Time and Resilient Master Clocks in Cyber-Physical Systems

Andrea Ceccarelli[1(✉)], Francesco Brancati[2], Bernhard Frömel[3], and Oliver Höftberger[3]

[1] Department of Mathematics and Informatics, University of Florence, Florence, Italy
andrea.ceccarelli@unifi.it
[2] Resiltech SRL, Pisa, Italy
francesco.brancati@resiltech.com
[3] Institute of Computer Engineering, Vienna University of Technology, Vienna, Austria
{froemel,oliver}@vmars.tuwien.ac.at

1 Introduction: Challenges for Time-Aware Cyber-Physical Systems-of-Systems

1.1 On the Role of Time in Cyber-Physical Systems-of-Systems

Since many years, it has been acknowledged that the role of time is fundamental to the design of distributed algorithms [21]. This is exacerbated in cyber-physical distributed systems, and consequently in Systems-of-Systems, where it is sometimes impossible to say which one of two observed environmental events occurred first.

Computers, and consequently constituent systems (CSs), use at least a basic mechanism for local time keeping, in the form of incremental timers. In fact, each autonomous CS has its own oscillator that swings freely and is uncoordinated with respect to the oscillation of the oscillator in any other autonomous CS. These oscillators are often adequate to make local duration measurements, generate alarms by time-out, etc. However, there are reasons for resorting to clocks that give you an absolute notion of time. Clocks are the only way to achieve tightly synchronized actions of an ensemble of nodes in a System-of-Systems [23].

For example, *synchronized clocks* make it possible to measure the duration of some action that starts in one node and ends in another node [22]. This is a common requirement in many applications, where the duration between events that occur in the environment of the different CSs of an SoS must be determined. It is not possible to measure the duration between events that occur in the physical environment of different CSs if no global notion of time of adequate precision is shared by all CSs of the SoS.

If a *global timestamp* is assigned to every significant event, then the duration between any two significant events occurring at any place within the whole SoS can be

This work has been partially supported by the FP7-610535-AMADEOS project.

A. Bondavalli et al. (Eds.): Cyber-Physical Systems of Systems, LNCS 10099, pp. 165–185, 2016.
DOI: 10.1007/978-3-319-47590-5_6

calculated easily. For example, we can consider the temporal validity of real-time data. An observation of a dynamic entity, e.g., the state of traffic light, e.g., green, can only be used for control purposes within a validity interval that depends on the dynamics of the entity (the traffic light). If the observation of the environment is performed by a CS that is different from the CS that uses the observation, then, based on the timestamp of the observation, the user can determine if the given observation is still valid to use at a particular later instant [1, 2]. Given that such a global SoS time is available, this global time can be used to radically simplify the solution of many other temporal coordination problems in an SoS [1, 2].

As an SoS is sensitive to the progression of time, such global notion of time is required in order to [2, 6], amongst other: (i) enable the interpretation of timestamps in the different CSs; (ii) limit the validity of real-time control data; (iii) synchronize input and output actions across nodes, with specific reference to *stigmergic* and *message-based* information exchange; (iv) specify the *temporal properties* of interfaces; (v) perform prompt error detection; (vi) strengthen security protocols; (vii) allocate resources *conflict-free* (e.g., in time-triggered communication, scheduling).

A *dependable global (physical) time* is thus needed to establish the backbone of the temporal infrastructure of an SoS. Every CS in the SoS that is subject to physical time requirements should be able to measure time with an appropriate precision, and achieve a quality of time synchronization which is deemed sufficient [24]. Such a dependable global (physical) time is a fundamental requirement for time-aware SoS, although we remark that it is well-known that it is impossible to precisely synchronize the clocks in a distributed computer system. A measurement error in the timestamps of events is unavoidable. This measurement error can lead to inconsistencies between the actual and recorded temporal order of events [1, 25].

1.2 Towards a Dependable Global Time-Base

In an SoS, *external clock synchronization* is the preferred alternative to establish a global time, since the scope of an SoS is often ill defined and it is not possible to identify *a priori* all CSs that must be involved in the (internal) clock synchronization. A CS that does not share the global time established by a subset of the CSs cannot interpret the timestamps that are produced by this subset. The preferred means of clock synchronization in an SoS is the external synchronization of the local clocks of the CSs with the standardized time signal distributed worldwide by satellite navigation systems, such as GPS, Galileo or GLONASS. Standalone satellite navigation systems are based on receivers processing GNSS (Global Navigation Satellite Systems) satellite signals. GNSS currently have two core constellations: Global Positioning System (GPS) of the United States and the Global Navigation Satellite System (GLONASS) of the Russian Federation. Other similar systems are the upcoming European Galileo positioning system, the Japanese Quasi-Zenith Satellite System (QZSS), and the proposed COMPASS-Bediou Navigation System of China.

Reasons which may affect the availability and signal quality of the standalone satellite navigation systems and related algorithms, and consequently quality of clock synchronization, have been extensively discussed in literature e.g., a comprehensive

overview can be found in [18, 19]. Special considerations related to the availability of GPS signal refer specifically to mobile CSs. GPS is not available in building or under roofing (e.g., in a wood), which is very likely to (temporarily) happen for mobile CSs. Such mobile CSs that operate on batteries are energy sensitive. Since a GPS sensor is very expensive in terms of consumed energy these mobile applications can benefit by switching off the GPS whenever possible and as long as possible, still maintaining the required synchronization quality.

Additionally, it should be considered that GPS may be subject to deliberate attacks, which even when detected timely they still make the GPS signal unavailable for the whole duration of the attack [5]. Amongst these, we mention [20] (i) jamming GNSS based vehicle tracking devices to prevent a supervisor's knowledge of a driver's movements, or avoiding road user charging; (ii) rebroadcasting ('meaconing') a GNSS signal maliciously, accidentally or to improve reception but causing misreporting of a position; (iii) spoofing GNSS signals to create a controllable misreporting.

In a report from the US Government Accountability Office (GAO) to the US Congress [3] on *"GPS-Disruption—Efforts to Assess Risks to Critical Infrastructure and Coordinate Agency Action Should be Enhanced"* it is pointed out that many of the large infrastructure SoSs in the US are already using GPS time synchronization on a wide scale and a disruption of the GPS signals could have a catastrophic effect on the infrastructure. In this report it is noted that a global notion of time is required in nearly all infrastructure SoSs, such as telecommunication, transportation, energy, etc. and this essential requirement has been met by gradually using more and more often the time signals provided by GPS, not considering what consequences a disruption of the time distribution, either accidental or intentional, has on the overall availability and function of the infrastructure [6].

Even more recently, on 4 February 2016 the BBC reported that "several companies were hit by hours of system warnings after 15 GPS satellites broadcast the wrong time, according to time-monitoring company Chronos" [4]. This led to serious problems to many companies that resulted in money loss.

These events just confirm existing warnings from different communities. For example, a Report from the UK Royal Academy of Engineering in 2011 [20] suggests that the U.K. may have become dangerously over-reliant on satellite-navigation signals, and too many applications have little or no back-up were these signals to go down. The report concludes that several concerns are bounded to GNSS. First, non-GNSS based back-ups are often absent, inadequately exercised or inadequately maintained". Second, that the jammers are easily available and that most jammers are able to block GPS, GLONASS and GALILEO. Third, a full picture of the dependencies on GPS and similar systems is missing.

Starting from the above concern, we can conclude that *we should not entirely rely on satellite navigation systems to build a dependable global time base in time-aware SoS*. Following this observation, this Chapter presents the design and development of a resilient fail-silent master clock based on satellite-based time synchronization (e.g., GPS or Galileo signals), to provide a dependable global time base for cyber-physical Systems-of-Systems. Such Resilient Master Clock (RMC) is intended to feature low power consumption, low weight and low cost. The RMC should be built with hardware

Off-the-Shelf (OTS), as for example COTS MEMS sensors, and whenever possible software OTS.

The RMC includes an independent oscillator and GPS devices complemented by acceptance tests. In fact, software clock control techniques are devised in order to:

1. Provide a self-estimation of the quality of clock synchronization. This is achieved via the Reliable and Self-Aware Clock (R&SAClock), which acts as an oracle of the quality of clock synchronization. R&SAClock keeps nodes of a network aware about the quality of synchronization: it monitors the synchronization level of the local clock with respect to a global time reference (like the Temps Atomique International, TAI).
2. Extend the holdover duration of the clock by compensation of local clock deviations, especially in case of absence of the GPS signals. In fact, clock (crystal oscillators) deviations may be caused by physical environmental variations, like temperature, pressure, humidity variations, voltage. Correction techniques based on COTS sensors are introduced to compensate local clock deviations and avoid unsynchronized clocks in SoSs in the absence of GPS signals.

When the satellite-based time synchronization signal fails (or it is corrupted by a security incident), for a certain amount of time, the RMC is able to maintain its clock close to the global time within a required accuracy until the satellites signal becomes available again.

The rest of the section is organized as follows. Section 2 presents the architecture design of the Resilient Master Clock. The successive Sects. 3 and 4 explore the main technical solutions that are included in the Resilient Master Clock, that are respectively intended to discipline the clock when the GPS signal is unavailable, and to provide a self-estimation on synchronization uncertainty. Section 5 presents conclusions.

2 Resilient Master Clock (RMC) Architecture

This section discusses the architecture of the RMC. The presentation of the architecture is generic i.e., it is not bound to a specific board or pieces of hardware. For example, in Sects. 3 and 4, two different instantiations of (part of) the RMC on two different boards will be described.

The architecture of the RMC is represented in Fig. 1, and described below. In Fig. 1, the components in light grey are required hardware and software components that should be available for the selected OTS board. Dark grey components identify instead the components that have been devised and developed when building the Resilient Master Clock.

The architecture of the RMC is divided in three layers: the *board* (or Hardware layer), the *Operating System* (or OS layer) and the *Middleware*. Each layer consists of the different constituent blocks which are herewith described.

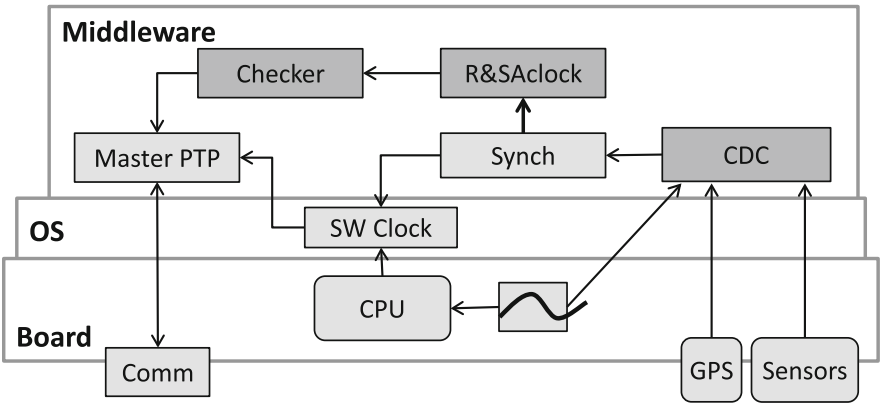

Fig. 1. Resilient Master Clock architecture.

Board Layer. The building blocks of this layer are hardware components. These are:

- A *GPS module* for receiving time messages by the GPS satellite constellation. The messages are then provided to any enquiring hardware or software along with a one-pulse-per-second (1PPS) signal.
- *Sensors*. Sensors for acquiring information about the environment. Examples are temperature and pressure sensors.
- *Comm*. This block refers to the communication interface. For example, an Ethernet Network Interface Card (NIC) can connect the RMC to a network in which CSs slaves wait for time synchronization packets from the RMC, which acts as a master clock node.
- *CPU, Memory and the physical oscillator*. These are standard components of any hardware board. The physical clock is particularly relevant in our context.

OS Layer. This layer includes a local software clock (*SW Clock*) which is usually created by Operating Systems starting from the hardware clock and that provides the timestamps to the services executing on the board.

Middleware Layer. This layer includes OTS SW components (*Synch*, *master PTP*) and SW components that are specific for the RMC. In particular, these are the following.

Synch. The Network Time Protocol (NTP [17]) is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks. NTP is intended to synchronize all participating computers to within a few milliseconds of Coordinated Universal Time (UTC). Since RMC is based on TAI, it should be remarked that conversion UTC-TAI is trivial [26]. A *synchronization module* (*Synch*) based on the Network Time Protocol (NTP) uses the GPS time signals to discipline the local clock.

MasterPTP. The Precision Time Protocol (PTP, [7]) is a protocol used to synchronize clocks throughout a computer network. On a local area network, it achieves clock accuracy in the sub-microsecond range, making it suitable for measurement and control

systems. PTP is defined in the IEEE 1588-2008 standard. A *master Precision Time Protocol (PTP)* module is available on the RMC, and it allows broadcasting a time synchronization packet according to the protocol IEEE 1588 PTP to the nodes of the subnetwork to which the board is connected through *Comm*.

R&SAClock. The *R&SAclock* uses the offset and drift obtained from the synchronization module to estimate the uncertainty of the time provided by the local clock over time.

Clock Drift Compensation (CDC). The *CDC* module generate a clock-drift compensated Pulse Per Second (PPS) signal when the GPS signal is unavailable. The compensation mechanism provided by the CDC module is based on: (i) the values measured by the dedicated sensors (e.g., temperature), and (ii) a-priori knowledge of the frequency deviation caused by environmental changes on the onboard crystal oscillator (e.g., temperature variations).

Checker. A *checker* module checks the uncertainty associated to the time of the local clock provided by the R&SAclock; consequently, it decides if the RMC can be considered a reliable time source and allows or blocks the PTP synchronization. For example, it can be implemented as a process which periodically checks the quality of the local time provided by the R&SAClock. On the other hand, when the quality of the local time is outside acceptable thresholds, the PTP synchronization beans must be stopped because the RMC cannot be considered a time reference.

3 The Clock Drift Compensation Module

This section discusses the Clock Drift Compensation (CDC) module that provides a periodic Pulse Per Second (PPS) time signal to the Synch module (cf. Fig. 1). During normal operation, the CDC module forwards the high quality, externally provided time signal (e.g., generated by a GPS receiver). In case the externally provided time signal becomes unavailable, the CDC module switches seamlessly without interruption of the output PPS signal to holdover mode until the external time signal becomes available again. During holdover mode the CDC module internally generates the output PPS time signal for the Sync module from a local clock based on a common (quartz) crystal oscillator. This local clock is drift compensated with respect to the – now in holdover mode unavailable – external time signal. The drift compensation improves the precision of the internally generated PPS time signal and consequently allows for a prolonged holdover duration compared to a crystal oscillator based clock that is not clock drift compensated.

In the following subsections we detail our clock drift compensation method and present a proof-of-concept prototype implementation which we used for calibrating and evaluating the CDC module.

3.1 Compensating the Drift of Clocks Based on Crystal Oscillators

Clocks in computer systems are usually realized by a digital counter register and a crystal oscillator whereas each oscillation generates a tick event that increments the counter register. The oscillator frequency output slightly deviates from its designed nominal frequency output, because of (1) mechanical imperfections introduced during manufacturing of the oscillator, (2) dynamic deviations caused by aging of the oscillator, and (3) environmental conditions (e.g., temperature, acceleration, humidity) acting on the oscillator. The static and dynamic deviations from the nominal frequency are the cause for clock drift: Any two clocks of the same design, even when perfectly started at the same instant, will eventually drift apart as time progresses.

For establishing a global time in Cyber-Physical Systems-of-Systems (CPSoSs), we need to periodically resynchronize the local clocks of the Constituent Systems (CSs) by external clock synchronization (e.g., synchronization with the GPS time source). The resynchronization is necessary to ensure a bounded offset, also called precision, among the clocks of the CSs. A critical parameter of clock synchronization is the resynchronization period which needs to be short enough to keep all clocks within the required precision. By actively compensating the clock drift, the duration between two resynchronization instants can be increased. This is of particular interest if the source for external synchronization is unavailable (e.g., losing the GPS signal when driving into a tunnel, turning the GPS receiver off to save power), while the synchronization precision of the clocks of the CSs has to be maintained until the external source becomes available again.

Drift Compensation Method. In order to compensate clock drift, the effects of internal and external sources of oscillator frequency deviations must be negated. This requires to measure these effects or know a-priori about them, but also to apply corrective actions on the clock. There are two options to apply corrections: (1) control the enclosing environment of the oscillator (e.g., oven-controlled or voltage-controlled oscillators) such that the output frequency is corrected, or (2) to periodically correct the counter register by adding each correction period a correction value that compensates for the frequency deviations.

The first approach requires, additionally to the measurement of the enclosing environment, possibly expensive and power-demanding actuation hardware (e.g., heat source), insulation, or is in case of some effects (e.g., acceleration) infeasible. However, for some effects (for example, temperature, humidity, pressure, aging) this approach is technically more simple, as it only requires to steer the oscillator frequency close to the external clock source and then maintain the same environmental conditions during holdover mode.

The clock drift compensation of the CDC module is based on the second correction approach which also depends on measuring the oscillator environment, but – besides that – can be realized purely in software. This software implements a compensation model which predicts for each correction period an accurate correction value in clock ticks. To achieve a high level of accuracy, the clock drift compensation model fuses several sources of information (a-priori knowledge, sensor observations).

Important parameters of our drift compensation method – besides the drift compensation model – are the tick rate of the external time signal which should be forwarded as the output tick by the CDC module during normal operation and generated when absent during holdover mode, the correction period, and the internal tick rate or oscillator frequency. The output tick rate determines the counter register size, i.e., this register needs to be able to count the number of internal ticks that correspond to one output tick. The output tick rate also determines the correction period, because corrections predicted by the compensation model should be applied for each output tick to avoid imprecision effects. Finally, in order to have only a small discretization error, the clock compensation method assumes that the output tick rate (e.g., 1 Hz) is much lower than the internal tick rate (e.g., at least a few kHz or MHz).

Compensation Model. The clock drift compensation model is based on (1) a-priori knowledge about the oscillator (e.g., manufacturing defects, aging behavior, known effects of environmental conditions), and (2) on the currently observed external time signal and the environmental conditions. Environmental conditions that can be observed by sensors are for example: temperature, barometric pressure, acceleration, air humidity.

Explicitly defining a compensation model would be a tremendous task, because it requires precise knowledge about all relevant physical properties of the crystal oscillator, their interrelationships, and the involved sensors which – similarly to the oscillator – also slightly deviate from their designed characteristics (measurement errors that depend on currently prevailing environmental conditions). Consequently, we focus on defining and parameterizing the compensation model by using classical machine learning techniques where, for example, regression (curve fitting) for independent input variables, or artificial neural networks for input variables with unknown dependency relationships are available. The selection of a concrete machine learning technique depends on the available sensors, available computational resources and the required compensation quality. Regardless of the technique, training of the compensation model is necessary by taking a set of input instances (e.g., an observation of the environmental conditions) and adjusting the model parameters such that it maps the input data to a known correction value for the counter register. The trained model is then available during holdover mode for predicting correction values for new input data where the correction value wasn't known before.

For model training there are two methods that should be applied in combination:

- *Offline-Learning/Calibration*: After manufacturing the CDC module, including its oscillator and the sensors, obtains training data by observing (using its own sensors) the controlled environmental conditions. Under these controlled environmental conditions, the oscillator frequency deviation is measured by external equipment (e.g., an oscilloscope that measures the difference of the external time signal with respect to the internally generated output tick during holdover). Training data is collected by doing various sweeps of the controlled environmental conditions through the range of expected environmental conditions and recording the sensor observations together with the necessary correction value. Offline learning initializes the compensation model with a-priori knowledge.

- *Online-Learning/Adaptation*: During operation of the CDC module the compensation model can be adjusted by constantly retraining it, when the external time signal is available. Online-learning compensates for aging effects of the oscillator and involved sensors. Also online-learning possibly allows for limiting the necessary offline-learning to a smaller sample size of a production series where only small model deviations are expected among individual CDC modules.

3.2 Proof-of-Concept Prototype

The proof-of-concept prototype is based on an implementation of the CDC module on the SmartAP 2.0[1], a small embedded system originally intended for auto-piloting small aircrafts. It consists of a STM32 ARM Cortex M4 microcontroller integrated with two external quartz oscillators (84 MHz, and 32.768 kHz), and sensors to measure acceleration (Invensense MPU-6050/ MPU-9150), barometric pressure (MS5611-01BA03). We customized the board by adding sensors to additionally measure temperature and air humidity (Sensirion SHT75). As an external time signal we used an UBLOX LEA-6H GPS receiver. For wirelessly communicating with the board we added a Bluetooth module (Microchip RN42).

The microcontroller implements a data recording functionality to obtain the measured training data, and the drift compensation method including a simple variant of a compensation model. In this prototype we did not implement online-learning, because the effects of online-learning are minor, if the compensation model is well calibrated and the CDC module prototype has not been left several months for aging between calibration and evaluation.

Compensation Model Implementation. Figure 2 illustrates a simple compensation model based on look-up tables where averaged training data can be deployed directly. For each environmental condition a look-up table exists from which the contribution of the measured parameter to the clock drift is obtained. Temperature is codependent with all other environmental conditions. Consequently, the lookup tables for pressure, humidity, and acceleration contain temperature dependent correction values. To estimate the clock drift for the current correction period, the individual contributions are summed up and added to the constant drift value of the crystal oscillator.

Training of the Compensation Model. The compensation model training data is obtained by placing the CDC module with its sensors in an experimental chamber within which the environmental conditions can be controlled. For each correction period an oscilloscope records the deviation of the internally generated output time signal from the external time signal (GPS receiver).

Figure 3 shows our prototype of such an experimental chamber. It consists of a thermally insulated box with a Peltier element for heating and cooling, an air pump to produce an over or under pressure inside, and a humidifier. To achieve a constant acceleration force on the oscillator crystal, the board is mounted on a plate that can be

[1] http://sky-drones.com/autopilots/9-smartap-autopilot-20.html.

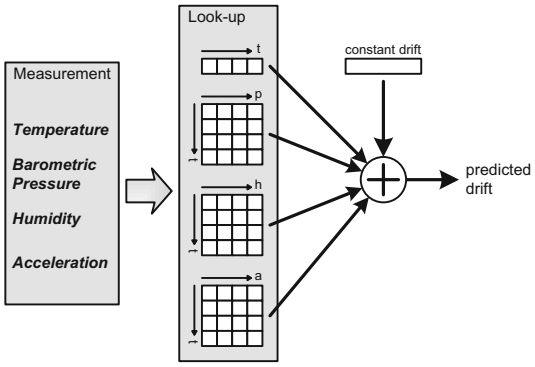

Fig. 2. Clock drift compensation model based on look-up tables.

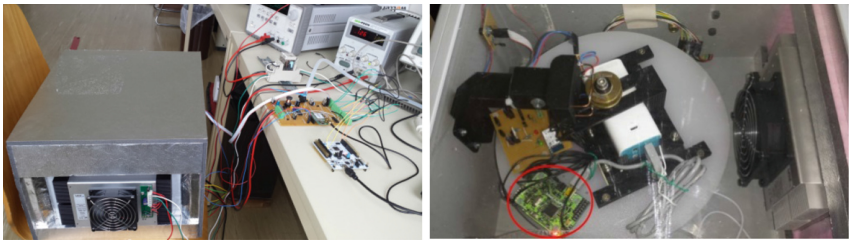

Fig. 3. Experimental chamber for obtaining training data for the compensation model. On the right the prototype of the CDC module has been highlighted with a red circle and is mounted on the rotating plate.

rotated. Acceleration forces can be applied on the oscillator in X and Y directions, depending on how the board is mounted on the rotating plate.

By using this box, one environmental condition after the other is varied from its minimum value up to the maximum, and vice versa. For each set point the number of oscillations of the crystal oscillator, which drives the clock, is counted several times. The reading and resetting of this counter is triggered every second by the PPS time pulse originating from the GPS receiver.

From the recorded training data, the constant drift at certain environmental conditions – i.e., the zero conditions C_0 – is obtained. C_0 can be selected arbitrarily. Furthermore, for each condition that can be observed a table entry (see lookup-based compensation model in the previous section) is derived by averaging over the training data that have been recorded for the same environmental conditions. These table entries indicate the additional drift if the conditions deviate from the zero conditions C_0.

3.3 Evaluation

The evaluation of the CDC module is conducted using the abovementioned experimental chamber. In particular, only results for temperature and acceleration in X

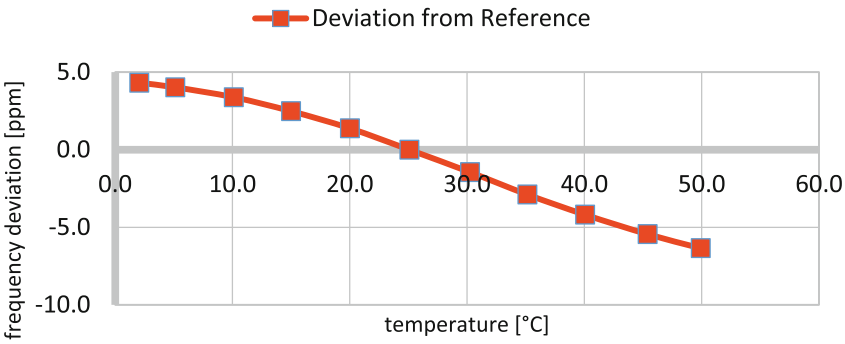

Fig. 4. Frequency deviation with respect to temperature.

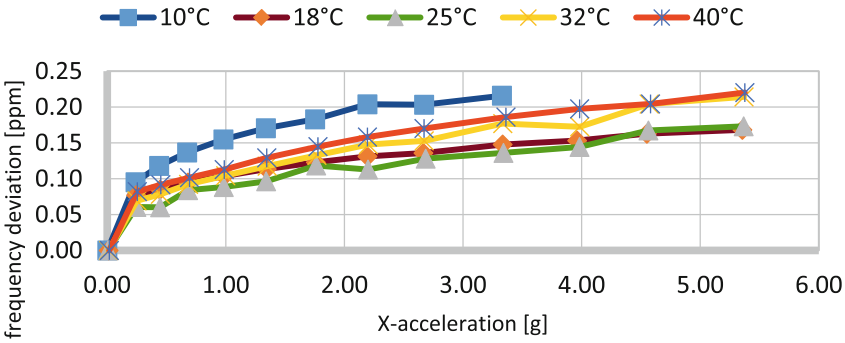

Fig. 5. Frequency deviation with respect to acceleration in X-axis at different temperatures.

direction for the 84 MHz quartz oscillator are shown here, while information also on other environmental conditions (Y-acceleration, air humidity, barometric pressure) and the second oscillator can be found in [15].

Obtaining Training Data. The investigated oscillator has a measured variance (i.e., variations in the number of oscillations when environmental conditions are stable) around its nominal frequency of approximately 0.02381 ppm. In the following we present frequency deviations of the oscillator when doing a temperature sweep from 0° C to 50° C and an acceleration sweep from 0 g to 5 g (g is the weight per mass unit).

Figure 4 depicts the frequency deviation over the temperature range. If, for instance, the frequency deviation is about 2.5 ppm (e.g., at 15° C), it means that in each second a clock driven by this oscillator deviates 2.5 µs from a clock with perfect accuracy. Without clock synchronization, this deviation sums up to 9 ms/h, or 0.216 s/day. In Fig. 4, a positive value of the deviation denotes an increase in the number of oscillations compared to the nominal value.

Figure 5 shows the frequency deviation of the oscillator concerning variations in the acceleration force and temperature.

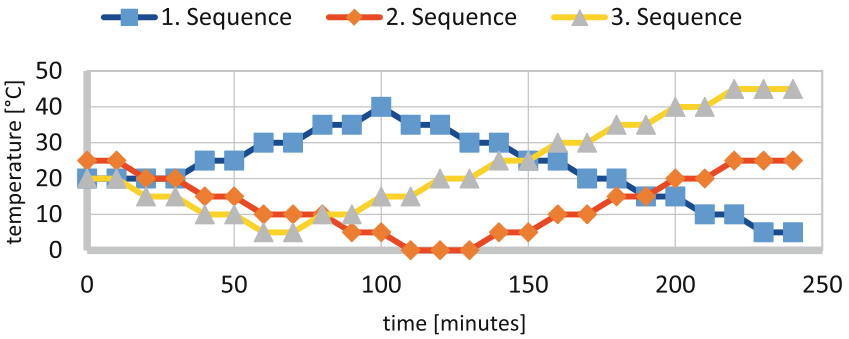

Fig. 6. Three test sequences of temperature set points.

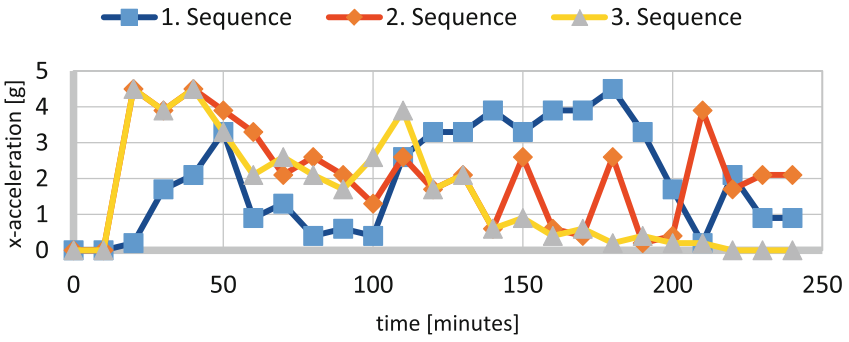

Fig. 7. Three test sequences of X-acceleration set points.

The results show that additionally to the significant correlation between the temperature and the frequency deviation, the oscillator exhibited deviations concerning changes in acceleration (and pressure and humidity, see [15]). All these deviations have been higher than the variance of the oscillator at stable environmental conditions.

Evaluation of Compensation Model. The evaluation of the compensation model carried out by recording the compensation performance over a set of test sequences where the controllable environmental conditions are varied. Each of the test sequences takes 240 min. Figures 6 and 7 show three of such sequences for the environmental conditions temperature and acceleration in X direction.

Figure 8 shows the compensation performance of the CDC module in holdover mode only for the first test sequence (1. Sequence), because all other test sequences gave similar results. In Fig. 8 we plotted the results for two different compensation models: basic compensation (only constant drift correction), and temperature, acceleration and pressure (T&A&P). Also for comparison reasons, the deviation is depicted, when no compensation mechanism is applied. Different effects during manufacturing or aging of the oscillator crystal lead to a permanent deviation from the nominal frequency (here it is faster than the reference), which is about 60 ppm.

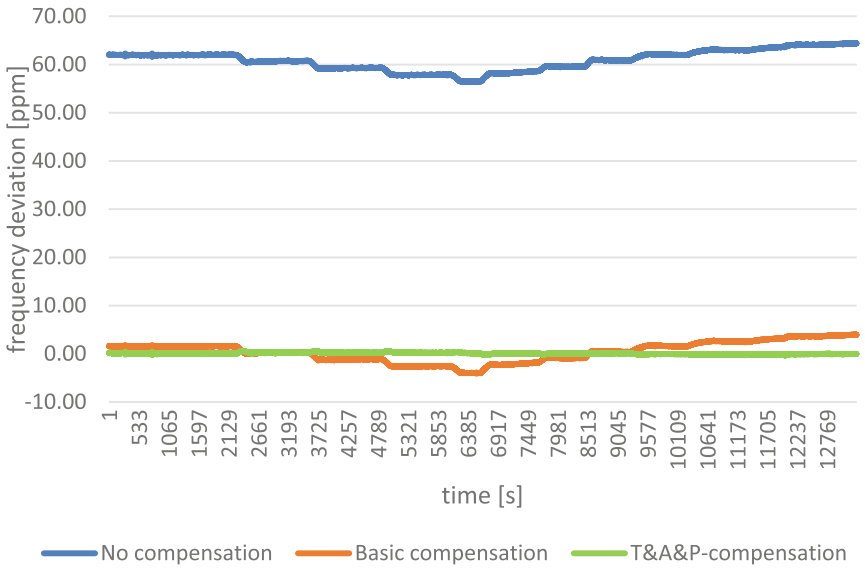

Fig. 8. Frequency deviation under different compensation models.

Clearly, basic compensation has the largest corrective effect and improved the mean drift rate of the oscillator from an order of magnitude of about 10^{-5} to approximately 10^{-6}. The compensation model that regarded more of the investigated environmental conditions (T&A&P) gave a better result, as the oscillator's drift rate is now in the 10^{-7} order of magnitude.

Even more improved results should be easily achievable when using more sophisticated compensation models, sensors of better quality, and implementing online-learning for self-fine-tuning. Consequently, this evaluation gives strong support to the benefits of our proposed clock drift compensation method.

3.4 Summary of Main Findings on the Clock Drift Compensation Module

The proof-of-concept prototype confirms that the compensation of frequency deviations of crystal oscillators by passive observation of the surrounding environmental conditions, and using a trained compensation model leads to a significant decrease of clock drift. The different environmental conditions indeed have an effect on the stability of the oscillator and some of these effects can be reduced when these conditions are known. Available protocols (e.g., the Network Time Protocol – NTP) are able to compensate constant drifts of the local clock of a computer system, if the environmental conditions are not changed after the reference clock is disconnected. However, many CPSoSs operate in environments, within which it is infeasible – or at least only with a considerable effort (e.g., by constantly heating the crystal oscillator) – to keep these conditions stable. In contrast, sensors to determine the environmental conditions are often already available in those systems, or can be installed at relatively low cost.

While the presented improvement by temperature, acceleration and pressure compensation are already promising, further experiments have to be performed with more sophisticated compensation models and a more advanced experimental equipment that allows higher ranges of pressure, the up- and down variation of humidity, as well as experiments under other environmental conditions (e.g., vibration, radiation, electromagnetic fields).

4 Reliable and Self-Aware Clock (R&SAClock) Module

The Reliable and Self Aware Clock (R&SAClock, [12]) is a software component that provides IEEE 1588 compliant techniques for the analysis and improvement of the synchronization quality among CSs interacting in SoS. The R&SAClock exploits statistical information in order to provide information about uncertainty of the current time view.

Generally, in several contexts such as industrial automation, telecommunication or energy distribution, SoSs require an accurate synchronization of their CSs in order to assure the adequate Quality of Service (QoS). The statistical information collected by R&SAClock to estimate synchronization uncertainty [8] is used as feedback about quality of synchronization. The CSs equipped with R&SAClock are continuously updated about the current synchronization performance.

4.1 General Concepts on R&SAClock

A CS uses R&SAClock to acquire both the time value and synchronization uncertainty associated with the time value.

For clarity, we report basic notions on time and clocks that are used in the rest of this section. Noteworthy, the terminology is consistent with the terms defined in Chap. 1. Figure 9 below is introduced to better clarify relevant aspects.

The *global time* is an abstraction of physical time in a distributed computer system; it is the unique time view shared by the CSs. The *reference clock* is a *working hypothesis* for measuring the instant of occurrence of an event of interest: it is a clock that always holds the global time. We can say that the *reference node* is the CS that owns the reference clock. Also, 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* of two events denotes the duration between two events and the position of the second event with respect to the first event on the timeline; the offset $\Theta_c(t) = t - c(t)$ is the actual distance of local clock c of the CS node n from the global time at time. This distance may vary through time.

Accuracy A_c of clock c denotes the maximum offset of a given clock from the external time reference, measured by the reference clock. An upper bound of the offset adopted in the definition of system requirements and therefore targeted by clock synchronization mechanisms.

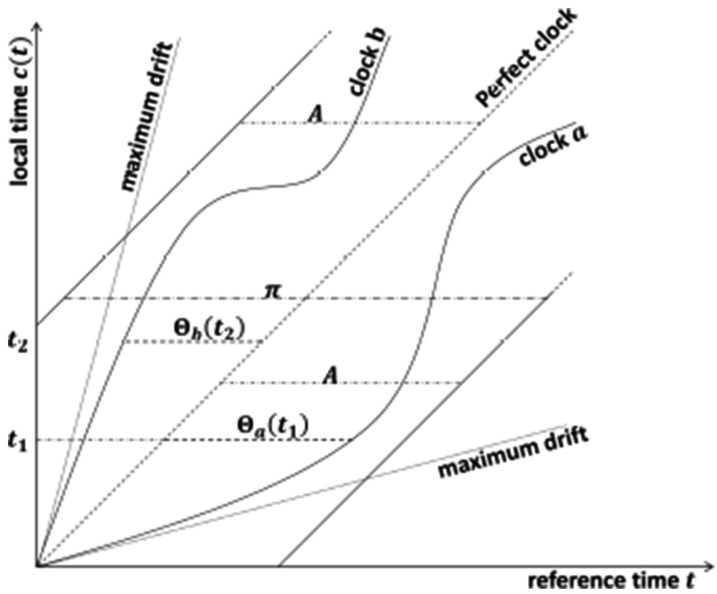

Fig. 9. Basic notions on time and clocks.

The precision π of an ensemble of synchronized clocks denotes the maximum offset of (distance) respective ticks of the global time of any two clocks of the considered clock ensemble.

The drift $\rho_c(t)$ of a physical clock describes the frequency ratio between the physical clock and the reference clock i.e., the rate of deviation of a local clock c at time t from global time [10].

Synchronization uncertainty $U_c(t)$ is defined as an adaptive and conservative evaluation of the offset $\Theta_c(t)$ at any time t; uncertainty is such that $A_c \geq U_c(t) \geq |\Theta_c(t)| \geq 0$ [8]. Hence, accuracy Ac is an upper bound of uncertainty $U_c(t)$ and consequently of the absolute value of the offset $\Theta_c(t)$.

When a CS asks the current time to R&SAClock, the latter provides an enriched time value useful for time synchronization. The enriched time value is composed of a set of values: *likelyTime*, *minTime*, *maxTime* and *FLAG*. *LikelyTime* is the time value computed reading the local clock. *minTime* and *maxTime* represent left and right synchronization uncertainty margins with respect to *likelyTime*. They are based on synchronization uncertainty provided by the internal mechanisms of R&SAClock. Finally, the *FLAG* takes the value 1 if requirements on uncertainty are satisfied, 0 otherwise. Details on R&SAClock and its implementation can be found in [8, 11].

It is evident that the main core of R&SAClock is the uncertainty evaluation algorithm that equips R&SAClock with the ability to compute the uncertainty. Such an algorithm relies on the Statistical Predictor and Safety Margin (SPS) algorithm.

Each CS that uses the R&SAClock *getTime* method for getting synchronization information and each CS has the two main expectations: (*i*) a request for the time value

Table 1. Requirements for R&SAClock

Req. ID	R&SAClock requirement description
REquation 1	The service response time provided by R&SAClock is bounded: there exists a maximum reply time ΔRT from a *getTime* request made by a CS user to the delivery of the enriched time value (the probability that the *getTime* is not provided within ΔRT is negligible)
REquation 2	For any *minTime* and *maxTime* in any enriched time value generated at time t, it must be *minTime* ≤ t ≤ *maxTime* with a coverage ΔCV (by coverage we mean the probability that this equation is true). In other words, given *likelyTime* = c(t), the true time t must be guaranteed within the interval [*minTime*, *maxTime*] with a coverage ΔCV

should be satisfied quickly, and (*ii*) the enriched time value should include the correct real time. These are formally expressed by the two requirements in Table 1.

4.2 The Statistical Predictor and Safety Margin (SPS)

In the following the SPS algorithm is briefly described for a local software clock c that is disciplined by an external clock synchronization mechanism. SPS computes the uncertainty at a time t with a coverage, intended as the probability that $A_c \geq U_c(t) \geq |\Theta_c(t)| \geq 0$ holds. The computed uncertainty is composed by three quantities: (i) the *estimated offset*, (ii) the output of a *predictor* function, P and (iii) the output of a *safety margin* function, SM. The computation of synchronization uncertainty requires a right uncertainty $U_r(t)$ and a left uncertainty $U_l(t)$: consequently, SPS has a right predictor with a right safety margin for right uncertainty, and a left predictor with a left safety margin for left uncertainty. The output of the SPS at $t \geq t_0$ is constituted by the two values:

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

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

The *estimated offset* $\tilde{\Theta}(t_0)$ is computed by the synchronization mechanism and can contain errors. If the estimated offset is positive, it influences the computation of an 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, $P_r(t)$ and $P_l(t)$, 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 oscillator drift is modelled with the random walk frequency noise model, one of the five canonical models used to model oscillators (the power-law models [14]), that we considered as appropriate and used. Obviously the parameters of this random walk are unknown and depend on the specific oscillator used. They are computed resorting to the observation of the last m samples of the drift (where m smaller or equal to the set-up

Table 2. SPS parameters

Symbol	Definition
t_0	time in which the most recent synchronization is performed
$\tilde{\Theta}(t_0)$	estimated offset at time t_0
$\tilde{\rho}(t_0)$	estimated drift at time t_0
M, m	maximum and current number of (most recent) samples of the estimated drift that the UEA collects ($0 < m \leq M$)
N, n	maximum and current number of (most recent) samples of the estimated offset that the UEA collects ($0 < n \leq N$)
p_{ds}	probability that the population variance of the estimated drift is smaller than a safe bound on such variance
p_{dv}	a safe bound of the drift variation since t_0 is computed with probability p_{dv}
$p_{\mathrm{ds}} \circ p_{\mathrm{dv}}$	the joint probability of these two values represents the coverage of the prediction function
p_{os}	probability that the population variance of the estimated offset is smaller than a safe bound on the variance
p_{ov}	a safe bound of the offset at t_0 is computed with probability p_{ov}
$p_{\mathrm{os}} \circ p_{\mathrm{ov}}$	the joint probability of these two values represents the coverage of the safety margin function

parameter M), and using a safe bound on the population variance of the estimated drift values. The coverage of this safe bound depends on the set-up probabilities p_{ds} and p_{dv} defined in Table 2 together other main quantities involved in the SPS algorithm.

The *safety margin* functions $SM_r(t_0)$ and $SM_l(t_0)$ aim at compensating possible errors in the prediction or in the offset estimation. The safety margin function is computed starting from the collection of the last n samples of the estimated offset (where n is smaller or equal to the set-up parameter N). A safe bound to the population variance of the estimated offset is computed. The coverage of this safe bound depends on the set-up probabilities p_{os} and p_{ov} (see Table 2).

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

4.3 Proof-of-Concept and Exemplary Runs

The R&SAClock has been implemented in the Beagle Bone [16] board with Debian OS as described in the AMADEOS deliverable D4.4 [15]. The R&SAClock uses GPS for clock synchronization; it acquires data on the estimated offset and drift from the Network Time Protocol (NTP) [9] component.

When the synchronization uncertainty exceeds a given threshold, the Checker module is notified by reading the FLAG field of the enriched time value. The communication channel between the Checker and the R&SAClock is socket-based.

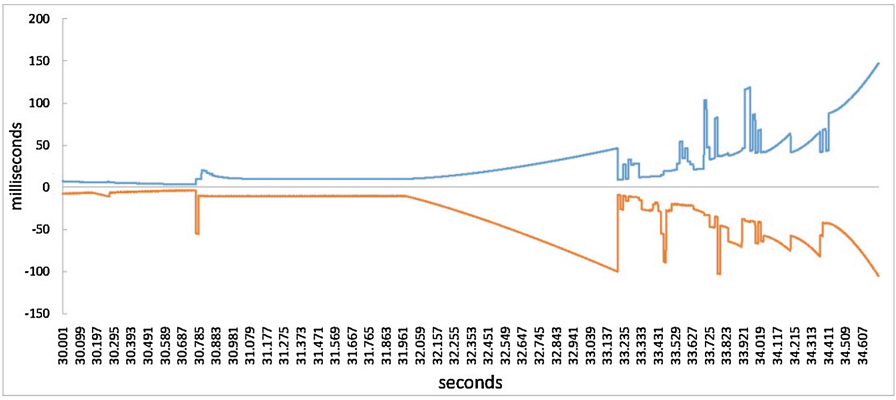

Fig. 10. Exemplary run of R&SAClock on the proof-of-concept.

In the following we show an exemplary run with the R&SAClock executing on the Beagle Bone board proof-of-concept. We acknowledge that an extensive assessment activity is required [13] (*i*) to give evidence that the defined software executing on the proof-of-concepts satisfies the identified requirements, and (*ii*) to opportunely tune the R&SAClock parameters (Table 2). Still, we present this exemplary run because it explains intuitively the behavior of R&SAClock.

In the considered run, reported in Fig. 10, the *x*-axis (in seconds) corresponds to the *likelyTime* collected reading the local clock. The *maxTime* and *minTime* computed by R&SAClock are respectively the two lines above and below the *x*-axis. In fact, the *y*-axis (in milliseconds) shows the *maxTime* and *minTime* with respect to *likelyTime* i.e., *maxTime-likelyTime* and *likelyTime* - *minTime*. The *FLAG* value is not shown in this figure.

In the present run, the R&SAClock was already running for 30000 s (see x-axis). As long as the connection with the GPS signal is stable, NTP reliably disciplines the local clock: an accurate estimation of offset is provided, and synchronization uncertainty is a small interval (in the order of few microseconds or less). As it can be shown in the time interval between second 30000 and 30600, the synchronization uncertainty is slightly reduced through time. In fact, the R&SAClock "studies" the past behavior of the local clock, understand that it is overall stable and trustable, and reduces synchronization uncertainty.

Instead, at approximately second 30700, an instability in the local clock and NTP is detected, most likely due to the temporary unavailability of the GPS signal (signal loss). Synchronization uncertainty is increased, because no fresh information on the clock behavior w.r.t. the reference time is provided.

At approximately second 31900, the synchronization uncertainty steadily increases through time. In fact, GPS signal is lost and no fresh information from the time source is provided. There are no guarantees that the clock is disciplined correctly.

When the GPS signal is newly available (approximately at second 33200), a new accurate estimation of the offset is provided. Consequently, the synchronization uncertainty is reduced again. However, from now on, the synchronization with the GPS

is unstable: there are only few, sparse synchronizations. This determines the behavior of the R&SAClock, which cannot trust the local clock and consequently the synchronization uncertainty grows.

4.4 Summary of Main Findings on R&SAClock

The R&SAClock was initially proposed in [8, 11, 12] to monitor the software clock in distributed system. In such works, R&SAClock was implemented and exercised on a fixed node, and with the intention of supporting only the node itself.

Instead, in the RMC, the R&SAClock is intended to operate as a failure detector for a Master clock: in other words, the Checker module of the RMC can read the FLAG value of the R&SAClock, and decide if the RMC can act as a master clock or not.

In addition, we implemented the R&SAClock on a light board which has small requirements for power consumption. This improves the range of applicability of the R&SAClock w.r.t. the previous environments described in [12], which are distributed servers. The experiments, although still preliminary, confirm that the R&SAClock behaves as expected also confirming, in a different environment, the results shown in [12].

5 Conclusions

This Chapter discussed the role of time in Systems-of-Systems. Building on the terms, definition and knowledge defined in Chap. 1, this Chapter identified motivation, with examples, for the prominent role of time and clocks in time-aware Systems-of-Systems. Further, the Chapter discussed the challenges of resilient time keeping and it presents the Resilient Master Clock (RMC), a hardware-software solution that acts as an accurate, fail-silent global time base which is externally synchronized to a satellite-based time source.

The design of the RMC is presented, its main algorithm illustrated including results from the execution on two different prototypes. Although significant work is still needed to consolidate results, the RMC appears a promising approach to provide a low-cost, low-power consumption solution for resilient time-keeping and resilient master clock in Systems-of-Systems.

References

1. Kopetz, H.: Why a Global Time is Needed in a Dependable SoS. CoRR abs/1404.6772 (2014)
2. AMADEOS Consortium, D2.3 – AMADEOS conceptual model – Revised (2016)
3. US Government Accountability Office: “GPS Disruptions: Efforts to Assess Risk to Critical Infrastructure and Coordinate Agency Actions Should be Enhanced”. Washington, GAO - 14-15 (2013)

4. GPS error caused '12 hours of problems' for companies, 4 February 2016. http://www.bbc.com/news/technology-35491962
5. Shepard, D.P., Humphreys, T.E., Fansler, A.A.: Evaluation of the vulnerability of phasor measurement units to GPS spoofing attacks. Int. J. Crit. Infrastruct. Prot. **5**(3), 146–153 (2012)
6. Cyber-physical Systems-of-Systems: the AMADEOS approach and Main Advances, Webinar for the INCOSE WG on Systems-of-Systems (2015)
7. 1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (2008)
8. Bondavalli, A., Ceccarelli, A., Falai, L.: Assuring resilient time synchronization. In: Proceedings of the 2008 Symposium on Reliable Distributed Systems, October 06–08, 2008, SRDS, pp. 3–12. IEEE Computer Society, Washington, DC
9. IEEE Standard for a Precision Clock Synchronization Protocol for Network Measurement and Control Systems, IEEE Std 1588-2008 (IEEE Std 1588-2002), p. cl-269, 24 July 2008
10. Verissimo, P., Rodriguez, L.: Distributed Systems for System Architects. Kluwer Academic Publisher, Dordrecht (2001)
11. Bondavalli, A., Brancati, F., Ceccarelli, A.: Safe estimation of time uncertainty of local clocks. In: Proceedings of International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication, ISPCS 2009, pp. 47–52
12. Bondavalli, A., Brancati, F., Ceccarelli, A., Falai, L., Vadursi, M.: Resilient estimation of synchronisation uncertainty through software clocks. Int. J. Crit. Comput. Based Syst. **4**, 301–322 (2013). doi:10.1504/IJCCBS.2013.059038. ISSN: 1757-8779
13. Bondavalli, A., Brancati, F., Ceccarelli, A., Vadursi, M.: Experimental validation of a synchronization uncertainty-aware software clock. In: The 29th IEEE Symposium on Reliable Distributed Systems, Delhi, India, October 31 – November 3, 2010, pp. 245–254 (2010). doi:10.1109/SRDS.2010.35. ISBN: 978-0-7695-4250-8
14. Barnes, J.A., et al.: Characterization of frequency stablity. IEEE Trans. Instrum. Meas. **IM-20**, 105–120 (1970)
15. AMADEOS project, deliverable D4.4 "Internal Delivery- Design of Resilient Master Clock" (2016)
16. BeagleBoard.org project Debian. http://beagleboard.org/project/debian
17. Mills, D.: Internet time synchronization: the network time protocol. IEEE Trans. Commun. **39**(10), 1482–1493 (1991)
18. Beekhuizen, J., Kromhout, H., Huss, A., Vermeulen, R.: Performance of GPS devices for environmental exposure assessment. J. Exposure Sci. Environ. Epidemiol. **23**(5), 498–505 (2013). ISSN 1559-0631
19. Kaplan, E.D., Hegarty, C.J. (eds.): Understanding GPS: Principles and Applications, 2nd edn. Artech House, Boston (2006)
20. The Royal Academy of Engineering, Global Navigation Space Systems: reliance and vulnerabilities, March 2011
21. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978)
22. Verissimo, P.: On the role of time in distributed systems. In: FTDCS (1997)
23. Verissimo, P., Rodrigues, L.: Distributed Systems for System Architects, vol. 1. Springer Science & Business Media, New York (2012)
24. Ceccarelli, A., et al.: Introducing meta-requirements for describing system of systems. In: 2015 IEEE 16th International Symposium on High Assurance Systems Engineering. IEEE (2015)

25. Kopetz, H.: Real-time Systems-Design Principles for Distributed Embedded Applications. Springer, New York (2011)
26. Levine, J., Mills, D.: Using the network time protocol (ntp) to transmit international atomic time (tai). In: 2nd Annual Precise Time and Time Interval Systems and Applications Meeting, 28–30, Reston, VA, USA, pp. 431–440 (2000)