On the Feasibility of Time Estimation under Isolation Conditions in Wireless Sensor Networks

Daniela Tulone
Department of Computer Science
University of Pisa, Italy

Abstract. We study the problem of providing a sensor with an accurate estimate of the time, from a *novel perspective* which is complementary to the well-studied clock synchronization problem. More precisely, we analyze the case in which a sensor node is temporarily unable to run a clock synchronization protocol due to failures or intermittent connectivity, or is willing to skip one or more clock adjustments to save energy, but still requires an accurate estimate of the *reference time*.

We propose and analyze two simple and efficient clock reading methods, one deterministic and the other probabilistic, which are designed to work in synergy with a clock synchronization protocol. Our deterministic method achieves a better time accuracy by exploiting information regarding the sign of the deviation of the hardware clock from the reference time. This algorithm leads to noticeable energy savings since it can be applied to reduce the frequency of the periodic clock adjustments by a factor of 2, while maintaining the same error bound. Moreover, our method is of theoretical interest since it shows how a stronger but realistic clock model leads to a refinement of the *optimality bound* for the maximum deviation of a clock that is periodically synchronized. We also propose two simple versions of this algorithm: a method that guarantees the monotonicity of the clock values, and a generalization that improves the accuracy in case of clock stability.

Our probabilistic method is based on *time series forecasting*, and provides a probabilistically accurate estimate of the reference time with a constant error bound. It is more flexible than our previous methods since it does not depend on the frequency at which clock synchronization occurs, and can be dynamically tuned according to the application requirements and resource availability. All these methods have broad applicability for their generality. In sensor networks they can be applied to improve the clock accuracy of a sensor node in conditions of network isolation, or to reduce the frequency of the clock adjustments, thus saving energy and increasing the system lifetime.

Keywords: Clock synchronization, clock drift, sensor networks, energy conservation, time series models, resource efficiency.

1. Introduction

As in any distributed computer system, the computation of an accurate estimate of the *reference time* is an important issue in wireless sensor networks (WSN). Time synchronization is critical in many sensornet tasks such as object tracking, surveillance, duplicate detection, power-saving duty cycling, or distributed beam-forming. It plays a crucial role

also in data integration and sensor reading fusion, which rely on data time-stamps. In fact, the lack of synchronization among sensor clocks can result in inaccurate time-stamping due to the different clock drift, and can lead to falsely reorder events or even reverse them, thus affecting data correctness. Time synchronization is relevant also for TDMA medium access scheduling for low-energy radio operation. In fact, since listening and transmitting are both energy-expensive operations in a low-power radio, a common technique is to turn the radio off, waking up only briefly to exchange short messages before going back to sleep [20]. Therefore, the lack of synchronization among sensor clocks can result in message lost, and consequently energy waste. Clearly, there are many other settings in which time synchronization plays a crucial role, since sensor applications are spawning in different domains (i.e. environmental, military, scientific, health, civilian [19]).

All these applications explain the growing attention for the clock synchronization problem in WSN, and the large volume of work appeared in the last few years [1–18]. In fact, as pointed out by Elson et al.[2], WSN show some unique characteristics that preclude the application of well-known clock synchronization protocols designed for wired networks, such as NTP [34]. A time synchronization service in WSN has to meet challenges that are substantially different from those present in infrastructure-based networks and that are related to the sensor's limited hardware and bandwidth. For instance, *energy conservation* is a critical issue for WSN due to the limited battery source of the sensor nodes. Other major issues in the design of clock synchronization protocols in WSN are *adaptability* to changes in the network topology and dynamic reconfigurability, and *scalability* since WSN can consist of few thousands of sensors (e.g., in environmental monitoring). In addition, due to battery depletion or destruction of the sensors, WSN show a *higher failure probability* than in wired networks, and changes in the environment can dramatically affect radio propagation causing frequent network topology changes and network partitions. Moreover, at high densities WSN become much more likely to suffer communication failures due to contention for their shared communication medium. All these elements rule out the appropriateness of existing clock synchronization protocols designed for wired networks to WSN, and show the need for *energy-efficient*, *adaptable* and *robust* time services. These requirements have been object of intense study in the last few years.

1.1. Motivations

In the last few years several clock synchronization protocols for WSN have been proposed [1–17] based on different approaches, such as the

Reference Broadcast Synchronization (RBS) proposed by Elson et al. [1] which exploits the physical radio broadcast, or hierarchical approaches [6, 5, 9], or interval-based [12, 13], or probabilistic approaches [4]. However, despite their diversity they all share a common viewpoint: each sensor derives a notion of time (global or relative) through messages exchanged with its neighbors, periodically or on demand.

Clearly, adjusting sensor clocks is energy consuming since it involves the transmission of several messages across the network. In WSN transmitting messages and listening are highly-expensive operations, and consume much more power than processing data. In [20] Pottie and Kaiser make concrete the effect of this constraint with an example: they compared the energy required to transmit 1 bit over 100 meters, to the energy used by a general purpose processor with 100 MIPS/W to execute 3 million instructions. Since the synchronization of each sensor clock is guaranteed by exchanging messages with its neighbors, the consumption of energy involved in synchronizing clocks is noticeable and can affect the system lifetime especially in case of long-term systems or high clock accuracy requirements. As a result, reducing the frequency of periodic clock synchronization results in noticeable energy savings and in a natural extension of the network lifetime. The works of Elson et al. [1] and Römer [3], develop this idea by avoiding clock synchronization, thus leaving sensor clocks run undisciplined. In RBS [1] each sensor builds a table consisting of relative clock offsets and drifts that relates the local clock with the other nodes in the network. This table is updated only when needed through expensive message exchanges among each neighbor. Therefore, energy is saved via a *post-facto synchronization* which exploits the fact that sensor clocks might not need to be synchronized all the time. However, this approach is not suitable for a number of WSN applications such as tracking, or surveillance, because of its latency in the convergence time. Recently, PalChaudhuri et al. [4] have proposed an adaptive clock synchronization protocol, based on the RBS, that provides a probabilistic bound on the clock accuracy, thus allowing for a trade-off between accuracy and resource requirements.

Energy-efficiency, mobility, and self-configuration requirements, as well as the high jitter in multi-hop transmissions, show the need for protocols which are *more local* in nature. The theoretical result of Fan and Lynch [18] emphasizes that. They introduced a property for clock synchronization, called gradient property, which requires that the skew (deviation) between two clocks forms a gradient with respect to the distance between nodes. Their worst case clock skew between two nodes shows that clock synchronization is not a local property, in the sense that the clock skew between two nodes depends not only on the distance

between nodes, but also on the size of the network. As a consequence, the TDMA will fail as the network grows.

There are other elements specific to WSN, that support the need for more local protocols. For instance, there are cases in which a sensor node is temporarily *unable* to synchronize its clock to its neighbors's due to process failures, or intermittent connectivity, or topological changes in the network, or temporary obstructions (e.g., in environmental monitoring [19]). Note that these situations are related to the intrinsic nature of the WSN, and therefore should be taken into consideration as well. All these considerations (energy requirements, failures, mobility, high jitter) motivated us to study the accuracy of a clock in case a node is unable to communicate with other nodes.

1.2. Internal vs. external clock synchronization

Several clock synchronization protocols in WSN [3, 16, 14, 13, 1, 4] focus on internal clock synchronization, which provides a constant bound on the deviation between any two clocks. This choice is motivated by energy reasons. In fact, external clock synchronization assumes the existence of some *time source* which provides an accurate estimate of the *reference time* (i.e., a node equipped with a GPS receiver [35]). Since this equipment is energy-hungry, it can induce extra energy requirements. In addition, clocks that are internally synchronized with maximum error λ are externally synchronized with error bound 2λ, however the error can grow unbounded in case of internal synchronization and network partitions. Note that external clock synchronization is more suitable for applications where sensors are tightly coupled to the physical world, and in some case it becomes necessary. For instance, in vehicle tracking applications the system can predict the vehicle moving direction and speed by matching the sensor location and sensing time at which it approached the sensor. Without an agreement on the global time, the data from different sensors cannot be matched up. Moreover, applications involving data coordination, or mobility, or detection of anomalies (e.g., gas leaks, seismic detection) require the notion of global time. In addition, we show here that clock information can be used to estimate the reference time in condition of node isolation or network partitions. All these reasons motivated us to analyze the external clock synchronization, that is the problem of estimating the reference time (or global time). This choice is also supported by recent work [15, 9, 11, 6, 17, 10].

1.3. Our perspective

The general goal of this paper is to try to maintain sensor clocks synchronized to the *reference time* within a constant error bound under all circumstances. We bring the idea of designing a *more local* time synchronization service discussed in Section 1.1 to the extreme consequences by studying the behavior of a single clock *without interactions* with other nodes, which is a novel perspective of studying the time estimation problem. In the literature the time accuracy has been always guaranteed through periodic clock adjustments to a more accurate time source. In fact, the lower bound for the maximum clock deviation [23, 24] takes into account a maximum *error growth* $\rho\Delta t$, with ρ maximum drift rate of the hardware clock and Δt time elapsed since the last synchronization. We investigate here ways to derive additional clock information that allow us to reduce the maximum error growth and improve the clock accuracy.

The problem of improving the clock accuracy *between effective synchronization* is of both theoretical and practical interest, as we show in Sections 5.4–5.5. It can be applied to any type of networks, but has a remarkable impact on WSN since it addresses the cases in which a sensor node is unable to communicate with other nodes due to failures or topology changes or intermittent connectivity, or it turns off its radio to conserve energy, thus skipping one or more clock adjustments. Therefore, this paper provides a first answer to the question: *"what time accuracy can be provided in case of node isolation or low-energy?"*

1.4. Contributions

The contributions of this work consist of our novel viewpoint mentioned in Section 1.3, a class of deterministic clock reading methods, and a probabilistic method.

Novel perspective. Our first contribution consists in studying the time synchronization problem in WSN from a different viewpoint, which is complementary to all previous work focusing on periodic clock adjustments. We tackle the problem of improving the accuracy of a clock between adjustments using information regarding the deviation of the hardware clock from the reference time. For this purpose we introduce a metric, called *cumulative hardware drift rate*, that provides information on the behavior of the hardware clock since its initialization, in contrast with the drift rate that provides only local information.

The deterministic methods. We propose a deterministic clock reading method, called DCR, that reduces the maximum error growth between clock adjustments by a factor of 2 using the *sign* of the clock deviation. It can be applied to any type of networks to improve the

clock accuracy. However, it is particularly suitable for WSN since it can be applied to (1) save energy and improve network bandwidth by reducing the frequency of the clock synchronization by a factor of 2 while maintaining the same time uncertainty, or (2) improve the clock accuracy in case a node is temporarily isolated. Note that the DCR method is of theoretical interest since it leads to a refinement of the *optimality bound* for the maximum clock deviation.

In addition, we show a simple version of the DCR method, which guarantees the monotonicity of the clock values, property that is crucial in several sensor applications (e.g., object tracking). We also propose a generalization of the DCR method, called GDCR, that does not depend on the sign of the *global deviation* (the deviation of the hardware clock from the reference time), and that leads to an improvement over the DCR method in case of clock stability.

The probabilistic method. We propose a generic probabilistic method, called PCR, that is based on an adaptation of *time series forecasting* for WSN, and provides a probabilistically accurate estimate of the reference time with a constant error bound. It computes a prediction of the clock deviation based on a narrow window of past events. The PCR method is more general than our deterministic methods since it does not assume a specific frequency at which the clock is synchronized (i.e., it provides a probabilistic time estimate with a constant error also in the case the node has "skipped" more than one clock adjustment). It is important to note that all our clock reading methods can be adapted to work in synergy with clock synchronization protocols (such as [12] that works for sporadic clock synchronization) to improve the clock accuracy and save energy.

Organization of the paper. The rest of the paper is organized as follows: Section 2 compares related work, and Section 3 presents our clock model and the techniques used. We define in Section 4 the cumulative drift rate used by the DCR and GDCR methods, and illustrate its differences with the hardware drift rate. In Section 5 we illustrate and analyze our DCR method and its revised versions, and discuss the refinement of the optimality bound for external clock synchronization. In Section 6 we present and analyze our PCR method based on time series forecasting, and then conclude the paper.

2. Related work

Clock synchronization protocols in WSN. In the last few years a very large volume of clock synchronization protocols have been proposed [1–17] which differ among them in their approach: some of them

focus mainly on time accuracy, others on energy conservation, or scalability, or mobility. For instance, proposals such as RBS [1] and [4, 3, 9] based on a *receiver-to-receiver* synchronization scheme, provide a very accurate estimate of the relative clock offset and skew since they reduce the time-critical path which contributes largely to non-deterministic latency. Other solutions such as [6, 11, 16, 14, 15] are based on the *sender-to-receiver* synchronization. To save energy, some protocols [1, 3, 4] do not synchronize local clocks but leave them run undisciplined; [1] adopts a *post-facto synchronization* to avoid unnecessary transmissions. This exploits the fact that sensor clocks do not need to be synchronized all the time. However, as mentioned in Section 1.1, because of its large convergence time this guarantee is too weak for applications such as tracking or TDMA. In addition, the number of messages exchanged in [1] is quite high and does not scale well as it is $O(m \cdot n^2)$ where n is the number of nodes and m the number of broadcasts needed to have an accurate time estimate. Our methods show some similarity with [1, 3] since they do not synchronize the local clock to avoid additional complexity, but provide a time estimation. To avoid unnecessary clock synchronization, some protocols such as TSync [5] adopt a *on-demand strategy* combining a push and pull mechanism. Some protocols [5, 6, 11, 17, 8, 14, 7] focus particularly on scalability and efficiency in a large-scale WSN and employ a hierarchical structure, others [6, 5, 11, 8, 3, 9] are particularly geared to mobility and focus on self-configurability, others such as [12, 13] employ, similarly to our work, an interval-based paradigm which seems more suitable to many WSN applications. However, despite their diversity all of these proposals provide a common notion of time (global or relative) via message transmissions. As mentioned in the Introduction, our view is very different since we focus on the clock behavior of a specific sensor node, and our algorithm is designed to work *between* clock synchronization. As a result, our algorithms are local and do not involve message transmissions. For this reason it is not possible to compare them to previous clock synchronization protocols, though our methods must work in synergy with a clock synchronization protocol.

Drift-based clock synchronization. Improving the accuracy of a set of distributed clocks by exploiting information related to the clock behavior is not a novel idea: for instance, Schossmaier et al. [31, 30] and NTP [34] include information regarding the hardware drift rate in their clock adjustments. Fetzer and Cristian show in [26] how to build clocks with bounded drift rate from components *off-the-shelf*, and achieve high accuracy by means of *drift synchronized clocks*. Their work diverges from ours for their *strong* assumptions (i.e. requiring hardware support). Our work differs from all previous proposals mainly

for two reasons: 1) for reducing the maximum error growth $\rho\Delta t$ between synchronization, and 2) for exploiting the deviation accumulated by the hardware clock since its initialization, in contrast with the hardware drift rate that captures the *local speed* of the clock.

Probabilistic approaches. There are few protocols that are probabilistic in nature: for instance, the probabilistic remote clock reading protocol proposed by Cristian [25], or Cristian and Fetzer's probabilistic clock synchronization protocol [33]. Statistical techniques are also applied in NTP [34] to compute, during synchronization, an accurate approximation of the time based on the timestamps received from the neighbors. As discussed before, Palchaundhuri et al. [4] have proposed a probabilistic protocol for WSN based on the RBS, to save energy. However, these solutions diverge significantly from ours for requiring message transmissions.

Time series forecasting techniques are broadly used in finance, to forecast physical phenomena, or to improve quality of service. For instance, they have been applied by Wolski [42] to provide dynamic resource performance forecast, such as to predict the TCP/IP end-to-end throughput and latency. However, the application of time series models to WSN is novel. We are also not aware of any probabilistic study regarding the clock deviation between synchronization. Vernotte et al. [36] apply an autoregressive linear model to estimate the time uncertainty of the on-board oscillator for different type of noise. However, their view is very different from ours since their target is to determine how the maximum error is related to the noise level of the clock in order to classify clocks.

3. Preliminaries

3.1. Clock model

Each sensor node has access to a local hardware clock whose function is denoted by $H(t)$. A hardware clock typically consists of an oscillator and a counting register that is incremented by the ticks of the oscillator. A hardware clock has a given *granularity* g: a tick of the oscillator increments the value of the hardware clock by g time units. Because of the impressions of the oscillator, variations in the temperature, pressure, and aging, a hardware clock can drift apart from the real time.

The *hardware drift rate* function, denoted by $\hat{\rho}(t)$, is defined as the first derivative of $H(t)$ with respect to time minus 1, that is $\hat{\rho}(t) = H'(t) - 1$, and provides a measure for the *punctual variation* of the

clock speed. It is common practice to assume a maximum drift rate ρ for the hardware clock. Note that ρ is a constant specified by the manufacturer, and represents the maximum difference between the expected frequency of the oscillator and its actual frequency. For most quartz clocks available on personal computers ρ is of the order of $10^{-4} - 10^{-6}$, it is larger in cheap sensor oscillators.

If ρ is a valid upper bound of the drift rate of the hardware clock, then for any real time $t_1 < t_2$ the hardware clock function $H(t)$ measures the passage of time in $[t_1, t_2]$ with an error of at most $\rho(t_2 - t_1) + g$. That is,

$$(1-\rho)(t_2 - t_1) - g \ \leq H(t_2) - H(t_1) \leq \ (1+\rho)(t_2 - t_1) + g \qquad (1)$$

For $t_1 = 0$ and for a negligible initial error, it implies that

$$(1-\rho)\ t - g \ \leq \ H(t) \ \leq \ (1+\rho)\ t + g$$

In our discussion we ignore terms of order ρ^2 or smaller, and equate $(1 \pm \rho)^{-1}$ to $(1 \mp \rho)$. For simplicity of presentation we ignore also the granularity of the hardware clock. However, our discussion remains valid for non-negligible granularity.

Note that due to the very restricted hardware sensor clocks behave differently than computer clocks. For instance, as pointed out by [6], Berkleley Mote sensors [21] with Tiny OS [22] do not have fine granularity. The lack of fine granularity is motivated by the limited energy supply, since high frequency of clock ticks leads to higher power consumption. This shows even at the hardware level a trade-off between accuracy and energy consumption. Therefore, a reasonable frequency should be determined in a task-directed fashion. Our methods can be extended to take into account the variability of the clock granularity.

3.2. The interval-based paradigm

We adopt the interval-based paradigm, originally proposed by Marzullo and Owicki [28] and later refined by Shmidt and Shossmaier [32], for its advantages in WSN over time estimates, as pointed out by [12, 13]. For instance, it allows to obtain guaranteed bounds from sensor data fusion, and to enter in a fail-safe state when the time uncertainty grows excessively large.

We assume that each sensor maintains a *virtual clock function* $C(t)$ that is synchronized periodically [1], namely every Γ time units, to a more accurate estimate of the current time. We assume for simplicity that the clock synchronization occurs at real time $t_1, t_2, \ldots, t_i, \ldots$ with

[1] Sections 5 and Section 6 shows how to relax this assumption.

$t_i \approx i\ \Gamma$, and that clock $C(t)$ is synchronized at the i^{th} synchronization to T_i (value derived by running a clock synchronization protocol or from a more accurate source), with maximum inherited error ε.

The time interval of $C(t)$, denoted by $I_C(t)$, is a function mapping real time values to clock values and such that $I_C(t) = [C(t) - E(t),\ C(t) + E(t)]$ for each $t \in [t_i, t_{i+1})$ and $i > 0$, with $C(t) = H(t) + (T_i - H(t_i))$ and error bound $E(t) = \varepsilon + \rho(H(t) - H(t_i))$. A time interval $I_C(t)$ is *correct* at time t if $t \in I_C(t)$.

Notations: from now on we denote for simplicity $H(t) - H(t_i)$ by ΔH, the time at which the last synchronization occurred by t_i, the time elapsed since the previous synchronization $t - t_i$ by Δt, the error growth since its last synchronization by $\rho\Delta H$, the maximum error in $[t_i,\ t]$, by $\eta = \varepsilon + \rho\Gamma$. Since the time Γ elapsed between two consecutive clock adjustments is not large, we can neglect terms of order $\rho^2\Gamma$ or smaller. This implies that $\forall t_1, t_2$ such that $t_2 - t_1 \leq \Gamma$, $\rho\Delta t \approx \rho\Delta H$.

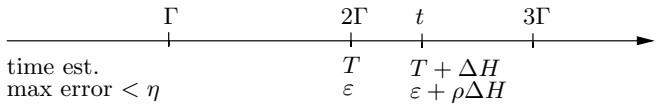

Figure 1. Periodic clock synchronization

Figure 1 illustrates the periodic clock adjustments performed at time Γ, 2Γ and 3Γ. Clearly, since $E(t)$ grows with the time elapsed since its last synchronization, if the time elapsed between two effective synchronization increases, then the maximum error bound $E(t)$ exceeds $\eta = \varepsilon + \rho\Gamma$.

3.3. Problem statement

Our goal is to try to provide a time estimate with the same error bound η in case the sensor node is unable or unwilling to synchronize its local clock. Note that $t_1, \ldots, t_i, \ldots$ represent the real time at which

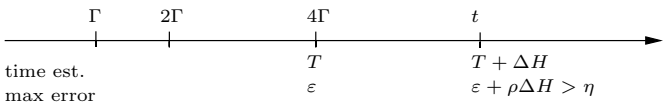

Figure 2. Irregular clock synchronization

the sensor clock is actually synchronized by exchanging messages with other nodes. For simplicity we assume that each node in the network tries to synchronize its clock at a regular basis, that is, $(t_i - t_{i-1})$ is a multiple of Γ. This assumption is reasonable in WSN since sensors sleep most of the time to save energy battery. Figure 2 shows a scenario in which clock synchronization occurs at time Γ, 2Γ, and 4Γ and the error associated with the evaluation of t is larger than η since $\Delta H > \Gamma$.

3.4. Time series forecasting

In this section we briefly review some basic concepts from time series forecasting that will be applied to the PCR method in Section 6. We refer the reader to [37] for further discussion on time series forecasting. A *time series* is a set of observations x_t, each one being recorded at a specific time t. An important part of the analysis of time series is the selection of a suitable probability model for the data. To allow for the possibly unpredictable nature of future observations, it is natural to suppose that each observation x_t is a realized value of a certain random variable X_t (often denoted as $X(t)$).

DEFINITION 1. *A time series model for the observed data* $\{x_t\}$ *is a specification of the joint distributions (or possibly only the means and covariances) of the sequence of random variables* $\{X_t\}$ *of which* $\{x_t\}$ *is postulated to be a realization.*

Clearly, if we wish to make predictions, then we must assume that something does not vary over time. Therefore, an important step to time series modeling is to remove *trend* and *seasonal* components to get a *stationary* time series (or *weakly stationary*). Loosely speaking, a time series $\{X_t\}$ is stationary if it has statistical properties similar to those of the time-shifted series $\{X_{t+h}\}$ for each integer h. More precisely, $\{X_t\}$ is *weakly stationary* if its mean function $\mu_X(t)$ and its covariance function $\gamma_X(t+h,t)$ are independent of t for each h.

The class of linear time series model, which includes the class of *autoregressive moving average* (ARMA) models, provides a general framework for studying stationary processes. The ARMA processes are defined by linear difference equations with constant coefficients. One of the key properties is the existence and uniqueness of stationary solutions of the defining equations.

DEFINITION 2. $\{X_t\}$ *is an ARMA(p, q) process if* $\{X_t\}$ *is stationary and for every t,*

$$X_t - \phi_1 X_{t-1} - \ldots - \phi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q}$$

where $\{Z_t\} \sim WN(0, \sigma^2)$ *and the polynomials* $(1 - \phi_1 z - \ldots - \phi_p z^p)$ *and* $(1 - \theta_1 z - \ldots - \theta_q z^q)$ *have no common factors.*

Notice that $\{Z_t\}$ is a series of *uncorrelated* random variables, each with zero mean and σ^2 variance. Such a sequence is referred as *white noise* and denoted by $WN(0, \sigma^2)$. An *autoregressive* model of degree p, denoted by $AR(p)$, is a particular type of ARMA model with $q = 0$. We use autoregressive models to predict the current deviation of the clock because of their simplicity and efficiency, which make them more suitable than general ARMA models in low-cost sensors.

4. Exploiting the global clock deviation

We study the deviation of the hardware clock from the reference time, accumulated since its initialization. This will allow us to derive information useful to maintain the error bound within a constant η. In fact, the global deviation of the hardware clock provides not only information regarding the behavior of the clock at that point in time, but also a sort of fingerprint of the *clock history* since its initialization. We exploit this fact by introducing a metric called *cumulative drift rate*.

4.1. The cumulative drift rate

We denote the clock deviation function by D. It is a function mapping real time values into reals and such that $D(t) = t - H(t)$ for any real time $t > 0$. Clearly, since the hardware clock runs undisciplined, $D(t)$ can grow unbounded. We write $D(t)$ as $D(t) = \delta(t) \cdot t$, where $\delta(t)$ is a function similar to the hardware drift rate function $\hat{\rho}(t)$, and it is called *cumulative hardware drift rate* function.

DEFINITION 3. *The cumulative hardware drift rate* δ *is a function mapping real time values into reals and such that* $\delta(t) = \frac{H(t)-t}{t}$ *for any real time value* $t > 0$.

To simplify the notation, we denote sometimes $\delta(t)$ with δ_t. Clearly, the hardware clock function can be written as $H(t) = (1+\delta_t)\, t$ for any real time $t > 0$. Similarly to the hardware drift rate, the cumulative hardware drift $\delta(t)$ is contained in $[-\rho,\ \rho]$ for any $t > 0$. This is a simple derivation from the stronger assumption (1) in Section 3.1.

4.2. Hardware drift rate vs. cumulative drift

As mentioned before, the cumulative drift provides information about the clock deviation *accumulated* since its initial time, while the hardware drift rate provides *local* information since it represents the speed of the clock at that time. Figure 3 illustrates graphically the differences

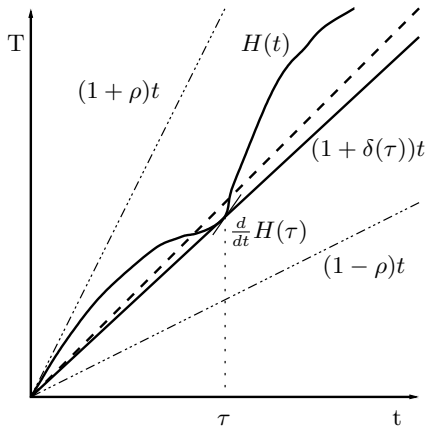

Figure 3. The hardware clock function $H(t)$, the cumulative drift $\delta(\tau)$, and the hardware drift rate $\rho(\tau)$ at time τ.

between the two metrics: the hardware drift rate $\hat{\rho}(t)$, and the cumulative drift rate $\delta(t)$. The graph shows the hardware clock function $H(t)$ limited by the lines $(1 \pm \rho)\ t$, the cumulative drift $\delta(\tau)$ at real time τ, and the hardware drift rate $\hat{\rho}(\tau)$. The dashed line bisector represents the reference time. The hardware drift $\hat{\rho}(\tau)$ represents the slope of the tangent of the clock function minus 1, while the cumulative drift $\delta(\tau)$ is the slope of the line passing by the origin O and $H(\tau)$, minus 1. The relation between $\hat{\rho}(t)$ and $\delta(t)$ is provided by the equation $\hat{\rho}(t) = \frac{d\delta(t)}{dt}\ t + \delta(t)$.

4.3. Cumulative drift variations

The most significant difference between the hardware drift rate and the cumulative hardware drift is represented by their variation with respect to the real time. As mentioned in Section 3.1, the variation of the hardware drift rate is related mainly to external factors, and therefore can occur at any time during the system lifetime. In contrast, the maximum variation of the cumulative drift during a fixed time

interval decreases over time. In fact, for any time t_2, t_1 with $0 < t_1 < t_2$, we can derive the following inequality based on inequality (1) in Section 3.1

$$-\rho(t_2 - t_1) - g \ \leq \ \delta_{t_2} t_2 - \delta_{t_1} t_1 \ \leq \ \rho(t_2 - t_1) + g$$

Let us write δ_{t_2} as $\delta_{t_2} = \delta_{t_1} + x$, where x is the variation of the cumulative drift in $[t_1,\ t_2]$, and replace it in the previous relation,

$$-(\rho + \delta_{t_1})(t_2 - t_1) - g \ \leq \ x\ t_2 \ \leq \ (\rho - \delta_{t_1})(t_2 - t_1) + g$$

Since $|\delta_{t_1}| \leq \rho$, then

$$|x| \leq \frac{2\rho(t_2 - t_1) + g}{t_2} \tag{2}$$

If $t_2 \geq \beta(t_2 - t_1)$ with $\beta > 1$, then $|x| \leq 2\rho\beta^{-1}$. This means that the variation of the cumulative drift in $[t_1,\ t_2]$ can be considered irrelevant after a sufficient large time t_2, intuitively after accumulating enough clock information. For instance, if $t_2 \geq \rho^{-1}(t_2 - t_1)$ we can neglect x in accordance with our assumption in Section 3.1, since it has order ρ^2. This is likely to occur faster in case of an inaccurate oscillator, such as a sensor oscillator which has a higher maximum drift rate.

Property (2) is useful for our DCR method to support the stability of the sign of the cumulative drift between synchronization. It will be discussed in Section 5.2. Note that this property cannot erroneously lead us to think that after a certain time it is possible to approximate the cumulative drift at time t_2 with the same precision relative to the previous synchronization, since approximating x to zero does not mean that $x \cdot t_2$ is negligible!

5. A deterministic clock reading method

We present in this section our deterministic clock reading methods, which are designed to work between synchronization and provide a more accurate estimate of the reference time, without adjusting the local clock. More precisely, the DCR method reduces by half the error growth since its last synchronization by exploiting the sign of the cumulative drift. The GDCR method additionally reduces the error growth in case of clock stability. They can be applied to any type of networks, but as discussed previously, they have a bigger impact in WSN due to its limited resources and characteristics.

5.1. The DCR method

Our DCR method is designed to reduce the maximum error growth $\rho\Delta H$ between adjustments without the aid of external data. It is based on the observation that increasing the inherited error ε by a quantity $\rho\Delta H$ proportional to its maximum drift rate, leads in most cases to a conservative approach, since it does not take into account any information regarding the clock deviation or the drift rate. We show here that such information can be used to improve the clock accuracy.

In the interval-based paradigm, improving the time accuracy is equivalent to reducing the size of the time interval $I(t)$, while maintaining its correctness. Our method is based on the intuition that the composition of two clocks, one proceeding faster than the reference time and the other slower, results in a more precise clock. Figure 4 illustrates that: it shows the time interval of clock $C(t)$ proceeding faster than the real time, and the time interval of clock $F(t)$ going slower.

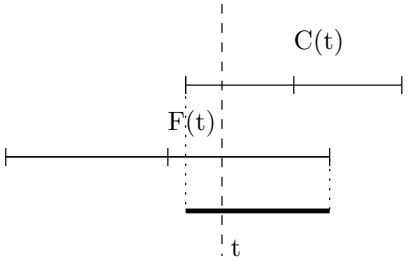

Figure 4. Intersection of time intervals $I_C(t)$ and $I_F(t)$

Clearly, the intersection of two correct time intervals, as the one shown in Figure 4, is correct and it is smaller in size than the previous intervals, and therefore more accurate. These are some of the intuitions that have driven us to study the sign of the cumulative drift and its variation between two consecutive clock adjustments.

The idea underlying our algorithm is very simple: if the node detects no variation in the sign of the cumulative drift since its last clock adjustment, it computes an *associate clock function* F proceeding in the opposite direction with respect to $C(t)$, and returns the time interval $I_C(t) \cap I_F(t)$. This simple idea brings two issues:

1. Detect the sign of the cumulative drift at time t_i and t;

2. Compute an associate clock function for $C(t)$.

The first issue can be solved by simply comparing the value of the hardware clock with the left and right side of the time interval of clock $C(t)$. Let us suppose that the last synchronization occurred at time t_i, and that T_i is an accurate approximation of real time t_i with maximum error ε. Because of the correctness of $I_C(t)$ at time t_i and t, if $H(t_i) > T_i + \varepsilon + \rho\Delta H$ (or $H(t_i) < T_i - \varepsilon - \rho\Delta H$) then the hardware clock is clearly faster (or slower) than the real time at time t_i, and it will be faster (or slower) at time $t > t_i$. Note that this is a feature of the DCR method and it can be applied to save energy, as shown in Section 5.2. In addition, the accuracy of this test improves over time, and since it exploits the clock deviation, it works better with cheap oscillators such as sensor oscillators. For instance, drifts smaller than $\frac{\rho}{2}$ can be detected after $2\varepsilon\rho^{-1}$ time units, that is less than $100s$ for $\rho = 10^{-4}$ and $\varepsilon = 5\mu s$.

The second issue is the most crucial for the DCR method. Before computing our associate clock function we define the clock properties that can lead to an improvement of the error growth *at least* by a factor of 2. We introduce this constraint because we are interested in the case in which a sensor has to skip at least one clock synchronization due to network isolation or low-energy. Therefore, we define a class of clock functions $\mathcal{A}_{C,t}$ called *associate clock functions* of $C(t)$ at time t.

DEFINITION 4. *$F \in \mathcal{A}_{C,t}$ at time $t \in [t_i, t_{i+1})$ if it satisfies the following properties:*

1. *F is a clock function defined in $[t_i, t_{i+1})$ with ρ valid drift bound at time t_i and t;*
2. *$F(t_i) = C(t_i)$;*
3. *$|F(t) - C(t)| \geq \rho\Delta H$.*

Note that condition 3 provides an upper bound $2\varepsilon + \rho\Delta H$ on the size of the time interval $I_C(t) \cap I_F(t)$, and condition 2 implies that clocks $C(t)$ and $F(t)$ have the same maximum error $E(t) = \varepsilon + \rho\Delta H$ during $[t_i, t_{i+1})$.

In this section we consider hardware clocks with a more restricted clock variation than (1). In Section 4.3, we have shown that the variation of the cumulative drift $x = \delta_t - \delta_{t_i}$ in $[t_i, t]$ is bounded as follows

$$-\frac{(\rho + \delta_{t_i})\Delta t}{t} \;\leq\; x \;\leq\; \frac{(\rho - \delta_{t_i})\Delta t}{t} \tag{3}$$

The DCR method relies on the following refinement of the previous bound. Its applicability is discussed in Section 5.2.

> Assumption $A1$: The variation of the cumulative drift in $[t_i,\ t]$ for any real time $t_i < t < t_{i+1}$, is bounded by condition (3) and by these additional constraints:

- If δ_t and $\delta_{t_i} \leq 0$ then $-\frac{(\rho+\delta_{t_i})\Delta t}{t} \leq x \leq -\frac{\delta_{t_i}\Delta t}{t}$
- If δ_t and $\delta_{t_i} > 0$ then $-\frac{\delta_{t_i}\Delta t}{t} \leq x \leq \frac{(\rho-\delta_{t_i})\Delta t}{t}$

The associate clock function of $C(t)$ in $[t_i,\ t_{i+1})$ computed by our DCR method is defined as followings:

$$A(t) := \begin{cases} C(t) + \rho\ \Delta H & \text{for } \delta_{t_i}, \delta_t \leq 0 \\ C(t) - \rho\ \Delta H & \text{for } \delta_{t_i}, \delta_t > 0 \end{cases}$$

Intuitively, if the hardware clock is going faster at time t_i and t, we compute a function which is slower with respect to the virtual clock $C(t)$ by a variable term $-\rho\Delta H$, proportional to the minimum drift rate and to the time elapsed since the last synchronization (note that $\rho\Delta t \approx \rho\Delta H$ because of our assumption in Section 3.2).

Figure 5 illustrates the DCR method. It checks if the sign of the cumulative drift of the hardware clock is unchanged in t_i and t, line (1). If this occurs, the method returns the midpoint of the time interval $I_C(t) \cap I_A(t)$ representing the time estimate, and its error bound, line (2). Otherwise it returns $C(t)$ with error bound $E(t)$, line (4). In Section 5.2 we discuss the meaning of condition at line (1), and show that experiments performed on sensors by Dan and Hai [5] confirm its verifiability.

DCR($t \in (t_i, t_{i+1})$)

1) if ($|H(t_i) - T_i| \ \geq\ \varepsilon + \rho\Delta H$)
2) return $\langle \frac{A(t)+C(t)}{2},\ \varepsilon + \frac{\rho\Delta H}{2} \rangle$
3) else
4) return $\langle C(t),\ \varepsilon + \rho\Delta H \rangle$

Figure 5. The DCR method.

The following lemmas prove the correctness of the DCR method: they show the correctness of the time interval $I_C(t) \cap I_A(t)$ when the sign of the cumulative drift is unchanged. The correctness of time interval $I_C(t)$ follows from the drift bound ρ, condition (1) in Section 3.1, as shown in [28].

LEMMA 1. *If the sign of the cumulative drift remains unchanged at time t_i and t, and A1 is verified, then $A(t) \in \mathcal{A}_{C,t}$.*

Proof Let us suppose $\delta_{t_i}, \delta_t > 0$. We show that ρ is the maximum drift rate of $A(t)$ in $[t_i, t_{i+1})$. The associate function $A(t)$ can be written in

$[t_i, t_{i+1})$ as following:

$$A(t) = (1-\rho)H(t) + T_i - (1-\rho)H(t_i)$$

Then, by writing $A(t)$ in terms of the cumulative drift

$$A(t) - A(t_i) = (1-\rho+\delta_t-\rho\delta_t)t - (1-\rho+\delta_{t_i}-\rho\delta_{t_i})t_i$$

Since $\delta_t, \delta_{t_i} > 0$ and $\delta_t t - \delta_{t_i} t_i \geq 0$ for assumption $A1$, then

$$A(t) - A(t_i) \geq (1-\rho)(t-t_1)$$

Clearly, since the associate function is slower than $C(t)$ by $\rho\Delta H$, then $A(t) - A(t_i) \leq t - t_1$, which is stronger than condition 1. Since $C(t_i) = A(t_i)$ and $C(t) - A(t) = \rho\Delta H$, then $A(t) \in \mathcal{A}_{C,t}$. Similarly, if $\delta_{t_i}, \delta_t \leq 0$. □

LEMMA 2. *For any real time $t \geq t_i$ if $|H(t_i) - T_i| \geq \varepsilon + \rho\Delta H$, and $A1$ is verified, then the DCR method returns an estimate of time t with maximum error $\varepsilon + \frac{\rho}{2}\Delta H$.*

Proof Let us suppose that $H(t_i) \geq T_i + E(t)$ for $t \geq t_i$. In this case the hardware clock is clearly proceeding faster at time t_i since $\delta_{t_i} t_i \geq T_i - t_i + \varepsilon + \rho\Delta t$ and $|T_i - t_i| \leq \varepsilon$. Therefore, $\delta_t t \geq 0$ since $\delta_{t_i} t_i \geq \rho\Delta t$, and because $-\rho\Delta t + \delta_{t_i} t_i \leq \delta_t t \leq \rho\Delta t + \delta_{t_i} t_i$ from inequality (1) in Section 3.1. Notice that $I_C(t)$ is a correct time interval because ρ is a valid drift bound of the hardware clock. Since the time interval $I_A(t)$ is correct because of Lemma 1, then $I_C(t) \cap I_A(t)$ is a correct time interval and its size is equal to $2\varepsilon + \rho\Delta H$. □

5.2. Considerations

Enhanced accuracy. A natural comment on the DCR method is that it works only if the sign of the cumulative drift is unchanged since the previous synchronization, and the clock drift does not vary too much. As shown in Section 4.3, since the hardware clock $H(t)$ runs undisciplined and the cumulative drift represents the global deviation of the clock, its maximum variation decreases with the time, in contrast with the drift rate that captures only the local behavior of the clock. That implies that after an initialization stage, if the global deviation of the clock is positive, it remains most likely positive in the following 2Γ time units, since in most cases the deviation accumulated by the hardware clock is larger than $2\rho\Gamma + \varepsilon$. In addition, assumption $A1$ refines the bound for the variation of the hardware drift, but this limitation is realistic in most

cases, as shown by the experiments performed by Dai and Han [5] on GPS-enabled sensors. They use the Mantis Nymph hardware platform integrated with a GPS chip with a time accuracy within $0.2\mu s$ to estimate the deviation of the actual duration between two consecutive pulses from the expected duration. A statistical analysis of their data revealed them a roughly normal distribution of the clock drift (local drift) with a mean value greater than the expected value, 1 sec, for some sensors (i.e. $1,000,009\ \mu s$) and smaller for others. This shows that on the average some sensor clocks drift faster than the reference time and others slower.

Efficiency and simplicity. The DCR method is extremely simple and efficient since it involves only numeric comparisons and the computation of $\frac{C(t)+A(t)}{2}$. As a result, it is particularly suitable to be implemented at low-cost sensors.

Monotonic clock function. It is known that instantaneous clock synchronization can cause time discontinuities if for example a virtual clock is going faster then the reference time and is adjusted to a value smaller than its current clock value. This can cause inconsistencies in time-stamping with serious consequences for the application. To avoid this problem some synchronization protocols such as [24], adjust the clock to the maximum clock value. However, this approach can increase the deviation of the virtual clock from the reference time. In [16] Mock et al. propose a *continuous clock* for wireless networks that avoids time discrepancies by gradually speeding up or slowing down the clock rate. However, this approach suffers from a run-time overhead since clocks need to be adjusted at every clock tick.

We adapt our DCR method to efficiently guarantee a weaker property, which ensures that all of the clock readings are monotonically increasing. We call this revised version IDCR, for *Increasing Deterministic Clock Reading Method*. The IDCR method builds on the DCR method and on the observation that a sensor does not require that its virtual clock function is monotonically increasing at *any time*, but that the clock readings are increasing and that their associate maximum error is bounded. In our case each clock reading is performed by invoking IDCR, and therefore value T_i received at the ith synchronization is not considered as a clock reading. This simple observation simplifies noticeably our problem since it is easier to guarantee that a clock reading is larger than its previous one (denoted as $lastValue$). Note that this simplification reduces the computational cost, thus conserving energy.

The values returned by the DCR method are clearly monotonically increasing between adjustments since both $C(t)$ and $A(t)$ are increasing between adjustments, and the auxiliary function $A(t)$ is computed during the entire interval (t_i, t_{i+1}), because of condition $|H(t_i) - T_i| \geq \varepsilon + \rho\Delta$ (see Figure 5:1). However, when considering time intervals larger than Γ, the DCR method can return values that are not monotonically increasing (e.g., if $C(t)$ is set back at some synchronization). This problem can be solved by slowing down the growth of the previous clock function, and more precisely by computing function $lastValue + \rho\Delta H$.

The IDCR method is illustrated in Figure 6. It invokes the DCR method, and compares it with the clock value previously read, Figure 6:1–2. If the value returned by the DCR method is not larger than the previous reading, it returns the previous value increased by term $\rho(H(t) - h)$, where $H(t) - h$ represents the time elapsed since the previous clock reading measured by the hardware clock, Figure 6:6. The purpose of this minimal increment is to slow down the growth of the previous virtual clock in order to be able to apply the new one. Lemma 3 provides bounds on the maximum time interval necessary for the new virtual clock function to "catch up" with the previous one, and the maximum error associated to the value returned by IDCR.

```
IDCR(t)
1)  ⟨T, e⟩ ← DCR(t)
2)  if T > lastValue
3)      lastValue ← T
4)      err ← e
5)  else
6)      lastValue ← lastValue + ρ(H(t) − h)
7)      err ← err + (1 + ρ)(H(t) − h)
9)  h ← H(t)
10) return ⟨lastValue, err⟩
```

Figure 6. The IDCR method.

LEMMA 3. *Function IDCR is a monotonically increasing function. When slowing down the virtual clock $C(t)$, the error bound is increased by less than $2\varepsilon + \rho(2\varepsilon + \Gamma)$, if the clock is adjusted every Γ time units and $\Gamma > 2\varepsilon + 1$.*

Proof IDCR is monotonically increasing between adjustments because both $C(t)$ and $A(t)$ are increasing in that interval. If $C(t)$ is set back at time t_i and $DCR(t) < lastValue$ at time $t > t_i$, then $lastValue + \rho(H(t) - h)$ is computed, which is clearly larger than $lastValue$.

In order to compute the maximum error associate to IDCR(t), we consider the worst case in which the DRC method returns $\langle C(t), E(t)\rangle$. Since $I_C(t)$ is a correct time interval, then the virtual clock cannot be set back at time t_i more than $\varepsilon + \rho\Gamma$ plus the error ε associate to T_i. Let us suppose that $lastValue$ is the last reading performed at time t_i, then since $\rho\Delta t \approx \rho\Delta H$, we want to compute the maximum time Δt elapsed since t_i such that

$$lastValue + \rho\Delta t < T_i + (1 + \rho)\Delta t$$

Since $lastValue - T_i \leq 2\varepsilon + \rho\Gamma$, this condition is always satisfied for $\Delta t \geq 2\varepsilon + \rho\Gamma$. Term $\rho\Gamma$ is less than 1 for reasonable choices of Γ. Therefore, the maximum error performed when slowing down the previous clock function is increased by at most $(1+\rho)(2\varepsilon+\rho\Gamma)$. □

5.3. A generalization of the DCR method

In this section we sketch a generalization of the DCR method, called GDCR, which is simpler than the DCR method, and it *does not* rely on the sign of the cumulative drift rate. The GDCR method is based on the observation, supported by experiments performed on computer clocks, that in most cases the variation of the hardware drift is much smaller than 2ρ. As a result, we consider the following more general assumption derived by inequality (1) in Section 3.1:

Assumption $A2$: the variation of the cumulative drift in $[t_i,\ t]$ for $0 < t_i < t$, is bounded by the following expression:

$$l\ \Delta t\ \leq\ \delta_t t - \delta_{t_i} t_i\ \leq\ u\ \Delta t \tag{4}$$

where $l \geq -\rho$, and $u \leq \rho$.

In contrast with the DCR method, GDCR does not need to compute the sign of the hardware clock deviation or an auxiliary clock function $A(t)$. It is not based on the intersection of two time intervals, one going faster than the real time and the other going slower, but on assumption A2. It computes the following clock function:

$$F(t) = C(t) - l\Delta H$$

for $t \in (t_i,\ t_{i+1})$, and returns the midpoint of the time interval

$$\widehat{I}(t) \ = \ [F(t) - \varepsilon - d\Delta H,\ F(t) + \varepsilon]$$

with $d = u - l$, and size $2\varepsilon + d\Delta H$. We show in Lemma 4 that interval $\widehat{I}(t)$ is always correct provided assumption A2.

LEMMA 4. *If assumption A2 is verified, then the GDCR method returns an accurate estimate of the reference time with maximum error* $\varepsilon + \frac{d}{2}\Delta H$.

Proof We have to show that $t \in \widehat{I}(t)$ for any $t_i < t < t_{i+1}$. We show first that

$$F(t) - \varepsilon - d\Delta H \ \leq \ t$$

Since $C(t) = T_i + H(t) - H(t_i)$, $T_i - t_i \leq \varepsilon$, and $\rho\Delta H \approx \rho\Delta t$ because of our assumption in Section 3.2, then

$$F(t) - \varepsilon - d\Delta H \leq t + \delta_t t - \delta_{t_i} t_i - (l + d)\Delta H$$

The condition is verified since $\delta_t t - \delta_{t_i} t_i \leq u\Delta t$ because of inequality (4) and since $d = u - l$.
On the other hand $F(t) + \varepsilon \geq t$ since

$$F(t) + \varepsilon \geq t + \delta_t t - \delta_{t_i} t_i - l\Delta t$$

and $\delta_t t - \delta_{t_i} t_i \ \geq \ l\Delta t$. Therefore, $t \ \in \ \widehat{I}(t)$, and the GDCR method returns the midpoint of $\widehat{I}(t)$ with maximum error $\varepsilon + \frac{d}{2}\Delta H$. □

We briefly discuss here the pro and counts of using GDCR over DCR.

1. The GDCR method does not rely on the sign of the cumulative drift. As a result, it is more general and suitable also in case of very precise oscillators.

2. If $u - l < \rho$, then the GDCR method reduces the error bound provided by DCR using assumption $A2$. Since a node is able to estimate the drift of the hardware clock and the drift bound at clock adjustments, if it detects that $u - l < \rho$ it can apply the GDCR method to estimate the time between synchronization. However, note that an incorrect evaluation of u and l can compromise the correctness of the GDCR method (e.g., in case of a sudden variation in the clock stability). This is not an issue for the DCR method, which is always correct.

5.4. A refinement of the lower bound

The DCR method shows that a *stronger* but realistic clock model can lead to a refinement of the *lower bound* for the external deviation of a clock that is periodically synchronized. Shrikanth and Toueg [24] showed that the bound on the drift rate of a virtual clock from the reference time is *at least as large as* the bound on the drift rate of the physical clock. The lower bound computed later by Fetzer and Cristian in [23] using a model similar to ours, is in line with the previous bound [24]. Fetzer and Cristian showed that at any real time the best maximum external deviation achievable by a clock synchronized at least every Γ time units is equal to

$$\Delta + \Lambda + \rho\ \Gamma$$

where Λ is the *remote clock reading error*, Δ is the error performed by the time source in estimating the reference time, and $\rho\Gamma$ is the maximum error growth during Γ time units. The DCR method shows that in most cases, after an initialization time it is possible to reduce the error growth $\rho\Gamma$ by half, thus leading to the following expression

$$\Delta + \Lambda + \frac{\rho}{2}\ \Gamma$$

The GDCR method can refine the previous expression to $\Delta+\Lambda+\frac{u-l}{2}\ \Gamma$ in case $u - l < \rho$. This implies that under that assumption, which is realistic in most cases, there exists a virtual clock with drift rate bound smaller than $\frac{\rho}{2}$. In conclusion, our deterministic methods are of theoretical interest because they suggest that a *stronger but realistic* clock model leads to a refinement of the optimality bound for the maximum deviation of a clock that is periodically synchronized.

5.5. Applications

Our methods are also of practical interest, and can be applied to WSN as well as to wired networks. We sketch below some applications.

1. **Energy saving.** They can be used in WSN to save energy and reduce communication since a node can decide at time t_i if to skip or not the next synchronization round at time $t_i + \Gamma$. In fact, if the deviation of the undisciplined hardware clock at time t_i is greater in absolute value than $\varepsilon + 2\rho\Gamma$, the node can surely apply the DCR method during $[t_i,\ t_i+2\Gamma]$ and estimate the time with error smaller than $\eta = \varepsilon + \rho\Gamma$. If $u - l < \rho$, the node can skip more than one synchronization by applying GDCR.

2. **Network isolation.** They can be applied in WSN to maintain the same accuracy η in case the sensor node cannot synchronize its clock because it is unable to communicate with other nodes due to failures, or intermittent connectivity, or temporary physical obstruction, or node mobility.

3. **Improve the time accuracy.** Because of our different viewpoint, both methods can be combined with most existing clock synchronization protocols in both wired and wireless networks to improve the time accuracy and save energy.

4. **Calibrated clocks.** In wired networks these methods can also coexist with *calibrated clocks* [26] to improve the time accuracy by replacing the hardware clock function $H(t)$ with a calibrated one. This can be useful in real-time critical applications.

6. A probabilistic clock reading method

As discussed in the previous section, the DCR method allows a node to reduce the frequency of its clock adjustments by a factor of 2, while the GDCR method allows it to skip more clock adjustments in conditions of clock stability. However, both methods rely on a stronger assumption of the variation of the hardware drift rate. Therefore, it is natural to question what time accuracy can be ensured if the node remains isolated for a period longer than 2Γ time units, or the clock oscillator is unstable such that our previous assumption cannot hold. In these extreme situations where it is not possible to guarantee high accuracy, probabilistic guarantees provide a reasonable answer for the needs of most sensor applications. In this section we illustrate our probabilistic PCR method that relaxes our assumption on the drift variation, and uses statistical properties on the deviation of the hardware clock from the reference time. More precisely, it is based on autoregressive (AR) models, which are simple linear time series models. This choice is motivated by their simplicity and efficiency, in terms of computational cost and memory storage, that make this type of model suitable for low-cost sensors. Clearly, there are machine learning and soft computing approaches based on non-linear techniques [41] that are more versatile than time series models and tolerate better chaotic components (e.g., artificial neural networks, recurrent neural networks [39], SVMP [40], hidden Markov models [38]), however these techniques are costly and not suitable to be implemented at low-cost sensor nodes.

6.1. Method overview

We propose here a *general framework* that provides a probabilistic estimate of the reference time with uncertainty $\eta = \varepsilon + \rho\Gamma$, in case the sensor node "skip" a *few* clock adjustments. It is more general than the DCR method since it is independent of the occurrences at which the clock was synchronized. In addition, PCR is more flexible than DCR since it allows the node to dynamically adjust the clock accuracy according to the application requirements and energy budget, and compute the frequency of the clock adjustments necessary to guarantee a given accuracy.

The PCR method computes an estimate of the reference time by predicting the deviation of the hardware clock based on a short window of past events. Therefore, the computation of a stationary autoregressive model that is able to predict the clock deviation plays a crucial role in the PCR method. A straightforward solution is to consider a time series $\{X_{t_i}\}_{i\geq 0}$ where X_{t_i} is the deviation $D(t_i)$ of the hardware clock at synchronization time t_i. However, this approach presents two main problems:

1. The time series $\{X_{t_i}\}$ contains some trend, since the deviation $D(t)$ varies over time, and also a seasonal component (e.g., related to daily variations in the temperature).

2. The other problem is related with the intermittent clock synchronization of the sensor node that does not necessarily provide measures on the clock deviation at regular intervals. This is a problem because time series forecasting requires periodic observations.

The first problem is overcome by differentiating data and replacing the original time series $\{X_{t_i}\}$ with $i \geq 0$, with $\{Y_{t_j} = X_{t_j} - X_{t_{j-1}}\}$ with $j \geq 1$. It is reasonable to assume that the time series $\{Y_{t_i}\}$ is weakly stationary and with zero mean. The second problem is solved by normalizing the data series to variations occurred during Γ time units. More precisely, if $(t_i - t_{i-1}) = \alpha_i\Gamma$, then Y_{t_i} is a random variable representing the average variation of the clock deviation during Γ time units in $[t_{i-1}, t_i]$. Therefore, y_{t_i}, which is the observed value of Y_{t_i}, is equal to $\frac{D(t_i)-D(t_{i-1})}{\alpha_i}$. To simplify the notation, we denote sometimes by y_i the ith observation y_{t_i} derived from the $(i-1)$th and ith clock synchronization. Figure 7 shows the clock deviation (its accurate approximation) measured at the synchronization time t_2, t_3, t_4 and t_5 and its variations y_3, y_4, y_5 relative to the intervals $[t_2, t_3]$, $[t_3, t_4]$, and $[t_4, t_5]$. Value t represents the current reference time at which a time estimate is requested.

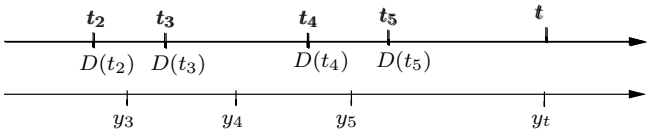

Figure 7. Series of the observed values

Our target is to predict variable Y_t, that is the average variation of the clock deviation during Γ time units since its last clock adjustment to provide a time estimate $T = T_i + \Delta H - Y_t \frac{\Delta H}{\Gamma}$ of t for any time $t \in (t_{i+1}, t_i)$, with uncertainty η. Intuitively, the time estimate T is equal to the value of the virtual clock $C(t)$ minus its predicted deviation since its last synchronization. Condition (1) in Section 3.1 provides a bound for $Y(t)$, that is $-\rho\Gamma \leq Y_t \leq \rho\Gamma$.

6.2. Our probabilistic model

Clearly, there are different ways to model our problem using the framework sketched in the previous section. Because of the limited resources of the sensors, we consider an autoregressive model AR(q) whose prediction value is based on the last q observed values, and with a Gaussian zero-mean white noise. The value at time t is given by the following equation,

$$Y(t) = \beta_1 Y(t-1) + \beta_2 Y(t-2) + \ldots + \beta_q Y(t-q) + b(\omega)W$$

with $\beta_1, \ldots, \beta_q$ constants derived from the observed values, and $b(\omega)W$ Gaussian white noise with zero mean. The choice of using this model, and particularly an $AR(3)$ model, was driven mainly by three factors crucial in a WSN:

- Computational efficiency. The coefficients of an AR model are more efficient to compute than in a general ARMA model. In fact, a node learns an AR(3) model by computing the coefficients β_1, β_2 and β_3. This can be done by calculating the least-square-error, and therefore by solving a linear system in three unknowns.
- Memory saving. Relying on the last three observed values not only improves the computational cost but also the memory usage of the sensor, since it needs to store only the last three observed values and its coefficients.
- Simplicity in the design and implementation, relevant in WSN.

– An AR model with a Gaussian white noise with zero mean allows to derive strong properties for our time series, useful to compute a bound for the error probability (see [37]).

It is important to note the inherent trade-off between efficiency (computational cost and memory) and precision of the time estimate. In fact, the time accuracy usually increases when using a higher number of observed values, that is an AR(p) model with $p > 3$. We are currently evaluating such a trade-off and the suitability of this model with respect to others. More precisely, we consider the following AR(3) model:

$$Y(t) = \beta_1 Y(t-1) + \beta_2 Y(t-2) + \beta_3 Y(t-3) + b(\omega)W$$

in which the prediction of $Y(t)$ depends on the last *three* observed values of $\{y_t\}$ derived by the last four clock synchronization. We denote by $t_{i-3}, t_{i-2}, t_{i-1}$, and t_i the time at which the last four synchronization occurred. Therefore, in our case $Y(t-1)$ refers to Y_{t_i}, $Y(t-2)$ to $Y_{t_{i-1}}$, and $Y(t-3)$ to $Y_{t_{i-2}}$. Clearly, the prediction Y_t of the variation of the clock deviation occurred in $[t_i, t]$, depends on its variation in $[t_{i-1}, t_i]$, $[t_{i-2}, t_{i-1}]$, and $[t_{i-3}, t_{i-2}]$. Notice that $b(\omega)W$ is a Gaussian *white noise* of $Y(t)$, with zero mean and standard deviation equal to $b(\omega)$ with $\omega = t - t_i$. This is because the variations of the *cumulative drift* are more likely to occur as ω grows. In this way we assume that drift variations are more likely to occur during longer time intervals. However, this does not affect the stationarity of the time series $\{Y_t\}$. Since $b(\omega)$ depends on the sensor clock (i.e. stability of the oscillator, thermal isolation) and on the environmental fluctuations, it should be tuned based on the network characteristics.

6.3. Predicting $Y(t)$

Our PCR method computes a prediction of $Y(t)$ based on the last three observed values of $\{y_t\}$. To do this, we need to compute $\beta_1, \beta_2, \beta_3$, that is to find the linear combination of the last three values that forecast Y_t with *minimum squared error*. Therefore, $\beta_1, \beta_2, \beta_3$ correspond to the coefficients of the *best linear predictor* and are obtained by computing the minimum of the following function

$$Q(\beta_1, \beta_2, \beta_3) = \sum_{i=4}^{N} (y_i - (\beta_1 y_{i-1} + \beta_2 y_{i-2} + \beta_3 y_{i-3}))^2$$

with N number of observations to consider. Therefore, $\beta_1, \beta_2, \beta_3$ are computed by solving the linear system of 3 linear equations in 3 un-

known

$$\begin{cases} \frac{\partial Q(\beta_1,\beta_2,\beta_3)}{\partial \beta_1} &= 0 \\ \frac{\partial Q(\beta_1,\beta_2,\beta_3)}{\partial \beta_2} &= 0 \\ \frac{\partial Q(\beta_1,\beta_2,\beta_3)}{\partial \beta_3} &= 0 \end{cases}$$

Clearly, for an accurate estimate of the coefficients $\beta_1, \beta_2, \beta_3$, the number N of observed values should be large (i.e Box and Jenkins suggest more than 50, see also [37]). However, storing these values might be too expensive for a sensor because of its limited resources. For this reason, during the learning phase, the sensor does not store these values but the matrix 3×4 associate to the linear system, and updates it at each reading.

6.4. The PCR method

The PCR method consists of two procedures:

- $setVariables(T_i, H(t_i))$ invoked at each clock synchronization t_i, with T_i accurate estimate of t_i computed by running a clock synchronization protocol,
- $PCR(t)$ returning an estimate of the current reference time t.

Figure 9 illustrates $setVariable()$, and function $PCR(t)$.

setVariables($T_i, H(t_i)$):

1) $y[1] \leftarrow y[2]$
2) $y[2] \leftarrow y[3]$
3) $y[3].val \leftarrow T_i - H(t_i) - lastDev$
4) $lastDev \leftarrow T_i - H(t_i)$
5) $y[3].m \leftarrow round(\frac{H(t_i) - lastClock}{\Gamma})$
6) $lastClock \leftarrow H(t_i)$
7) $N \leftarrow N + 1$
8) if $N < \Psi$
9) updateMatrix()
10) else if $N = \Psi$
11) computeCoeff()

PCR(t):

1) if $N < \Psi$
2) $l \leftarrow y[1].m + y[2].m + y[3].m$
3) $\beta[1] \leftarrow \frac{2}{3} - \frac{y[1].m}{l}$
4) $\beta[2] \leftarrow \frac{2}{3} - \frac{y[2].m}{l}$
5) $\beta[3] \leftarrow \frac{2}{3} - \frac{y[3].m}{l}$
6) $dev \leftarrow \beta[1]y[1].val + \beta[1]y[2].val + \beta[1]y[3].val$
7) $\Delta H \leftarrow H(t) - lastClock$
8) $T \leftarrow T_i + \Delta H - dev\frac{\Delta H}{\Gamma}$
9) return T

Figure 9. The PCR method.

The data structures used are the followings:

- an array $y[0 \,..\, 2]$ of records with two fields: $y[j].val$ representing the observed value y_{i-j} relative to the time interval $[t_{i-j}, t_{i-j-1}]$ for $j = 0, 1, 2$, and $y[j].m = \frac{t_{i-j} - t_{i-j-1}}{\Gamma}$ representing the times in which the sensor was unable to get its clock synchronized;
- an integer matrix $M = M(3 \times 4)$ of the coefficients of the linear system.

Array $y[0 \,..\, 2]$ is updated *every time* the sensor clock is synchronized (Fig 9 $setVariables$:1–5), while M is updated until the number of observed data is sufficiently reasonable to compute the coefficients $\beta_1, \beta_2, \beta_3$ (Fig 9 $setVariables$: 9,11), that is until it reaches a constant threshold Ψ (i.e. $\Psi = 50$).

Usually services built on top of statistical models involving a learning phase become active only after the learning phase. However, since time estimation is crucial in many sensor applications, and the clock adjustments represent the inputs of the learning phase, and they might be missing for the reasons discussed in the Introduction, we want to provide some *basic* time estimation during the learning phase, but with no bounded accuracy. During the initial phase the coefficients $\beta_1, \beta_2, \beta_3$ are the weights of the data y_i, y_{i-1}, y_{i-2} based on the time interval associate to each increment. This is based on the consideration that larger variations are more likely to occur in larger time intervals. Therefore, values associated to smaller time intervals are considered more accurate than values associated to larger ones. More precisely, if $J = [t_i,\ t_{i-3}]$ is the interval involved in the prediction, then weight $w_1 = \frac{2}{3} - \frac{t_i - t_{i-1}}{t_i - t_{i-3}}$ is associate with value $y[1].val$, $w_2 = \frac{2}{3} - \frac{t_{i-1} - t_{i-2}}{t_i - t_{i-3}}$ with $y[2].val$, and $w_3 = \frac{2}{3} - \frac{t_{i-2} - t_{i-3}}{t_i - t_{i-3}}$ with $y[3].val$ (Fig 9 PCR:2–5). Clearly $w_1 + w_2 + w_3 = 1$.

The following lemma provides a bound for the error probability of the time estimate $T = PCR(t)$. It says that after the learning phase, the time interval $[T - \eta, T + \eta]$ is correct with probability at least $1 - \frac{1}{\zeta^2}$ provided $b(\omega) < \frac{\rho\Gamma}{\zeta}$.

LEMMA 5. *If* $b(\omega) < \frac{\rho\Gamma}{\zeta}$ *then* $P((T - t) > \eta) \leq \frac{1}{\zeta^2}$

Proof If the ARMA process is driven by a Gaussian white noise, as in our case, then $Y(t)$ has a normal distribution $N(\lambda, b(\omega)^2)$ with $\lambda = \beta_1 y_i + \beta_2 y_{i-1} + \beta_3 y_{i-2}$. By applying Chebychev inequality we have $P(|Y - \lambda| \geq \rho\Gamma) \leq \frac{1}{\zeta^2}$, and therefore $P((T - t) > \eta) \leq \frac{1}{\zeta^2}$. □

Since function $b(\omega)$ depends on the network and on the environmental conditions, it is hard to write it in a generic analytical form, and compute the maximum number of clock adjustments that a sensor

can skip maintaining the same accuracy η. However, we can provide a condition on the number x of skips that a sensor can perform with maximum error η. In fact, a sensor node is able to compute at the end of the learning phase the standard deviation of the white noise at different data rate, that is, it can compute $b(\Gamma), b(2\Gamma), \ldots, b(a\Gamma)$. Therefore, given a constant $\zeta > 1$, if the standard variation of the white noise during $x\Gamma$ time units, with x positive integer, is less than $\frac{\rho\Gamma}{\zeta}$, the sensor can skip x clock adjustments with error probability at most $\frac{1}{\zeta^2}$. This relation shows an inherent trade-off between time accuracy and energy efficiency.

Similarly to the DCR method, the PCR method can be applied to to reduce the frequency of the clock synchronization, thus conserving energy, or in case of node isolation. Note that it is highly flexible since it allows the sensor node to tune the degree of the time accuracy (η, ζ) according to the application requirements and resource optimization.

7. Conclusions and future works

We have studied the time synchronization problem from a novel perspective, which consists of reducing the clock error between synchronization by exploiting information of the hardware clock. This perspective has a noticeable impact on WSN since it leads to energy conservation and enhances the robustness of the sensor clock in the presence of communication and node failures and mobility. For instance, our DCR method can be applied to WSN to reduce by a factor of 2 the frequency at which clocks are synchronized with noticeable energy and network bandwidth savings.

We have proposed also a probabilistic framework for time estimation which is based on time series forecasting and it is highly flexible. We believe that this scheme is general and can be applied to other problems in sensor networks to save energy, such as the problem of efficiently answering user queries at the sink regarding data produced. We are currently working on this problem.

Both methods are novel, and leave several open issues both on the theoretical and practical side. Because of the numerous differences between computer and sensor clocks, we believe it is important to study the behavior of the sensor oscillator and its cumulative drift on real sensors and under different conditions. We plan to implement and evaluate our methods on sensors and verify the suitability of our AR model.

Acknowledgements

The author would like to thank Susanna Pelagatti for introducing her to time series forecasting, Alessio Micheli, for helpful discussions on time series forecasting and statistical models, and the anonymous referees for their valuable comments.

References

1. J. Elson, L. Girod, and D. Estrin, Fine-grained network time synchronization using Reference Broadcasts, In Proc. of the 5th Symp. on Operating Systems Design and Implementation (OSDI '02), Dec 2002.
2. J. Elson and K. Römer, Wireless sensor networks: a new regime for time synchronization, ACM SIGCOMM Computer Communication Review, 33(1), pp. 149–154, Jan 2003.
3. K. Römer, Time synchronization in ad hoc networks, In Proc. of the 2nd Intl. Symp. on Mobile ad Hoc Networking and Computing (MobiHoc '01), Oct 2001.
4. S.Palchaudhuri, A. Saha, and D. Johnson, Adaptive clock synchronization in sensor networks, In Proc. of the 3rd Intl. Symp. on Information Processing in Sensor Networks (IPSN '04), Apr 2004.
5. H. Dai, and R. Han, TSync: a lightweight bidirectional time synchronization service for wireless sensor networks, ACM SIGMOBILE Mobile Computing and Communications Review, 8(1), pp. 125–139, Jan 2004.
6. S. Ganeriwal, R. Kumar, and M. B. Srivastava, Timing-sync protocol for sensor networks, In Proc. of the 1st Conf. on Embedded Networked Sensor Systems (SenSys 2003), Nov 2003.
7. A. Hu and S. D. Servetto, Time synch and localization: Asymptotically optimal time synchronization in dense sensor networks, In Proc. of the 2nd ACM Intl. Conf. on Wireless Sensor Networks and Applications, Sept 2003.
8. J.P. Sheu, C.M. Chao, and C.W. Sun, A clock synchronization for multi–hop wireless ad hoc networks, In Proc. of the 24th Intl. Conf. on Distributed Computing Systems (ICDCS '04, pp. 574-581, Mar 2004.
9. G.Gao and J. Welch, Accurate multi–hop clock synchronization in mobile ad hoc networks, In Proc. of the 33rd Intl. Conf. on Parallel Processing Workshops (ICPP '04), pp. 13–20, Aug 2004.
10. J. Elson, R. M. Karp, C. H. Papadimitriou, and S. Shenker, Global Synchronization in Sensornets, In Proc. of the 6th Latin American Symp. on Theoretical Informatics (LATIN '04), pp. 609–624, Apr 2004.
11. J. van Greunen, and J. Rabaey, Time synch and localization: lightweight time synchronization for sensor networks, In Proc. of the 2nd Intl. Conf. on Wireless Sensor Networks and Applications, Sept 2003.
12. P.Blum, L.Meier, and L.Thiele, Improved interval–based clock synchronization in sensor networks, In Proc. of the 3rd Intl. Symp. on Information Processing in Sensor Networks (IPSN '04), pp. 349-358, Apr 2004.
13. L.Meier, P.Blum, and L.Thiele, Internal synchronization of drift–costraint clocks in ad–hoc sensor networks, In Proc. of the 5th Symp. of Mobile ad Hoc Networking and Computing (MobiHoc '04), pp. 90–97, May 2004.

14. M. L. Sichitiu and C. Veerarittiphan Simple, accurate time synchronization for wireless sensor networks, In Proc. of the Wireless Communications and Networking Conference (WCNC '03), Mar 2003.
15. Q. Li and D. Rus, Global clock synchronization in sensor networks, In Proc. of the 23rd Conf. of the Communications Society (INFOCOM '04), Mar 2004.
16. M. Mock, R. Frings, E. Nett, and S.Trikaliotis, Continuous clock synchronization in wireless real–time applications, In Proc. of the 19th Symp. on Reliable Distributed Systems (SRDS'00), pp. 125–134, Oct 2000.
17. W.Su, and I. Akyldiz, Time–diffusion sensor prootocol for sensor networks Technical report, Georgia Institute of Technology, 2002.
18. R.Fan and N. Lynch, Gradient clock synchronization, In Proc. of 23rd Annual Symp. on Principles of Distributed Computing (PODC '04), pp. 320–327, Jul 2004.
19. I. Akyildiz , W. Su , Y. Sankarasubramaniam , and E. Cayirci, Wireless sensor networks: a survey, Computer Networks, Intl. Journal of Computer and Telecommunications Networking, 38(4), pp.393-422, Mar 2002.
20. G.J.Pottie, and W.J.Kaiser, Wireless integrated network sensors, Comm. of the ACM, 43(5), pp. 551–558, 2000.
21. J. Polastre, R. Szewczyk, C. Sharp, and D. Culler The Mote Revolution: Low Power Wireless Sensor Network Devices, In Proc. of Hot Chips 16: A Symposium on High Performance Chips, Aug 2004.
22. TinyOS, http://webs.cs.berkeley.edu/tos/
23. C. Fetzer and F.Cristian, Integrating external and internal clock synchronization, Journal of Real-Time Systems, 12(2), pp. 123-171, Mar 1997.
24. T.K. Srikanth, S. Toueg, Optimal clock synchronization, Journal of ACM, 34(3), pp. 626–645, 1987.
25. F.Cristian, Probabilistic clock synchronization, Distributed Computing 3(3), pp. 146–158, 1989.
26. C.Fetzer, and F.Cristian, Building fault-tolerant hardware clocks, In Proc. of the 7th IFIP Intl. Working Conf. on Dependable Computing for Critical Applications, pp. 59-78, Jan 1999.
27. C.Liao, M.Martonosi, and D. Clark, Experience with adaptive globally–synchronizing clock algorithm, In Proc. of the 11th Symp. on Parallel Algorithms and Architectures (SPAA '99), pp. 106–114, Jun 1999.
28. K.Marzullo, and S.Owicki, Maintaining the time in a distributed system, In Proc. of the 2nd Annual Symp. on Principles of Distributed Computing (PODC '83) pp. 295–305, Aug 1983.
29. P. Verissimo, L. Rodrigues, and A.Casimiro, Cesiumspray: a precise and accurate global clock service for large-scale systems, Journal of Real-Time Systems, 12(3), pp. 243-294, May 1997.
30. K.Schossmaier, B.Weiss. An Algorithm for Fault–Tolerant Clock State&Rate Synchronization. In Proc. of the 18th IEEE Symp. on Reliable Distributed Systems (SRDS '99), pp. 36-47, Oct 1999.
31. K. Schossmaier, An interval-based framework for clock rate synchronization, In Proc. of the 16th Symp. on Principles of Distributed Computing (PODC '97), pp. 169–178, Aug 1997.
32. U.Schmid, and K.Schossmaier, Interval-based clock synchronization, Journal of Real-Time Systems 12(2), pp. 173-228, Mar 1997.
33. F. Cristian, and C. Fetzer, Probabilistic internal clock synchronization, In Proc. of the 13th Symp. on Reliable Distributed Systems (SRDS '94), pp. 22–31, Oct 1994.

34. D.L.Mills, Adaptive hybrid clock discipline algorithm for the Network Time Protocol. Trans. Networking 6(5), pp. 505–514, Oct 1998.
35. The Science of Time–keeping. Hewlett Packard, application Note 1289.
36. F. Vernotte, J. Delporte, M. Brunet, and T. Tournier, Uncertainties of drift coefficients and extrapolation errors: application to clock prediction, Metrologia, Feb 2001.
37. P. J. Brockwell, and R. A. Davis, Introduction to Time Series and Forecasting, Springer Text in Statistics, 1994.
38. L. R. Rabinier, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, In Proc. of IEEE, 77(2), pp. 257–285, 1989.
39. J. L. Elman, Finding structure in time, Cognitive Science, 14:179.211, 1990.
40. V. N. Vapnik, The nature of statistical learning theory, Springer, New York, 1995.
41. J. Hertz, A. Krogh, and R. Palmer, Introduction to the Theory of Neural Computation, Addison–Wesley: Redwood City, California, (1991).
42. R.Wolski, Dynamically forecasting network performance using the Network Weather Service, Journal of Cluster Computing 1, pp. 119–132, 1998.
43. P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, Energy–efficient computing for wildlife tracking: design trade–offs and early experience with Zebranet. Proc. 10th Conf. on Architectural Support for Programming Languages and Operating Systems, Oct 2002.