ACES: Adaptive Clock Estimation and Synchronization Using Kalman Filtering

Benjamin R. Hamilton
School of ECE
Georgia Institute of Technology
Atlanta, GA 30332, USA
bhamilton3@ece.gatech.edu

Xiaoli Ma
School of ECE
Georgia Institute of Technology
Atlanta, GA 30332, USA
xiaoli@ece.gatech.edu

Qi Zhao
AT&T Labs - Research
180 Park Ave.
Florham Park, NJ 07932, USA
qzhao@research.att.com

Jun Xu
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
jx@cc.gatech.edu

ABSTRACT

Clock synchronization across a network is essential for a large number of applications ranging from wired network measurements to data fusion in sensor networks. Earlier techniques are either limited to undesirable accuracy or rely on specific hardware characteristics that may not be available for certain systems. In this work, we examine the clock synchronization problem in resource-constrained networks such as wireless sensor networks where nodes have limited energy and bandwidth, and also 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 applicability. We design efficient algorithms based on this model to achieve high synchronization accuracy given limited resources. These algorithms apply the Kalman filter to track the clock offset and skew, and adaptively adjust the synchronization interval so that the desired error bounds are achieved. We demonstrate the performance advantages of our schemes through extensive simulations obeying real-world constraints.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms

Design, Measurement, Theory

MobiCom'08, September 14–19, 2008, San Francisco, California, USA.

Keywords

Kalman filter, clock synchronization, clock offset, clock skew, resource-constrained network

1. INTRODUCTION

The availability of an accurately synchronized clock enables and enhances a wide range of applications in distributed environments that facilitate pervasive computing and communications. For example, Internet measurements, which rely 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, power mode energy saving, 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 is described Mills' classic papers [13, 14]. Several techniques [6, 12, 18, 25] have been proposed to improve its synchronization accuracy with local hardware enhancement or specially designed network infrastructure 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 a packet delay trace [16, 27].

In the Internet, each node is either a router or a host which is wired to a constant power source and has one or more stable and powerful CPU's. In contrast, some other networks have only very limited resources such as scarce energy, unstable processors, and unreliable and low communication bandwidth, which we refer to as "resource-constrained networks". The unique characteristics of these resource-constrained networks make it difficult to directly apply the aforementioned network clock synchronization approaches.

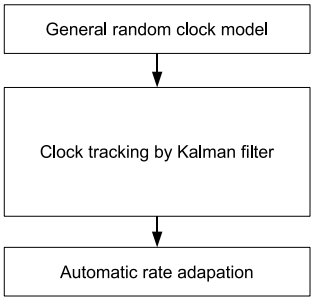


Figure 1: Block diagram of this work

They need a more energy- and communication-efficient way, a challenge we are going to address in this work.

WSN is a representative example of resource-constrained networks. In a WSN, the vast majority of sensors are battery-powered so that a desirable clock synchronization scheme must preserve energy to prolong the battery life. Pottie et al. [20] shows that transmitting 1 bit over 100 meters requires 3 joules, which can be used for executing 3 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 [19, 26]. The situation could become even worse under catastrophic conditions such as earthquake, battlefield, and forrest fire.

In this work, we design energy- and bandwidth-efficient clock estimation and synchronization techniques that can work with low precision oscillators under resource-constrained environments such as sensor networks. The main flow of this paper is summarized in Figure 1.

First, we decompose the clock uncertainty into multiple independent components and use these components to construct general models for a real clock. None of the previous synchronization protocols take into account all these components. The proposed models are also general enough to subsume all of the existing 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 performance. To our best knowledge, we are the first who try to model the random drift of the clock using Kalman filter which tracks the variation of the clock drift and thus enhances the synchronization performance. Furthermore, we derive the theoretical steady-state Mean Square Error (MSE), which is an important performance metric for evaluating the proposed scheme under various parameter settings and for comparing it with competing algorithms.

Third, we present an automatic rate adaptation method to dynamically tune the synchronization interval in order to quickly recover from a clock offset violating the pre-defined MSE bound with minimal communication cost. Since a shorter synchronization interval enhances clock tracking performance at the expense of increased communication cost, our method only expends energy when it is needed to improve synchronization. Our Kalman filter provides a general framework to achieve the optimal performance-overhead tradeoffs.

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 [4, 15, 17, 23] and sender-to-receiver modes [5, 11, 21, 22, 28].

The rest of the paper is organized as follows. We first summarize the related work in Section 2. Section 3 describes the general models we develop to model a clock in detail. In Section 4 we present and analyze our algorithms to track clock skew and offset. In Section 5 we demonstrate the need for dynamic adjustment of the synchronization period and present our rate adaptation algorithm. We conclude the paper in Section 6.

2. 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 [13, 14], which uses NTP packets containing timestamp information exchanged between 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 [12] the synchronization is offloaded into a programmable network interface card. This card would autonomously perform synchronization by sending periodic messages and perform timestamping when packets arrive. In [18, 25], the methods of using a clock based on the more accurate CPU oscillator were 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 improvements rely on some specific hardware enhancements which may not be available in a resource-constrained environment.

Since some passive network monitoring tasks do not require real-time synchronization, a few algorithms have emerged for synchronizing data captures. In [16] a linear-programming based algorithm for estimating and removing the skew and offset of a dataset was proposed. Convex hulls were used in [27] to estimate the clock skew and offset within a dataset. 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 synchronizing the clock.

WSNs bring a host of issues such as device quality and cost, many of which are not associated with wired networks. These issues lead to a number of new challenges to clock synchronization such as unstable oscillators, limited energy and communication bandwidth. Most proposed methods syn-

chronize a sender with a receiver by transmitting the current clock values as timestamps [5, 11, 21, 22, 28]. In this regard, these methods are vulnerable to variance in message delay between the sender and the receiver due to network delays and the involved workloads. Some other methods [4, 15, 17, 23] 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 message delay variance due to the reduced time-critical path which is the path of a message that contributes to non-deterministic errors. Our proposed scheme is independent of the above synchronization modes. It can be easily adapted into both modes as far as we obtain reasonably good parameter estimates.

Kalman filtering has been used in the context of clock synchronization [2, 3, 10] for packet-switched networks. In [10] Kalman filter was used to model the packet jitter after shaping its characteristics by low-pass prefiltering. In contrast, our work focuses on modeling a clock itself. [3] presents a Kalman filtering algorithm for end-to-end time synchronization. The proposed algorithm assumes the constant clock skew in a long term, which is not valid in most resource-constrained networks, and relies on NTP to exchange timestamp information. Our scheme instead assumes time-varying clock skew and uses an adaptive mechanism to tune the synchronization interval for minimizing the communication/energy cost. [2] also assumes the constant clock skew and relies on TSC register [18], a specific hardware found in Pentium class PC's to count CPU cycles.

3. 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 which 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. The derived models will be adopted by Kalman filter for clock synchronization in Section 4.

3.1 Terminology

As we have mentioned, this work takes the unique viewpoint from physical (hardware) perspective and addresses what causes clock drifting and how to model and track it.

- **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 some related phenomena 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 at 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.

3.2 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 real world. Considering all these factors, we define and model the clock offset next.

3.2.1 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 α:

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

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

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

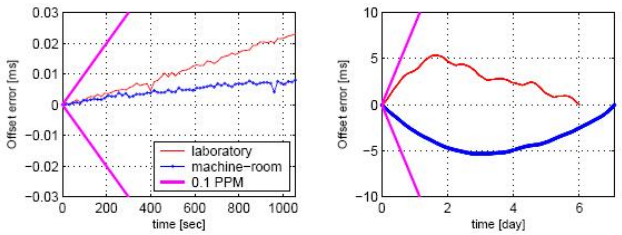


Figure 2: Illustration of clock drift (Figure 2 in [25])

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 resource-constrained networks such as WSN. 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 [1]. Additionally, physical effects such as temperature and age can change the oscillator frequency. For the remainder of the 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 the 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.

Here we borrow a figure from [25] to illustrate the randomness of clock offset and skew. The authors of [25] examined the clock offset of the same 600Mhz CPU host in two different temperature environments, *laboratory* which is an open plain area in a building without air conditioner, and *machine-room* which a closed temperature controlled room. In Figure 2 it is clear that the constant skew model fails over day timescale in both environments (the right figure), as the residual errors are far from linear. Recall that [25] adopts highly reliable CPU oscillators. We can expect that in a resource-constrained network low-cost quartz oscillators would generate more severe time-varying skews which are observable over smaller timescales (e.g., seconds or minutes).

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 Eq. (2) reduces to the simple skew model in [25].

3.2.2 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 timestamped message exchange. The timestamps are nothing but discrete samples of the continuous time. Based on Eq. (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, $\tau[k]$ is the sampling period at the k^{th} sample. Here note that our discrete-time model is also quite general. It covers not only uniform sampling, but also non-uniform 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 Gaussian distributed with variance σ_w^2. The variance of $w[n]$ depends on the time-critical path [4]. In general, the time-critical path in a sender-to-receiver synchronization consists of four factors [24]: i) the time for message construction and sender's system overhead, ii) the time to access the transmit channel, iii) propagation delay, and iv) the time spent by the receiver to process the message. In contrast, a receiver-to-receiver synchronization is only impacted by iii) and iv) and hence has smaller variance. In either case, the variance value can be estimated using samples. Notice that our tracking scheme in Section 4 is not sensitive to the accuracy of this estimate.

We can rewrite this model using 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 Gaussian random variable with mean 0 and variance $\sigma_v^2 = 2\sigma_w^2$. This is not surprising since Eq. (4) is the differential form of the observation equation for the clock, and differential forms are well-known to double the noise variance. If the observation noise $w[n]$ has non-zero mean, thanks to the differential format in Eq. (4), it is not difficult to verify that $v[n]$ still has zero mean.

3.3 Clock Skew Modeling

When using the recursive model in Eq. (4), to synchronize the clock, 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 i (constant skew): Suppose the clock skew $\alpha[n]$ is constant as in [16, 25]. From Eq. (4), since $\theta[n]$'s are known for $k = 1, \ldots, n$, if the sampling period $\tau[n]$'s are also known, the optimal clock skew estimator (in terms of 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 ii (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 of the existing schemes are based on these two simple cases without considering any statistical and time-series model of the clock skew. Because of the phase noise of oscillator, the clock skew has certain randomness, but is not completely independent for each sample. In the following, we give 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 to have random offset and skew [19]. 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., [25]) which adopt constant skew. We also assume the smoothness (order of autoregression model) of the clock skew is just first order due to the randomness of the phase noise.[1] Therefore it is reasonable to adopt first-order Gauss-Markov model for the time-varying clock skew. This means that the clock skew satisfies the auto-regressive (AR) relation as:

$$\alpha[n] = p\alpha[n-1] + \eta[n], \tag{7}$$

where p is a positive number less than but close to 1, $\eta[n]$ is model noise with zero mean and variance $\sigma_\eta^2 = (1-p^2)\sigma_\alpha^2$ with σ_α^2 being the variance of $\alpha[n]$.

Note that to obtain the variance, we only need to assume that $\alpha(t)$ is wide sense stationary. 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). This model is general and practical. It subsumes the two extreme cases as special cases. Although it is just first order, it quantifies the slow drifting of the clock frequency, captures the main variation of the skew and also takes into account the randomness.

For this model to be useful, the parameter p needs to be determined. This parameter depends 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 AR(1) parameter p can be obtained as:

$$p = \frac{r_\alpha(\tau)}{r_\alpha(0)} = \frac{r_\alpha(\tau)}{\sigma_\alpha^2}. \tag{8}$$

It is clear that as time goes on, the auto-correlation of the clock skews becomes weaker. In [8], the auto-correlation function is modeled as a decaying exponential as:

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

where ν denotes the normalizer to model different decaying rates, and ρ is a positive number close to 1. Here in this paper, we also adopt this exponential decaying model for the autocorrelation function. As τ increases, the auto-correlation $r_\alpha(\tau)$ decreases, and thus p reduces. Since the driven noise variance σ_η^2 also depends on p, when p is too small, the AR process is dominated by the noise and the tracking protocol may fail since the current value becomes too uncorrelated from the previous value.

To estimate the parameter p of AR(1) model, we need to estimate the auto-correlation function and variance of $\alpha[n]$. Given the clock observations θ_n in Eq. (1), one can obtain samples of the clock skew α_n's. Thus, the autocorrelation $r_\alpha(\tau_0)$ 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 p for the desired sampling period τ. Theoretically $\alpha(t)$ is non-stationary, and thus parameter p may change along with time. However, p changes quite slowly relative to the clock offset and thus we can still take it as quasi-stationary.

Given the AR model in Eq. (7) and the recursive observation model in Eq. (4), we are finally prepared to consider using Kalman filter as a frame work to track the variation of the clock skew and synchronize the clocks. In the following section, we will introduce how to apply Kalman filter for time synchronization.

[1] We tried the higher order models. But the improvement is quite marginal.

4. TRACKING CLOCK SKEW AND OFFSET

In this section, we design Kalman filters to track the time-varying clock skew and offset based on the proposed skew and offset models. We first introduce a scalar Kalman filter to track clock skew. Then we propose a vector Kalman filter to track both clock skew and offset meanwhile, which enhances the synchronization performance. For both cases we discuss the impact that sampling rate has on the estimation error and convergence time.

4.1 Skew Estimation with a Scalar 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 update coefficient p is known. The state equation is given in Eq. (7). The observation equation is obtained from Eq. (4) as

$$\Delta\theta[n] = \theta[n] - \theta[n-1] = \alpha[n]\tau_0 + v[n]. \tag{9}$$

The Kalman filtering process is summarized as follows (cf. [9, Chp. 13]).

$$\text{Prediction MSE}: \quad \Sigma[n] = p^2 M[n-1] + \sigma_\eta^2 \tag{10}$$

$$\text{Update}: \quad \hat{\alpha}[n] = p\hat{\alpha}[n-1] + G[n](\Delta\theta[n] - \tau_0 p\hat{\alpha}[n-1]) \tag{11}$$

$$\text{MMSE}: \quad M[n] = (1-\tau_0 G[n])\Sigma[n] \tag{12}$$

$$\text{Kalman Gain}: \quad G[n] = \frac{\Sigma[n]\tau_0}{\sigma_v^2 + \Sigma[n]\tau_0^2}, \tag{13}$$

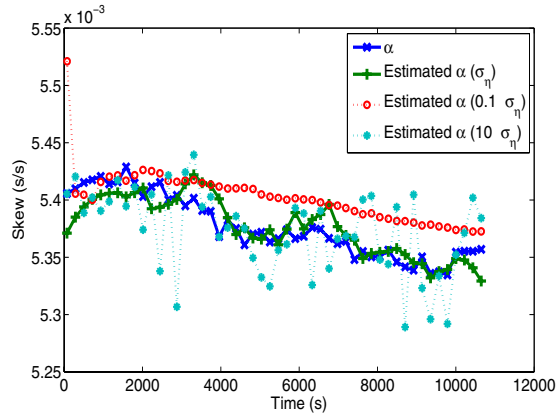
where $\Sigma[n]$ is the prediction MSE when $\alpha[n]$ is predicted from previous observations with the AR(1) model in Eq. (7), $\hat{\alpha}[n]$ is the estimate of the skew state at the n^{th} sample, $M[n]$ is the minimum mean-square error (MMSE) of the estimate, and $G[n]$ is the so-called Kalman gain. σ_v^2 is the observation noise variance, and σ_η^2 is the variance of the driven noise in the state equation. Both are given in Section 3. The recursion of the Kalman filter is initialized by

$$\hat{\alpha}[0] = \mathrm{E}\{\alpha[n]\}, \quad M[0] = \sigma_\alpha^2, \quad \theta[0] = \theta_0, \tag{14}$$

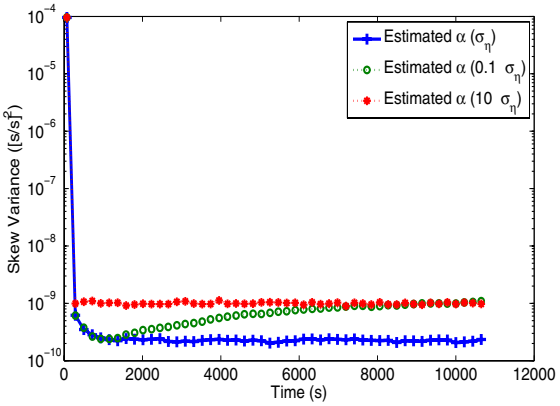
where $E\{\cdot\}$ denotes the statistical expectation, and σ_α^2 is the variance of $\alpha[n]$. Since Kalman filter does not strongly depend on the initial conditions, the statistical mean and variance can be replaced by sample mean and sample variance. For example, $\hat{\alpha}[0]$ can be initialized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

We can use the obtained estimated skew $\hat{\alpha}[n]$ to predict the clock offset recursively as

$$\hat{\theta}[n+1] = \hat{\theta}[n] + \hat{\alpha}[n]\tau_0, \tag{15}$$

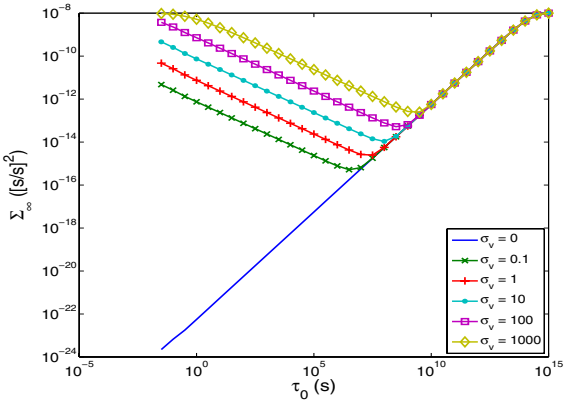


(a) Tracking of the time-varying skew $\alpha(t)$

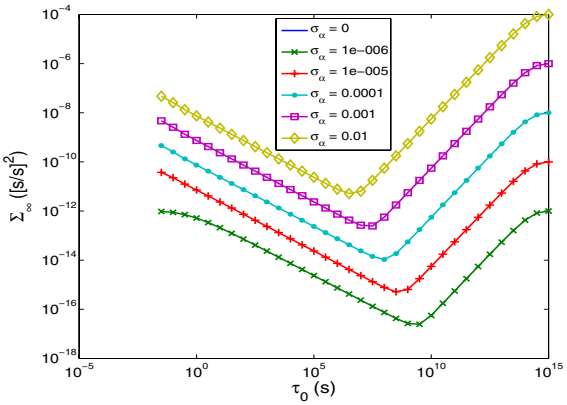


(b) MSE of the skew tracking performance

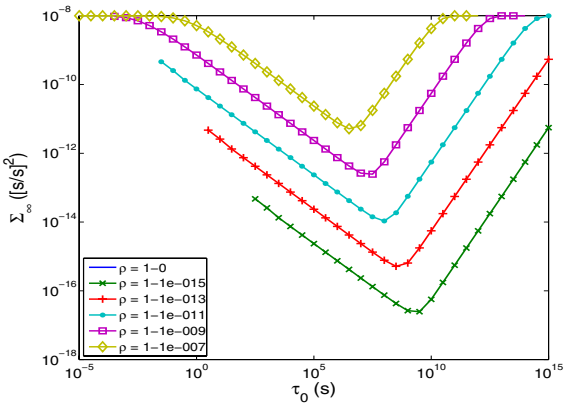
Figure 3: Clock Skew Tracking Example



(a) $\sigma_\alpha = 0.0001$, $\rho = 1 - 10^{-11}$, and varying σ_v



(b) $\sigma_v = 10$, $\rho = 1 - 10^{-11}$, and varying σ_α



(c) $\sigma_v = 10$, $\sigma_\alpha = 0.0001$, and varying ρ

Figure 4: Theoretical Σ_∞

where $\hat{\theta}[0] = \theta[0]$. There are other different ways to predict $\theta[n+1]$. In Section 4.3 we will present a more complicated estimator which uses a Kalman filter to track $\theta[n]$.

A simulated example of the performance of this skew tracking Kalman filter is shown in Figure 3. In Figure 3(a), the skew and its estimate are shown as time progresses. Note that even though the initial estimate $\hat{\alpha}[0]$ was set to 0, which is relatively far from the true value, the filter was still able to track the true state closely. Since the estimate of the clock skew closely tracks the real skew, this verifies our estimator described in Eq. (10)-Eq. (13). Figure 3(b) shows the MSE ($M[n]$ in Eq. (12)) of the clock skew estimate during this interval. This plot shows that the Kalman filter converges rapidly with high accuracy.

Here we also evaluate the performance of our skew tracking processes when the clock skew model is not estimated accurately. This could happen when the AR(1) model is not exact or we are short of prior information about $\alpha(t)$ or p gradually deviates from the real value over time without recalibration. The resulting curves in Figure 3 show that the tracking performance when σ_η^2 is set to be either 0.01 or 100 times its true value, which represents quite large modeling errors. Note that since $\sigma_\eta^2 = (1-p^2)\sigma_\alpha^2$, this is equivalent to if there was modeling error in either p or σ_α^2. In the case of σ_α^2, the modeling error is equivalent under or overestimating σ_α^2 by a factor of 0.01 or 100, respectively. This corresponds to a significant error in estimating p. If we write $p = 1 - \epsilon$ where ϵ is very small, we find $\sigma_\eta^2 \approx (2\epsilon - \epsilon^2)\sigma_\alpha^2$. Since $\epsilon \gg \epsilon^2$, this means that deviation of the parameter p from 1 was under or overestimated by a factor of 10. Our simulation in Figure 3 shows that our method still performs quite well despite these large modeling errors. This shows the robustness of our method in non-ideal situations.

Based on the Kalman filter setup in Eq. (10)-Eq. (13), we have an interesting observation. For uniform sampling, *if the*

AR parameter p does not change with τ_0, as the sampling period τ_0 increases, the MMSE ($M[n]$) for the estimate of $\alpha[n]$ approaches zero faster with the same number of samples. Initially, this claim may sound counter-intuitive since one may think the faster the sampler is, the better the estimator performs. However, here it is not this case. The explanation for this is that increasing τ_0 reduces the variance of the effective observation. If we take the observation equation in Eq. (9) we can solve for α and write:

$$\alpha[n] = \frac{\theta[n] - \theta[n-1] + v[n]}{\tau_0} = \frac{\theta[n] - \theta[n-1]}{\tau_0} + \tilde{v}[n], \quad (16)$$

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 MMSE for larger sampling period is purely due to reduced effective observation noise. While one may be tempted to opt for a large τ_0 to take advantage of the improved MMSE, greater τ_0 values imply a larger sampling period which means that even though the filter converges with fewer samples, it may actually have a longer convergence time. This motivates us to study the relationship between steady-state MSE and the sampling period τ_0.

4.2 Steady State of the Prediction MSE

The steady-state of the prediction MSE is the prediction MSE that the algorithm asymptotically converges toward as time progresses. It provides an important metric to evaluate the performance of the algorithm, allowing comparison of the algorithm when parameters are varied. The reason we consider the steady state of the prediction MSE instead of MMSE is two-fold: (i) MMSE goes to zero as the number of samples goes to infinity and/or the observation noise variance goes to zero; therefore, it is not so informative; (ii) prediction error is more important to fully utilize the AR model to design the sampling rate adaptively (Section 5).

4.2.1 Theoretical Analysis

Let Σ_∞ denote the steady-state prediction MSE (i.e., $\lim_{n\to\infty} \Sigma[n]$). From Eq. (10), Eq. (12) and Eq. (13), we obtain:

$$\Sigma_\infty = p^2 \left(\frac{\sigma_v^2 \Sigma_\infty}{\sigma_v^2 + \tau_0^2 \Sigma_\infty} \right) + \sigma_\eta^2, \quad (17)$$

where $\Sigma[n] = \Sigma[n-1] = \Sigma_\infty$. Recalling $\sigma_\eta^2 = (1-p^2)\sigma_\alpha^2$ and $p = \rho^{\frac{\tau_0}{\nu}}$, we obtain:

$$\Sigma_\infty = \frac{(1-\rho^{\frac{2\tau_0}{\nu}})(\sigma_\alpha^2\tau_0^2 - \sigma_v^2)}{2\tau_0^2} + \frac{\sqrt{(1-\rho^{\frac{2\tau_0}{\nu}})^2(\sigma_\alpha^2\tau_0^2 - \sigma_v^2)^2 + 4(1-\rho^{\frac{2\tau_0}{\nu}})\tau_0^2\sigma_v^2\sigma_\alpha^2}}{2\tau_0^2}. \quad (18)$$

It is difficult to analyze the relationship between Σ_∞ and the parameters (σ_v, ρ, ν, and τ_0) in closed forms. Here we study them numerically in Figure 4. For all these figures we fix the normalization factor $\nu = 1$ hour ($= 3,600$ seconds).

The steady state of the prediction MSE versus the sampling period τ_0 is plotted in Figure 4(a) with varying observation noise variation σ_v^2. When $\sigma_v^2 = 0$, $\Sigma_\infty = \sigma_\eta^2 = (1-\rho^{2\tau_0/\nu})\sigma_\alpha^2$ and thus Σ_∞ is a linear function of τ_0 in log scale. We observed that for fixed non-zero σ_v^2, as τ_0 increases, prediction MSE (Σ_∞) decreases and then increases.

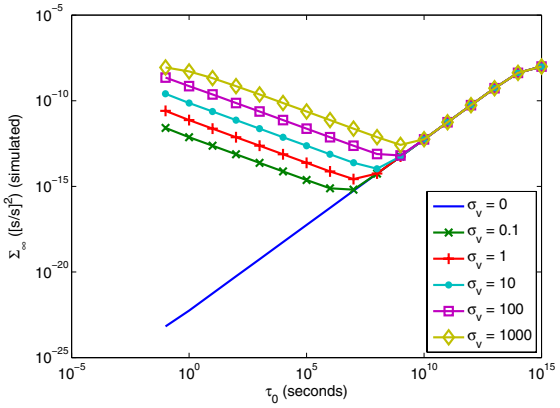


Figure 5: Simulated Σ_∞ for Skew Tracking

This is reflected in Eq. (17) as the MSE drops until the first RHS term become negligible in comparison to the second RHS term. This explains why as τ_0 increases, all of the curves converge to the noiseless case ($\sigma_v^2 = 0$). We also observe that as σ_v^2 increases (more severe noise for the observation), the steady-state prediction MSE level is higher, reflecting the effect increased observation noise has on the quality of the estimate.

Figure 4(b) illustrates the link between σ_α^2 and the prediction MSE. As σ_α^2 increases, that means the state (clock skew) has stronger randomness, and thus the tracking performance in terms of MSE is worse. This is easy to be confirmed from Eq. (18). When $\sigma_\alpha^2 = 0$, the state $\alpha[n]$ becomes deterministic and hence the tracking performance reaches the optimum. The prediction MSE converges to zero which was not shown in Figure 4(b).

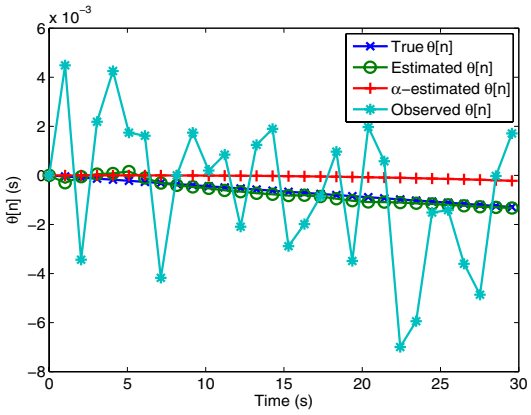
In Figure 4(c), the effect of changing ρ is shown. When $\rho = 1$, i.e., the driven noise $\sigma_\eta^2 = 0$ and thus the skew is constant, Σ_∞ is zero which is not shown in the figure. This means as the number of samples increases, for a constant skew, the steady-state MSE converges to zero. From Eq. (18), it is clear that when ρ decreases, the prediction MSE Σ_∞ increases. This is also consistent with intuition: as ρ decreases, the skew α has more severe randomness and thus more difficult to predict and track.

4.2.2 Numerical Examples

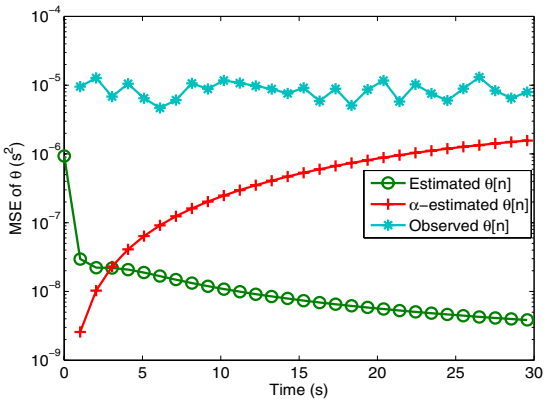
We also run a simulation to verify the theoretical results derived above where $\sigma_\alpha = 0.0002$, $\rho = 0.998$ and $\nu = 1$ hour. The simulation was continued for 20,000 synchronization periods and averaged over 50 runs. The steady-state prediction MSE from this simulation is plotted against τ_0 for various σ_v in Figure 5. If we compare this plot to those in Figure 4 (with particular attention to the plot corresponding to this one, Figure 4(a)), we can see that our theoretical analysis matches the simulation result well.

4.3 Skew and Offset Estimation with a Vector Kalman Filter

In Section 4.1, we design a scalar Kalman filter to track the variation of the clock skew $\alpha(t)$. However, since most applications need to estimate the clock offset, next we design another Kalman filter with a vector-matrix form to track the variation of the clock offset.

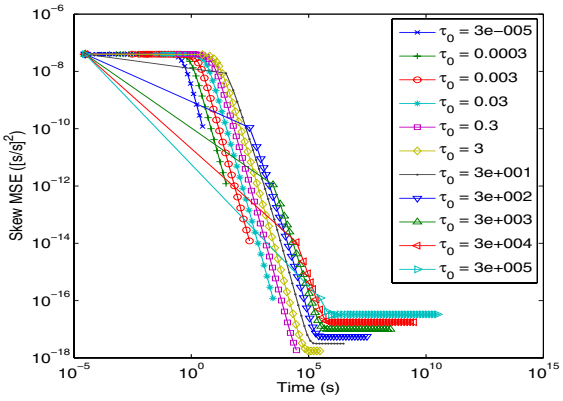


(a) Clock offset tracking by Kalman filter

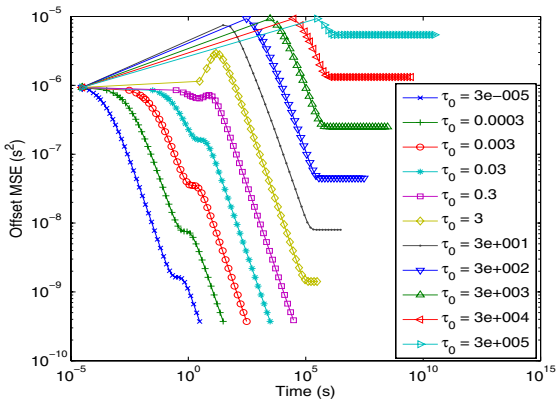


(b) MSE comparisons

Figure 6: Clock Offset Tracking Example



(a) Skew MSE vs. time for various τ_0



(b) Offset MSE vs. time for various τ_0

Figure 7: Simulated MSEs vs. time

4.3.1 *Kalman Filter*

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 $\tilde{\theta}[n] = \tilde{\theta}[n-1] + \alpha[n]\tau_0$. Based on the AR(1) model in Eq. (7), we can define an extended state equation as

$$\boldsymbol{x}[n] = \boldsymbol{A}\boldsymbol{x}[n-1] + \boldsymbol{u}[n], \tag{19}$$

where $\boldsymbol{x}[n] = \begin{bmatrix}\tilde{\theta}[n] & \alpha[n]\end{bmatrix}^T$, $\boldsymbol{A} = \begin{bmatrix}1 & \tau_0 \\ 0 & p\end{bmatrix}$, and $\boldsymbol{u}[n] = \begin{bmatrix}0 & \eta[n]\end{bmatrix}^T$. The observation equation is the noisy observation of the offset:

$$\theta[n] = \tilde{\theta}[n] + v[n] = \boldsymbol{b}^T\boldsymbol{x}[n] + v[n], \tag{20}$$

where $\boldsymbol{b}^T = \begin{bmatrix}1 & 0\end{bmatrix}$. In this case, the Kalman filter design is summarized as follows.

$$\text{Update}: \hat{\boldsymbol{x}}[n] = \boldsymbol{A}\hat{\boldsymbol{x}}[n-1] + \boldsymbol{G}[n](\theta[n] - \boldsymbol{b}^T\boldsymbol{A}\hat{\boldsymbol{x}}[n-1]) \tag{21}$$

$$\text{MSE}: \boldsymbol{\Sigma}[n] = \boldsymbol{A}\boldsymbol{M}[n-1]\boldsymbol{A}^T + \boldsymbol{C}_u \tag{22}$$

$$\boldsymbol{M}[n] = (\boldsymbol{I} - \boldsymbol{G}[n]\boldsymbol{b}^T)\boldsymbol{\Sigma}[n] \tag{23}$$

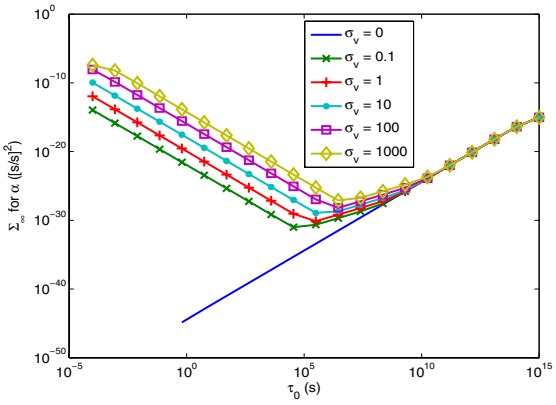
$$\text{Kalman Gain}: \boldsymbol{G}[n] = \boldsymbol{\Sigma}[n]\boldsymbol{b}\left(\sigma_v^2 + \boldsymbol{b}^T\boldsymbol{\Sigma}[n]\boldsymbol{b}\right)^{-1}, \tag{24}$$

where $\boldsymbol{\Sigma}[n]$ is the prediction MSE of the estimate when the observation is not considered, $\boldsymbol{A}\hat{\boldsymbol{x}}[n-1]$. The recursion of the Kalman filter is initialized by

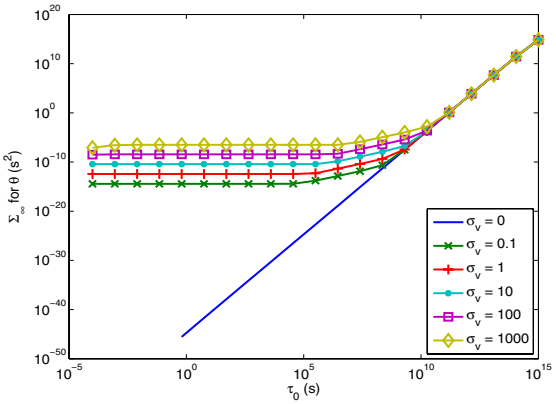
$$\hat{\boldsymbol{x}}[0] = \begin{bmatrix}\mathrm{E}\{\theta[n]\} \\ \mathrm{E}\{\alpha[n]\}\end{bmatrix}, \quad \boldsymbol{M}[0] = \begin{bmatrix}\sigma_v^2 & 0 \\ 0 & \sigma_\alpha^2\end{bmatrix}$$

An example profile of this algorithm is shown in Figure 6. The observation noise variance is $\sigma_v^2 = 10^{-5}$ s^2, and the parameter ρ is chosen as $1 - 2 \cdot 10^{-6}$ with $\nu = 1$ hour. Figure 6(a) shows the offset estimated by our algorithm from Eq. (21) - Eq. (24) ("Estimated $\theta[n]$"), the true offset $\tilde{\theta}[n]$ ("True $\theta[n]$"), the estimated offset based on Eq. (15) ("α-estimated $\theta[n]$"), and the offset from observation ("Observed $\theta[n]$"). Note that even though the observed offset is completely dominated by the observation noise, the Kalman filter is able to extract the true value with only small deviations. This vector Kalman filter outperforms the scalar version in Section 4.1. We compare their MSEs in Figure 6(b). It is observed that our vector Kalman filter quickly converges to achieve a 3 order of magnitude reduction in MSE compared to the scalar one[2].

[2]The scalar version remains valuable due to its simplicity for the applications where the clock skew is the only concern such as RTT estimation performed at a single measurement point.



(a) Σ_∞ for α using offset tracking model



(b) Σ_∞ for θ using offset tracking model

Figure 8: Simulated Σ_∞ for offset tracking

4.3.2 MSE Analysis

The convergence speed of the vector Kalman filter method is greatly affected by the synchronization period chosen. This feature is analyzed through simulation. The tracking algorithm is run for 100,000 synchronization periods with several different synchronization period lengths. The results are shown in Figure 7. The skew MSE is shown in Figure 7(a) and the offset MSE is shown in Figure 7(b). It was mentioned in Section 4.1, that for the scalar Kalman filter the skew MSE converges in fewer synchronization periods if the synchronization periods are longer. Figure 7(a) shows that this is still true for the vector Kalman filter. This can also be seen to be true for the offset MSE as well. In spite of this rapid (in terms of synchronization periods) convergence, the longer synchronization periods result in a longer time before the nodes are synchronized. Additionally, the longer synchronization periods offer worse steady-state offset MSE.

This motivates us to once again investigate the steady-state prediction MSE for this model. Unfortunately, no closed form solution could be found. Instead, we run a simulation to estimate the final steady-state MSE. Figure 8shows the effects of sampling frequency and measurement noise. In Figure 8(a) the MSE of the clock skew estimate is shown. It is very similar in shape to that shown for the skew tracking model in Figure 5. This is expected since this tracking algorithm is an extension of the previous skew tracking algorithm. If we consider the effect of the reduced MSE in the estimate of α with increasing τ_0, we would expect this to cause the offset estimate's MSE to decrease. Note from Figure 8(b) that this decrease in MSE for α does not appear to significantly affect the offset estimate. In Section 4.2.1 we claimed that the decrease in MSE for the skew estimate with increasing sampling period was due to the decrease in the $\frac{\sigma_v^2}{\tau_0^2}$ term, which was the effective variance of the noise in the observation equation. 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 MSE is relatively constant here.

Table 1: Skew and Offset Estimation Accuracy

Sampling Period τ_0 (s)	Oscillator Instability σ_α (ppm)	Steady-state MSE of θ (s^2)	Steady-state MSE of α $([s/s]^2)$
1	2	$1.9 * 10^{-15}$	$4.25 * 10^{-23}$
60	2	$4.21 * 10^{-14}$	$1.19 * 10^{-22}$
900	2	$3.25 * 10^{-13}$	$2.37 * 10^{-22}$
1800	2	$5.54 * 10^{-13}$	$2.85 * 10^{-22}$
3600	2	$9.50 * 10^{-13}$	$3.46 * 10^{-22}$
1	20	$6 * 10^{-15}$	$1.34 * 10^{-21}$
60	20	$1.34 * 10^{-13}$	$3.77 * 10^{-21}$
900	20	$1.07 * 10^{-12}$	$7.77 * 10^{-21}$
1800	20	$1.87 * 10^{-12}$	$9.58 * 10^{-21}$
3600	20	$3.36 * 10^{-12}$	$1.21 * 10^{-20}$
1	200	$2 * 10^{-14}$	$4.25 * 10^{-20}$
60	200	$4.3 * 10^{-13}$	$1.21 * 10^{-19}$
900	200	$3.83 * 10^{-12}$	$2.75 * 10^{-19}$
1800	200	$7.35 * 10^{-12}$	$3.63 * 10^{-19}$
3600	200	$1.57 * 10^{-11}$	$5.09 * 10^{-20}$

Table 1 provides some numerical results on the steady-state prediction MSEs for several different sampling periods and oscillator characteristics. The observation noise variance is $\sigma_v = 10^{-5}$ s^2, and the parameter ρ is chosen as $1 - 2 \cdot 10^{-6}$ with $\nu = 1$ hour. This table shows the high performance of our method. As the oscillator accuracy increases, the tracking performance is also improved. It can also be used to determine the clock synchronization period required for a desired offset tracking accuracy for a system that uses oscillators with a certain characteristic. In the next section we will introduce a method to adaptively select the synchronization period to achieve less than a certain probability of synchronization error.

5. ADAPTIVE SAMPLING RATE

Many applications can tolerate time synchronization offset of up to a pre-defined value, ϵ. Since the system is nondeterministic, there exists some probability P_ϵ that the offset will exceed this threshold, i.e.,

$$P_\epsilon = Pr(|\theta[n]| > \epsilon).$$

When this occurs it is called an *outage*, since the offset is "outside" the operating region. P_ϵ is called the *outage probability*. The average length of time that an outage lasts is called the average outage duration.

In Section 4.2 we have seen that the sampling rate has a strong effect on the steady-state MMSE. The sampling rate also has a profound effect on both the outage probability and outage duration. For example, an estimator with a faster sampling rate may recover from an outage more rapidly. With this in mind we turn our attention to how choosing a sampling period affects performance.

5.1 Existing Methods

In [7] an adaptive sampling algorithm was proposed. This algorithm uses a moving average of the normalized differences between the predicted and observed values to determine how much to increase or decrease the sample rate. First the algorithm generates the current normalized difference:

$$\delta[n] = \left| \frac{\theta[n] - \boldsymbol{b}^T \hat{\boldsymbol{x}}[n]}{\theta[n]} \right|. \tag{25}$$

Then the W most recent of these normalized differences terms are combined using weighted averaging to increase stability:

$$\Delta = \frac{\sum_{j=1}^{W} \delta(n-j+1)/j}{\sum_{j=1}^{W} 1/j}.$$

This weighted average is used to calculate the new sampling interval:

$$\tau_0 = \tau_0 + \gamma(1 - e^{\frac{\Delta - \lambda}{\lambda}}), \tag{26}$$

where λ is the normalized target error, and γ is a parameter that controls the rate the sampling interval changes.

There are a few aspects of this method that limit its applicability to the problem at hand. The division by $\theta[n]$ in the Eq. (25) will causes the error to increase as time passes. This is because the normalization step divides by the observed offset, which will increase in time. Additionally, this method performs poorly in noisy environments, because δ will be high even when the true error is low. Increasing W will not help to ensure convergence.

5.2 Proposed Method

In light of the problems present in existing works we design a modified approach. We assume there exists a target MSE that our algorithm should achieve. In order to limit the average outage duration we would like the algorithm to adjust rapidly when it overshoots the target, but to dampen oscillations, we will allow it to adjust slowly when it undershoots the target. Note that Eq. (26) has these characteristics. Since the Kalman filter algorithm already calculates an estimate of the current error in the form of the MMSE matrix $\boldsymbol{M}$, we can use the entry in the matrix corresponding to the error in the current estimate of $\theta[n]$ in place of Δ in Eq. (26). Our adaptive rate equation becomes:

$$\tau_0 = \tau_0 + \gamma \left(1 - e^{\frac{M_{1,1} - \lambda}{\lambda}}\right),$$

where λ is now the target MSE that we would like to achieve, and $M_{1,1}$ is the upper-left entry of the MMSE matrix $\boldsymbol{M}$.

Recall that the goal of having an adaptive sampling rate is to reduce both the probability and expected duration the error spends above some maximum allowed threshold ϵ. The target MSE λ for a given maximum error ϵ and outage probability P_ϵ can be easily calculated. If the error is a zero-mean Gaussian random variable, we let it have some variance λ.

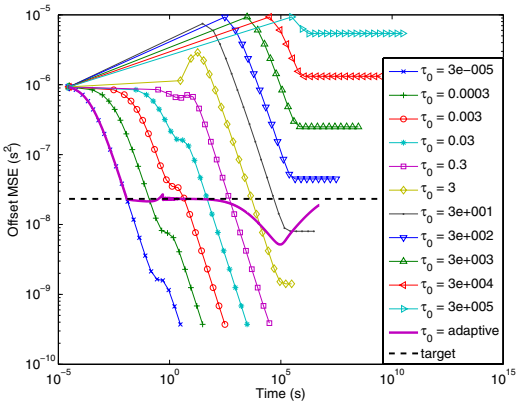


Figure 9: MSEs of offset for the fixed-rate and adaptive algorithms

Then we find λ such that the probability of the error exceeds ϵ is less than P_ϵ:

$$\lambda = \frac{-\epsilon}{Q^{-1}(\frac{P_\epsilon}{2})},$$

where $Q^{-1}(x)$ is the inverse of the cdf for the standard normal distribution function.

We use simulations to compare the uniform sampling (constant update) to adaptive sampling methods in Figure 9. As expected, the adaptive method starts with a high sampling rate, achieving an offset comparable to most frequent uniform sampling allowed. Then, after the offset MSE has achieved the target MSE, it reduces the sampling rate to that necessary to maintain the target MSE. The adaptive method overshoots the target rate slightly, but rapidly recovers (around 10^0-th second in the figure). Then later, as the estimates of α improve and the state updates become more accurate the adaptive method undershoots the target and the adaptive method makes a slow recovery as designed (around 10^5-th second in the figure).

6. CONCLUSIONS AND FUTURE WORK

Efficient and accurate network time synchronization is a challenging problem in resource-constrained networks. This paper has described a novel Kalman filter based approach to achieve high efficiency in terms of both energy and communication bandwidth. Considering the inherent instability of the inexpensive oscillators, we propose the general models to capture the time-varying behavior of clock offset and skew. Then, applying these models we generated clock skew and offset estimation algorithms based on the Kalman filter. We analyze the performance of this algorithm in detail for both the time-varying and steady-state cases. Motivated by a desire to increase efficiency, we propose a novel algorithm to adaptively adjust the synchronization interval and provide probabilistic performance guarantees, and demonstrate its superior performance through simulation. This extension achieves highly desirable synchronization accuracy with less overhead.

Several directions are open for further exploration. We mention one here. In most synchronization protocols (including ours), the clock offset is assumed as a real number. But only limited bits can be used to transmit timestamps

in practical systems. This is particularly true in resource-constrained networks due to scarcity of energy. While the small number of bits allocated to timestamps saves energy and communication bandwidth consumption, the synchronization performance could degrade significantly. It is worth studying the tradeoff between the synchronization accuracy and the space of a timestamp [3].

7. ACKNOWLEDGMENTS

The work of B. R. Hamilton, X. Ma, and J. Xu is supported in part by NSF grant 0626979, by a gift from Cisco University Research Fund at Silicon Valley Community Foundation, and through collaborative participation in the Collaborative Technology Alliance for Communications & Networks sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAD19-01-2-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

8. REFERENCES

[1] Clock oscillator stability. Cardinal Components Inc. Applications Brief No. A.N. 1006, www.cardinalxtal.com/docs/notes.

[2] L. Auler and R. d'Amore. Adaptive kalman filter for time synchronization over packet-switched networks (an heuristic approach). In *IEEE COMSWARE*, Jan. 2007.

[3] A. Bletsas. Evaluation of kalman filtering for network time keeping. *IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control*, 52(9), Sept. 2005.

[4] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.

[5] S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-sync protocol in sensor networks. In *ACM SENSYS*, Nov. 2003.

[6] IEEE std. 1588 - 2002 IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2002*, pages i–144, 2002.

[7] A. Jain and E. Y. Chang. Adaptive sampling for sensor networks. In *Proc. DMSN*, volume 72, pages 10–16, Aug. 2004.

[8] R. H. Jones and F. Boadi-Boateng. Unequally spaced longitudinal data with ar(1) serial correlation. *Biometrics*, 47(1):161–175, 1991.

[9] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, first edition, 1993.

[10] K. Kim and B.G.Lee. Kalp: A kalman filter-based adaptive clock method with low-pass prefiltering for packet networks use. *IEEE Trans. on Comm.*, 48(7), July 2000.

[3]The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

[11] Q. Li and D. Rus. Global clock synchronization in sensor networks. In *IEEE INFOCOM*, Mar. 2004.

[12] C. Liao, M. Martonosi, and D. W. Clark. Experience with an adaptive globally-synchronizing clock algorithm. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 106–114, 1999.

[13] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Trans. on Comm.*, 39(10):1482–1493, Oct. 1991.

[14] D. L. Mills. Improved algorithms for synchronizating computer network clocks. In *IEEE/ACM Trans. Networking*, volume 3, pages 245–254, June 1995.

[15] M. Mock, R. Frings, E. Nett, and S. Trikaliotis. Continuous clock synchronization in wireless real-time applications. In *IEEE SRDS*, pages 125–133, Oct. 2000.

[16] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *INFOCOM 1999*, volume 1, pages 227–234, Mar. 1999.

[17] S. PalChaudhuri, A. Saha, and D. Johnson. Adaptive clock synchronization in sensor networks. In *IEEE IPSN*, Apr. 2004.

[18] A. Pàsztor and D. Veitch. PC based precision timing without GPS. In *ACM SIGMETRICS Performance Evaluation Review*, volume 30, pages 1–10, June 2002.

[19] J. Phillips and K. Kundert. Noise in mixers, oscillators, samplers, and logic: An introduction to cyclostationary noise. In *IEEE CICC*, pages 431–438, May 2000.

[20] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of ACM*, 43(5):51–58, May 2000.

[21] K. Romer. Time synchronization in ad hoc networks. In *ACM MobiHoc*, Oct. 2001.

[22] M. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *IEEE WCNC*, 2003.

[23] W. Su and I. Akyildiz. Time-diffusion synchronization protocols for sensor networks. *IEEE/ACM Trans. on Networking*, 2005.

[24] B. Sundararaman, U. Buy, and A. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3), Feb. 2005.

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

[26] J. R. Vig. Introduction to quartz frequency standards. Technical Report SLCET-TR-92-1 (Rev. 1), Army Research Laboratory, Oct. 1992.

[27] L. Zhang, Z. Liu, and C. H. Xia. Clock synchronization algorithms for network measurements. In *INFOCOM 2002*, volume 1, pages 160–169, 2002.

[28] D. Zhou and T. Lai. A scalable and adaptive clock synchronization protocol for ieee 802.11-based multihop ad hoc networks. In *IEEE Mobile Adhoc and Sensor Systems*, Nov. 2005.