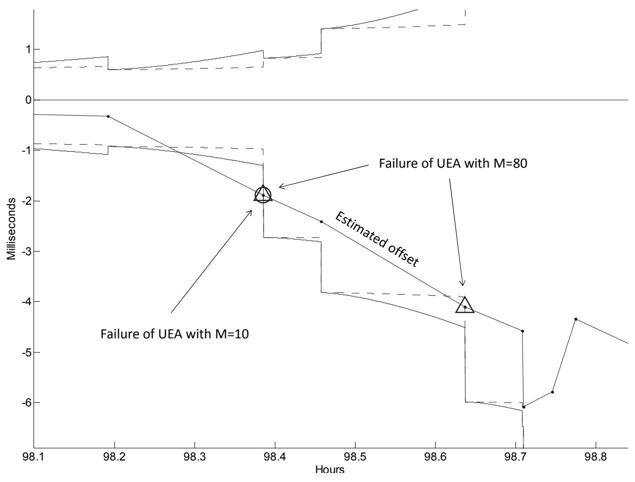


Figure 4. Synchronization uncertainty varying the prediction function parameter without taking account of initial drift.



a)



b)

Figure 5: Synchronization uncertainty varying the M parameter: a) behavior of synchronization uncertainty in a slot of about 12 hours, and b) hour 98, enlarged to show details and violation of the requirements on the synchronization uncertainty.

While for $t \leq 72{,}5$ Hours it is difficult to see relevant differences between the two, for $t > 72{,}5$ hours in absence of synchronizations it is easy to see the that the dashed line grows much faster.

Another important observation is that the left uncertainty is much bigger (in absolute values) than the right one, due to the fact that uncertainty has been computed assuming possible failures of the process that disciplines the local clock (i.e., considering the drift at the beginning of each synchronization period).

Releasing such assumption we obtain the curves depicted in Fig. 4. The two uncertainties are symmetric and overall much less pessimistic than before. It is evident that much stricter bounds are achieved if one can guarantee that the initial drift is compensated. From now on, the following analyses will be performed by neglecting the drift at the beginning of each synchronization period.

B. Results varing the M parameter

Fig. 5a and Fig. 5b report the synchronization uncertainty obtained using two different values for the parameter *M* of the estimated drift samples used in the algorithm. The dashed line represents the synchronization uncertainty computed with *M*=80, while the continuous one has been obtained with *M*=10 samples. The other parameter have the default values.

As one can expect low values for *M* imply a larger but more reactive synchronization uncertainty. High values reduce the size of the interval but at the price of a slower reaction to sudden changes of the drift. In fact, Fig. 5b (in which we observe sudden drift changes) shows that the dashed line fails twice in containing the estimated offset while the continuous one fails only once.

C. Results varing the parameters of safety margin function

Fig. 6 reports the synchronization uncertainty obtained using two different values for the coverage of the safety margin function. The dashed line represents the synchronization uncertainty computed setting p_{os}, p_{ov} to 0,999999, while the continuous one is computed leaving p_{os}, p_{ov} set to 0,99 (the other parameters are set at their default values).

Using a low coverage safety margin we obtain a small synchronization uncertainty at steady state but, as shown by the circles in Fig. 6, it increases the probability that the estimated

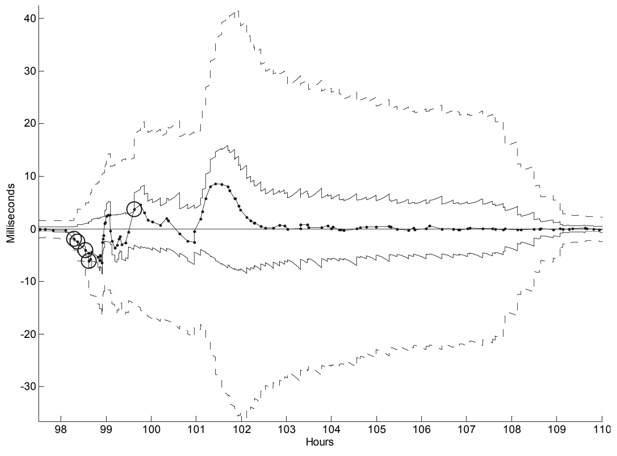


Figure 6. Synchronization uncertainty varying the parameters of the safety margin function.

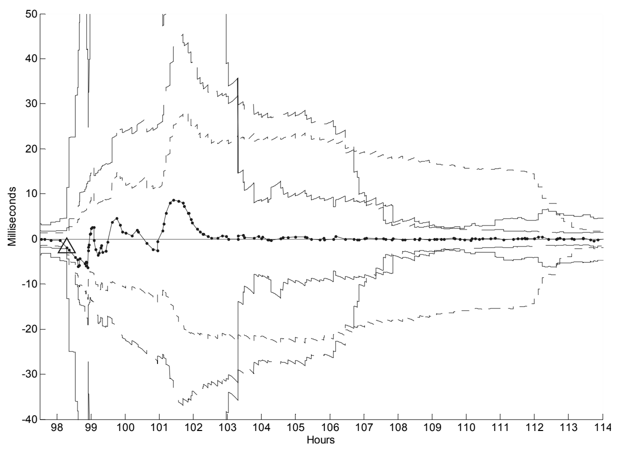


Figure 7. Synchronization uncertainty varying the *N* parameter.

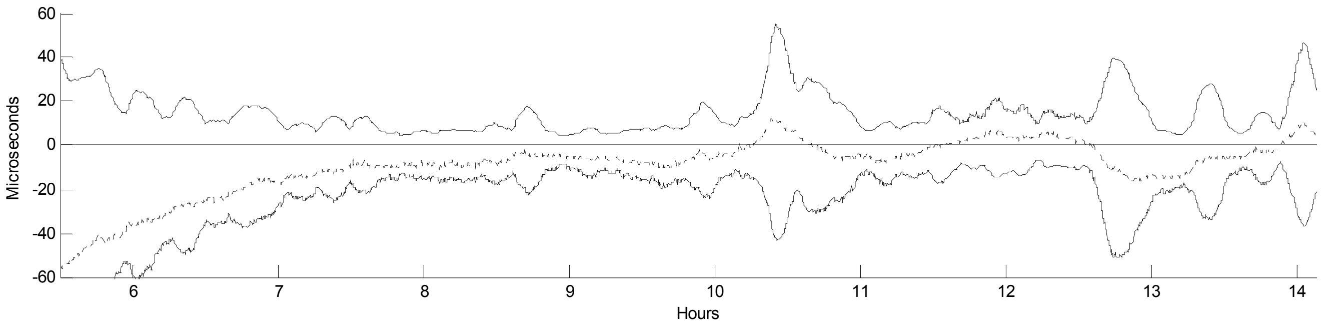


Figure 8. Synchronization uncertainty computed on a PC equipped with a GPS receiver.

offset goes outside the synchronization uncertainty (this happened 5 times in the figure). Hence, it increases the probability that the R&SAClock requirements are violated.

D. *Results varing the N parameter*

Fig. 7 reports the synchronization uncertainty obtained using three different values for the parameter *N* used in the algorithm. The dotted line represents the synchronization uncertainty computed with *N*=80 samples, the dashed one has been computed with *N*=40 samples, while the continuous one has been obtained with *N*=10 samples. The other parameters have the default values.

As it happens for the *M* parameter, decreasing the number of samples produces a larger but more reactive synchronization uncertainty, while considering more samples reduce the size of the interval but at the price of a slower reaction to sudden changes of the offset. In Fig. 7 the triangle at the beginning of the figure shows a failure for the uncertainty computed with *N*=80.

E. *SPS on a GPS-Synchronized pc*

Figures from 2 to 7 show the behavior of the SPS algorithm running on PCs synchronized using remote reference time servers. Fig. 8 shows instead the behavior of this algorithm when the PC is equipped with a GPS receiver. The dashed line represents the estimated offset while the two continuous ones represent the synchronization uncertainty computed with $p_{dv}, p_{ds}, p_{ov}, p_{os} = 0.999999$ and $N, M = 50$.

We observe that while in the other scenarios presented the synchronization data was taken by NTP with variable synchronization interval, in this scenario, using a GPS receiver physically connected to the PC, synchronization is always performed every 16 seconds Thanks to this and to the very short transmission delay from the reference node, the computed synchronization uncertainty is at steady state within $50 - 60\mu s$.

V. CONCLUSIONS

The R&SAClock is a new software clock for resilient time information that provides both current time and current synchronization uncertainty. In this paper we presented the SPS algorithm that, through statistical analysis, provides synchronization uncertainty within a specific coverage; a set of simulations shows the algorithm behavior and its potential in a real usage scenario, showing useful insight about the influence of the initial setup parameters of the algorithm. To determine the specific impact of each initial setup parameter and their interplay is a complex problem that requires deep further analyses. Simulations presented in this paper suggest that other important parameters are the frequency and variance of the synchronization. A more frequent and constant synchronization (16 seconds) coupled with a quite short transmission delay from the reference node allowed to obtain a very small synchronization uncertainty. Moreover, the safety margin parameters are more critical in short-term synchronization, while drift predictor parameters are critical for long time intervals without synchronizations. The number of samples used in the algorithm makes it less or more reactive to sudden variation of estimated offset and estimated drift.

We intend to continue our work moving to validate the behavior of SPS, by testing in a real execution environment. Moreover we will investigate simpler versions of safety margin to trade off complexity and usability of the SPS.

REFERENCES

[1] K. Römer, P. Blum, and L. Meier, “Time synchronization and calibration in wireless sensor networks,” in *Handbook of Sensor Networks: Algorithms and Architectures*, I. Stojmenovic, Ed. John Wiley & Sons, Sep. 2005, pp. 199–237.

[2] BIPM, IEC, IFCC, ISO, IUPAC, and OIML. *ISO International Vocabulary of Basic and General Terms in Metrology (VIM)*, third edition, 2004.

[3] A. Bondavalli, A. Ceccarelli, and L. Falai, "Assuring Resilient Time Synchronization," *Reliable Distributed Systems, 2008. SRDS '08. IEEE Symposium on*, pp.3-12, 6-8 Oct. 2008.

[4] D. Mills. Internet time synchronization: the network time protocol. *IEEE Trans. Communications 39*, 10, pages 1482-1493, 1991.

[5] P. Verissimo and L. Rodriguez. Distributed Systems for System Architects. Kluwer Academic Publisher, 2001.

[6] J. A. Barnes et al, “Characterization of frequency stablity,” *IEEE Trans. Instrum. Meas.*, vol. IM-20, pp. 105-120, 1970.

[7] S.M. Ross, Introduction to probability and statistics for engineers and scientists, 3rd edition, Elsevier Academic Press, 2004.

[8] H. Risken, The Fokker-Planck equation: methods of solutions and applications, Springer, Berlin, 2nd edition, 1989.

[9] N.M. Freris and P.R. Kumar, "Fundamental limits on synchronization of affine clocks in networks," *Decision and Control, 2007 46th IEEE Conference on* , vol., no., pp.921-926, 12-14 Dec. 2007.

[10] D. Veitch, S. Babu, and A. Pàsztor, “Robust synchronization of software clocks across the internet”. In *Proceedings of the 4th ACM SIGCOMM Conference on internet Measurement*, pp 219-232, 2004.