Tracking Low-Precision Clocks With Time-Varying Drifts Using Kalman Filtering

Hayang Kim, Xiaoli Ma, *Senior Member, IEEE*, and Benjamin Russell Hamilton

***Abstract*—Clock synchronization is essential for a large number of applications ranging from performance measurements in wired networks to data fusion in sensor networks. Existing techniques are either limited to undesirable accuracy or rely on specific hardware characteristics that may not be available in certain applications. In this paper, we examine the clock synchronization problem in networks where nodes lack the high-accuracy oscillators or programmable network interfaces some previous protocols depend on. This paper derives a general model for clock offset and skew and demonstrates its application to real clock oscillators. We design an efficient algorithm based on this model to achieve high synchronization accuracy. This algorithm applies the Kalman filter to track the clock offset and skew. We demonstrate the performance advantages of our schemes through extensive simulations and real clock oscillator measurements.**

***Index Terms*—Clock offset, clock skew, clock synchronization, Kalman filter, oscillator.**

I. Introduction

THE AVAILABILITY of an accurately synchronized clock enables and enhances a wide range of applications in distributed environments. For example, Internet measurements relying on either passively monitoring network events (e.g., packet loss) or actively probing network conditions (e.g., end-to-end delay and loss rate) implicitly require a common notion of time among all participating measurement points. Another example lies in wireless sensor networks (WSNs). Sensor network applications need a common notion of time for precise data integration and sensor reading fusion. Clock synchronization is also essential in network and communications protocols such as TDMA medium access scheduling, node sleep scheduling, and scheduling for directional antenna reception.

Many clock synchronization techniques for the Internet have been proposed over the past few decades, among which the most popular and widely used is Network Time Protocol (NTP). The development and evolution of NTP are described in Mills' classic papers [1], [2]. Several techniques [3]–[6] have been proposed to improve its synchronization accuracy when NTP is not able to satisfy the requirements of demanding applications. Additionally, in applications that do not require real-time synchronization, several other techniques are proposed to estimate and remove clock offset and skew offline in captured data sets such as packet delay traces [7], [8].

In the Internet, each node is either a router or a host that is wired to a constant power source and has one or more stable and powerful CPUs. In contrast, some other networks have only very limited resources such as scarce energy, unstable processors, and unreliable low-bandwidth communications. WSN is a representative example of this type of network. In a WSN, since the vast majority of sensors are battery-powered, a desirable clock synchronization scheme must preserve energy to prolong the battery life. Pottie *et al.* [9] shows that transmitting 1 bit over 100 m requires 3 J, which can be used for executing three million instructions. Therefore, a successful clock synchronization scheme must minimize the amount of message exchange and at the same time maintain high synchronization accuracy. Scarcity of power on sensor nodes, however, is not the only resource constraint. Due to its small size and low cost, the clock readings in a sensor are derived from oscillators with only limited stability (due to phase noise, thermal noise, aging, etc.). Consequently, clocks on sensors are easily affected by temperature variations, vibration, and interference and can significantly deviate from the reference sources [10], [11]. The situation could become even worse under catastrophic conditions such as earthquakes, battlefields, or forest fires. All these affect the oscillation period and make the clock drift nonlinear and time-varying.

In this paper, we design clock estimation and synchronization techniques that work on low-precision oscillators with time-varying drift rates.

First, we decompose the clock uncertainty into multiple independent components and use these components to construct general models for real clocks. The proposed models are also general enough to subsume the existing models. Additionally, we introduce techniques to choose the number of parameters used in the models.

Second, a Kalman filter is designed to track the clock uncertainty based on the aforementioned models. In fact, most of the prior protocols fail when the clock has some time-varying drift. We model the random drift of the clock using a Kalman filter that tracks the variation of the clock drift and thus enhances the synchronization performance.

Third, we measure the time offset from two low-cost oscillators and use the measured data to estimate a clock model. The model order is determined using information criteria. Comparisons between the model and measurements show that the derived clock models closely track the real clock drift.

Manuscript received October 21, 2010; revised April 19, 2011; accepted May 17, 2011; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Kasera. Date of publication June 27, 2011; date of current version February 15, 2012. This work was supported in part by the MKE/KEIT under the IT R&D Program [10035172, Development of fundamental technology on next-generation adaptive wireless mesh communication system for on-the-move nodes] and the NSF under Grant 0626979. Part of this work was presented at the IEEE Military Communications Conference (MILCOM), San Jose, CA, October 31–November 3, 2010.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: hkim369@gatech.edu; xiaoli@gatech.edu; bhamilton3@gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2011.2158656

Fourth, we further evaluate the clock model and the tracking method by considering missing or corrupt observations that can occur in unreliable links. Interestingly, we find that thanks to the accuracy of our tracking model, the Kalman-filter-based clock-tracking method is not only robust to missing data, but also integrates well with least absolute shrinkage and selection operator (LASSO) to detect the corrupt data.

Note that our clock synchronization scheme establishes local clock models and a Kalman-filter-based tracking algorithm, which does not require any particular message exchange mode. Therefore, our scheme can be easily adapted into both receiver-to-receiver [12]–[15] and sender-to-receiver modes [16]–[21].

The rest of the paper is organized as follows. We first summarize the related work in Section II. Section III describes the autoregressive models we develop for low-precision clocks. In Section IV, we present and analyze our algorithms to track clock skew and offset. In Section V, we demonstrate our tracking method with the real measurements. We examine the clock-tracking performance in systems with missing or corrupt observations in Section VI. We conclude the paper in Section VII.

II. Related Work

Clock synchronization mechanisms ensure that physically dispersed processors have a common knowledge of time. This topic is well studied in wired networks such as the global Internet. The most common and widely used mechanism is NTP [1], [2], which uses NTP packets containing timestamp information exchanged between the NTP server and the host across a network to perform time synchronization. NTP is designed to provide clock offset accuracy bounded by the round-trip time (RTT) between the server and the client.

However, NTP provides insufficient accuracy and robustness for many demanding applications. A few techniques have been proposed to improve measurement accuracy or clock stability. In [3], synchronization is offloaded onto a programmable network interface card. This card autonomously performs synchronization by sending periodic messages and performs time-stamping when packets arrive. In [4] and [5], a method of using a clock based on the more accurate CPU oscillator was proposed. This method relies on the high reliability of the processor oscillator and the availability of a timestamp counter (TSC) register. The Precision Time Protocol [6] was drafted into the IEEE 1588 standard for synchronization of network measurement and control systems. It uses a specially designed network infrastructure to achieve high synchronization accuracy. Unfortunately, all these techniques may not be effective for low-cost devices.

Since some passive network monitoring tasks do not require real-time synchronization, a few algorithms have emerged for synchronizing data captures. In [7], a linear-programming based algorithm for estimating and removing the skew and offset of a data set was proposed. Convex hulls were used in [8] to estimate the clock skew and offset within a data set. This was shown to perform better than the linear regression methods, but suffers increased computational complexity. All these algorithms are offline, which means they deal with the saved measurement data such as network packet delay traces instead of online clock synchronization.

Most existing methods synchronize a sender with a receiver by transmitting the current clock values as timestamps [16]–[21]. In this regard, these methods are vulnerable to variance in message delay between the sender and the receiver due to network delays and processing overhead. Some other methods [12]–[15] perform receiver-to-receiver synchronization. These methods exploit the property of the physical broadcast medium where any receivers one hop away receive the same message at approximately the same time. Such an approach reduces the message delay variance because nondeterministic delays at the transmitter no longer affect accuracy of the timestamp. Our proposed scheme is independent of the above synchronization modes. It can be easily adapted into both modes as long as we obtain reasonably good parameter estimates for the clock models.

Recent works have attempted to deal with the clock synchronization for WSNs from a signal processing perspective. A survey in [22] categorizes the existing protocols and clock estimation results. In [23], assuming the delay model is known in two-way message exchanges, the maximum likelihood estimator (MLE) and its corresponding Cramér–Rao lower bound (CRLB) of clock offset and clock skew are derived. The authors proposed an ML-like skew estimator, which is simple and more suitable for WSNs. Later, the ML-like skew estimator in [23] was generalized in [24], where another estimator that reduces the complexity while bringing comparable performance to ML is proposed. These approaches estimate time-invariant clock skew and clock offset based on different delay models.

Kalman filtering has been used in the context of clock synchronization [25]–[27] for packet-switched networks. In [25], a Kalman filter was used to model the fluctuation in packet inter-arrival times, after shaping this fluctuation with low-pass pre-filtering. Reference [26] presents a Kalman filtering algorithm for end-to-end time synchronization. This algorithm assumes a constant clock skew in the long term, which is not valid in most resource-constrained networks, and relies on NTP to exchange timestamp information. Reference [27] also assumes constant clock skew and relies on the TSC register, found in Pentium class CPUs to count CPU clock cycles. Kalman filtering was also employed in [28] to estimate clock skew and offset by assuming that clock skew has a first-order Gauss–Markov model, but lacks any model validation.

III. Clock Modeling

Network time synchronization at its simplest is not a difficult problem to understand. It is simply the problem of setting two or more clocks with the same notion of time and performing updates to ensure this continues to occur. This problem becomes complicated, however, when the characteristics of the network and clocks themselves are considered. Information sent over a network is subject to random, variable delays that add significant measurement noise to time measurements. Oscillators in clocks suffer from skew, drift, and jitter. All of these cause the clocks to progress somewhat erratically. In this section, we carefully study these characteristics. Let us first demonstrate some important terms used in this paper. Then, we show how to model the uncertainty of a clock.

A. Terminology

As we have mentioned, we examine clock drift from a physical (hardware) perspective and consider the causes of clock drift and how it can be modeled.

- *Oscillator:* An oscillator is an electronic circuit that produces a periodic electronic signal, often a sinusoidal waveform. Oscillators are important in many different types of electronic equipment. For example, a quartz watch uses a quartz oscillator to keep track of time. An AM radio transmitter uses an oscillator to create the carrier wave for the station, and an AM radio receiver uses a special form of oscillator called a resonator to tune to a station. There are oscillators in computers, sensors, metal detectors, and even stun guns.
- *Phase noise:* Phase noise is a common type of noise existing in oscillators. An ideal oscillator would generate a pure sinusoid waveform. However, because of time domain instabilities (e.g., "jitter"), the frequency generated cannot be restricted to a single sine. That means the phase noise components spread the power of the signal to adjacent frequencies, and the frequency domain representation of the signal shows some rapid, short-term, random fluctuations in the phase.
- *Clock:* A clock is a device that measures time. It generally consists of a periodic component (e.g., an oscillator) and a counting component (e.g., a hardware register). Their combination determines the resolution (i.e., the smallest measurable time unit) and accuracy.
- *Clock drift:* Clock drift refers to the phenomenon where a clock does not run at the correct speed compared to the actual time. The phase noise in oscillators is an important component of clock drift. Because phase noise is random, the clock drift is also random. Clocks often drift differently depending on their oscillator quality, the exact power they get from the battery, temperature, pressure, humidity, age, and so on. Thus, the same clock could have different clock drift rates on different occasions. Usually, the instantaneous clock drift rate is called *clock skew*, and the time difference with the actual time is called *clock offset*. Their formal definitions follow.

B. Clock Offset Modeling

A clock is merely the combination of an oscillator and a counter. The characteristics of the oscillator and the counter define the clock's behavior. The starting values of the counters control the initial relative offset between clocks. The frequency of the oscillator controls the rate that the clock advances. Since it is impossible to create oscillators that oscillate at exactly the same rate, every clock advances at a different rate in the real world. Considering all these factors, we define and model the clock offset.

1) Continuous-Time General Clock Model: The time reported by a clock at some ideal time t is written as $C(t)$. We will write $C_A(t)$ as the time given by clock A at time t. The difference between the time of an ideal clock and a given clock is said to be the offset $\theta(t)$, which is defined as

$$\theta(t) = C(t) - t.$$

The relative offset from node B to node A, $\theta_A^B(t)$, is defined as

$$\theta_A^B(t) = C_A(t) - C_B(t) = \theta_A(t) - \theta_B(t).$$

The oscillator in a clock produces periodic pulses. The difference between the rate these pulses are produced and the rate an ideal clock counts the desired interval is called the skew, denoted by $\alpha(t)$

$$\alpha(t) = \frac{d\theta(t)}{dt} \approx \frac{\theta(t+\tau) - \theta(t)}{\tau}. \tag{1}$$

The skew of a clock is the slope of the change in offset compared to the ideal clock. The slope of the relative offset $\theta_A^B(t)$ is relative skew $\alpha_A^B(t)$. This is defined as

$$\alpha_A^B(t) = \alpha_A(t) - \alpha_B(t).$$

If the oscillator were perfectly stable, the slope of $\theta(t)$ would reflect a constant skew α. However, this is not the case, especially in low-cost devices. Oscillators do not produce perfectly periodic pulses. The oscillator's nonlinearity and the phase noise alter the pulse period, making the clock rate time-varying [29]. Additionally, physical effects such as temperature and age can change the oscillator frequency. For the remainder of this paper, we assume the reference node has a perfect clock (i.e., zero offset and skew) so that all the offset and skew notations lack subscripts without loss of generality. It is straightforward to adapt all of derivations into the relative sense when the reference node deviates from the actual time.

Having a complete understanding of clock drift, we decompose its variations into three independent components: the instantaneous clock skew $\alpha(t)$, the initial clock offset θ_0, and the random measurement and other types of additive noise $w(t)$. The instantaneous clock offset $\theta(t)$ at time t is given as

$$\theta(t) = \int_0^t \alpha(\tau)\, d\tau + \theta_0 + w(t). \tag{2}$$

This model is quite general and subsumes all those existing simpler clock models. For example, if the clock skew $\alpha(t)$ does not change along with time t, the model in (2) reduces to the simple skew model in [4].

2) Discrete-Time Clock Model: After sampling, the continuous-time model becomes discrete-time model. In most cases, a discrete clock model is desirable since the synchronization is typically achieved by time-stamped message exchange. The timestamps are nothing but discrete samples of the continuous time. Based on (2), the discrete-time clock model is obtained as

$$\theta[n] = \sum_{k=1}^{n} \alpha[k]\tau[k] + \theta_0 + w[n] \tag{3}$$

where k is the sample index, "$[\cdot]$" is adopted for discrete indexing, and $\tau[k]$ is the sampling period at the kth sample.

Here, note that our discrete-time model is also quite general. It covers not only uniform sampling, but also nonuniform sampling (by choosing different $\tau[k]$). Since $w[n]$ is mainly caused by the observation and measurement noise, it is reasonable to assume $w[n]$'s are independently distributed with variance σ_w^2. The variance of $w[n]$ depends on the time-critical path [12]. In general, the time-critical path in a sender-to-receiver synchronization consists of four factors [30]: 1) the time for message construction and sender's system overhead; 2) the time to access the transmit channel; 3) propagation delay; and 4) the time spent by the receiver to process the message. In contrast, a receiver-to-receiver synchronization is only impacted by 3) and 4), and hence has smaller variance. In either case, the variance

value can be estimated using samples. We can rewrite this model using a recursive form as

$$\theta[n] = \theta[n-1] + \alpha[n]\tau[n] + v[n] \tag{4}$$

where $v[n] = w[n] - w[n-1]$. Clearly, $v[n]$ is a random variable with mean 0 and variance $\sigma_v^2 = 2\sigma_w^2$. This is not surprising since (4) is the differential form of the observation equation for the clock, and differential forms are well known to double the noise variance. Even if the observation noise $w[n]$ has nonzero mean, it is not difficult to verify that $v[n]$ still has zero mean because of the differential format in (4).

C. Clock Skew Modeling

When using the recursive model in (4) to synchronize clocks, we need to estimate the clock skew $\alpha[n]$, which is also time-varying. Before we establish the clock skew model, we look at two extreme cases of clock skew.

Case 1) (constant skew): Suppose the clock skew $\alpha[n]$ is constant as in [4] and [7]. From (4), since $\theta[k]$'s are known for $k = 1, \ldots, n$, if the sampling period $\tau[k]$'s are also known, the optimal clock skew estimator in terms of mean-square error (MSE) is

$$\hat{\alpha}[n] = \frac{\sum_{k=2}^{n} (\theta[k] - \theta[k-1])\tau[k]}{\sum_{k=2}^{n} \tau^2[k]}. \tag{5}$$

Case 2) (independent skew): If the clock skew $\alpha[n]$ changes completely from one sample to another, the optimal estimator becomes

$$\hat{\alpha}[n] = \frac{\theta[n] - \theta[n-1]}{\tau[n]}. \tag{6}$$

These two cases are simple, but neither of them is practical. Most existing schemes are based on these two simple cases without considering any statistical and time-series models of the clock skew. Because of the phase noise in the oscillator, clock skew has certain randomness, but is not completely independent for each sample. Fig. 1 is an example of the real clock skew behaviors in a resource-constrained network. It shows the clock skews of the two clocks used in a low-powered micro-controller platform, which are 32.768 kHz and 16 MHz. We examined the clock skews using the platform in a temperature-controlled room over 1.5 months. The variation of clock skew is observable over small timescales, and it is clear that the constant skew model fails over timescales of several hours even in air-controlled environments. It is expected that clock skew would vary severely in the real environment due to lack of energy or large temperature variations. Therefore, we investigate a model that can reflect these time-varying characteristics. In the following, we derive a model for the random clock skew starting from the phase noise.

Phase noise in oscillators has different representations. Here, we consider a simple way to model it through jitter. It is natural to think of it as a noisy random offset in the timing of events. If the unperturbed oscillator output is $s(t)$, the jitter perturbed output is $s(t + \phi(t)/2\pi f_o)$, where $\phi(t)$ is random and f_o is the center frequency of the oscillator. Clearly, the jitter $\phi(t)$ affects the frequency of the oscillator $\frac{d\phi(t)}{dt}\frac{1}{2\pi f_o}$ and thus causes clocks

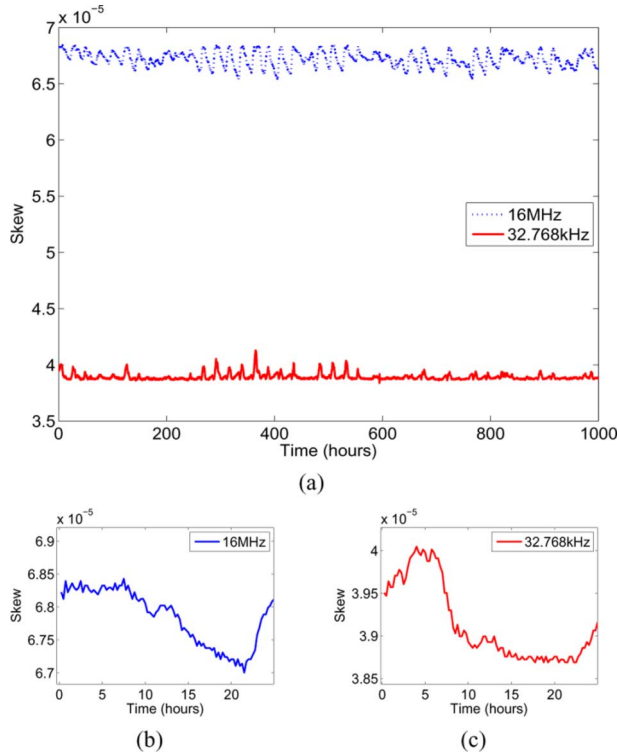


Fig. 1. Measurement of time-varying clock skews.

to have random offset and skew [11]. In general, phase noise is not stationary, but only cyclostationary.

To model the time-varying clock skew as a random process, we assume clock skew is a random process with zero mean and a small perturbation around the mean. This assumption has been observed by some previous works (e.g., [4]) that adopt constant skew. Here, we start modeling the time-varying skew as an auto-regressive (AR) process. We assume the smoothness (order of AR model) of the clock skew is P, which is more general than the previous works [4], [28]. Later, real measurements show that AR model order P can be reasonably small. Thus, the complexity of the model is limited. Given an AR model with order P, the time-varying clock skew process satisfies the relation as

$$\alpha[n] = \sum_{i=1}^{P} c_i \alpha[n-i] + \eta[n] \tag{7}$$

where c_i's are AR coefficients, and $\eta[n]$ is model noise with zero mean and σ_η^2 variance. Usually, we model $\eta[n]$ as Gaussian noise because, in general, the phase noise derivative $\Delta\phi(t)$ is unbounded, but the frequency drift is focused within a certain range (which is usually specified by the oscillator manufacturer).[1] The skew model in (7) is general and practical. It subsumes the two extreme cases in (5) and (6) as special cases. It also quantifies the drifting of the clock frequency, captures the main variation of clock skew, and also takes into account the randomness. To apply this model, the parameters c_i's and the model order P need to be determined. In this section, we consider two ways to estimate c_i's and leave the model order selection to Section III-D.

[1]Note that this assumption is not required for the derivation of the following Kalman filter.

One way to estimate these coefficients is based on the statistical properties of $\alpha[n]$. Define the auto-correlation function of $\alpha(t)$ as $r_\alpha(\tau) = E\{\alpha(t)\alpha(t+\tau)\}$, and then the $\mathrm{AR}(P)$ coefficients c_i's can be derived as

$$\begin{bmatrix} r_\alpha(P\tau) \\ r_\alpha((P-1)\tau) \\ \vdots \\ r_\alpha(\tau) \end{bmatrix} = \begin{bmatrix} r_\alpha(0) & \cdots & r_\alpha((1-P)\tau) \\ r_\alpha(\tau) & \cdots & r_\alpha((2-P)\tau) \\ \vdots & \vdots & \vdots \\ r_\alpha((P-1)\tau) & \cdots & r_\alpha(0) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{bmatrix}. \quad (8)$$

To estimate the parameters c_i's of $\mathrm{AR}(P)$ model, we need to estimate the auto-correlation function and variance of $\alpha[n]$. It is clear that as time goes on, the auto-correlation of the clock skew becomes weaker. In [31], the auto-correlation function is modeled as a decaying exponential as

$$r_\alpha(\tau) = \sigma_\alpha^2 \rho^{\frac{\tau}{\nu}} \quad (9)$$

where ρ denotes the normalized decay in time period ν. We also adopt this exponential decay model for the auto-correlation function. Given the clock observations $\theta[n]$ in (3), one can obtain samples of the clock skew $\alpha[n]$ using the independent skew model in (6). Thus, the auto-correlation $r_\alpha(\tau_0)$ and the variance can be estimated using sample means. Once we obtain the auto-correlation, we can find its parameters by setting $\nu = \tau_0$ and solving for ρ. Then, we can use this auto-correlation to estimate the c_i's for the desired sampling period τ. Theoretically, $\alpha(t)$ is nonstationary, and thus the coefficients c_i's may change along with time. However, the c_i's change quite slowly relative to the clock offset, and thus we can still take them as quasi-stationary.

Another way is to use some of the measurements as training data. Suppose that we collect T observations, where $T > P$ as the model fitting data, and then (7) can be rewritten as

$$\begin{bmatrix} \alpha[P+1] \\ \alpha[P+2] \\ \vdots \\ \alpha[T] \end{bmatrix} = \begin{bmatrix} \alpha[P] & \cdots & \alpha[1] \\ \alpha[P+1] & \cdots & \alpha[2] \\ \vdots & \vdots & \vdots \\ \alpha[T-1] & \cdots & \alpha[T-P] \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{bmatrix} + \begin{bmatrix} \eta[P+1] \\ \eta[P+2] \\ \vdots \\ \eta[T] \end{bmatrix}. \quad (10)$$

Let $\mathbf{z}$, $\mathbf{Z}$, $\mathbf{c}$, and $\boldsymbol{\eta}$ denote the four column vectors/matrix in (10) for simplicity. Then, performing QR-decomposition on $\mathbf{Z}$, we obtain an upper triangular matrix $\mathbf{R}$. We can rewrite (10) as $\mathbf{Rc} = \mathbf{d}$, where $\mathbf{d} = \mathbf{Q}^T\mathbf{z}$ ($\mathbf{Q}$ is an orthogonal matrix). $\mathbf{Rc} = \mathbf{d}$ is solved by backward substitution. In this way, given a model order P, we can find the AR coefficients in (10) using a least-squares approach.

D. Information Criteria for Model Order Selection

To model the time-varying clock skew as an AR process, the selection of the model order P in (7) is an important step. As the model order increases, the model more closely fits the measurements. However, if model order is too high, there is a danger of overfitting the data. Overfitting results in a model that is more complex than necessary, and while the model may perform better on the training data, it will perform poorly on new data. Following the principle of Occam's razor, we adopt information criteria to select the proper model order.

One widely adopted information criterion is Akaike Information Criterion (AIC) [32]. When using finite dimensional AR models, the AIC provides an asymptotically efficient solution under a quadratic loss function. AIC finds an appropriate order of a model to select the best fit to the data. It was originally derived from maximizing the likelihood function of the model while adding a modeling penalty function, which is a term to represent the complexity generated as the modeling order increases. The modeling penalty function in AIC is the number of estimated parameters in the model (see [32]–[35]). AIC is defined as

$$\begin{aligned} \mathrm{AIC}(P) = &-2(\text{maximized log-likelihood of the model}) \\ &+2(\text{the number of parameters in the model}). \end{aligned} \quad (11)$$

When the noise is Gaussian and the least-squares coefficient estimator in (10) is used, the AIC function can be simplified as (cf. [32])

$$\mathrm{AIC}(P) = T\log\left(2\pi\hat{\sigma}_P^2\right) + 2P \quad (12)$$

where $\hat{\sigma}_P^2$ is an estimate of the variance of the modeling error $\eta[n]$ in (7) as

$$\hat{\sigma}_P^2 = \frac{1}{T-P}\sum_{n=P+1}^{T}\left(\alpha[n] - \sum_{i=1}^{P}\hat{c}_i\alpha[n-i]\right)^2 \quad (13)$$

with $\hat{c}_i$ denoting the estimate of the coefficients from (10).

Another often used information criterion is called minimum description length (MDL). This criterion also consists of the maximized log-likelihood function and a penalty function that is $(\log T)P$ as

$$\mathrm{MDL}(P) = T\log\left(2\pi\hat{\sigma}_P^2\right) + (\log T)P. \quad (14)$$

By multiplying P by $\log T$ in the penalty function, the penalty term becomes larger as the number of observations increases. MDL tries to find the smaller number of parameters by having a heavier penalty compared to AIC when there are a large number of observations.

The third information criterion is AIC_c, which is modified based on AIC in (12) and gives a better modeling order when the number of samples is small. AIC may perform poorly when the number of parameters in the model under consideration is a substantial fraction of the sample size. AIC_c adds $\frac{2P(P+1)}{T-P-1}$ to AIC when a sample size is small as

$$\mathrm{AIC}_c(P) = T\log\left(2\pi\hat{\sigma}_P^2\right) + \frac{2T}{T-P-1}P. \quad (15)$$

The optimal order of AR processes can be decided when the results of these information criteria are minimized. Using this optimal order and the estimates of AR parameters, the time-varying clock skew model in (7) is generated. Note that the optimal model order does not mean the best fit with the measurements (i.e., data). It balances the model fitting and data fitting, i.e., the smoothness of the process and the randomness of the

measurements. Of the two methods to estimate the AR coefficients, the statistical method in (9) is more robust and provides better performance for long data sets, but the training-based method in (10) is easier to implement and feasible even for a small amount data. In this paper, we adopt the training-based method with our experimental measurements.

Given the recursive observation model in (4) and the AR model in (7), we are finally prepared to construct Kalman filter as a framework to track the variation of the clock skew and synchronize clocks. In Section IV, we will describe how the Kalman filter can be applied to time synchronization.

IV. Clock Skew and Offset Tracking

Ideally, clock behavior could be estimated accurately if an exact model is derived. In the real world, however, clocks are affected by environmental factors such as temperature variations, vibration, and humidity. Moreover, resource-constrained networks, such as WSNs, usually consist of inexpensive devices that have unstable oscillators, vulnerable to interference. The factors have nondeterministic effects on clock behavior, which prevent even an exact clock model from tracking clock behaviors exactly. In this section, we design Kalman filters to track the time-varying clock skew and offset based on the proposed clock skew and clock offset models.

A. Skew and Offset Estimation With a Kalman Filter

Suppose that the sampling rate is fixed, i.e., uniform sampling with $\tau[n] = \tau_0$. The sampling period τ_0 is known, and the coefficients c_i's of the $\mathrm{AR}(P)$ model in (7) are estimated. Let $\tilde{\theta}[n]$ denote the true clock offset (i.e., $\tilde{\theta}[n] = \sum_{k=1}^{n} \alpha[k]\tau[k] + \theta_0$). It is clear that $\tilde{\theta}[n] = \tilde{\theta}[n-1] + \alpha[n]\tau_0$. We can define an extended state equation as

$$\boldsymbol{x}[n] = \boldsymbol{A}\boldsymbol{x}[n-1] + \boldsymbol{u}[n] \tag{16}$$

where $\boldsymbol{x}[n] = [\,\tilde{\theta}[n] \quad \alpha[n] \quad \ldots \quad \alpha[n-P+1]\,]^T$

$$\boldsymbol{A} = \begin{bmatrix} 1 & \tau_0 & 0 & \cdots & 0 \\ 0 & \hat{c}_1 & \hat{c}_2 & \cdots & \hat{c}_P \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \quad \boldsymbol{u}[n] = \begin{bmatrix} 0 \\ \eta[n] \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where $\boldsymbol{A}$ is a $(P+1)\times(P+1)$ transition matrix and $\boldsymbol{u}[n]$ is a $(P+1)\times 1$ driving noise vector. The observation equation is defined as

$$\theta[n] = \tilde{\theta}[n] + v[n] = \boldsymbol{b}^T\boldsymbol{x}[n] + v[n] \tag{17}$$

where $\boldsymbol{b}^T = [\,1 \quad 0 \quad \ldots \quad 0\,]$, which is a $1\times(P+1)$ vector, and $v[n]$ is observation noise. In this case, the Kalman filter design is summarized as follows (cf. [36, Ch. 13]):

$$\text{Update}: \hat{\boldsymbol{x}}[n] = \boldsymbol{A}\hat{\boldsymbol{x}}[n-1] + \boldsymbol{G}[n](\theta[n] - \boldsymbol{b}^T\boldsymbol{A}\hat{\boldsymbol{x}}[n-1]) \tag{18}$$

$$\text{MSE}: \boldsymbol{\Sigma}[n] = \boldsymbol{A}\boldsymbol{M}[n-1]\boldsymbol{A}^T + \boldsymbol{C}_u \tag{19}$$

$$\boldsymbol{M}[n] = (\boldsymbol{I} - \boldsymbol{G}[n]\boldsymbol{b}^T)\boldsymbol{\Sigma}[n] \tag{20}$$

$$\text{Kalman Gain}: \boldsymbol{G}[n] = \boldsymbol{\Sigma}[n]\boldsymbol{b}\left(\sigma_v^2 + \boldsymbol{b}^T\boldsymbol{\Sigma}[n]\boldsymbol{b}\right)^{-1} \tag{21}$$

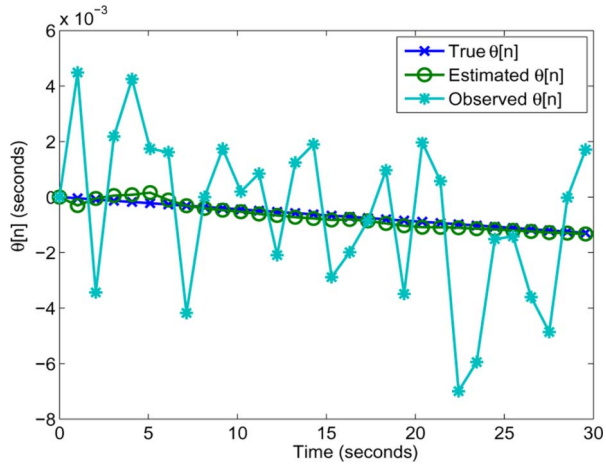


Fig. 2. Example: tracking clock offset.

where $\boldsymbol{\Sigma}[n]$ is the prediction MSE of the estimate when the current observation is not considered, $\boldsymbol{A}\hat{\boldsymbol{x}}[n-1]$. $\hat{\boldsymbol{x}}[n]$ is the estimate of the offset and skew state at the nth sample, $\boldsymbol{M}[n]$ is the minimum mean-square error (MMSE) of the estimate, and $\boldsymbol{G}[n]$ is the so-called Kalman gain. σ_v^2 is the observation noise variance, and $\boldsymbol{C}_u$ is the covariance matrix of $\boldsymbol{u}[n]$ when σ_η^2 is the variance of the driving noise in the state equation. The recursion of the Kalman filter is initialized by

$$\hat{\boldsymbol{x}}[0] = \begin{bmatrix} \mathrm{E}\{\theta[n]\} \\ \mathrm{E}\{\bar{\boldsymbol{\alpha}}[n]\} \end{bmatrix} \quad \boldsymbol{M}[0] = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\alpha^2\boldsymbol{I}_P \end{bmatrix}$$

where $E\{\cdot\}$ denotes the statistical expectation, and $\bar{\boldsymbol{\alpha}}[n]$ is a $P\times 1$ vector consisting of $\alpha[n]$. σ_α^2 is the variance of $\alpha[n]$, and $\boldsymbol{I}_P$ is a $P\times P$ identity matrix. Since the Kalman filter does not strongly depend on the initial conditions, the statistical mean and variance can be replaced by sample mean and sample variance. For example, $\hat{\alpha}[0]$ can be initialized as $(\theta[1]-\theta[0])/\tau_0$.

An example profile of this algorithm is shown in Fig. 2. The observation noise variance is $\sigma_v^2 = 10^{-5}\ s^2$, and the parameter ρ in (9) is chosen as $1 - 2\cdot 10^{-6}$ with $\nu = 1$ h. For simplicity, we assume an AR(1) model for clock skew in (7) and recursive observation model in (4). Fig. 2 shows the offset estimated by our algorithm from (18)–(21) ("Estimated $\theta[n]$"), the true offset $\tilde{\theta}[n]$ ("True $\theta[n]$"), and the offset from observation ("Observed $\theta[n]$"). Note that even though the observed offset is dominated by the observation noise, the Kalman filter is able to extract the true value with only small deviations.

Fig. 3 shows the effects of sampling frequency and measurement noise. As shown in Fig. 3(a), while the sampling period τ_0 increases, the root mean square error (RMSE) for the estimate of $\alpha[n]$ decreases. Initially, this claim may sound counterintuitive since one may think the faster the sampler is, the better the estimator performs. However, increasing τ_0 reduces the variance of the effective observation. If we take the observation equation in (4), we can solve for $\alpha[n]$ and write

$$\alpha[n] = \frac{\tilde{\theta}[n] - \tilde{\theta}[n-1] + v[n]}{\tau_0} = \frac{\tilde{\theta}[n] - \tilde{\theta}[n-1]}{\tau_0} + \tilde{v}[n] \tag{22}$$

where $\tilde{v}[n] = \frac{v[n]}{\tau_0}$ has variance $\frac{\sigma_v^2}{\tau_0^2}$. This $\tilde{v}[n]$ is the effective noise when used as an estimate of $\alpha[n]$. This means the reduced RMSE for larger sampling period is purely due to reduced effective observation noise. Note from Fig. 3(b) that this decrease in

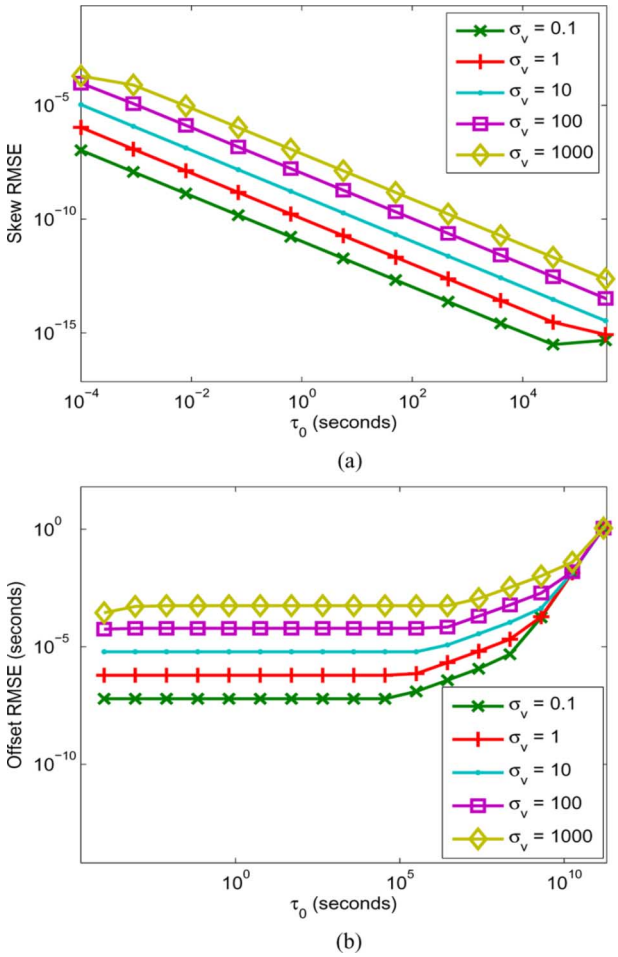


Fig. 3. Example: steady-state prediction RMSEs. (a) Steady-state error for α (b) Steady-state error for θ.

RMSE for $\alpha[n]$ does not appear to significantly affect the offset estimate. The observation equation for offset θ in this model has noise with variance σ_v^2, which does not depend on τ_0. From this, it is clear why the RMSE is relatively constant here.

V. Experiment and Model Validation

In this section, we verify the proposed clock models and the Kalman filtering method empirically with measurements of real low-cost clocks and show their performance.

A. Clock Skew and Offset Measurement

In order to ensure applicable results, we use a low-powered micro-controller platform similar to that deployed in sensor networks. Like the Mica2 mote [37], the device we tested uses an Atmel ATmega128L processor with a 32.768-kHz crystal oscillator. Our hardware platform differs in that it uses a 16-MHz primary crystal oscillator to drive the processor instead of the 8-MHz present on the Mica2 mote, and it lacks the RF interface.

The device was programmed to maintain two counters representing the time in 1/32768 of a second. One counter was driven by the 16-MHz external crystal oscillator, and the other counter was driven by the external 32.768-kHz crystal oscillator. The device was connected to a PC over an RS232 serial interface running at 38400 baud and was programmed to respond to timestamp queries. The timestamp queries consisted of the PC sending a single byte as a synchronization point followed by a 4-B timestamp. The device would then reply by sending a 4-B timestamp for each of the counters in response. Upon reception of the single byte, the device disables interrupts, copies the current time value from both counters to temporary registers, enables interrupts, and transmits the timestamps a byte at a time.

Timestamps were recorded on a 2.4-GHz Athlon 64 Dual-Core PC running Ubuntu 8.04. A program was run on the PC to send timestamp requests and record the replies in a binary file. In order to reduce latency on the PC, the program used the "mlockall" call, preventing the program from being swapped to disk, to avoid paging delays and enabled real-time scheduling using the SCHED_FIFO scheduler. Additionally, the serial port involved was set to low-latency mode using the setserial program. CPU frequency scaling and NTP were disabled on the PC to eliminate their effects on measurements.

Time was estimated on the PC using the "gettimeofday" call (which used the processor's TSC register for accurate timing) and converting the time to 1/32768 of a second. The PC also recorded the amount of time spent for each timestamp request, to measure the influence of variable delays on the measurement accuracy. For 99.8% of the measurements, processing the timestamp request took within 1/32768 s of the average processing time. Therefore, the variation of the noise from the duration of the measurement processes is bounded and negligible in most cases. The device was placed under direct sunlight in the room during the test to better simulate outdoor environment of sensors. Timestamps were collected using this setup every second for over 1.5 months.

We calculate the clock skew between samples separated by several minutes over the entire data set. This is equivalent to applying a moving average filter of the same width to the clock skew calculated from adjacent samples. This technique removes the effects of quantization noise, but larger filter widths will remove some dynamic components. The instantaneous offset $\theta[n]$ was obtained as $\theta[n] = t_{\mathrm{PC}}[n] - t_{\mathrm{device}}[n]$, where $t_{\mathrm{PC}}[n]$ is the instantaneous time of PC, and $t_{\mathrm{device}}[n]$ is that from counters of the device, every 900 s to reduce the effects of noise for our simulation. The skew $\alpha[n]$ was obtained as $\alpha[n] = \frac{\theta[n+1]-\theta[n]}{t_{\mathrm{PC}}[n+1]-t_{\mathrm{PC}}[n]}$ for the same interval as the offset. The measured results are shown in Fig. 1. Over the course of the measurement, the clock skew was around 40 ppm for the 32.768-kHz clock and around 70 ppm for the 16-MHz clock. Additionally, the skew of each clock varies as time, and the variation is as much as 3 ppm over this course. Moreover, the accumulated offset was about 150 s for the 32.768-kHz clock and about 250 s for the 16-MHz clock.

B. Model Order Selection and $\mathrm{AR}(P)$ Model

Since our clock-tracking method highly depends on the $\mathrm{AR}(P)$ model, we first use the real measurements to validate the $\mathrm{AR}(P)$ model and select a reasonable model order P. To find an appropriate order of an AR model for each clock skew, we employ the information criteria that were introduced in Section III-D. We train the model using a single day's measurements, i.e., $T = 96$. For each information criterion, we use the one-day data to calculate the cost of each fitting order of the AR model. As Fig. 4 shows, the resulting values

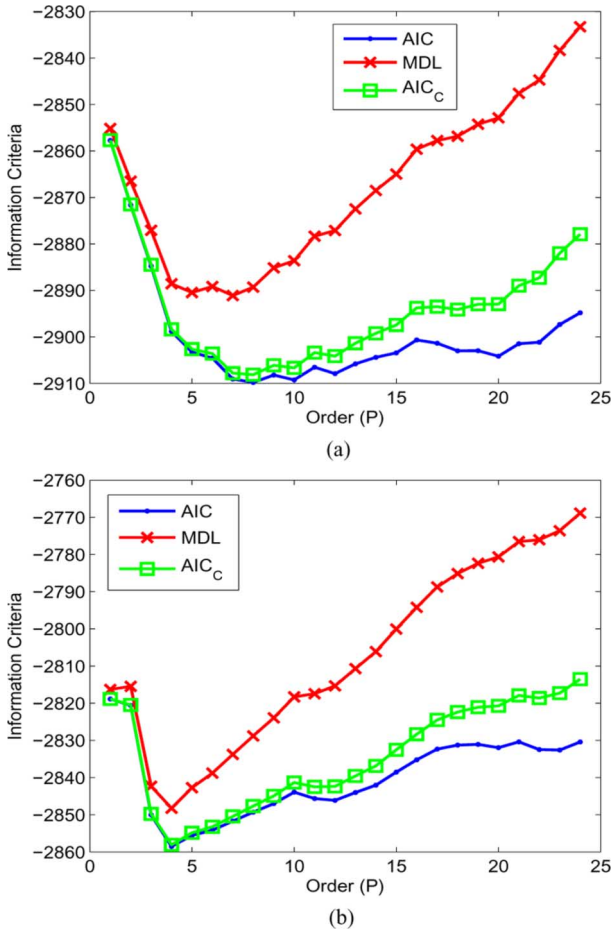


Fig. 4. Model order selection. (a) 32.768-kHz clock. (b) 16-MHz clock.

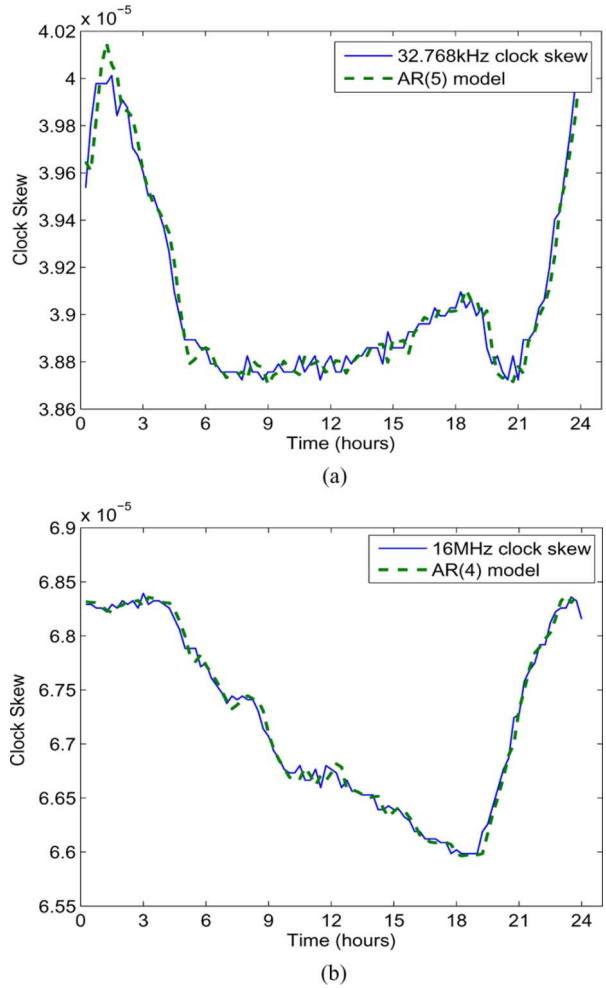


Fig. 5. Comparison of an AR model and measurement of clock skew. (a) 32.768-kHz clock skew. (b) 16-MHz clock skew.

of AIC, MDL, and AIC_c have similar patterns. The optimum orders of model based on AIC, MDL, and AIC_c are 8, 7, and 8, respectively, for the 32.768-kHz clock (see Fig. 4(a)) and 4 for the 16-MHz clock [see Fig. 4(b)]. Comparing different criteria, we observe that MDL criterion brings a sharper curve due to the heavier cost function compared to AIC and AIC_c. This makes the MDL criterion more consistent and reliable when the data set is really large and the measurement duration is long [34], [38]. However, based on our real measurements for clock skews, all criteria provide the same model order [see Fig. 4(b)]. Therefore, one can choose any of these information criteria to find the optimal model order.

Fig. 5 illustrates that the derived AR processes match each clock behavior with model order 5 or 4. The optimal order 4 from information criteria is employed as the model order for the 16-MHz clock, while order 5 is chosen for the 32.768-kHz clock. Note that we did not choose an optimum from information criteria as the model order of the 32.768-kHz clock. Since clock models in resource-limited networks need to be as simple as possible, we choose the order at which a downward tendency in the information criteria curves decreases. Our numerical simulation also shows that the clock-tracking performance is not highly sensitive to the model order. Therefore, in the following, we fix the model order to 5 or less to keep the complexity low. In Section V-C, we will show that with an AR(5) model, the proposed method tracks real clock behaviors well.

C. Clock Skew and Offset Tracking

Here, we estimate the real clock skew and offset based on the Kalman filter approach developed in Section IV and evaluate our method in various cases.

1) Tracking Performance: Suppose that the sampling period τ_0 is fixed as 900 s, which is a known value. The AR model derived in Section V-B is used as a clock skew model. When $\tilde{\theta}[n]$ is the true clock offset and $\alpha[n]$ is the true clock skew from the measurements (i.e., $\alpha[n] = \frac{\tilde{\theta}[n]-\tilde{\theta}[n-1]}{\tau_0}$), the state equation is defined as (16), where $\boldsymbol{x}[n] = \left[\tilde{\theta}[n]\ \alpha[n]\ \alpha[n-1]\ \alpha[n-2]\ \alpha[n-3]\ \alpha[n-4]\right]^T$

$$\boldsymbol{A} = \begin{bmatrix} 1 & \tau_0 & 0 & 0 & 0 & 0 \\ 0 & \hat{c}_1 & \hat{c}_2 & \hat{c}_3 & \hat{c}_4 & \hat{c}_5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \boldsymbol{u}[n] = \begin{bmatrix} 0 \\ \eta[n] \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where $\hat{c}_i$'s are the estimates of AR coefficients. The observation equation is defined as (17), where $\boldsymbol{b}^T = [1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$, $\theta[n]$ is the observation of the clock offset, and $v[n]$ is an observation noise that represents any kinds of uncertainties including

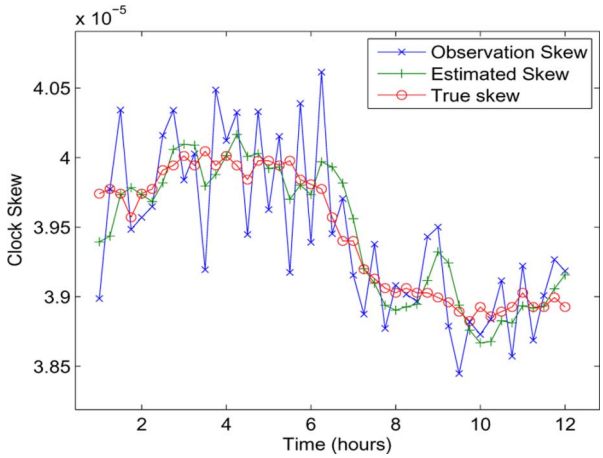


Fig. 6. Tracking clock skew using Kalman filter.

network delay and measurement errors that can be added to the true data when the clock offset is observed.

With these conditions, we design a Kalman filter as (18)–(21), where $\hat{\boldsymbol{x}}[n]$ is the estimate of $\boldsymbol{x}[n]$, and $\boldsymbol{\Sigma}[n]$ is the prediction MSE of the estimate when the current observation is not considered, $\boldsymbol{A}\hat{\boldsymbol{x}}[n-1]$. $\boldsymbol{C}_u$ is the covariance matrix of $\boldsymbol{u}[n]$, and $\eta[n]$ is the modeling error whose variance is $3.91502 \cdot 10^{-15}$. The variance is calculated by (13) using the measurements. The recursion of the Kalman filter is initialized by

$$\hat{\boldsymbol{x}}[0] = \begin{bmatrix} \tilde{\theta}[0] \\ \bar{\boldsymbol{\alpha}}[0] \end{bmatrix} \quad \boldsymbol{M}[0] = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\alpha^2 \boldsymbol{I}_5 \end{bmatrix}$$

where $\bar{\boldsymbol{\alpha}}[0]^T = [\,\alpha[0] \quad \alpha[0] \quad \alpha[0] \quad \alpha[0] \quad \alpha[0]\,]$, σ_v^2 is the variance of the measurement noise of clock offset, and σ_α^2 is the variance of clock skew, $\alpha[n]$, which is $1.29446 \cdot 10^{-13}$. The estimates of AR coefficients, which are components of $\boldsymbol{A}$, are obtained from (10).

Fig. 6 shows the performance of the Kalman filter estimating clock skew from noisy offset measurements. The signal-to-noise ratio (SNR) of clock offset was set to -20 dB by setting the standard deviation of observation noise of clock offset, σ_v, to $3 \cdot 10^{-4}$ s. Even with this low SNR, the Kalman filter was able to estimate the true skew quite closely. This means the derived skew model is able to accurately track the real clock measurement.

To validate our Kalman filtering method, more simulations are conducted and RMSE of the skew and offset is used as a performance metric for evaluation. This RMSE is calculated as the square root of the prediction MSE. First, we analyze the effects of the varying sampling rate on clock estimation performance. Since clock synchronization consumes resources such as power, bandwidth, and processing time, many applications will need to trade clock estimation accuracy against resource consumption. Fig. 7 depicts the performance of estimating clock skew and clock offset as synchronization period varies. The simulation was continued for 800 synchronization periods with several different sampling periods, and these curves represent the prediction RMSEs of skew and offset every four samples. The RMSE of the skew in Fig. 7(a) becomes smaller when sampling rate is lower. This is because that longer sampling period cannot capture the small and fast variations of the skew, and thus the modeling error is smaller. This can also be explained by the calculation of skew from the offset observations as $\alpha[n] = \frac{\theta[n]-\theta[n-1]}{\tau_0}$. When τ_0 is large, the noise variance in observed clock skew is reduced to $\frac{\sigma_v^2}{\tau_0^2}$. For the offset in Fig. 7(b), the RMSEs change little, around $10^{-3.6}$ s, even as sampling period varies. This means tracking offset performance is unaffected by sampling period. Fig. 7 also shows that both the RMSEs of clock skew and offset converge to steady state after several samples. As the sampling period τ_0 increases, the prediction RMSE approaches its steady-state value in fewer samples. However, since greater τ_0 implies a larger sampling period, it actually has a longer convergence time. The performance shown is only at sampling instants, where new data has just arrived, rather than at arbitrary time instants. This performance will therefore be better than the true time-averaged performance.

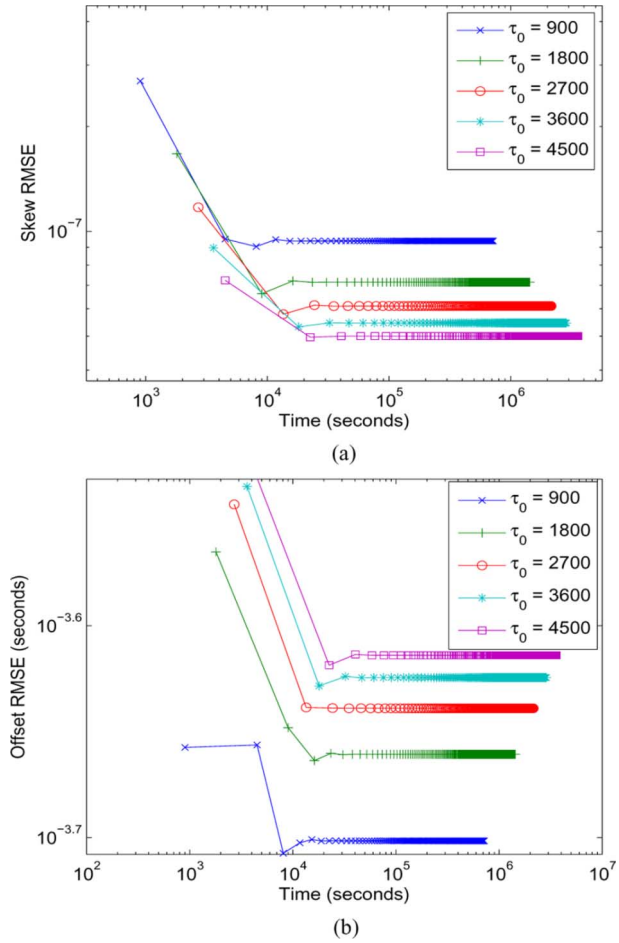


Fig. 7. RMSEs for various sampling rates. (a) Skew RMSE. (b) Offset RMSE.

Fig. 8(a) and (b) further clarifies the relation between sampling rate and clock-tracking error by showing how the steady-state RMSE[2] varies with the sampling period (τ_0). The simulations were run 200 times and averaged for each point. The skew RMSE decreases as sampling period increases, while the offset RMSE is nearly same. Both of them are more strongly

[2]Note that Figs. 7 and 8 show that the required granularity is around 10^{-6} s for the offset measurement and $5 \cdot 10^{-9}$ (i.e., 0.005 ppm) for the skew when the observation noise variance is really low $\sigma_v^2 = 9 \cdot 10^{-12} s^2$. This granularity may be difficult (particularly considering our measurement resolution of 1/32768 s $\approx 3 \cdot 10^{-5}$) to achieve. Recall in our measurement, we employed a moving average filter to average at least 900 measurements for the skew and offset estimates, which enhances the resolution of the observation to about 10^{-8} s for the offset and 10^{-12} for the skew. These resolutions match with our simulation results.

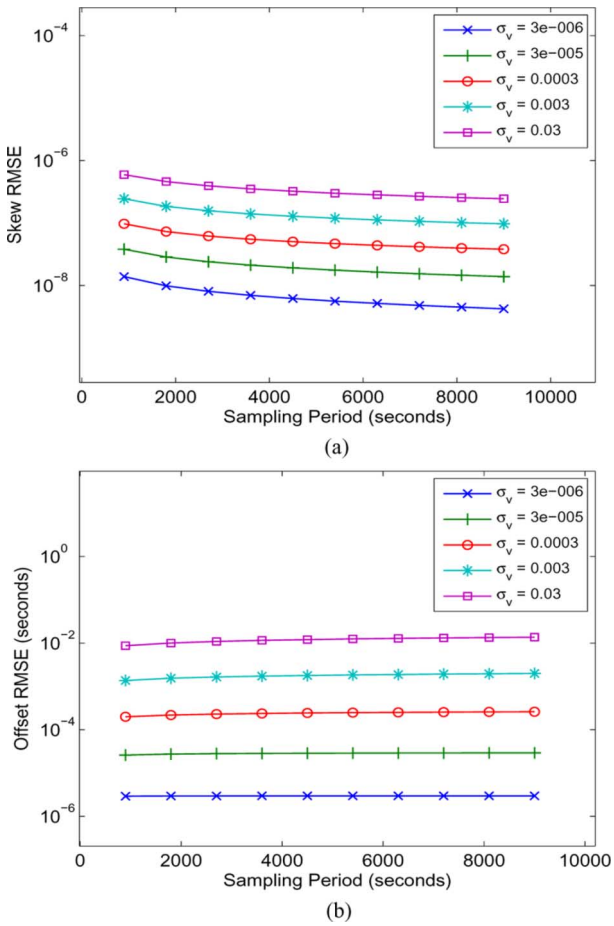


Fig. 8. Steady-state RMSE versus sampling period. (a) Skew RMSE versus sampling period. (b) Offset RMSE versus sampling period.

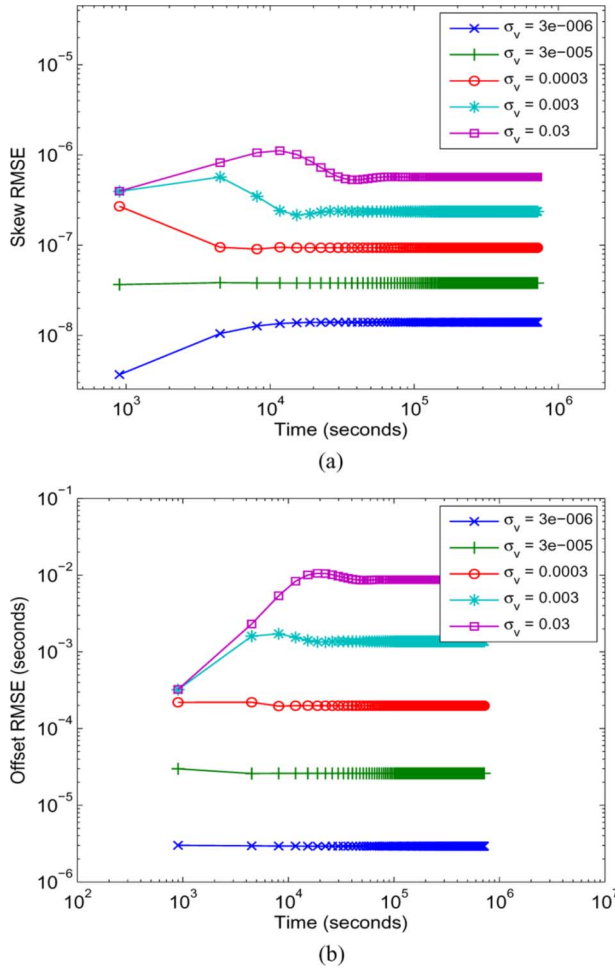


Fig. 9. RMSEs for various observation noise variances. (a) Skew RMSE. (b) Offset RMSE.

influenced by observation noise variance than sampling period. This result is compatible with the previous section (see Fig. 3), which shows an example of using the AR(1) model of the clock skew. We have shown that our modeling and tracking methods are quite robust to different sampling periods.

We also show that our proposed clock-tracking algorithm is robust to observation noise and converges after several samples even when the observation noise is large. This is illustrated in Fig. 9(a) and (b). To obtain the RMSE curves, the tracking algorithm was run for 800 synchronization periods with several different noise variances. RMSE of the estimated clock skew, $\hat{\alpha}[n]$, is less than 10^{-6} even when σ_v is as high as 0.03 s, and the RMSE of the estimated clock offset, $\hat{\theta}[n]$, is less than 10^{-2} s at the same condition. Even if the variance of observation noise is extremely high, the prediction RMSE of our Kalman filter curves converge in several samples.

2) Comparison: In this section, to evaluate our clock models, we borrow two other skew models and compare them. The simulation was run 200 times for 4000 synchronization periods with $\tau_0 = 900$ s and $\sigma_v = 3 \cdot 10^{-4}$ s. RMSE calculated by (23) and (24) is used as a performance metric for evaluation

$$\mathrm{RMSE}_{\mathrm{offset}}[n] = \sqrt{\frac{\sum_{i=1}^{n} (\hat{\theta}[i] - \tilde{\theta}[i])^2}{n}} \tag{23}$$

TABLE I
PERFORMANCE COMPARISON

	Skew RMSE	Offset RMSE (seconds)
AR(5) Skew Model	$1.1133 * 10^{-7}$	$2.1307 * 10^{-4}$
AR(1) Skew Model	$1.3282 * 10^{-7}$	$2.5998 * 10^{-4}$
Constant Skew Model	$2.8806 * 10^{-7}$	$6.6122 * 10^{-2}$

$$\mathrm{RMSE}_{\mathrm{skew}}[n] = \sqrt{\frac{\sum_{i=1}^{n} (\hat{\alpha}[i] - \tilde{\alpha}[i])^2}{n}}. \tag{24}$$

First, an AR(1) skew model was generated and used as a skew model instead of the AR(5) model, while (4) was used as the offset model as before. The skew and offset estimates found using both the AR(1) and AR(5) models are used to calculate their respective skew and offset RMSE. As shown in Table I, both the skew RMSE and the offset RMSE are higher than those with the AR(5) skew model. This shows that using a clock skew model with the model order chosen using information criteria enhances tracking performance.

Another skew model for performance comparison is the constant skew model in (5). When the clock skew is constant, the clock offset represents a linear model. RMSEs in Table I are calculated with a constant skew and a linearly increasing offset. Both RMSEs are greater than those of the AR(1) and AR(5) models. The offset RMSE is especially affected and is 10^2 times

worse than that of the AR(1) and AR(5) models. This shows the importance of using a time-varying model for higher-precision clock skew estimates. Observation noise was not added for simulations of a constant skew model, while σ_v is $3 \cdot 10^{-4}$ s for simulations of AR(1) and AR(5). Therefore, a constant skew model will perform even worse if observation noise is considered. For this reason, the constant skew model is not suitable for clocks with varying skew.

VI. Tracking With Corrupted Data

In sensor networks, the links are unreliable and prone to interference. Packets that contain timestamps may get lost or suffer from collisions. To handle message loss, traditional wired networks just send extra messages. However, this is not desirable for WSNs mainly because transmission of each bit consumes more energy than computational cost and the retransmission cannot guarantee reliability either [30]. Additionally, retransmitted timestamps suffer from increased and variable delay from retransmission. In this case, the received timestamps may not be uniform as one expected and/or impaired by network uncertainty, called corrupted data. In this section, we investigate tracking methods with missing or corrupted data.

A. Tracking With Missing Data

We test the tracking performance with three types of missing data: 1) uniform loss; 2) consecutive loss; and 3) random loss. Again, the Kalman filter in Section IV is adopted with the model from Section V-C-I.

When an observation is missing, we cannot estimate clock skew and clock offset using the introduced Kalman filtering method. However, the previous estimates can be used instead of the observation to find new estimates of clock skew and offset. Fig. 10(a) shows the tracking performance of the clock skew when one out of every five observations is dropped. Vertical dotted lines indicate the points of missing data. Clock skew and clock offset still track closely true clock skew based on the AR model derived in Section V-B even when the clock information is absent.

Next, let us suppose packets do not arrive at the destination node for a while due to mechanical problems or channel congestion. Fig. 10(b) shows an example of tracking clock skew when six consecutive measurements are missing. Even if several consecutive data packets are missing, the proposed method follows the true skew closely. Since there are no observed measurement, clock skew and offset are tracked only with estimates based on the model during the missing period. The estimated skew is close to the true skew even for consecutive missing observations. However, the gap between the estimate of clock skew and the true skew becomes larger as more samples are missed. This means our approach tracks the unreliable clock even when a system misses clock information for a while. However, if packets that include clock information are missed continuously for a long period, our method cannot track clocks. Because the AR model is a statistical forecasting model in which future values are computed only on the basis of past values of a time series data, our method cannot make reasonable estimates without valid previous data. Therefore, the tracking performance may decrease significantly when packet losses sequentially occur for a long period.

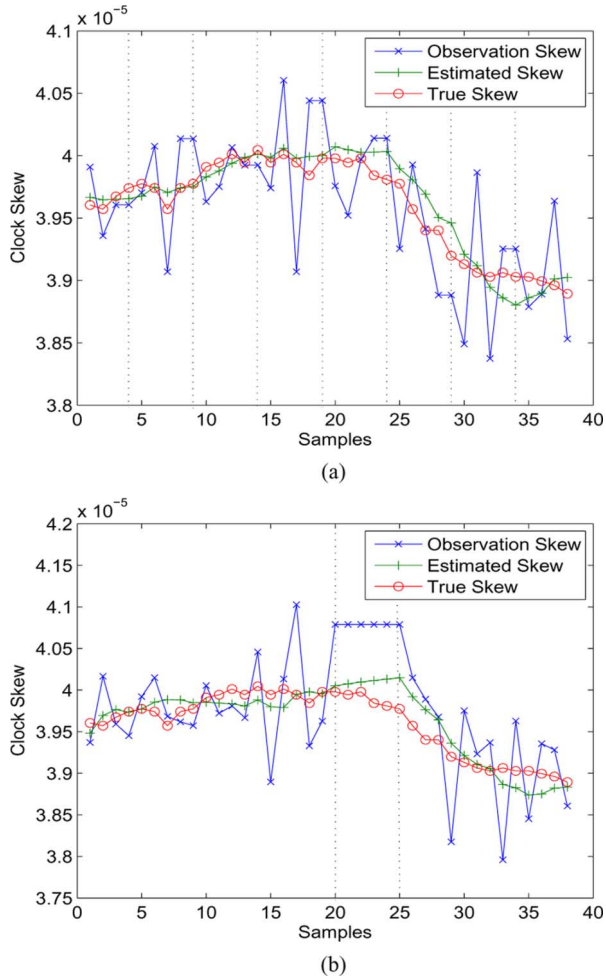


Fig. 10. Tracking clock skew with missing data. (a) Uniform missing data. (b) Consecutive missing data.

For randomly missing data, we compare the RMSEs of each sample, calculated by (23) and (24). Here, we assume that every packet has the same probability to be lost (i.e., the network packet loss rate) and every event is independent. The simulation was continued for 100 synchronization periods and averaged over 200 runs when the sampling period was 900 s and the standard deviation of observation noise of clock offset was set to $3 \cdot 10^{-4}$ s. We compare the tracking performance of two methods. The first method used here is a "with tracking" method that is applied in the previous two missing cases. This method estimates clock skew and offset using the clock models and tracks them if there is no available observation. The other method is a "without tracking" method. Since this method does not process anything when observations are missing, the estimates are not updated, so the previous estimates are used instead. By comparing the two methods, we show that our "with tracking" method is powerful in unstable networks. In Fig. 11, we compare the skew and offset RMSE of the two methods with different packet loss rates (10%, 20%, and 50%). Both methods have performance degradation compared to RMSE curves without missing observations. However, the "with tracking" method is robust to missing data compared to the "without tracking" method. The "without tracking" method has a significant performance degradation as depicted in Fig. 11. When the loss rate is 20%, both RMSEs of the "without tracking"

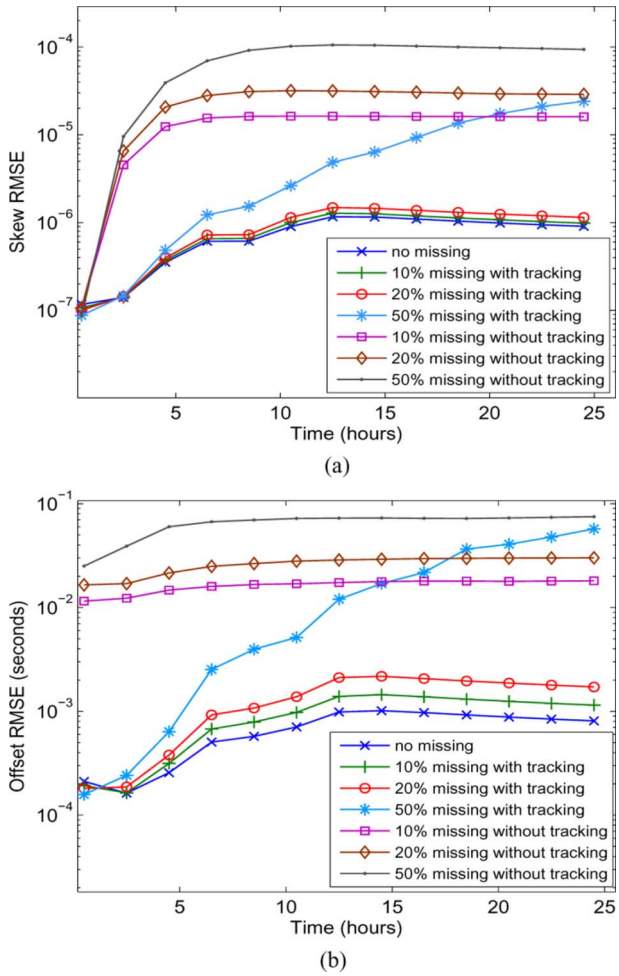


Fig. 11. RMSEs with random missing data. (a) Skew RMSE with random missing data. (b) Offset RMSE with random missing data.

method are 10 times higher than the "with tracking" method. Even using the "with tracking" method, the Kalman filter may not converge when 50% of the observations are missing, and thus the performance is poor. We observe the following: 1) as packet loss rate increases, the tracking performance gets worse, but our tracking method is fairly robust to missing data; and 2) performance "with tracking" always outperforms "without tracking."

B. Tracking With Dirty Data

In wireless networks, received data can be corrupted by other data packets, channel noise, or jamming. To differentiate the case considered here from the missing data in the previous section, the corrupted data are called dirty data. In this case, the proposed Kalman filtering method alone cannot perform effectively since it cannot differentiate dirty data from normal data. Thus, we adopt some dirty data detection methods (e.g., LASSO method) and combine them with Kalman filtering. Simulations show the effectiveness of our method.

The model of the clock offset observation with possible dirty component is redefined as

$$\theta[n] = \tilde{\theta}[n] + v[n] + \xi[n] \tag{25}$$

where $v[n]$ is Gaussian observation noise that usually has small variance, and $\xi[n]$ is a dirty component that is 0 for the majority of samples, but can randomly assume very large values.

1) Threshold Based Method: Dirty data can be detected by a threshold-based method, and the removal of dirty data processes as follows.

1) Compute measurement noise; $\hat{v}[n] = \theta[n] - \hat{\theta}[n]$.
2) Compute $\frac{|\hat{v}[n]|}{\sigma_v}$ and compare to a predefined threshold; we determine it as dirty data when the value is greater than the threshold.
3) Set $\theta[n] = \hat{\theta}[n]$ when the data is determined as dirty.
4) Repeat every synchronization period.

This method is intuitive, yet assumes that σ_v is known. The performance depends on the selection of the threshold.

2) LASSO: Dirty data rarely happen. To identify dirty data, we can apply the LASSO method by adding a regularization term to a standard least-squares problem [39]. Note that to fit in the tracking algorithm proposed in the previous section and also keep the complexity low, we propose scalar LASSO algorithm for each observation instead of performing LASSO for vector observations. Assuming that $\psi[n]$ represents measurement noise and dirty part, i.e., $\psi[n] = v[n] + \xi[n]$, we formulate the problem as a constrained ℓ_0 norm minimization problem [40]

$$\arg\min_{\psi[n]} \left(\|\hat{\theta}[n] + \psi[n] - \theta[n]\|_2^2 + \lambda \|\psi[n]\|_0 \right) \tag{26}$$

where $\|\cdot\|_0$ is the ℓ_0 norm, $\|\cdot\|_2$ is the ℓ_2 norm, and $\lambda \in [0, \infty)$ controls the sparsity. However, since ℓ_0 is not convex in general and is difficult to minimize, a common approach is to approximate the ℓ_0 norm as an ℓ_1 norm as

$$\hat{\psi}[n] = \arg\min_{\psi[n]} \left(\|\hat{\theta}[n] + \psi[n] - \theta[n]\|_2^2 + \lambda \|\psi[n]\|_1 \right) \tag{27}$$

where $\|\cdot\|_1$ is the ℓ_1 norm.

Since we use scalar observation, the solution of (27) is expressed in a soft-thresholded version of the least-squares estimate as explained in [41] and [42]

$$\hat{\psi}[n] = \text{sign}(\theta[n] - \hat{\theta}[n]) \left[\left| \theta[n] - \hat{\theta}[n] \right| - \frac{\lambda}{2} \right]_+ \tag{28}$$

where $\text{sign}(\cdot)$ denotes the sign operator, and $[\chi]_+ := \chi$, if $\chi > 0$, and zero otherwise. The regularization weight factor λ controls the sparsity of the results. If $\lambda = 0$, the result of (27) is least-squares solution, i.e., no sparsity is considered. If λ goes to infinity, more entries of $\psi[n]$ become zero. The optimal value of λ depends on the sparsity of the dirty data. Due to limited space, in this paper, we will not compare different methods on how to choose optimal λ, but choose it based on some numerical tests. If the estimate $\hat{\psi}[n]$ by LASSO is not zero, this observation is assumed to have nonzero dirty component and determined as dirty data. Once a data is claimed as dirty, it is removed and treated as missing data.

We adopt LASSO since it does not require any knowledge of the noise variance to detect dirty data. The above process is applied to every observation. Thus, it is possible to determine if an incoming sample is corrupted or not for real-time synchronization. Combined with LASSO, the tracking method avoids estimating clock behavior with large errors and becomes more robust to network uncertainties.

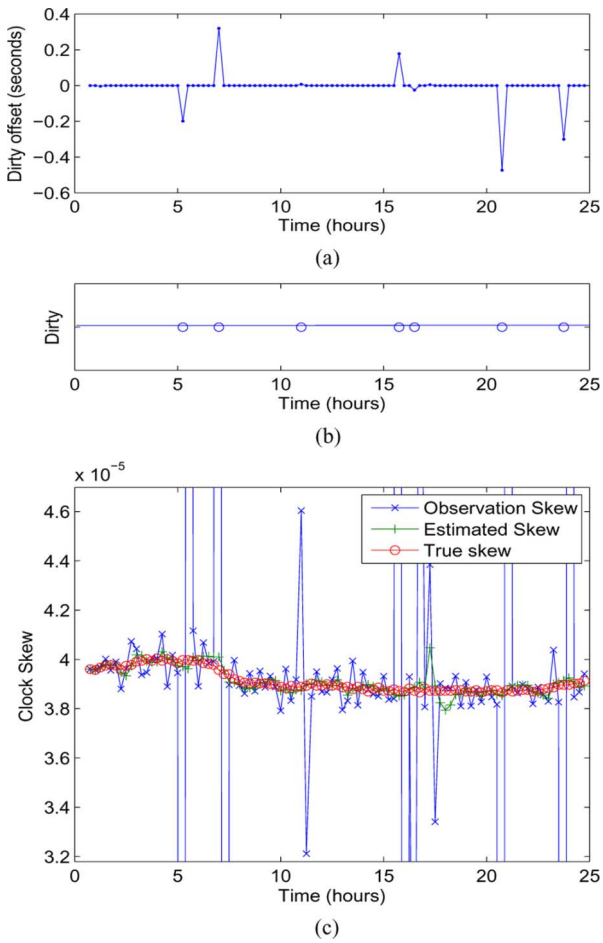


Fig. 12. Tracking with dirty data. (a) Dirty component (ξ). (b) Estimated dirty data position. (c) Tracking of clock skew.

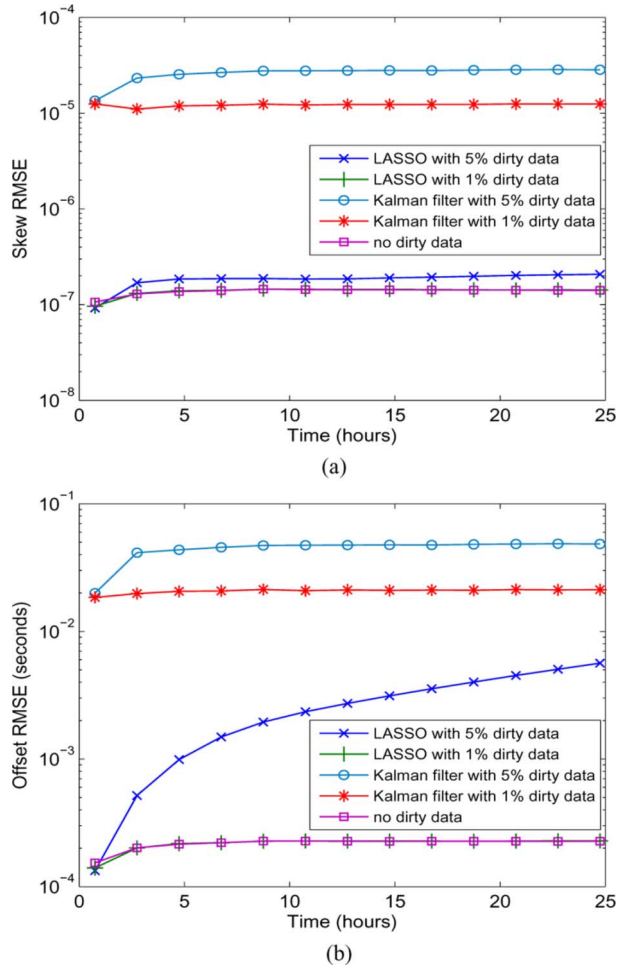


Fig. 13. RMSEs with dirty data. (a) Skew RMSE with dirty data. (b) Offset RMSE with dirty data.

In the simulation, we assume the dirty components are injected at a random time instance. The amplitude of the dirty component is Gaussian distributed with zero mean and variance 1. The simulation was continued for 100 synchronization periods and averaged over 500 runs when a synchronization period is 900 s and σ_v is $3 \cdot 10^{-4}$ s. In the case that dirty data are generated as in Fig. 12(a), our method using LASSO detects the position of dirty data exactly shown in Fig. 12(b). We set λ as 0.01 for simulation. Our method discards the possible dirty data and tracks the clock skew and offset as replacing them by estimates based on the time-varying clock models as explained in Section VI-A. Fig. 12(c) represents that the estimated skew tracks the true clock skew well in spite of large values of the dirty components.

Fig. 13 presents the tracking performance when every data has 5% or 1% probability to be dirty data. The above two curves are the RMSEs when dirty observations are not detected and the next two curves are those when dirty observations are detected and removed using LASSO, and the last curve represents the RMSEs when dirty observation does not exist as a reference. Without removal of dirty data, the performance becomes much worse since the Kalman filter uses dirty data, which are incorrect clock information as observation. However, our approach prevents this decrease in clock-tracking performance by applying LASSO to detect dirty data. This example shows that our tracking method using LASSO is robust to dirty data.

VII. Conclusion

Efficient and accurate network time synchronization is a challenging problem in resource-constrained networks. Considering the inherent instability of the inexpensive oscillators, we derive a general model to capture the time-varying behavior of clock offset and skew. Then, applying this model, we develop a clock skew and offset tracking method based on the Kalman filter. This tracking method is analyzed in detail and evaluated with measurement of real low-cost clock behaviors. Lastly, we demonstrate that our tracking method is robust to unreliable observations by combining with LASSO algorithm.

References

[1] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.

[2] D. L. Mills, "Improved algorithms for synchronizing computer network clocks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 3, pp. 245–254, Jun. 1995.

[3] C. Liao, M. Martonosi, and D. W. Clark, "Experience with an adaptive globally-synchronizing clock algorithm," in *Proc. ACM Symp. Parallelism Algor. Archit.*, 1999, pp. 106–114.

[4] D. Veitch, S. Babu, and A. Pàsztor, "Robust synchronization of software clocks across the internet," in *Proc. ACM SIGCOMM IMC*, Oct. 2004, pp. 219–232.

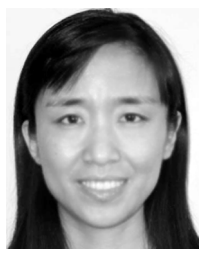
[5] A. Pàsztor and D. Veitch, "PC based precision timing without GPS," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 30, pp. 1–10, Jun. 2002.

[6] *IEEE Std. 1588-2002 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std 1588-2002, 2002, pp. i–144.

[7] S. B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *Proc. IEEE INFOCOM*, Mar. 1999, vol. 1, pp. 227–234.
[8] L. Zhang, Z. Liu, and C. H. Xia, "Clock synchronization algorithms for network measurements," in *Proc. IEEE INFOCOM*, 2002, vol. 1, pp. 160–169.
[9] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, May 2000.
[10] J. R. Vig, "Introduction to quartz frequency standards," Army Research Laboratory, Adelphi, MD, Tech. Rep. SLCET-TR-92-1 (Rev. 1), Oct. 1992.
[11] J. Phillips and K. Kundert, "Noise in mixers, oscillators, samplers, and logic: An introduction to cyclostationary noise," in *Proc. IEEE CICC*, May 2000, pp. 431–438.
[12] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.
[13] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Continuous clock synchronization in wireless real-time applications," in *Proc. IEEE SRDS*, Oct. 2000, pp. 125–133.
[14] S. PalChaudhuri, A. Saha, and D. Johnson, "Adaptive clock synchronization in sensor networks," in *Proc. IEEE IPSN*, Apr. 2004, pp. 340–348.
[15] W. Su and I. Akyildiz, "Time-diffusion synchronization protocols for sensor networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 384–397, Apr. 2005.
[16] K. Romer, "Time synchronization in ad hoc networks," in *Proc. ACM MobiHoc*, Oct. 2001, pp. 173–182.
[17] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol in sensor networks," in *Proc. ACM SenSys*, Nov. 2003, pp. 138–149.
[18] M. Maròti, G. S. B. Kusy, and A. Lèdeczi, "The flooding time synchronization protocol," in *Proc. ACM SenSys*, Nov. 2004, pp. 39–49.
[19] M. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Proc. IEEE WCNC*, 2003, vol. 2, pp. 1266–1273.
[20] Q. Li and D. Rus, "Global clock synchronization in sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2004, vol. 1, pp. 564–574.
[21] D. Zhou and T. Lai, "A scalable and adaptive clock synchronization protocol for ieee 802.11-based multihop ad hoc networks," in *Proc. IEEE Int. Conf. Mobile Adhoc Sensor Syst.*, Nov. 2005, pp. 1–8.
[22] Y. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 124–138, Jan. 2011.
[23] K. Noh, Q. M. Chaudhari, E. Serpedic, and B. W. Suter, "Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 766–777, Apr. 2007.
[24] M. Leng and Y. Wu, "On clock synchronization algorithms for wireless sensor networks under unknown delay," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 182–190, Jan. 2010.
[25] K. Kim and B. Lee, "KALP: A Kalman filter-based adaptive clock method with low-pass prefiltering for packet networks use," *IEEE Trans. Commun.*, vol. 48, no. 7, pp. 1217–1225, Jul. 2000.
[26] A. Bletsas, "Evaluation of Kalman filtering for network time keeping," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 52, no. 9, pp. 1452–1460, Sep. 2005.
[27] L. Auler and R. d'Amore, "Adaptive Kalman filter for time synchronization over packet-switched networks (an heuristic approach)," in *Proc. IEEE COMSWARE*, Jan. 2007, pp. 1–7.
[28] B. R. Hamilton, X. Ma, Q. Zhao, and J. Xu, "ACES: Adaptive clock estimation and synchronization using Kalman filtering," in *Proc. ACM MobiCom*, 2008, pp. 152–162.
[29] "Clock oscillator stability," Cardinal Components Inc., Wayne, NJ, Applications Brief No. A.N. 1006 [Online]. Available: http://www.cardinalxtal.com/docs/notes/cardinal_clock_stability.pdf
[30] B. Sundararaman, U. Buy, and A. Kshemkalyani, "Clock synchronization for wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 3, no. 3, pp. 281–323, Feb. 2005.
[31] R. H. Jones and F. Boadi-Boateng, "Unequally spaced longitudinal data with AR(1) serial correlation," *Biometrics*, vol. 47, no. 1, pp. 161–175, Mar. 1991.
[32] G. Kitagawa and W. Gersch, *Smoothness Priors Analysis of Time Series*, 1st ed. New York: Springer, 1996.
[33] P. Stoica and Y. Selen, "Model-order selection: A review of information criterion rules," *IEEE Signal Process. Mag.*, vol. 21, no. 4, pp. 36–47, Jul. 2004.
[34] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2743–2760, Oct. 1998.
[35] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, 1st ed. New York: Wiley, 1996.
[36] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, 1st ed. Upper Saddle River, NJ: Prentice Hall, 1993.
[37] "Mica2 datasheet," MEMSIC, Inc., Andover, MA [Online]. Available: http://www.mem sic.com/products/wireless-sensor-networks/wireless-modules.html
[38] F. D. Ridder, R. Pintelon, J. Schoukens, and D. P. Gillikin, "Modified AIC and MDL model selection criteria for short data records," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 1, pp. 144–150, Feb. 2005.
[39] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.
[40] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, "Network anomography," in *Proc. ACM SIGCOMM IMC*, Oct. 2005, p. 30.
[41] J. Friedman, T. Hastie, H. Hofling, and R. Tibshirani, "Pathwise coordinate optimization," *Ann. Appl. Statist.*, vol. 1, no. 2, pp. 302–332, 2007.
[42] H. Zhu, G. B. Giannakis, and G. Leus, "Weighted and structured sparse total least squares for perturbed compressive sampling," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, May 2011, pp. 3792–3795.

Hayang Kim received the B.S. and M.S. degrees in electrical engineering from Ewha Womans University, Seoul, Korea, in 2003 and 2005, respectively, and is currently pursuing the Ph.D. degree in electrical engineering at the Georgia Institute of Technology, Atlanta.

From 2005 to 2008, she worked as a Research Engineer with LG Electronics, Seoul, Korea, and developed PDP controllers. Her research interests include wireless sensor networks and clock synchronization.



Xiaoli Ma (M'03–SM'09) received the B.S. degree in automatic control from Tsinghua University, Beijing, China, in 1998, the M.S. degree in electrical engineering from the University of Virginia, Charlottesville, in 2000, and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2003.

From 2003 to 2005, she was an Assistant Professor of electrical and computer engineering at Auburn University, Auburn, AL. Since 2006, she has been with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, where she is now an Associate Professor. Her research interests include network performance analysis, wireless routing, clock synchronization, and cooperative networks.

Benjamin Russell Hamilton received the B.S.E.E. degree from Auburn University, Auburn, AL, in 2005, and the M.S.E.E. degree from the Georgia Institute of Technology, Atlanta, in 2007, and is currently pursuing the Ph.D. degree in electrical engineering at the Georgia Institute of Technology.

He performed research on routing in ad hoc and wireless sensor networks from 2006 to 2009. He has interned at General Dynamics C4 Systems, Scottsdale, AZ, where he conducted research on peak-average power ratio and pilot design for OFDM systems. He is currently researching location estimation and tracking in ad hoc networks.

Mr. Hamilton was awarded the Georgia Tech Research Institute Shackleford Fellowship in 2009.