A Resource–efficient Time Estimation for Wireless Sensor Networks

Daniela Tulone
Department of Computer Science
University of Pisa
via F. Buonarroti, 2
56124 Pisa, Italy
tulone@di.unipi.it

ABSTRACT

We study the problem of providing a sensor node with an accurate estimate of the current time, from a *novel prospective* 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 low–energy, or intermittent connectivity, or process failures, but still requires an accurate estimate of the time. We propose and analyze two *efficient* clock reading methods, one deterministic and the other probabilistic, which are designed to work in synergy with clock synchronization protocols to provide a better time estimate. Our deterministic method achieves a better accuracy by exploiting the *sign* of the *global deviation* of the hardware clock from the reference time, and can be applied to reduce the frequency of the periodic clock adjustments by a factor 2, while maintaining the same *time uncertainty*. The second method based on time series forecasting, is more flexible than the previous one since it is independent of the frequency at which clock synchronization occur.

Categories and Subject Descriptors

G.3 [**Probability and statistic**]: Time series analysis; C.4 [**Performance of systems**]: Measurement techniques; C.2.1 [**Network Architecture and Design**]: Wireless communication.

General Terms

Algorithms, Performance, Measurement.

Keywords

Time estimation, clock drift rate, clock synchronization, time series forecasting, 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). For instance, it plays a crucial role in data integration and sensor reading fusion. 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 [26]. Time synchronization plays a crucial role in other WSN settings, such as object tracking, duplicate detection and in distributed beam–forming. These applications motivate the increasing attention to the clock synchronization problem in WSN in the last few years, and the numerous protocols that have been proposed such as [10, 11, 2, 4, 5, 7, 8, 9, 12, 13]. In fact, as pointed out by Elson et al. [1], WSN show some unique characteristic that makes hard the application of well–known clock synchronization protocols such as NTP [22]. Time synchronization service in WSN has to meet challenges which are substantially different from those in infrastructure–based networks. For instance, as each sensor has a finite battery source and communication is expensive in terms of energy, an important issue of WSN is *energy efficiency.* In addition, WSN show a higher failure probability over the time than in traditional networks due to battery depletion or destruction of the sensors, 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. These elements lead to strong *energy–efficiency*, *self–configuration* and *robustness* requirements. In the last few years several clock synchronization protocols for WSN have been proposed based on different approaches, such as the Reference Broadcast Synchronization (RBS) proposed by Elson et al. [10], or hierarchical approaches [13], or interval–based [8, 9], or probabilistic approaches for energy efficiency [7]. However, despite their diversity, these proposals share a common viewpoint: they provide an accurate time estimate by means of *periodic* synchronization performed by each sensor node and based on messages exchanged with its neighbor nodes. Clearly, each clock adjustment is energy–consuming since it involves transmitting messages and listening, besides the computational cost. Therefore, reducing the frequency of periodic clock synchronization would result in a noticeable energy saving, and in a natural extension of the network lifetime. In addition, for the reasons previously discussed, 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 [6] (i.e. environmental sensors, or sensors thrown by a plane). Notice that these situations are related to the intrinsic nature of the WSN, and therefore should be taken into consideration as well. Therefore, it is also important to study the time accuracy of a sensor clock in case of complete temporary isolation of the sensor. The problem of improving the clock accuracy *between synchronization* has not been taken into consideration by previous works on WSN. This paper provides the first answer to the question: *"what time accuracy can be provided in case of node isolation or low–energy?"*

Our contributions Our first contribution consists in pointing out an unexplored aspect of the time synchronization problem in WSN. We tackle the problem of providing an accurate approximation of the reference time from a *novel prospective* which is complementary to the view analyzed by previous works focusing on clock synchronization, and that consists in improving the accuracy of a clock *between synchronization.* In fact, all previous works take into account a maximum error growth $\hat{\rho}\Delta t$ between synchronization, with $\hat{\rho}$ maximum hardware drift rate and Δt time elapsed since the last synchronization.

We propose two methods: a deterministic clock reading method called DCR, and a probabilistic one called PCR, that provides a probabilistically accurate time estimate with constant *time uncertainty* (error bound). Our DCR method improves in the vast majority of cases, the clock accuracy between synchronization by exploiting the *sign of the deviation* of the hardware clock from the real time at a given point in time. Despite the broad literature on clock synchronization related to the hardware drift rate [22, 20, 16, 19], this represents a novel approach. We introduce a new metric, called *cumulative hardware drift rate*, that provides, in contrast with the traditional drift rate, *global information* on the clock behavior. It measures the clock deviation from the reference time accumulated since its initial time. Our DCR method is general, and can be applied to WSN to reduce by a factor 2 the frequency of the clock synchronization while maintaining the same time uncertainty, with noticeable energy saving and enhancement of the network bandwidth. It can also be applied to improve the time accuracy by reducing the maximum error by a factor 2. Our probabilistic method PCR proposes a *general framework* that is based on an adaptation of *time series forecasting* to predict the value of the clock deviation at the current time based on a narrow window of past events. Time series forecasting is a powerful statistical technique that has been applied to provide dynamic resource performance forecasts, such as TCP/IP end–to–end throughput and latency, and quality–of–service guarantees [25]. However, it has not been applied to our context. The PCR method returns a time value, which is probabilistically accurate, with constant time uncertainty. Clearly there is a trade–off between accuracy and flexibility. It is more flexible that the DCR method since it does not assume a specific frequency in the clock synchronization (it provides a time estimate with a constant time uncertainty even when the sensor has "skipped" more than 2 clock adjustments). Clearly, the DCR and the PCR methods can be combined and work in synergy with a clock synchronization protocol (such as [8] that works for *sporadic* clock synchronization) to improve the clock accuracy. Our methods show some attractive features for WSN: they are *computationally efficient*, *general*, and *simple* to implement.

The rest of the paper is organized as follows: Section 2 compares related works and Section 3 presents our model and the techniques used. We present in Section 4 our DCR method and analyze it. Section 5 introduces our PCR method based and analyzes it. Section 6 discusses future works.

2. RELATED WORKS

Clock synchronization protocols in WSN In the last few years several clock synchronization protocols have been proposed such as [10, 11, 2, 4, 5, 7, 8, 9, 12, 13]. Among these, some protocols [7] are based on the Reference Broadcast Synchronization (RBS) proposed by Elson et al. [10], others on a hierarchical structure [2, 13], or very recently on the interval–based paradigm [8, 9], originally introduced by Marzullo and Owicki [17]. All of these proposals provide time synchronization by periodically adjusting each clock to its neighbors. As mentioned in the Introduction, our approach is very different since it does not synchronize the sensor clock, but is designed to work *between* clock synchronization. Therefore, it is not possible to compare our algorithms 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 their hardware drift rate is not a novel idea: for instance, Schossmaier et al.[19] and NTP [22] include drift information in their clock adjustments. Fetzer and Cristian show in [16] how to build clocks with bounded drift rate from components *off–the–shell*, and achieve high accuracy by means of *drift synchronized clocks.* Their work is quite close to our DCR method for analyzing the behavior of the hardware drift rate, but diverges for its *strong* assumptions (i.e. requiring hardware support). Our work diverges from all previous proposals mainly for two reasons: 1) for exploiting the *global* deviation of the clock, the deviation from the reference time accumulated since the initial time, in contrast with the hardware drift rate that captures the *local speed* of the clock, and 2) for reducing the maximum error growth $\hat{\rho}\Delta H$ between synchronization. As discussed in Section 4.2, this novel prospective can lead to an improvement of the Cristian and Fetzer [14] optimality result for external clock synchronization.

Probabilistic approaches There are few protocols that are probabilistic in nature: for instance, the probabilistic remote clock reading protocol proposed by Cristian [15], or Cristian and Fetzer probabilistic clock synchronization protocol [21]. Statistical techniques are also applied in NTP [22] at synchronization time to compute an accurate approximation of the time based on the timestamps received from the neighbors. Very recently Palchaundhuri et al [7] have proposed a probabilistic protocol for WSN based on the RBS, for energy efficiency. However, these approaches diverge from ours for solving the clock synchronization problem.

Statistical techniques such as time series forecasting, has been applied by Wolski [25] to provide dynamic resource performance forecast, such as to predict the TCP/IP end–to–end throughput and latency. We are not aware of any probabilistic study on the clock deviation between synchronization. Vernotte et al. [23] apply an auto–regressive linear model to estimate the time uncertainty of the on–board oscillator for different type of noise. However, 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 clock function is denoted by $H(t)$. Because of the impressions of the oscillator, variations in the temperature, pressure, and aging, a hardware clock can drift apart from real time. The *hardware drift rate* function $\rho(t)$, defined as the first derivative of $H(t)$ over the time minus 1, provides a measure for the punctual variation of the clock speed. It is common practice to assume a maximum drift rate $\hat{\rho}$ for the hardware clock. This implies that for any real time $t_1 < t_2$ the hardware clock function measures the passage of time in $[t_1, t_2]$ with an error of at most $\hat{\rho}(t_2 - t_1)$. That is,

$$(1-\hat{\rho})(t_2 - t_1) \le H(t_2) - H(t_1) \le (1+\hat{\rho})(t_2 - t_1) \quad (1)$$

For $t_1 = 0$ and for a negligible initial error, this implies that $(1-\hat{\rho})\ t \ \le\ H(t)\ \le\ (1+\hat{\rho})\ t$.

As mentioned before, our algorithms exploits the behavior of the *deviation* of the hardware clock from the real time. The clock deviation function is denoted by D, it is a function mapping real time values into reals and such that $D(t) = t - H(t)$ for each time t. Clearly, $D(t)$ can grow unbounded with the time. It is useful for our further discussion to write $D(t)$ as $D(t) = \delta(t)\ t$ where $\delta(t)$ is a function *similar* to $\rho(t)$ and called *cumulative hardware drift rate* function. Function δ maps real time values to real values in $[-\hat{\rho}, \hat{\rho}]$, and is such that $\delta(t)$ is the *slope* of the line passing through the origin O and point $H(t)$, minus 1. Figure 1 illustrates the hardware clock function $H(t)$ limited by the lines $(1 \pm \hat{\rho})\ t$, the cumulative drift at time τ and the hardware drift rate $\rho(\tau)$. The dashed line bisector represents the reference time.

Notice that the cumulative drift provides information about the clock deviation *accumulated* since its initial time, while the hardware drift rate provides a *local measure* representing the speed of the clock at that time. Figure 1 shows the difference between the two metric, the hardware drift represents the slope of the tangent of the clock function minus 1. The cumulative drift seems a more appropriate metric for our study whose target is to improve the accuracy of the clock by exploiting past information on the its deviation. The hardware clock function can be written as $H(t) = (1 + \delta(t))\ t$, and $|\delta(t)| < \hat{\rho}$ by condition (1). The relation between $\rho(t)$ and $\delta(t)$ is provided by the equation $\rho(t) = \frac{d\delta(t)}{dt}t + \delta(t)$.

3.2 The interval–based paradigm

We adopt the interval–based paradigm, originally proposed by Marzullo and Owicki [17] and later refined by

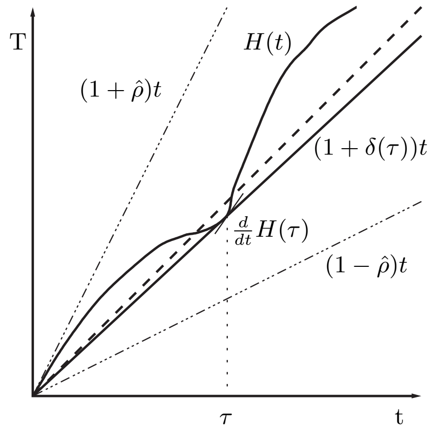

Figure 1: The hardware clock function $H(t)$, the cumulative drift $\delta(\tau)$, and the hardware drift rate $\rho(\tau)$ at time τ.

Shmidt and Shossmaier [20], for its advantages in WSN over time estimates [8]. 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. Each node maintains a *virtual clock function* $C(t)$ that is synchronized periodically, namely every Γ time units, to a more accurate estimate of the current time. We assume that the clock synchronization occurs at real time $t_1, t_2, \ldots, t_i, \ldots$ with $t_i \approx i\ \Gamma$, and that clock $C(t)$ is synchronized at the i^{th} synchronization to T_i (value obtained by running a clock synchronization protocol among the sensor neighbor or from a more accurate source) with maximum inherited error ε. The time interval of clock $C(t)$, denoted by $I_C(t)$, is a function mapping real time values to clock values and such that $I(t) = [C(t) - E(t), C(t) + E(t)]$ for each $t \in [t_i, t_{i+1})$, with $C(t) = H(t) + (T_i - H(t_i))$ and $E(t) = \varepsilon + \hat{\rho}(H(t) - H(t_i))$. A time interval $I_C(t)$ is correct at time t if $t \in I_C(t)$. 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 error growth since its last synchronization by $\hat{\rho}\Delta H$, the maximum time uncertainty by $\eta = \varepsilon + \hat{\rho}\Gamma$, and the maximum hardware drift rate $\hat{\rho}$ by ρ. We neglect terms smaller than $\rho^2\Gamma$. Clearly, if the time interval between two synchronization increases, then the maximum error bound $E(t)$ exceeds η. Figure 2 illustrates the periodic clock adjustments performed at time Γ, 2Γ and 3Γ.

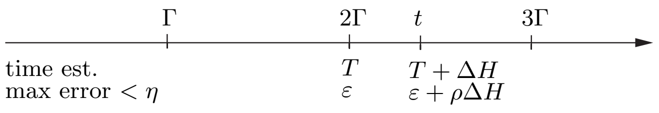

Figure 2: Periodic clock synchronization

3.3 Problem statement

Since the error bound $E(t)$ grows with the time elapsed since its last synchronization, our target is to provide a time estimate with the *same* time uncertainty η in case the sensor node is unable to synchronize its clock due to low–battery or intermittent connectivity, or network partition or process failures. Clearly, if the clock synchronization does not oc-

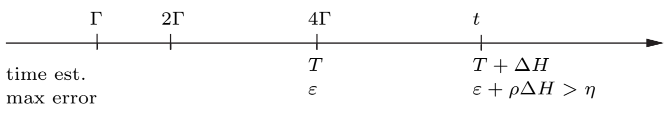

Figure 3: Irregular clock synchronization

cur necessarily regularly, the time period between two consecutive synchronization $[t_{i-1},\ t_i]$ can vary. However, for simplicity we assume that clock synchronization occur at a regular basis in the WSN, and that therefore $(t_i - t_{i-1})$ is a multiple of Γ. This assumption is reasonable in WSN since sensors sleep most of the time to save energy battery. Figure 3 shows a scenario in which clock synchronization occur at time Γ, 2Γ, and 4Γ and the time evaluation of t is requested after more Γ time units since its last synchronization.

3.4 Time series forecasting

In this subsection we briefly review some basic concepts on time series forecasting that will be applied in the PCR method in Section 5. Time series seem to suit well our problem because of their temporality. We refer the reader to [24] 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 the 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 with the time. Therefore, an important step to time series modelling 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 will adopt this model to predict the current deviation of the clock because it is simpler, more efficient and therefore more suitable to our discussion than general ARMA models.

4. A DETERMINISTIC CLOCK READING METHOD

We propose here a deterministic clock reading method that improves in most cases the accuracy of the time estimate between synchronization by exploiting the sign of the clock deviation at the last synchronization. More precisely, the DCR method reduces by half the error growth since its last synchronization. It is general and applicable to both traditional networks and WSN, to improve the time accuracy. However, as discussed previously it has a bigger impact in WSN due to its limited resources.

4.1 Method overview

Our DCR method is designed to reduce the maximum error growth $\rho\Delta H$ between adjustments without the aid of additional inputs. 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 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 lead to a better 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.

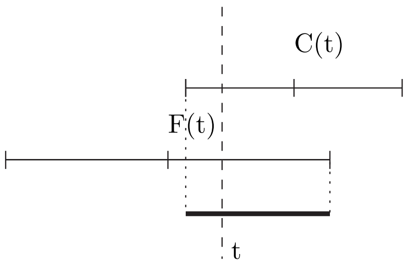

Figure 4: Intersection of time intervals $I_C(t)$ and $I_F(t)$

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. Notice that it would be tempting to think that if a clock C proceeds slower than the real time at time t, then $[C(t), C(t) + E(t)]$ is a correct time interval of t. In fact, there are cases in which this does not hold due to the inherited error and the variation of the hardware cumulative drift in $[t_i, t]$. These are some of the intuitions that has 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 sensor 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 brings two issues:

1. Detect the sign of the cumulative drift at 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. Let us suppose that the last synchronization occurred at time t_i and that T_i is the accurate approximation of t_i. Since the time interval of $C(t)$ is correct 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 than the real time at t_i and t. Notice that the fact that a sensor can detect it at time t_i is a feature of the DCR method which can be applied to save energy, as show in Section 4.2. Notice that the accuracy of this test improves with the time, and since it exploits the clock deviation, paradoxically it works better with cheap clocks such as in WSN. For instance, drifts smaller than $\frac{\rho}{2}$ can be detected after $2\varepsilon\rho^{-1}$ time units, that is less than 100 sec for $\rho = 10^{-4}$ and $\varepsilon = 5$ msec.

The second issue is the most crucial for our DCR method. Before computing our associate clock function we define the clock properties that can lead to optimal accuracy. We define a class of clock functions $\mathcal{A}_{C,t}$ called *associate clock functions* of $C(t)$ at time t.

DEFINITION 3. *$F \in \mathcal{A}_{C,t}$ at time $t \in [t_i, t_{i+1})$ if satisfies the following properties:*

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

Notice that condition 3 provides an upper bound $\varepsilon + \rho\Delta H$ for 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 $E(t) = \varepsilon + \rho\Delta H$ in the interval $[t_i, t_{i+1})$.

The associate clock function of $C(t)$ computed by our DCR method is defined as followings:

$$A(t) := \begin{cases} (1+\rho)H(t) + \gamma - \rho H(t_i) & \text{for } \delta(t_i), \delta(t) \leq 0 \\ (1-\rho)H(t) + \gamma + \rho H(t_i) & \text{for } \delta(t_i), \delta(t) \geq 0 \end{cases}$$

where $\gamma = T_i - H(t_i)$ is a constant set at the last synchronization.
Figure 5 illustrates the DCR method. Line (2) checks the sign of the cumulative drift, and line (3) returns the midpoint of the time interval $I_C(t) \cap I_F(t)$ representing the time estimate, and its error bound.

The following lemmas prove the correctness of the DCR method, they show that $I_C(t) \cap I_A(t)$ is a correct time interval. The correctness of time interval $I_C(t)$ follows from the drift bound ρ, condition (1) at Section 3.1, as shown in [17].

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

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

1)  $err \leftarrow \varepsilon + \rho(H(t) - H(t_i))$
2)  if $(H(t_i) - T_i > err) \vee (H(t_i) - T_i < -err)$
3)      return $\langle \frac{A(t)+C(t)}{2}, \frac{err}{2} \rangle$
```

Figure 5: The DCR method.

PROOF. Let us suppose $\delta(t_i), \delta(t) \geq 0$. Since the cumulative drift of $A(t)$ is equal to $\delta(t) - \rho - \rho\delta(t)$ and $\delta(t) - \rho\delta(t) > 0$, then ρ is a valid bound for the cumulative drift of $A(t)$ at real time t_i and t. Therefore, 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 $\delta(t_i), \delta(t) \leq 0$. $\square$

LEMMA 2. *If the sign of the cumulative drift remains unchanged at time t_i and t, then DCR returns a correct estimate of t with maximum error $\varepsilon + \frac{\rho}{2}\Delta H$.*

Remark A natural comment on the DCR method is that it works only if the sign of the cumulative drift is unchanged. Notice that since the hardware clock $H(t)$ is undiscipline, and the cumulative drift represents the deviation of the clock in a *global scale*, its variations are *minimal* and decrease with the time, in contrast with the drift rate that captures the local clock behavior. In fact, the fluctuation of the drift rate is larger in small intervals (i.e. seconds). We have introduced the cumulative drift with this purpose.

4.2 Considerations

Our DCR method shows the following properties, particularly attractive for WSN.

- **Generality and wide applicability** It is important to notice that our DCR method is not tight to a specific hardware drift distribution, and works for any type of hardware clocks. It assumes only a valid upper drift bound ρ of the hardware clock (common assumption). However, since our method exploits the clock deviation it works better with cheap crystal clocks, such as sensor clocks. Because of its generality and its different view, our DCR method can be encapsulated in previous well–known clock synchronization protocols. In traditional networks the DCR method can also coexist with *calibrated clocks* [16] by replacing the hardware clock function with a calibrated one to improve the time accuracy.
- **Efficiency and simplicity** Computational and resource efficiency and simplicity in the implementation, are critical in WSN for its limited resources.

The DCR method has the two main applications in WSN:

1. **Save energy and improve network bandwidth**: after synchronizing its clock, a sensor node can decide if to skip the following synchronization if its hardware clock (undiscipline) deviates from the real time more than $\varepsilon + \rho\Gamma$.
2. **Improve the time accuracy**: it provides a way to reduce the error growth by half.

Optimality bound It is interesting to see how it is possible to lower the *optimality bound* for the external deviation of a clock that is periodically adjusted, by strengthening the clock model using information regarding the deviation of the hardware clock. In [14] Fetzer and Cristian have showed that 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 reading error* and Δ the error performed by the reference source in reading the reference time. The DCR method shows that refining the hardware clock model by adding realistic information about its behavior, can lead to a better time accuracy. This would be particularly advantageous for WSN since would save energy and resources.

5. A PROBABILISTIC CLOCK READING METHOD

5.1 Method overview

We propose here a *general framework* that provides a probabilistic estimate of the time with uncertainty $\eta = \varepsilon + \rho\Gamma$, in case the sensor node "skipped" *few* clock adjustments due to intermittent connectivity or process failures or low–energy. In the previous section we have proposed a deterministic method that can be applied to refine the accuracy of the clock reading by reducing the maximum error growth $\rho\Delta H$ by a factor 2. However, if $\Delta H > 2\Gamma$ the time uncertainty exceeds the requested bound η, even when applying the DCR method. Our PCR method provides a probabilistic time estimate also in this case with time uncertainty η. It is more flexible than the DCR method since it is independent of the occurrences (maybe irregular) of the clock synchronization. It predicts the current deviation of the clock based on some past events by applying time series forecasting.

A straightforward solution is to consider a time series $\{X_{t_i}\}$ where X_{t_i} is the deviation $D(t_i)$ of the hardware clock at the synchronization time t_i. However, this approach presents two main problems:

1. The time series $\{X_{t_i}\}$ contains some trend, since the deviation $D(t)$ is proportional to the time t, and also some seasonal component, due for instance to periodic (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 can be 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 can be overcome 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} that 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 i^{th} observation y_{t_i} derived from the $(i-1)^{th}$ and i^{th} clock synchronization. Figure 6 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.

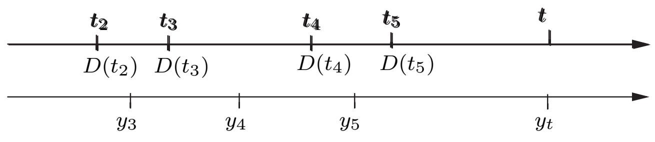

Figure 6: Series of the observed values

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 $T = T_i + \Delta H - Y_t\frac{\Delta H}{\Gamma}$ for time $t \in (t_{i+1}, t_i)$ with uncertainty η. Condition (1) at Section 3.1 provides a bound for $Y(t)$, that is $-\rho\Gamma \leq Y_t \leq \rho\Gamma$.

5.2 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, more precisely an AR(3) model in which the prediction of a value depends on the last three observed values, and with a Gaussian white noise. The choice of using this model and relying only on three values 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 addition, in an AR(3) model the sensor needs to compute only three coefficients, and therefore solves a linear system in three unknown.

- Memory saving. Relying on the last three observed values not only improves the computational cost, but also enhances the memory usage of the sensor, since it needs to store only three observed values.

- Simplicity in the design and implementation, relevant in WSN.

- An AR model with a Gaussian white noise allows to derive strong properties for our time series, useful to compute a bound for the error probability.

However, it is important to notice the inherent trade–off between efficiency, in terms of computational cost and memory, and precision of the time estimate. Clearly, the time accuracy increases by 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 $\{y_t\}$ obtained 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 deviation equal to $b(\omega)$ with $\omega = t - t_i$. Intuitively, since the forecasting *does not* occur at regular intervals, the variations of the *cumulative drift* are more likely to occur as the probability (and magnitude) of variations in the temperature or pressure increase, and therefore as ω grows. This does not affect the stationarity of the time series $\{Y_t\}$. Since $b(\omega)$ depends on the sensor clock (i.e. stability, thermic isolation) and on the environmental fluctuations, it should be tuned by the application. Our model allows us to derive strong properties for the time series $\{Y_t\}$, useful to compute a bound for the error probability (see [24]).

5.3 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, $\beta_1, \beta_2, \beta_3$ corresponds to the coefficients of the *best linear predictor* and are obtained by computing the minimum of 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 the 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 unknown

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

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). It is important to notice that because of the limited resources, the sensor does not maintain these values but the matrix 3×4 of the system coefficients (see Fig 7:9).

5.4 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) that returns an estimate of the current reference time.

Figure 7 illustrates *setVariable*(), and Figure 8 function *PCR*(t). The data structures used are the followings:

- an array $y[0 \mathrel{..} 2]$ 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 $y[0 \mathrel{..} 2]$ is updated *every time* the sensor clock is synchronized (Fig 7:1–5), while M is updated until the number of observed data is sufficiently reasonable to compute the coefficients $\beta_1, \beta_2, \beta_3$ (Fig 7: 9,11), that is until it reaches a constant threshold Ψ.

```
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()
```

Figure 7: setVariables invoked at time t_i.

```
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)  $\Delta H \leftarrow H(t) - lastClock - dev$
8)  $T \leftarrow T_i + \Delta H - dev\frac{\Delta H}{\Gamma}$
9)  return $T$
```

Figure 8: The PCR method.

In the initial phase coefficients $\beta_1, \beta_2, \beta_3$ are set to be the weights of the data y_i, y_{i-1}, y_{i-2} and are based on the accuracy of these observations. More precisely, if $J = [t_i,\ t_{i-3}]$ is the interval involved in the prediction, then $y[1].val$ is given weight $w_1 = \frac{2}{3} - \frac{t_i - t_{i-1}}{t_i - t_{i-3}}$, $y[2].val$ weight $w_2 = \frac{2}{3} - \frac{t_{i-1} - t_{i-2}}{t_i - t_{i-3}}$, and $y[3].val$ weight $w_3 = \frac{2}{3} - \frac{t_{i-2} - t_{i-3}}{t_i - t_{i-3}}$ (Fig 8:2–5), thus giving more weight to more accurate observations (with smaller intervals). 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 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 3. *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 $N(\lambda, b(t)^2)$ with $\lambda = \beta_1 y_i + \beta_2 y_{i-1} + \beta_3 y_{i-2}$, and the bound for the error probability is obtained by applying Chebychev inequality. $\Box$

6. CONCLUSIONS AND FUTURE WORKS

We have studied the time synchronization problem from a novel prospective that consists in improving the clock accuracy between synchronization by exploiting global and not local information regarding the behavior of the hardware clock. This prospective has a noticeable impact on WSN for their strong energy–efficiency, robustness and self–configuration requirements. For instance, our DCR method can be applied to WSN to reduce by a factor 2 the frequency at which clock synchronization occur, with noticeable energy and network bandwidth savings.

We have presented also a general probabilistic framework based on time series forecasting and highly flexible. Both approaches (sign–based and time–series–based) are novel, and leave several issues opened both on the theoretical and practical side. For instance, a refinement over the optimality bound by Fetzer and Cristian [14] obtained by means of a stronger but realistic clock model. It is also very important to evaluate our PCR framework through experiments on sensor clocks under different conditions and verify the suitability our model with respect to other statistical models. We are currently working on this.

Acknowledgments The author would like to thank Susanna Pelagatti for introducing me to time series forecasting, and Maurizio Pratelli and Alessio Micheli for very helpful discussions on time series forecasting and statistical models.

7. REFERENCES

[1] J. Elson, K. Römer *Wireless sensor networks: a new regime for time synchronization.* ACM SIGCOMM Computer Communication Review, 33(1), pp. 149–154, Jan 2003.

[2] H. Dai, 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.

[3] J. Elson, R. M. Karp, C. H. Papadimitriou, S. Shenker *Global Synchronization in Sensornets* In Proc. of the 6^{th} Latin American Symp. on Theoretical Informatics (LATIN '04), pp. 609–624, Apr 2004.

[4] J. van Greunen, J. Rabaey *Time synch and localization: Lightweight time synchronization for sensor networks* In Proc. of the 2nd ACM Intl. Conf. on Wireless sensor networks and applications, Sept 2003.

[5] A. Hu, 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.

[6] I. F. Akyildiz , W. Su , Y. Sankarasubramaniam , E. Cayirci *Wireless sensor networks: a survey.* Computer Networks, Intl. Journal of Computer and Telecommunications Networking, 38(4), pp.393-422, Mar 2002.

[7] S.Palchaudhuri, A. Saha, 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.

[8] P.Blum, L.Meier, L.Thiele *Imporved interval–based clock synchronization in sensor networks* In Proc. of the 3rd Intl. Symp. on Information Processing in Sensor Networks (IPSN '04), Apr 2004.

[9] L.Meier, P.Blum, 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), May 2004.

[10] 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.

[11] K. Römer *Time synchronization in ad hoc networks* In Proc. of the 2nd ACM Intl. Symp. on Mobile ad Hoc Networking and Computing (MobiHoc '01), Oct 2001.

[12] M. L. Sichitiu and C. Veerarittiphan *Simple, Accurate Time Synchronization for Wireless Sensor Networks* In Proc. of the IEEE Wireless Communications and Networking Conference (WCNC 2003), Mar 2003.

[13] S. Ganeriwal, R. Kumar, M. B. Srivastava *Timing-sync Protocol for Sensor Networks* In Proc. of the 1st Conf. ACM SenSys, Nov 2003.

[14] C. Fetzer and F.Cristian. *Integrating external and internal clock synchronization.* Journal of Real-Time Systems, 12(2), pp. 123-171, Mar 1997.

[15] F.Cristian. *Probabilistic clock synchronization.* Distributed Computing 3(3), pp. 146-158, 1989.

[16] C.Fetzer, 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.

[17] K.Marzullo, S.Owicki. *Maintaining the Time in a Distributed System.* In Proc. of the 2nd Annual ACM Symp. on Principles of Distributed Computing (PODC '83) pp. 295-305, Aug 1983.

[18] P. Verissimo, L. Rodrigues, A.Casimiro. *Cesiumspray: a precise and accurate global clock service for large-scale systems.* Real-Time Systems, 12(3), pp. 243-294, May 1997.

[19] K. Schossmaier. *An Interval-based Framework for Clock Rate Synchronization.* In Proc. of the 16th ACM Symp. on Principles of Distributed Computing (PODC '97), pp. 169-178, Aug 1997.

[20] U.Schmid, K.Schossmaier. *Interval-based Clock Synchronization*, Journal of Real-Time Systems 12(2), pp. 173-228, Mar 1997.

[21] F. Cristian, C. Fetzer: *Probabilistic Internal Clock Synchronization.* In Proc. of the 13th Symp. on Reliable Distributed Systems (SRDS '94), pp. 22-31.

[22] D.L.Mills. *Adaptive hybrid clock discipline algorithm for the Network Time Protocol.* IEEE/ACM Trans. Networking 6(5), pp. 505-514, Oct 1998.

[23] F. Vernotte, J. Delporte, M. Brunet, T. Tournier *Uncertainties of drift coefficients and extrapolation errors: application to clock prediction* Metrologia, Feb 2001.

[24] P. J. Brockwell, R. A. Davis *Introduction to Time Series and Forecasting* Springer Text in Statistics, 1994.

[25] R.Wolski *Dynamically forecasting network performance using the Network Weather Service* Journal of Cluster Computing 1, pp. 119–132, 1998.

[26] G.J.Pottie, W.J.Kaiser *Wireless integrated network sensors.* Comm. of the ACM, 43(5), pp. 551–558, 2000.