MODELING AND TRACKING TIME-VARYING CLOCK DRIFTS IN WIRELESS NETWORKS

A Thesis
Presented to
The Academic Faculty

by

Ha Yang Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2014

MODELING AND TRACKING TIME-VARYING CLOCK DRIFTS IN WIRELESS NETWORKS

To my family.

ACKNOWLEDGEMENTS

I am very pleased and honored to write this dissertation. This long course could not have been completed without many people, and their prayers and support. I am happy to have an opportunity to express my gratitude to them.

First and foremost, I am deeply grateful to my advisor, Prof. Xiaoli Ma for her invaluable advice for these years. Her enthusiasm and insights to new ideas have stimulated and encouraged me. Because of her patience and support, I could have proceeded through the doctoral program and completed my dissertation. She will be one of very influential people in my life.

I would like to thank my committee members, Prof. Mary Ann Weitnauer, Prof. David V. Anderson, Prof. Vijay K. Madisetti, and Prof. Ellen Witte Zegura, whose thoughtful comments and suggestions helped the research developed and smoothly finished. I am honored to have them serve on my committee. I also would like to thank Dr. Guotong Zhou for her helpful comments.

I wish to thank my colleagues who have spent long time together, had already graduated and are just starting in my group, Dr. Wei Zhang, Dr. Sungeun Lee, Dr. Giwan Choi, Dr. Qi Zhou, Zhenhua Yu, Malik Muhammad Usman Gul, Andrew Dawson Harper, Lingchen Zhu, Qingsong Wen, Brian Beck, Kai Ying, Yiming Kong, Hyunwoo Cho. Special thanks to Marie Shinotsuka for being a good friend, and to Dr. Benjamin Russell Hamilton for providing his measurement data.

Finally, many thanks from the bottom of my heart to my family for their prayers and love: my parents, Hyuncheol Kim and Eunhee Park, who have shown unconditioned love and thought of their daughter as a pride and joy, and my husband, Dr. Dongwon Kwon, who is always on my side with love and support. They deserve the deepest appreciation of all you have done for me. I also express my gratitude to my parents-in-law who understand and support a long course of my study, and to my sister who always encourages and trusts me. I, sincerely and deeply, thank God to guide me on all my journey.

TABLE OF CONTENTS

DEDICATION iii
ACKNOWLEDGEMENTS iv
LIST OF TABLES ix
LIST OF FIGURES x
SUMMARY xiii
I INTRODUCTION 1
1.1 Motivation 1
1.2 Objectives 3
1.3 Outlines 4
II THE IMPERFECTION OF CLOCK OSCILLATORS 6
2.1 Clock Frequency Uncertainties 6
2.2 Clock Definitions 7
2.3 Clock Synchronization Techniques 9
2.3.1 Clock Synchronization Protocols in Wireless Sensor Networks 9
2.3.2 Signal Processing for Clock Synchronization 12
2.3.3 Oscillator Frequency Characteristics 14
III MODELING TIME-VARYING CLOCK DRIFTS 16
3.1 The Derivation of Clock Models 16
3.2 Information Criteria for Selecting a Model Order 19
3.3 The Evaluation of Clock Modeling 21
3.3.1 The Measurement of Clock Skew and Offset 21
3.3.2 Model Validation 24
IV TRACKING CLOCK OFFSET AND SKEW USING KALMAN FILTERING 29
4.1 Design of a Kalman Filter for Tracking Clocks 29
4.1.1 Clock Skew and Offset Tracking 31

4.1.2 Performance of Clock Tracking 33
4.2 Clock Tracking over Unreliable Networks 39
4.2.1 Tracking Clocks with Missing Data 39
4.2.2 Tracking Clocks with Dirty Data 42
4.3 Clock Tracking with Periodic Training 46

V TEMPERATURE EFFECTS ON CLOCK DISCREPANCY IN WIRELESS NETWORKS 52

5.1 The Evaluation of Clock Error 52
5.1.1 Experimental Setup for a Wireless Connection 52
5.1.2 Time Difference 54
5.2 The Effects of Environmental Changes 61
5.2.1 Observed Temperature and Humidity 62
5.2.2 Time-varying Clock Skew 63
5.3 Temperature-correlated Clock Skew 65
5.3.1 The Effect of Temperature on Clock Skew 65
5.3.2 Correlation Analysis 68
5.4 Maximum-likelihood-type Clock Estimator (M-estimator) 71
5.4.1 Skew-Temperature Modeling 71
5.4.2 Clock Error Estimation 74

VI CLOCK TRACKING IN TEMPERATURE-VARYING ENVIRONMENTS 77

6.1 Kalman Filter Representation 77
6.2 Performance Analysis 79

VII IMPLEMENTING KALMAN FILTER-BASED CLOCK SYNCHRONIZATION 89

7.1 Implementation 89
7.2 System and Network Setup 91
7.3 Real-time Clock Tracking 92

VIII FIXED-POINT CLOCK SYNCHRONIZATION 96

8.1 Fixed Point Representation 96

8.2 Range Estimation and Wordlength Decision 98

8.3 Accuracy Evaluation 99

IX CONCLUDING REMARKS AND FUTURE DIRECTIONS . . 103

9.1 Contributions 103

9.2 Future Directions 104

REFERENCES 106

VITA 115

LIST OF TABLES

1 The comparison of tracking errors for different model orders. 37

2 The comparison of tracking performance based on clock models derived with missing observations 51

3 Temperature and clock drifts for each experiment. 61

4 The parameters and accuracy of the derived STMs. 76

5 One example of fixed point representation for each variable. $a_{max} = max(abs(a))$. 99

LIST OF FIGURES

1 Time-critical path of a message delivery. 11

2 Measurements of time-varying clock skew. 23

3 Measurements of time-varying clock skew. 24

4 Model order selection for the 32.768 kHz clock with different training periods. 25

5 Model order selection for 16 MHz clock with one-day measurement. . 26

6 An AR model and the measurement of clock skew. The models follow the measurements very closely, which verifies the proposed clock skew modeling process. 28

7 Tracking clock skew using the Kalman filter. 33

8 RMSEs for various sampling rates. 35

9 Steady state RMSE vs. sampling period. 36

10 RMSEs for various observation noise variances. 38

11 Uniform loss of data. 40

12 Consecutive loss of data. 41

13 RMSEs with random loss of data. 43

14 Tracking clock skew when dirty observations exist. 47

15 The RMSEs of clock skew and offset when dirty data exist. 48

16 Normalized MSE with Various Periodic Training 50

17 One-way timestamp dissemination. Each transmission suffers from a different, unpredictable delay. 54

18 Time differences between two wireless nodes show lots of peaks as well as floating. 55

19 Allan deviation from a three-month experiment illustrates that clock difference between the two nodes was not stabilized even with long time intervals because of the uncertainties of timestamp transmissions. 57

20 The best linear fit derived using a least squares approach does not well fit the time differences between two wireless nodes. 58

21 Delay estimates under an assumption that offset increases at a constant rate. $t_{error}[n] - y[n]$ contains time difference induced by skew variation as well as link delay. 59

22 Clock skew is Gaussian-like distributed and has heavy tails. It is fitted to a Gaussian distribution whose probability density function (PDF) is $N(-5.7644, 2.9408)$. 60

23 The distributions of clock skew estimates for different temperature conditions. 61

24 Varying environmental factors surrounding two nodes for five days. . 62

25 Instantaneous clock skew from timestamps (measured skew) is affected by unpredictable noise, and the smoothed skew follows the trend of the instantaneous skew. 64

26 Clock drifting and temperature gap are correlated in a negative way. . 65

27 A heater and a cooler controlled temperature and humidity for an hour. 66

28 Clock drifting affected by varying environmental factors lasting an hour. 67

29 Linear relationship between a temperature gap and clock skew. The error variance is 3.5306×10^{-11}. 69

30 CPU temperature by a temperature sensor inside the OMAP chip on a Pandaboard. 70

31 Linear relationship between a CPU temperature gap and clock skew. The error variance is 6.1734×10^{-12}. 71

32 Comparison between clock skew estimates by timestamp and by temperature. 75

33 Signal flow for TCKF. 80

34 Tracking performance regarding observation noise variance when σ_n=6.8578$\times 10^{-6}$. The designed Kalman filter is combined with an optimal skew model (AR(3)) and the derived STM. 82

35 Tracking performance regarding measurement noise variance when σ_v=0.833$\times 10^{-2}$. 83

36 The MSEs regarding observation noise variances for different clock model orders when σ_n=6.8578 $\times 10^{-6}$. An accurate clock model can improve tracking performance. 86

37 The MSEs regarding process noise variances for different clock model orders when σ_v=0.833 $\times 10^{-2}$. 87

38 The proposed Kalman filter architecture. 88

39 The functional flow of Kalman filter-based clock synchronization. . . 90

40 Real-time clock synchronization. 93

41 Comparison between MATLAB simulation and C implementation at $\tau_0 = 1$ minutes, $\sigma_v^2 = 1.161635 \times 10^{10}$, and $\sigma_\eta^2 = 7.421050 \times 10^5$. . . . 95

42 The conversion from floating-point to fixed-point programs. 97

43 Tracking performance by using the fixed-point Kalman filter. 100

44 The MSEs for different wordlengths in fixed-point numbers. 102

SUMMARY

Clock synchronization is one of fundamental requirements in distributed networks. However, the imperfection of crystal oscillators is a potential hurdle for network-wide collaboration and degrades the performance of cooperative applications. Since clock discrepancy among nodes is inevitable, many software and hardware attempts have been introduced to meet synchronization requirements. Most of the attempts are built on communication protocols that demand timestamp exchanges to improve synchronization accuracy or resource efficiency. However, link delay and environmental changes sometimes impede these synchronization efforts that achieve in desired accuracy.

First, the clock synchronization problem was examined in networks where nodes lack the high accuracy oscillators or programmable network interfaces some previous protocols depend on. Next, a stochastic and practical clock model was developed by using information criteria which followed the principle of Occam's razor. The model was optimized in terms of the number of parameters. Simulation by using real measurements on low-powered micro-controllers validated the derived clock model. Last, based on the model, a clock tracking algorithm was proposed to achieve high synchronization accuracy between unstable clocks. This algorithm employed the Kalman filter to track clock offset and skew. Extensive simulations demonstrated that the proposed synchronization algorithm not only could follow the clock uncertainties shown in real measurements but also was tolerant to corrupted timestamp deliveries.

Clock oscillators are vulnerable to noises and environmental changes. As a second approach, clock estimation technique that took circumstances into consideration was proposed. Through experiments on mobile devices, the obstacles were clarified in synchronization over wireless networks. While the causes of clock inaccuracy were focused on, the effect of environmental changes on clock drifting was investigated. The analysis of the observations inspired an M-estimator of clock error that was accurate but under dominant disturbances such as oscillator instability and random network delay. A Kalman filter was designed to compensate with temperature changes and estimate clock offset and skew. The proposed temperature-compensated Kalman filter achieved the better estimates of clock offset and skew by adjusting frequency shifts caused by temperature changes.

The proposed Kalman filter-based clock synchronization was implemented in C. A real-time operation was proved by clock tracking between two mobile platforms that the synchronization technique was implemented on. Moreover, the technique was converted to fixed-point algorithm, which might degrade performance, to evaluate the synchronizing operation on fixed-point processors. The fixed-point simulation reported performance degradation caused by limited hardware resources; however, it also corroborated the applicability of the synchronization technique.

CHAPTER I

INTRODUCTION

1.1 Motivation

Modern technology advances have enabled cheap and small electronic devices capable of data processing and communications. Thousands of dispersed devices make up networks in diverse fields such as military, industrial, and environment monitoring [4, 96]. In many of these applications, it is required that nodes in a network operate independently, yet at the same time cooperate with one another. For the cooperative tasks, clock synchronization is a fundamental requirement.

Clock synchronization aims to keep a common notion of time among individual nodes in a network. Each node participating network-wide collaborations has a local clock that indicates and keeps time at its own rate. A synchronized network can be achieved by that dispersed nodes exchange their local time and estimate an accurate time. However, synchronization performance is restricted by the inherent inaccuracy and instability of clock oscillators. The accuracy of a clock is also affected by some limitations such as device size, cost, and power supply in many applications [4, 96]. Since an accurate clock tends to be high-priced and consume a considerable amount of energy [86], selecting a more accurate clock may not be an optimal solution for establishing a synchronized network. Instead, network-based synchronization techniques can help to keep being synchronized among nodes, which are equipped with cheap oscillators.

Performance in network synchronization depends on not only oscillator imperfection, but also transmission delay and measurement error. Unreliable wireless links delay message transmissions or lose the messages. Moreover, the network topology may change because of the mobility of wireless nodes. All these uncertainties in networks and oscillators may disturb to keep consistent in time among nodes. These uncertainties degrade the performance of cooperative networks.

Many software and hardware attempts have been introduced to meet synchronization requirements in a wide range of applications. Most clock synchronization techniques are built on communication protocols that demand timestamp exchanges, and improve synchronization accuracy or resource efficiency. For example, measurement and control systems implemented with networking technologies such as Ethernet communications are required to synchronize all participating systems in decentralized architectures [37, 12, 100, 109, 36]. In telecommunications, as network operators migrate to packet-switched next-generation networks, synchronization will become even more important than traditional synchronization of circuit-switched links in time-division multiplexing (TDM) [25, 64].

Particularly in wireless sensor networks (WSNs), each node needs a common notion of time for data fusion, which is an important operation to integrate the collected data from each node into meaningful information [4, 96]. The same notion of time enables and enhances network and communications protocols such as scheduling for time division multiple access (TDMA) and for directional antenna reception. Clock synchronization is also a critical factor in power management such as duty cycling. Moreover, localization, security, and tracking protocols also demand nodes to record messages and sensing events by using the same reference time.

However, clock synchronization in WSNs is challenging due to their decentralized nature and uncertainties introduced by imperfections in hardware oscillators. Clocks in low-cost sensors tend to be unreliable and vulnerable to noise and environmental changes. The clocks are easily affected by temperature variations, vibration, and interference, and as a result, they progress erratically [101, 80]. Catastrophic conditions such as earthquakes, battlefield activity, or forest fires also cause the clocks to deviate significantly from the reference sources. In addition, limited energy sources, unstable processors, and unreliable low-bandwidth communications, which WSNs suffer from, impair reliability on the clocks. Therefore, clock synchronization is an important prerequisite for coordinating distributed nodes, and at the same time, it is one of research challenges in the design of energy-efficient WSNs.

Several reports have detailed the characteristics of crystal oscillator [2, 1]. Each oscillator has different stability and accuracy while they are being manufactured, and possesses different sensitivity to various factors such as temperature, humidity, shock, and aging [103, 40]. Several oscillator designs reduce the perturbation of these environmental factors. Temperature-compensated crystal oscillators (TCXOs) with good temperature characteristics contain temperature-compensation circuits to cancel out frequency offset. Oven-controlled crystal oscillators (OCXOs) add a heater and heater control to the oscillator circuit and protect the temperature influenced elements in a thermally insulated container. These approaches may result in that devices are sufficiently accurate and tolerant to temperature changes, but the approaches may prove unaffordable for the production of large networks.

1.2 Objectives

The objective of this research is to develop clock synchronization algorithms in wireless networks under unreliable and resource-constrained conditions. Below are specific goals:

- Address the synchronization problem by exploring the behavior of crystal oscillators and derive clock models that reflect the instability and inaccuracy of the oscillators.

- Propose clock-tracking algorithms combined with accurate, statistical models. It is required that the proposed tracking techniques estimate a reference clock and track time-varying drifts under non-deterministic transmission delay, measurement noise, and environmental changes.

- Verify the proposed clock synchronization algorithms through experiments on wireless devices and prove that the algorithms are suitable for practical applications.

1.3 Outlines

The rest of this dissertation is organized as follows.

Chapter 2 defines the general terms of clocks and presents the characteristics of crystal oscillators. This chapter also summaizes the literature survey of clock synchronization and estimation techniques.

Chapter 3 shows that clock oscillators have time-varying drifts over WSNs, and derives clock models that follow time-varying drifts by utilizing experimental measurements.

In Chapter 4, a Kalman filter-based tracking technique for low-precision clocks is developed. The technique is applicable for tracking clocks using the timestamp-exchange mechanisms, and for enhancing performance. It is proved that this technique can track clock behavior even when messages are missing or corrupted under unreliable links.

Since environmental changes perturb the oscillating frequencies of clocks, Chapter 5 investigates the impacts of temperature on clock inconsistency in networks. As a result of the investigation based on time and temperature measurements, clock error estimates are derived based on temperature information.

Chapter 6 proposes a clock-tracking technique that compensates clock drifts in environmental changes. The designed Kalman filter tracks clock drifting caused by heat as well as clock instability.

Chapter 7 evaluates the realization of the proposed synchronization technique.

The real-time tracking between two wireless nodes corroborates the applicability of the proposed technique.

Chapter 8 introduces the fixed-point simulation of the proposed Kalman filtering. The simulation casts light on the practical problems of hardware implementation.

Chapter 9 summarizes the contributions of this research and suggests future research directions.

CHAPTER II

THE IMPERFECTION OF CLOCK OSCILLATORS

2.1 Clock Frequency Uncertainties

A *clock* consists of a periodic component (e.g., an oscillator) and a counting component (e.g., a hardware register). The resolution, which is the smallest measurable time unit is determined by the combination of the two components. *Clock drift* refers to the phenomenon where a clock does not run at the correct speed compared to the actual time. All clocks drift differently depending on the quality of their oscillators [101, 80], the power provided from their batteries, temperature, pressure, humidity, age and so on [103]. A clock could have different clock drift rates on different occasions.

One can grasp clock behavior by examining the physical characteristics of crystal oscillators. A timekeeping element (resonant) in each elestronic device vibrates or oscillates repetitively at a certain frequency. A control circuit keeps the element running and converts its vibrations into a series of pulses. A counter register counts the pulses and adds them up to convert to time units. The frequency of a crystal oscillator is determined by the cut, size, and shape of a quartz crystal. An ideal oscillator would generate a pure sinusoid waveform as

$$v_o(t) = V_o cos(2\pi f_0 t + \phi_0), \tag{1}$$

where f_0 is an ideal frequency for the oscillator, and ϕ_0 is the initial phase at $t = 0$. V_o is a tuning voltage, and $v_o(t)$ is an output. However, in the real world, oscillating frequencies are not stabilized at f_0 and drift. The frequency of clock A can be written with a variable term:

$$f_A(t) = (1 + \delta_A(t))f_0, \tag{2}$$

where $\delta_A(t)$ is a varying frequency error [1]. Clock drifts can be explained from *phase noise* that is a common type of noise existing in oscillators. However, because of time domain instabilities (e.g., jitter), the practical oscillators generate other frequency components than the intended one. This phenomenon indicates that the phase noise components spread the power of the signal to adjacent frequencies. The frequency domain representation of the oscillator output shows some rapid, short-term, random fluctuations in the phase.

Many factors such as climate changes, aging, driving voltage, and other environmental conditions influence an oscillating rate and cause frequency error ($\delta_A(t)$). Since these factors are variable in time, a resonant frequency fluctuates. According to a reference [2], the frequency error by temperature change ($0°C - 50°C$) reaches 2×10^{-6}, and the error is up to 3^{-7} per month by aging and 1×10^{-7} by line voltage for crystal oscillators (XOs).

Each clock has a different accuracy and stability. The characteristics of a clock are highly correlated with its power consumption and its cost. For example, GPSs are accurate (10^8 10^{11}), but they demand high power (180 mW) and still are expensive for many applications. Therefore, TCXOs (temperature compensated XOs) may be an affordable choice since it consumes less energy (6 mW), but less accurate (6×10^6) [86].

2.2 Clock Definitions

The instantaneous clock drift rate is called *clock skew* and the time difference with the actual time is called *clock offset*. The starting values of the counters determine the initial relative offset between clocks. As a matter of fact, any of two oscillators can not swing at exactly the same rate, thus, every clock advances at a different rate. A clock reading comes from a clock counter which increments its value every rising or falling edge of an output wave of a crystal oscillator. The number of ticks, which is a clock counter value at t, is

$$k(t) = \Big\lfloor \int_0^t f_A(\tau)\,\tau \Big\rfloor, \tag{3}$$

where $\lfloor a \rfloor$ rounds a to the nearest integer less than or equal to a. The current time by clock A is decided by the clock register value (the number of ticks) and an expected nominal frequency:

$$C_A(t) = \frac{k_A(t)}{f_0}. \tag{4}$$

The clock error of clock A from the absolute time (t), which is called clock offset, is defined as $C_A(t) - t$. Along this, the relative clock offset between clock A and clock B is

$$\begin{align}
\theta(t) &= C_B(t) - C_A(t) \tag{5}\\
&= \frac{k_B(t) - k_A(t)}{f_0} \tag{6}\\
&= \frac{1}{f_0}\Big\lfloor \int_0^t f_B(\tau)\,d\tau \Big\rfloor - \frac{1}{f_0}\Big\lfloor \int_0^t f_A(\tau)\,d\tau \Big\rfloor. \tag{7}
\end{align}$$

Clock skew is the derivative of clock offset by time, and the relative clock skew is

$$\begin{align}
\alpha(t) &= \frac{d\theta(t)}{dt} \tag{8}\\
&= \frac{1}{f_0}[f_B(t) - \epsilon_B - f_A(t) + \epsilon_A] \tag{9}\\
&\approx \Big[\frac{f_B(t) - f_0}{f_0}\Big] - \Big[\frac{f_A(t) - f_0}{f_0}\Big] \tag{10}\\
&= \delta_B(t) - \delta_A(t), \tag{11}
\end{align}$$

where ϵ_A and ϵ_B are less than 1. Thus, they are usually very small compared to f_A and f_B, and they can be ignored. Therefore clock skew assessed by time measures conjectures the frequency error of an oscillator.

In a discrete system, clock offset is defined as a time difference between clocks, and it is assessed by instantaneous time measurements as

$$\theta[n] = C_A[n] - C_B[n], \tag{12}$$

where n is the sample index, "$[\cdot]$" is adopted for discrete indexing, $C_i[n]$ is the n^{th} measurement of clock i. A challenge for clock synchronization is that clock offset is not constant and varying because of the influences on the oscillating frequency of each clock. The drifting rate of clock offset is called clock skew, which is defined as the derivative of clock offset by time. Accumulated clock offset is then updated by the estimate of clock skew ($\hat{\alpha}[n]$) and an observation interval as

$$\theta[n] = \theta[n-1] + \hat{\alpha}[n]\tau[n] + v[n], \tag{13}$$

where $v[n]$ is a random variable with zero mean. This recursive form of clock offset model covers not only uniform interval but also non-uniform interval synchronization by choosing a different $\tau[n]$.

A crystal oscillator can be characterized by frequency accuracy, which is the offset from the specified target frequency, and stability, which is the spread of the measured oscillator frequency about its operational frequency in a period time. The defects in an oscillator definitely cause clock discrepancy in a network. Therefore, it is important to understand oscillator characteristics to achieve network synchronization.

2.3 Clock Synchronization Techniques

2.3.1 Clock Synchronization Protocols in Wireless Sensor Networks

Many clock synchronization techniques have been proposed over the past few decades. The most popular and widely used protocol for the Internet is Network Time Protocol (NTP) [69, 70]. However, NTP provides insufficient accuracy and robustness for many applications such as network measurement. Several techniques [37, 100, 60, 79] have improved clock stability and accuracy to satisfy the requirements of demanding applications. Unfortunately, all these techniques may be inappropriate for WSNs because of limitations in low-cost devices [4, 96].

With use of the global positioning system (GPS), network devices may be synchronized to fulfill the requirements of network and communications protocols such as carrier sense multiple access with collision avoidance (CSMA/CA) [89, 5]. In vehicular ad-hoc networks (VANETs), each device is equipped with a GPS receiver, and it can generate a synchronized clock with accurate clock information from satellites. However, the GPS can not be a solution of network-wide clock synchronization in some applications because of its cost. WSNs usually consist of low-cost nodes that do not include a GPS receiver, and they require an affordable way to keep the same notion of time over the networks. Even when a device receives a GPS signal, it may not record timestamps stably at its application layer. The random delay from an antenna to higher layers prevents devices from recording an accurate receiving time using a GPS-synchronized software clock. Timestamping at a physical layer may help to avoid a nodal processing delay [37, 24, 106, 62, 3], but a local clock, which is used for timestamping at the physical layer and not synchronized with a GPS clock, is still affected by drifts [9]. Hence, still clock corrections or estimation algorithms should be employed to avoid clock drifts.

Clock synchronization methods for WSNs in [27, 66, 59, 30, 84, 67, 90, 58, 111] synchronize a sender with a receiver by exchanging current clock values as timestamps. The sender-to-receiver protocols are vulnerable to variance in message delays between a sender and a receiver due to network delay and processing overhead. In general, the time-critical path in a sender-to-receiver synchronization consists of four factors [96]: (i) the time for message construction and the system overhead of the sender; (ii) the time to access the transmission channel; (iii) propagation delay; and (iv) the time spent by the receiver to process the message. The authors of [27] propose the timing-sync protocol for sensor networks (TPSN) based on a two-way message exchange mechanism. This protocol consists of two phases. First, each node decides its level, and then, the nodes correct their clocks through timestamp exchanges by estimating the clock offset. Another example of synchronization protocols based on a two-way message exchange mechanism is introduced in [66]. The main difference with TPSN is that the network structure is assumed to be a mesh type topology instead of a tree topology. The approach estimates clock offset, but not clock skew while resulting in frequent re-synchronization. Another variation of TPSN is proposed in [30] that builds a tree structure within the network and achieves reasonable accuracy, while using modest computational resources. Some references [84, 67, 90] also introduce protocols suitable for environments with strict resource constraints based on sender-to-receiver synchronization schemes.

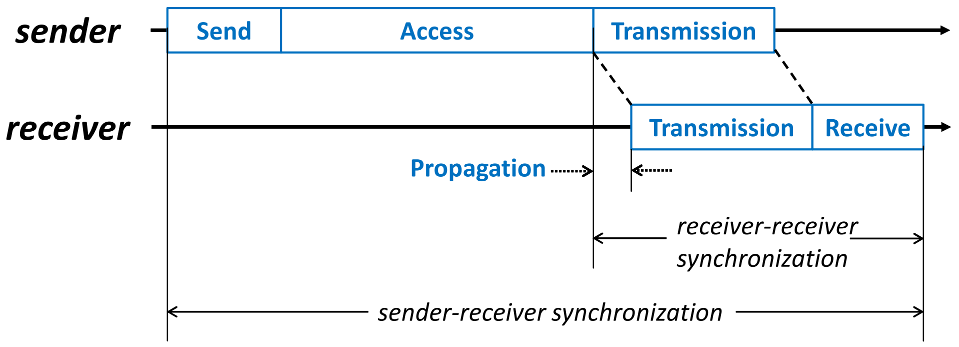


Figure 1: Time-critical path of a message delivery.

Unlike the previous sender-to-receiver synchronization, some other methods perform receiver-to-receiver synchronization [23, 77, 71, 95]. These synchronization methods exploit the property of the physical broadcast medium, where any receivers in one-hop away receive the same message at approximately the same time. Such an approach reduces the message delay variance because non-deterministic delay at the transmitter no longer affects the accuracy of timestamps. A receiver-to-receiver synchronization is only impacted by propagation delay and processing time spent by the receiver and hence exhibit smaller variance. The reference broadcast synchronization (RBS) introduced in [23] seeks to reduce non-deterministic latency using receiver-to-receiver synchronization and to conserve energy via post-facto synchronization. RBS is extended in [77] by providing an adaptive feature of allowing to trade-off dynamically synchronization accuracy for computational and energy resources. As an extension of the IEEE 802.11 standard for wireless local area networks, the authors of [71] define a continuous clock synchronization protocol in WSNs while mitigating the transmission path delay. The protocol introduced in [95] operates in alternating active and inactive phases periodically to enable all the sensors in the network to have a local time around the network-wide equilibrium time.

2.3.2 Signal Processing for Clock Synchronization

In a WSN, a desirable clock synchronization scheme contemplates energy efficiency to prolong a battery life. Since transmission costs are much higher than computation costs as presented in [81], some researchers have recently attempted to deal with the clock synchronization for WSNs from the signal processing perspective to achieve high synchronization accuracy.

By assuming that non-deterministic network delay is a random variable, many clock estimators were proposed [105]. This kind of clock estimation was started in [38, 42] by deriving maximum likelihood estimators (MLEs) of clock offset in exponential network delay. In the two-way message exchange mechanism (TPSN), the authors of [76] derive estimation of clock skew as well as that of clock offset. The joint MLEs of clock skew and offset are derived when the delay is Gaussian distributed, and a new estimator is proposed when the delay is exponentially distributed. Under an assumption of the exponential delay, the MLEs of clock skew and offset estimators are derived though iterative methods in [14]. In [55], the estimator of [76] is generalized to lower the complexity and enhance the performance. The authors of [13] employ this estimator [55] for pairwise broadcast synchronization. In [35], the performance of clock estimation is analyzed in both the cases of a one-way message transmission and a two-way message exchange. They prove that a one-way dissemination approach provides more accurate estimation for clock skew, and a two-way exchange scheme yields better clock offset estimation. Clock drifts are also estimated by using best linear unbiased estimator (BLUE) [15] and minimum-variance unbiased estimator

(MVUE) [16].

Kalman filtering has been used in the context of clock synchronization. In [49], a Kalman filter was used to model the fluctuation in packet inter-arrival times, after shaping the fluctuation with low-pass pre-filtering. The authors of [11] introduced a Kalman filtering algorithm for end-to-end time synchronization. The algorithm assumed a constant clock skew in the long term, and relies on NTP to exchange timestamp information. The authors of [6] also assumed a constant clock skew and relied on the TSC register to count CPU clock cycles. In [31], Kalman filtering was also employed by assuming that clock skew has a first-order Gauss-Markov model. Kalman filtering was also used as a clock servo in IEEE 1588 networks to enhance synchronization accuracy [28]. Instead of 32-bit timestamps, a sign of innovations was used for Kalman filtering [82], which could be achieved with low-cost communications.

Other than Kalman filtering, many signal estimation techniques were adopted for synchronization. In [17], a clock synchronization problem that was caused by sleep nodes was solved by a closed-loop control called a feedback-based synchronization (FBS) scheme. On the other hand, duty cycling could be used as a time-domain reference for detecting clock drifts [34]. In [54, 57], the estimation of clock skew, clock offset and fixed delay was formulated as a linear programming problem. A joint estimation of synchronization and localization for underwater wireless networks was proposed in [61]. The joint estimation used feedback loops to each other and improved accuracy by compensating a stratification effect in the underwater. In [48], a particle filter was adopted. Some synchronization techniques used some off-line methods such convex hulls [108], principal component analysis [10], Fourier analysis [7]. However, some of them demanded high computation loads, which might not be proper for resource-constrained applications.

Some synchronization algorithms introduce combinations of synchronization protocols and signal processing techniques. These algorithms pursue network-wide synchronization and solve network-specific problems. An average consensus algorithm [65] finds clock offset and skew compensation parameters. As nodes repeat a synchronization round, the network is gradually synchronized. In [56], a belief propagation is used for network-wide distributed synchronization, which outperforms an average consensus principle. A distributed asynchronous clock synchronization (DCS) protocol [18] also achieves global synchronization based on clock reading exchanges for delay tolerant networks (DTNs). A weighting coefficient is introduced to indicate the accuracy level of propagated clock information. Moreover, in [92, 29], a spatial smoothing algorithm is proposed based on an idea that clock offsets between neighboring nodes on a loop in a multi-hop network add up to zero.

2.3.3 Oscillator Frequency Characteristics

Some approaches focus on the characteristics of oscillators in frequency [112]. A time synchronization algorithm proposed in [88] compensates for the frequency drift of an oscillator caused by temperature changes. As storing the pairs of temperature and relative drift into a table, each node can achieve a longer synchronization period. In [107], assuming that clock skew is a non-stationary random process, a Kalman filter-based algorithm is proposed to dynamically compensate clock skew. Based on temperature measurements, this algorithm prolongs the re-synchronization period. The temperature characteristics of clocks are adopted to security research in [74]. The load on the CPU affects its heat output, and the result of temperature on clock skew can be remotely detected through observed timestamps.

In [87], instead of clock readings, which are expensive to access and transmit, manufacturing parameters of crystal oscillators are exploited for clock compensating and estimating. However, this approach is developed based on an assumption that the parameters are known. The combination of manufacturing parameters and timestamps is used [63]. The proposed algorithm requires significant computation resources, but achieves network synchronization in a distributed way.

CHAPTER III

MODELING TIME-VARYING CLOCK DRIFTS

3.1 The Derivation of Clock Models

With an understanding of clock drifts, the variation of a clock is broken down into three independent components: clock skew $\alpha(t)$, an initial clock offset θ_0, and random, additive noise $w(t)$. The clock offset $\theta(t)$ at time t is given as

$$\theta(t) = \int_0^t \alpha(\tau)\ d\tau + \theta_0 + w(t). \tag{14}$$

This model is quite general and subsumes all other existing simple clock models. For example, if the clock skew $\alpha(t)$ does not change along with time t, the model in (14) reduces to the constant skew model in [100].

Since clock synchronization is typically achieved by exchanging timestamps, which are discrete samples of the continuous time, the discrete-time clock model in (15) is more useful than the model in (14).

$$\theta[n] = \sum_{k=1}^{n} \alpha[k]\tau[k] + \theta_0 + w[n], \tag{15}$$

where k is the sample index, and $\tau[k]$ is the sampling period at the k^{th} sample. Note that this discrete-time model covers not only uniform sampling, but also non-uniform sampling by choosing a different $\tau[k]$. Since $w[n]$ is caused mainly by the observation and measurement noise, it is reasonable to assume $w[n]$'s are independently and identically distributed (i.i.d) with variance σ_w^2. The variance of $w[n]$ depends on the time-critical path [96]. The discrete-time clock model can be rewritten in a recursive form as (13), where $v[n] = w[n] - w[n-1]$. Clearly, $v[n]$ is a random variable with mean 0 and variance $\sigma_v^2 = 2\sigma_w^2$.

Before the clock skew can be estimated, two extreme cases of clock skew must be considered.

Constant skew: Suppose the clock skew $\alpha[n]$ is constant as in [100, 73]. From (13), 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

$$\hat{\alpha}[n] = \frac{\sum_{k=2}^{n}(\theta[k] - \theta[k-1])\tau[k]}{\sum_{k=2}^{n}\tau^2[k]}. \tag{16}$$

Independent skew: If the clock skew $\alpha[n]$ changes completely from one sample to another, the optimal estimator becomes

$$\hat{\alpha}[n] = \frac{\theta[n] - \theta[n-1]}{\tau[n]}. \tag{17}$$

These two models are simple, but neither one is practical. Most existing schemes are based on these two simple models without consideration of any statistical or time-series models of clock skew. Because of phase noise in the oscillator, clock skew has a certain randomness. The clock skew is not constant, but time-varying in the real environment, as shown in the measurement results (see Figure 2). It is not completely independent for each sample. The constant skew model fails over timescales of several hours. It is expected that clock skew would vary severely in any harsh environment as a result of energy insufficiency, or temperature variations. Therefore, it is necessary to investigate clock models that reflect these time-varying characteristics for clock skew.

Time-varying clock skew is assumed to be a random process with zero mean and a small perturbation around the mean. This assumption has been applied in some previous works (e.g., [100]), which adopt a constant skew. Here the time-varying skew is modeled as an auto-regressive (AR) process [99]. The smoothness (order of AR model) of the clock skew is P, which is more general than in the previous works

[100, 31]. Later, it will be shown that AR model order P can be reasonably small as a result of real measurements. Thus, the complexity of the model is limited. Given an AR model with an order P, the time-varying clock skew satisfies the relation as

$$\alpha[n] = \sum_{i=1}^{P} c_i \alpha[n-i] + \eta[n], \tag{18}$$

where c_i's are AR coefficients, $\eta[n]$ is modeling noise with zero mean, and σ_η^2 is variance. Usually $\eta[n]$ is modeled 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 (18) is general and practical and subsumes the two special cases in (16) and (17). This model quantifies the drifting of the clock frequency, captures the main variation of clock skew, and also takes into account the randomness.

The parameters c_i's and the model order P need to be properly determined to construct a clock skew model. One way to estimate these coefficients is based on the statistical properties of $\alpha[n]$. If the auto-correlation function of $\alpha(t)$ is defined as $r_\alpha(\tau) = E\{\alpha(t)\alpha(t+\tau)\}$, the AR($P$) coefficients c_i's can be derived as

$$\begin{bmatrix} r_\alpha(P\tau) \\ r_\alpha((P-1)\tau) \\ \vdots \\ r_\alpha(\tau) \end{bmatrix} = \begin{bmatrix} r_\alpha(0) & \dots & r_\alpha((1-P)\tau) \\ r_\alpha(\tau) & \dots & r_\alpha((2-P)\tau) \\ \vdots & & \vdots \\ r_\alpha((P-1)\tau) & \dots & r_\alpha(0) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{bmatrix}. \tag{19}$$

It is clear that as time goes on, the auto-correlation of the clock skew becomes weaker. In [39], the auto-correlation function is modeled as a decaying exponential as $r_\alpha(\tau) = \sigma_\alpha^2 \rho^{\frac{\tau}{\nu}}$, where ρ denotes the normalized decay in time period ν. Given the clock observations $\theta[n]$, one can obtain samples of the clock skew $\alpha[n]$ using the independent skew model in (17). Thus, the auto-correlation $r_\alpha(\tau_0)$ and the variance

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

can be estimated using sample means. Once the auto-correlation is produced, parameters are estimated by setting $\nu = \tau_0$ and solving for ρ. Then, this auto-correlation is used to estimate the c_i's for the desired sampling period τ. Theoretically, $\alpha(t)$ is non-stationary, and thus the coefficients c_i's may change over time. However, the c_i's change quite slowly relative to the clock offset and thus are taken as quasi-stationary.

Another way to estimate AR coefficients is to use some of the measurements as training data. Suppose that T observations where $T > P$ were collected as the model fitting data. Then, (18) can be rewritten as

$$\begin{bmatrix} \alpha[P+1] \\ \alpha[P+2] \\ \vdots \\ \alpha[T] \end{bmatrix} = \begin{bmatrix} \alpha[P] & \dots & \alpha[1] \\ \alpha[P+1] & \dots & \alpha[2] \\ \vdots & & \vdots \\ \alpha[T-1] & \dots & \alpha[T-P] \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{bmatrix} + \begin{bmatrix} \eta[P+1] \\ \eta[P+2] \\ \vdots \\ \eta[T] \end{bmatrix}. \tag{20}$$

For simplicity, let $\mathbf{z}$, $\mathbf{Z}$, $\mathbf{c}$ and $\boldsymbol{\eta}$ denote the four column vectors/matrices in (20). Then, performing QR-decomposition on $\mathbf{Z}$ gives an upper-triangular matrix $\mathbf{R}$. (20) can be written 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, the AR coefficients in (20) are obtained by a least squares approach. The derived model is general enough to subsume the existing skew models.

3.2 Information Criteria for Selecting a Model Order

The selection of the model order P in (18) is an important step to model the time-varying clock skew as an AR process. As the order of a model increases, the model usually better fits the measurements; however, if a 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 performs well on the training data, it will perform poorly on new data. Following the principle of Occam's razor, information criteria is adopted to select a proper model order [51, 94, 8, 33, 43, 44].

One widely adopted information criterion is Akaike information criterion (AIC) [51]. With finite dimensional AR models, the AIC provides an asymptotically efficient solution under a quadratic loss function. AIC finds an appropriate order to select the best fit to the data. It was originally derived from maximizing the likelihood function of a given model while adding a modeling penalty function, which is a term that represents the complexity generated as the modeling order increases. The modeling penalty function in AIC is the number of estimated parameters in the model as

$$\mathrm{AIC}(P) = -2 \times [\text{maximized log likelihood}] \quad + \quad 2 \times [\text{the number of parameters}]. \tag{21}$$

When the noise is Gaussian, the AIC function can be simplified by the least squares coefficient estimator in (20) as (c.f. [51])

$$\mathrm{AIC}(P) = T \log(2\pi\hat{\sigma}_P^2) + 2P, \tag{22}$$

where $\hat{\sigma}_P^2$ is an estimate of the variance of the modeling error $\eta[n]$ in (18) as

$$\hat{\sigma}_P^2 = \frac{1}{T-P} \sum_{n=P+1}^{T} \left(\alpha[n] - \sum_{i=1}^{P} \hat{c}_i \alpha[n-i] \right)^2, \tag{23}$$

with $\hat{c}_i$ denoting an estimate of coefficients for an AR model.

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:

$$\mathrm{MDL}(P) = T \log(2\pi\hat{\sigma}_P^2) + (\log T)P. \tag{24}$$

By multiplying P by $\log T$, the penalty term becomes larger as the number of observations increases. When a large number of observations are given, MDL tries to find the smaller number of parameters by adding a heavier penalty than AIC.

The third information criterion used is AIC_c, which is a modification based on AIC. AIC_c gives a better modeling order than AIC 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.

$$\mathrm{AIC}_c(P) = T\log(2\pi\hat{\sigma}_P^2) + \frac{2T}{T-P-1}P. \tag{25}$$

An optimal order of an AR process can be decided when the results of these information criteria are minimized. This optimal order and the parameter estimates of an AR model are used to construct the time-varying clock skew model in (18). 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 for estimating the AR coefficients, the statistical method in (19) is more robust and provides better performance for long data sets, but the training-based method in (20) is easier to implement and more feasible even for a small amount of data. Here, the training-based method was adopted for the experimental measurements.

3.3 The Evaluation of Clock Modeling

This sub-chapter verifies the introduced clock modeling process with measurements of real low-cost clocks and evaluates the derived clock models by showing their performance.

3.3.1 The Measurement of Clock Skew and Offset

To ensure applicable results, a low-powered micro-controller platform similar to that deployed in sensor networks is used. Like the Mica2 mote [68], the device tested uses an Atmel ATmega128L processor with a 32.768 kHz crystal oscillator. The hardware platform differs in that it uses a 16 MHz primary crystal oscillator to drive the processor instead of the 8MHz oscillator present on the Mica2 mote and lacks the RF interface.

The device was programmed to maintain two counters representing the time in

1/32768th of a second. One counter was driven by the 16 MHz external 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 four-byte timestamp. The device would then reply by sending a four-byte timestamp for each of the counters in response. Upon reception of the single byte, the device disabled interrupts, copied the current time value from both counters to temporary registers, enabled interrupts, and transmited the timestamps a byte at a time.

Timestamps were recorded on a 2.4GHz 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 an "mlockall" call, which preventing the program from being swapped to disk, avoided 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 by using the "gettimeofday" call (which uses the TSC register of the processor for accurate timing) and converting the time to 1/32768th of a second. The PC also recorded the amount of time spent for each timestamp request to measure the influence of variable delay on the accuracy of the measurements. For 99.8% of the measurements, processing the timestamp request took within 1/32768th of a second 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 the outdoor environment of sensors. Timestamps were collected using this setup every second for over 1.5 months.

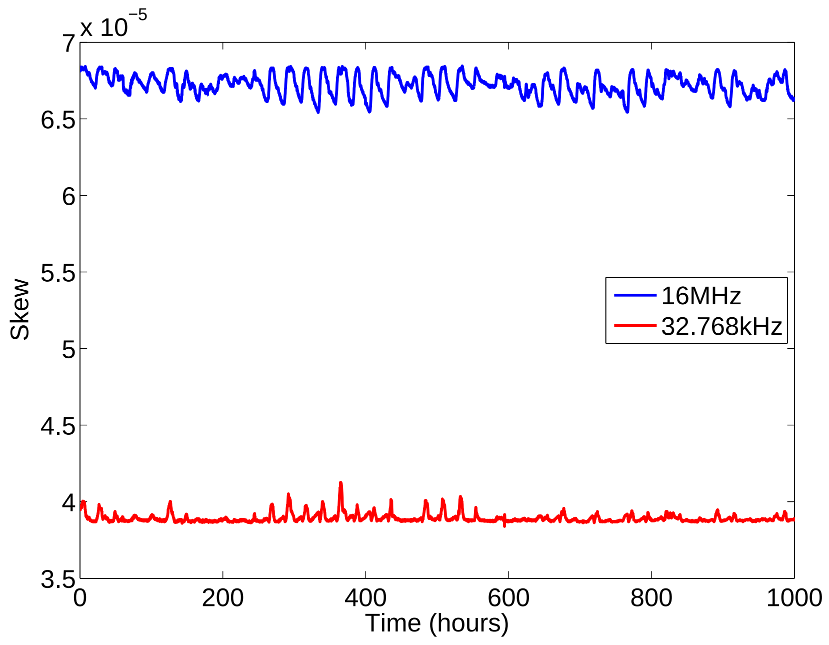


Figure 2: Measurements of time-varying clock skew.

Clock skew between samples separated by a sampling interval was calculated over all the dataset. The clock skew, which was derived by the measurements, is shown in Figure 2. This technique removes the effects of quantization noise, but a larger interval would remove some dynamic components. The instantaneous offset $\theta[n]$ was obtained as $\theta[n] = t_{\text{PC}}[n] - t_{\text{device}}[n]$, where $t_{\text{PC}}[n]$ is the instantaneous time of the PC and $t_{\text{device}}[n]$ is that from the counters of the device, every 900 seconds (four samples per hour) to reduce the effects of noise in simulations. The skew $\alpha[n]$ was obtained as $\alpha[n] = \frac{\theta[n+1]-\theta[n]}{t_{\text{PC}}[n+1]-t_{\text{PC}}[n]}$ for the same interval as the offset. Over the course of the measurements, 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 varied over time, as much as 3 ppm over this course. Moreover, the accumulated offset was about 150 seconds for the 32.768 kHz clock and about 250 seconds for the 16 MHz clock. The variation of clock skew is also noticeable in one day, as shown in Figure 3. It is observed that the pattern in this figure tend to be repeated daily.

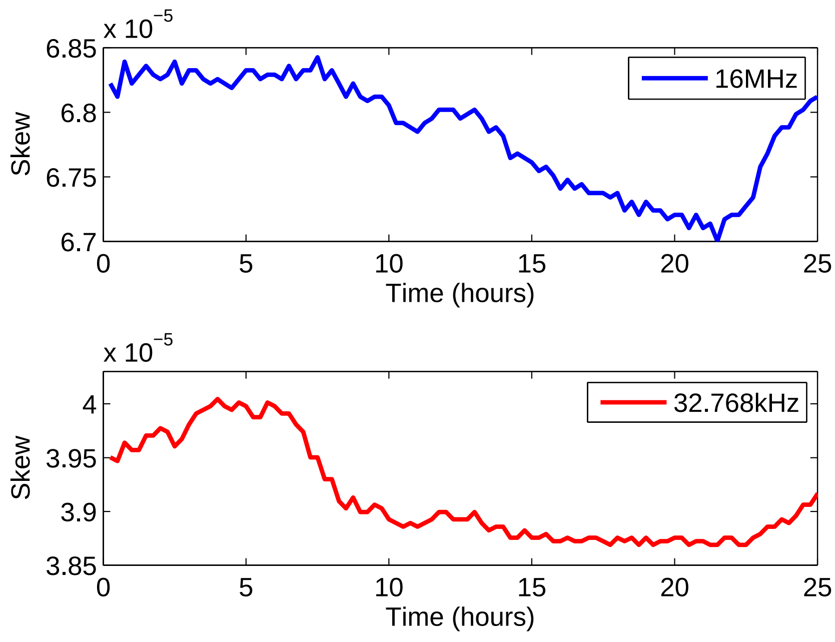
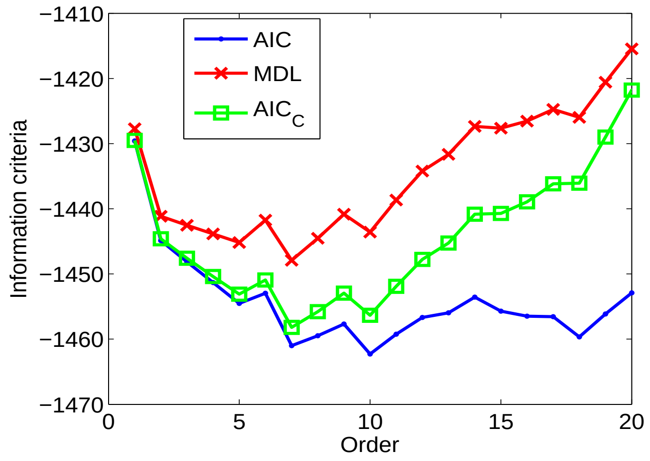


Figure 3: Measurements of time-varying clock skew.

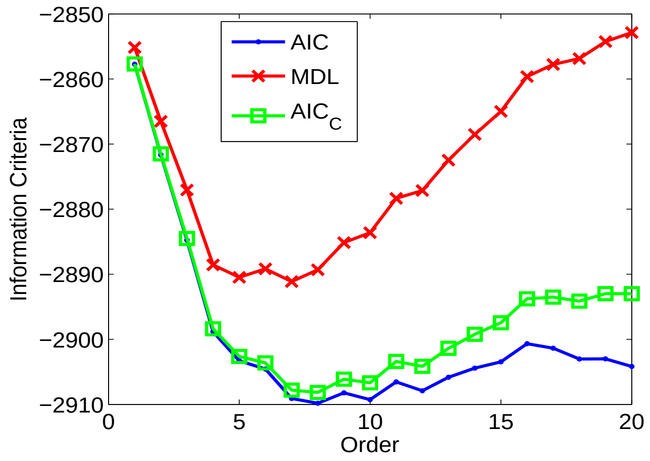
3.3.2 Model Validation

The real measurements were used to select a reasonable model order P and validate the AR(P) model. The information criteria were employed to find an appropriate order of an AR model for each clock skew. For the 32.768 kHz clock, the measurements of a half-day, one day, and three days, respectively, were trained to derive a model and find an optimal model order. As shown in Figure 4, if more data were trained, the proper number of parameters for an AR model decreased and the curves of information criteria apparently revealed the proper number. The optimum orders of the models based on AIC, MDL, and AIC$_c$ were 8, 7, and 8, respectively, when the measurements from a single day, i.e., $T = 96$ were trained.

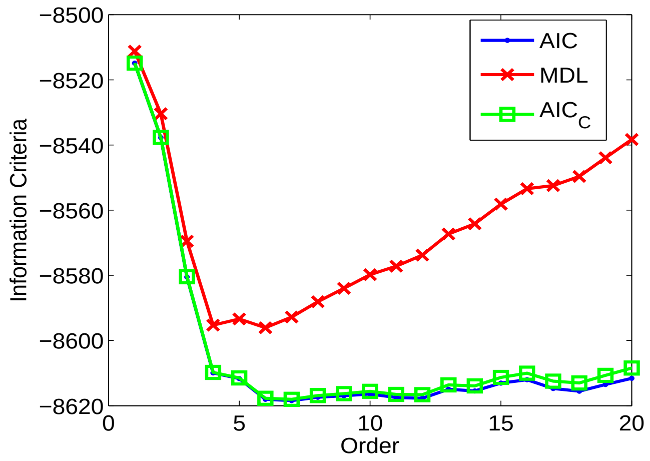
A training length may affect an optimal order of a model. It is observed that a longer training length tends to find a sharper curve with a smaller model order while a shorter training length results in unstable curves of information criteria. This observation concludes that it is reasonable to train one-day data in order to obtain



(a) By training 12 hours-data, the optimal orders are 7, 7, and 10 for AIC, MDL, and AIC$_c$, respectively.



(b) By training one day-data, the orders are 7, 8, and 8.



(c) By training three days-data, the orders are 6, 6, and 6.

Figure 4: Model order selection for the 32.768 kHz clock with different training periods.

an AR model order for the given dataset.

Similarly, for the 16 MHz clock, the measurements over one day are used to calculate the cost of each fitting order of the AR model for each information criterion. As Figure 5 shows, the results of AIC, MDL, and AIC$_c$ are similar, and the optimum order of the model is four. Comparing different criteria, it is observed that the MDL criterion results in a sharper curve due to the heavier cost function. This indicates that the MDL criterion is more consistent and reliable when the data set is large and the measurement duration is long [8, 83]. However, based on the real measurements for clock skew, all criteria provide similar results. Therefore, one can choose any of these information criteria to find the optimal model order.

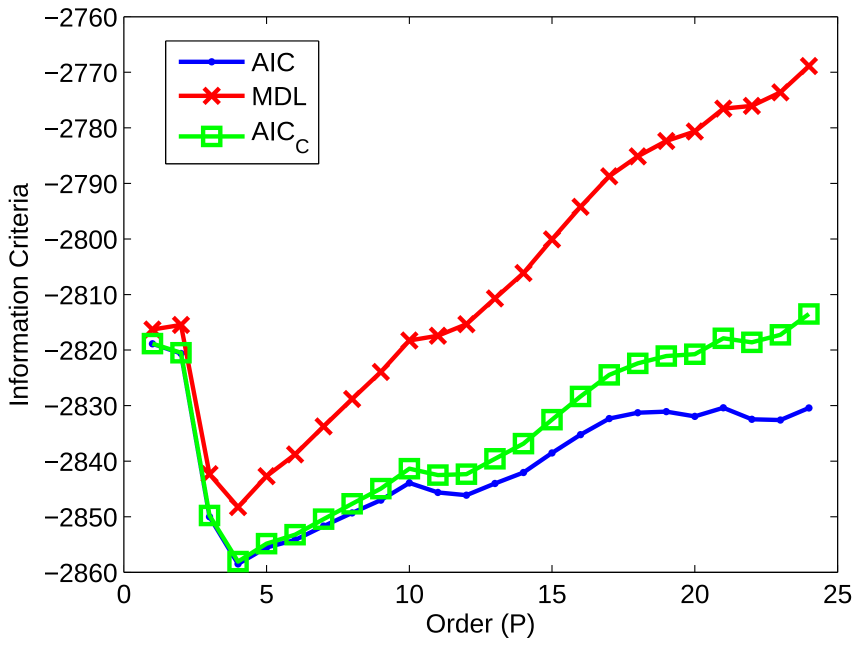
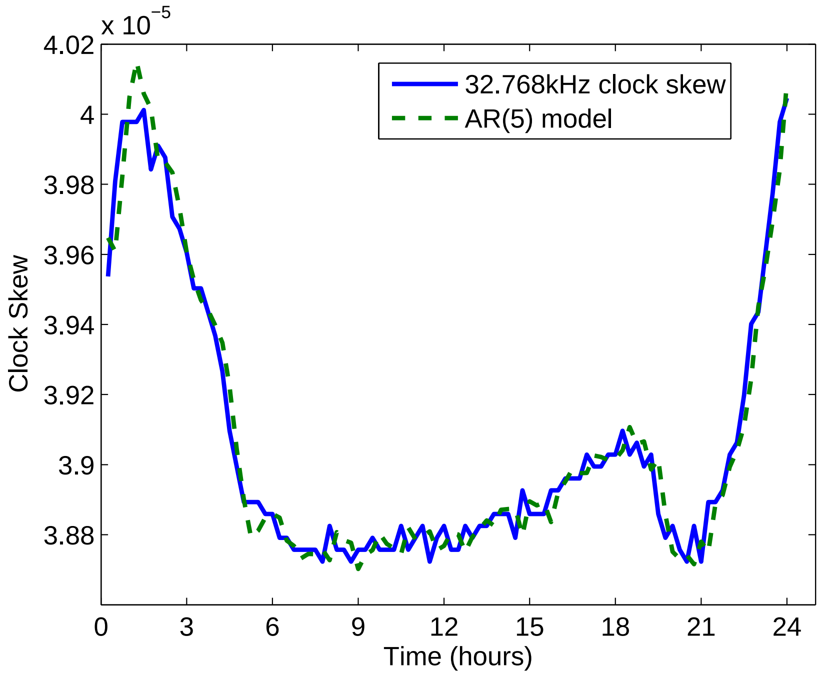
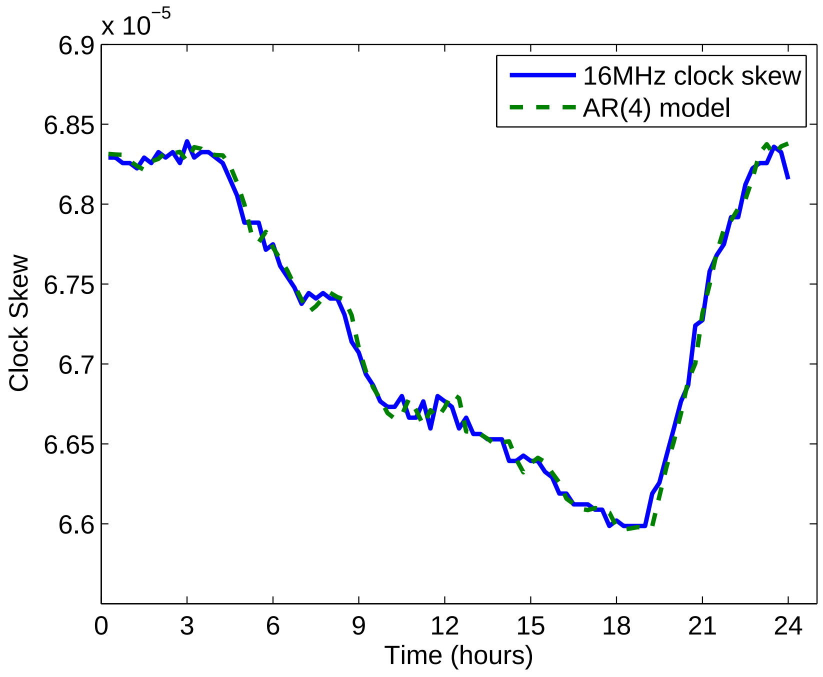


Figure 5: Model order selection for 16 MHz clock with one-day measurement.

Figure 6 illustrates that the derived AR processes match each clock behavior with a small order. As an optimal order, $P = 4$ is employed as the model order for the 16 MHz clock (see Figure 6b), while $P = 5$ is chosen for the 32.768 kHz clock (see Figure 6a). Note that the chosen order as the model order of the 32.768 kHz clock is not an optimum from information criteria. Since clock models in resource-limited networks need to be as simple as possible, the order at which a downward tendency in the information criteria curves decreases was chosen. Numerical simulation shows that the clock model follows the clock skew closely for the given order. Therefore, in the following, the model order can be limited to five or less to keep the complexity low.



(a) The derived model of 32.768 kHz clock skew.



(b) The derived model of 16 MHz clock skew.

Figure 6: An AR model and the measurement of clock skew. The models follow the measurements very closely, which verifies the proposed clock skew modeling process.

CHAPTER IV

TRACKING CLOCK OFFSET AND SKEW USING KALMAN FILTERING

Most clock synchronization techniques assume that clock skew is constant; however, clocks drift at varying rates. This is because of the impacts of environmental changes as well as the inherent instability of oscillators. This chapter introduces a clock estimation and synchronization technique that works on low-precision oscillators with time-varying drift rates.

4.1 Design of a Kalman Filter for Tracking Clocks

Kalman filtering is used for clock synchronization [11, 31, 28]. Here, a Kalman filter is designed to track the clock uncertainty. In fact, most of the prior protocols fail when the clock has time-varying drifts. However, as incorporating the clock models introduced in Chapter 3, a Kalman filter enhances synchronization accuracy. Based on the measurement results, clock drifting rate, clock skew, was modeled as an AR process. The optimal parameters could be found by training the measured data.

Suppose that the sampling rate is fixed, i.e., uniform sampling with $\tau[n] = \tau_0$. Let $\theta[n]$ denote the true clock offset (i.e., $\theta[n] = \sum_{k=1}^{n} \alpha[k]\tau[k] + \theta_0$). An extended state equation is defined as

$$\boldsymbol{x}[n] = \boldsymbol{A}\boldsymbol{x}[n-1] + \boldsymbol{w}[n], \tag{26}$$

where $\boldsymbol{A}$ is a $(P+1) \times (P+1)$ transition matrix, and $\boldsymbol{w}[n]$ is a $(P+1) \times 1$ driving noise vector. Each matrix and vector can be represented as

$$\boldsymbol{x}[n] = \begin{bmatrix} \theta[n] \\ \alpha[n] \\ \vdots \\ \alpha[n-P+1] \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 1 & \tau_0 & 0 & \dots & 0 \\ 0 & \hat{c}_1 & \hat{c}_2 & \dots & \hat{c}_P \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \quad \boldsymbol{w}[n] = \begin{bmatrix} 0 \\ \eta[n] \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $\hat{c}_i$'s are the estimates of AR coefficients, and $\eta[n]$ is the modeling noise.

The observation equation is defined as

$$\tilde{\theta}[n] = \theta[n] + v[n] = \boldsymbol{b}^T \boldsymbol{x}[n] + v[n], \tag{27}$$

where $\tilde{\theta}[n]$ is an observation of clock offset, $\boldsymbol{b}^T = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$, which is a $1 \times (P+1)$ vector, and $v[n]$ is observation noise.

With these conditions, the Kalman filter design is summarized as follows (cf. [45, Chapter. 13]).

$$\text{Update}: \hat{\boldsymbol{x}}[n] = \boldsymbol{A}\hat{\boldsymbol{x}}[n-1] + \boldsymbol{G}[n](\tilde{\theta}[n] - \boldsymbol{b}^T \boldsymbol{A}\hat{\boldsymbol{x}}[n-1]) \tag{28}$$

$$\text{MSE}: \boldsymbol{\Sigma}[n] = \boldsymbol{A}\boldsymbol{M}[n-1]\boldsymbol{A}^T + \boldsymbol{C}_w \tag{29}$$

$$\boldsymbol{M}[n] = (\boldsymbol{I} - \boldsymbol{G}[n]\boldsymbol{b}^T)\boldsymbol{\Sigma}[n] \tag{30}$$

$$\text{Kalman Gain}: \boldsymbol{G}[n] = \boldsymbol{\Sigma}[n]\boldsymbol{b}\left(\sigma_v^2 + \boldsymbol{b}^T \boldsymbol{\Sigma}[n]\boldsymbol{b}\right)^{-1}, \tag{31}$$

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 n^{th} 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 $\boldsymbol{C}_w$ is the covariance matrix of $\boldsymbol{w}[n]$when σ_η^2 is the variance of the driving noise in the state equation. The recursion of the Kalman filter is initialized by

$$\hat{\boldsymbol{x}}[0] = \begin{bmatrix} \mathrm{E}\{\theta[n]\} \\ \mathrm{E}\{\alpha[n]\}\mathbf{1}_{P\times 1} \end{bmatrix}, \quad \boldsymbol{M}[0] = \begin{bmatrix} \sigma_v^2 & \mathbf{0}_{1\times P} \\ \mathbf{0}_{P\times 1} & \sigma_\alpha^2 \boldsymbol{I}_P \end{bmatrix},$$

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

The introduced Kalman filter-based clock synchronization predicts the clock offset and skew of a local clock for a new incoming clock measurement of a reference. Clock measurements can be transmitted from a reference node through wireless links in a network. Therefore, the clock measurements are prone to be distorted by link failure, and the clock prediction would be affected by this distortion.

4.1.1 Clock Skew and Offset Tracking

A Kalman filter approach will search the behavior of a clock oscillator. Here, the AR(5) model derived by training the measurements of a single day of the 32.768 kHz clock (Chapter 3) is integrated into the Kalman filter. Suppose that the sampling period τ_0 is fixed as 900 seconds. The derived AR model is used as a clock skew model. The measured clock offset and skew are defined as the true clock offset ($\theta[n]$) and the true clock skew ($\alpha[n]$) since measurement delay is less than one granularity of the clock, which is 1/32768 second. In other words, based on the experimental results, it is inferred that uncertainties in a wired connection hardly disturbed clock information.

The state equation is defined as (26), where

$$\boldsymbol{x}[n] = \begin{bmatrix} \theta[n] \\ \alpha[n] \\ \alpha[n-1] \\ \alpha[n-2] \\ \alpha[n-3] \\ \alpha[n-4] \end{bmatrix}, \ \boldsymbol{A} = \begin{bmatrix} 1 & \tau_0 & 0 & 0 & 0 & 0 \\ 0 & \hat{c}_1 & \hat{c}_2 & \hat{c}_3 & \hat{c}_4 & \hat{c}_5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \ \boldsymbol{w}[n] = \begin{bmatrix} 0 \\ \eta[n] \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The estimates of the AR coefficients ($\hat{c}_i$'s) obtained by a training-based approach (20) are $\begin{bmatrix} 0.9271 & 0.4163 & 0.07483 & -0.387 & -0.03118 \end{bmatrix}$. The observation equation is defined as (27), where $\boldsymbol{b}^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, $\tilde{\theta}[n]$ is the observation of the clock offset, and $v[n]$ is observation noise that represents any kind of uncertainties, including network delay and measurement errors, which can be added to the true data when the clock offset is observed.

With these conditions, a Kalman filter is designed as (28) - (31). $\eta[n]$ is the modeling error whose variance is 3.91502×10^{-15}. The variance is calculated by using the measured samples. The recursion of the Kalman filter is initialized by

$$\hat{\boldsymbol{x}}[0] = \begin{bmatrix} \theta[0] \\ \alpha[0]\mathbf{1}_{5\times1} \end{bmatrix}, \ \boldsymbol{M}[0] = \begin{bmatrix} \sigma_v^2 & \mathbf{0}_{1\times5} \\ \mathbf{0}_{5\times1} & \sigma_\alpha^2 \boldsymbol{I}_5 \end{bmatrix},$$

where σ_v^2 is the variance of the observation noise of clock offset which is additional perturbation, and σ_α^2 is the variance of clock skew, $\alpha[n]$, which is 1.29446×10^{-13}.

Figure 7 shows the performance of the Kalman filter to estimate clock skew from noisy offset observations. Even when the signal-to-noise ratio (SNR) of clock offset was set to -20 dB by setting σ_v to 3×10^{-4} seconds, the Kalman filter was able to estimate the true skew quite closely. This is because the derived skew model reflects the characteristics of the real clock measurement.

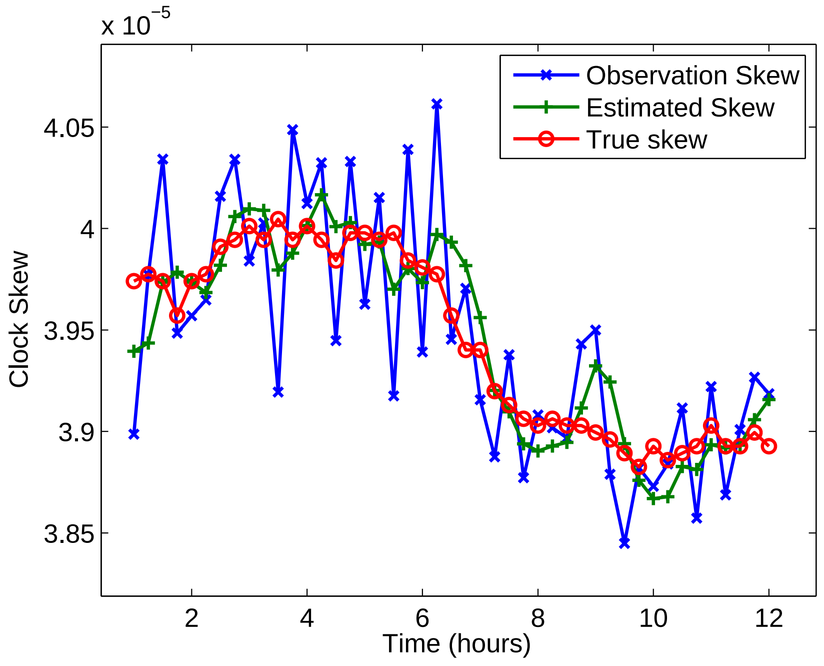


Figure 7: Tracking clock skew using the Kalman filter.

4.1.2 Performance of Clock Tracking

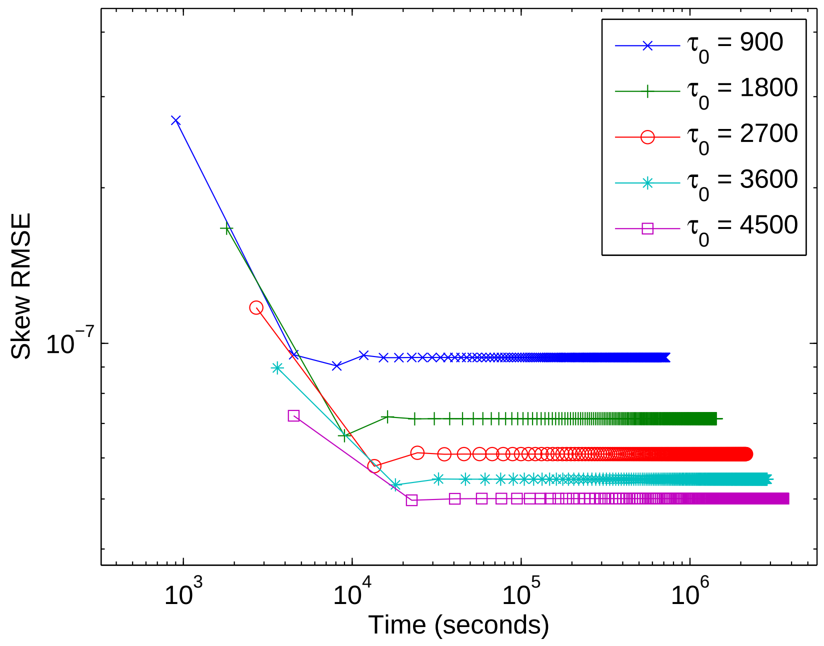
More simulations validate the Kalman filtering method. The root mean square error (RMSE) of the prediction MSE ($\mathbf{\Sigma}[n]$) is used as a performance metric for evaluation. Since clock synchronization consumes resources such as power, bandwidth, and processing time, many applications need to trade clock estimation accuracy for low resource consumption. If a re-synchronization rate is high, the accuracy would be improved, but consume more energy and resource. Therefore, it is important to determine synchronization periods when a synchronization algorithm is designed.

The performance of the proposed clock tracking is analyzed by looking into the effects of various sampling rates. Figure 8 depicts the performance of estimating clock skew and clock offset as synchronization periods vary. The simulation was run for 800 synchronization periods with several different sampling periods. The curves are drawn by the prediction RMSEs of skew and offset every four samples. The RMSE of the skew in Figure 8a becomes smaller when the sampling rate is lower. This is because a longer sampling period cannot capture the small and fast variations of the skew; thus the modeling error is smaller. This can also be explained by the calculation of skew from the offset observations as $\alpha[n] = \frac{\tilde{\theta}[n]-\tilde{\theta}[n-1]}{\tau_0}$. Since each offset observation suffers from noise, $v[n]$, when τ_0 is large, the noise variance in the observed clock skew is reduced to $\frac{\sigma_v^2}{\tau_0^2}$. For the offset in Figure 8b, the RMSEs change little around $10^{-3.6}$ seconds even as for the different sampling periods. This means the performance of tracking the clock offset is not significantly affected by the sampling period.

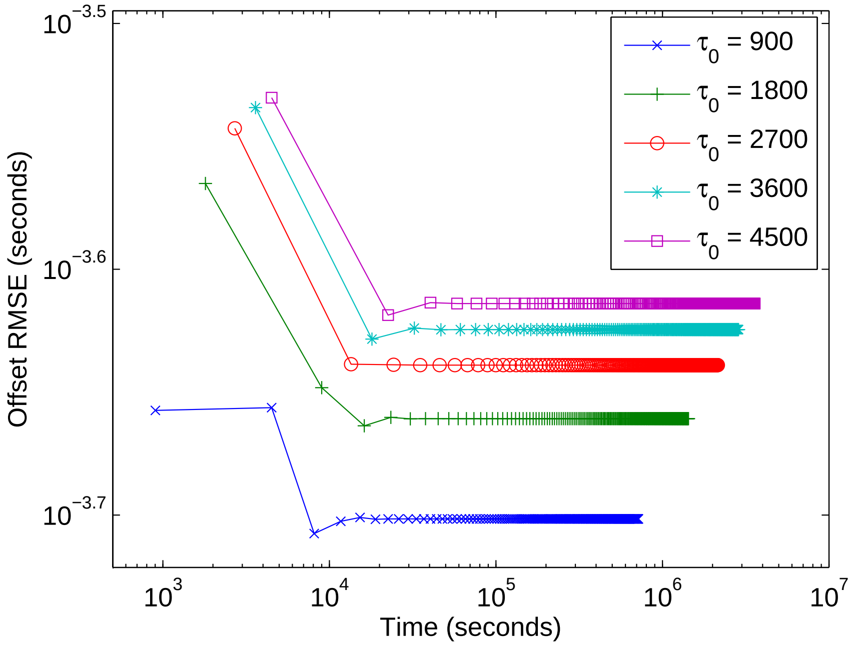
Both the RMSEs of clock skew and offset in Figure 8 converge to a steady state after several samples. As the sampling period τ_0 increases, the prediction RMSE approaches its steady-state value in fewer samples. However, since a greater τ_0 implies a larger sampling period, the actual convergence time is longer. Note the performance shown is only at sampling instants, when new data have just arrived, rather than at arbitrary time instants. This performance will therefore be better than the true time-averaged performance.

Figure 9 further clarifies the relationship between sampling rates and clock tracking errors by showing how the steady state RMSE varies with the sampling period (τ_0). The simulation was run 200 times, and an average for each sampling point was obtained. The skew RMSE decreased as a sampling period increased, while the offset RMSE remained nearly the same. The observation noise variance, rather than the sampling period, more strongly influenced both the clock offset and the skew. It was shown that the modeling and tracking method was quite robust for different sampling periods. Here, a sampling rate means how often the nodes measure local clocks and exchange the clocks for synchronizing between them. Therefore, the tracking method, which is less harmed for a long rate may accomplish network synchronization by using less hardware resource usage and energy consumption.

As illustrated in Figure 10, the proposed clock tracking algorithm performs regarding different observation noise variances and converges after several samples even

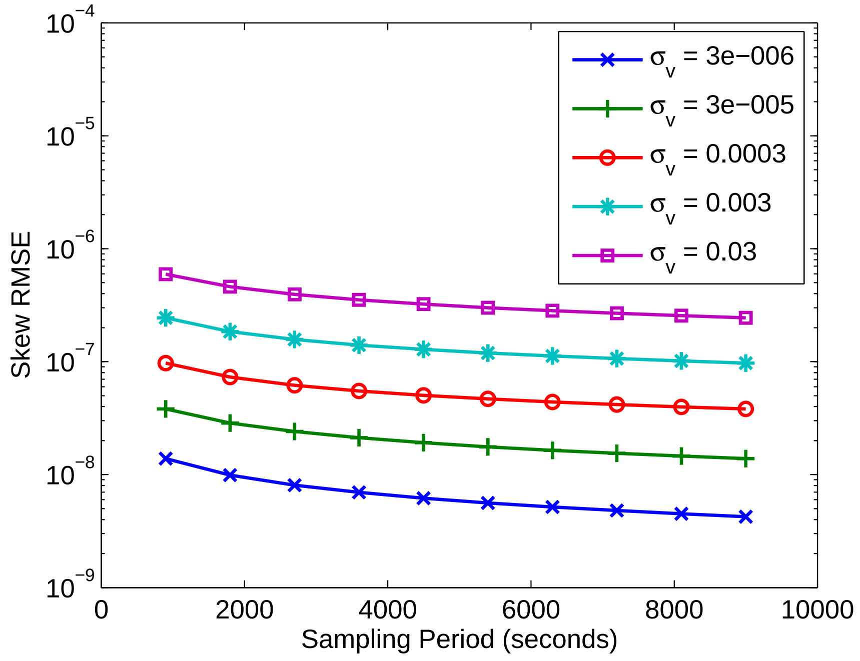


(a)

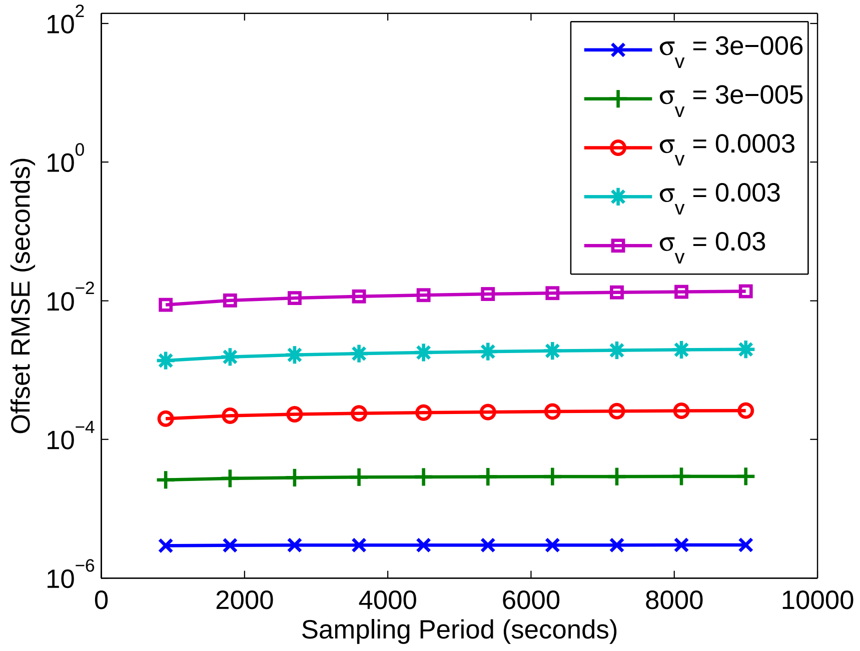


(b)

Figure 8: RMSEs for various sampling rates.



(a)



(b)

Figure 9: Steady state RMSE vs. sampling period.

when the observation noise is large. To obtain the RMSE curves, the tracking algorithm was run 800 synchronization periods with several different noise variances. The RMSE of the estimated clock skew, $\hat{\alpha}[n]$, is less than 10^{-6} even when σ_v is as high as 0.03 seconds is shown in Figure 10a; the RMSE of the estimated clock offset, $\hat{\theta}[n]$, is less than 10^{-2} seconds under the same condition, as shown in Figure 10b. Even if the variance of observation noise is extremely high, the prediction RMSE of the Kalman filter curves converges in several samples.

For the further evaluation of the clock models, two other skew models were borrowed here and compared to each other. The simulation was repeated 200 times for 4000 synchronization periods with $\tau_0 = 900$ seconds and $\sigma_v = 3 \times 10^{-4}$ seconds. The RMSEs used as performance metrics for evaluation were defined as

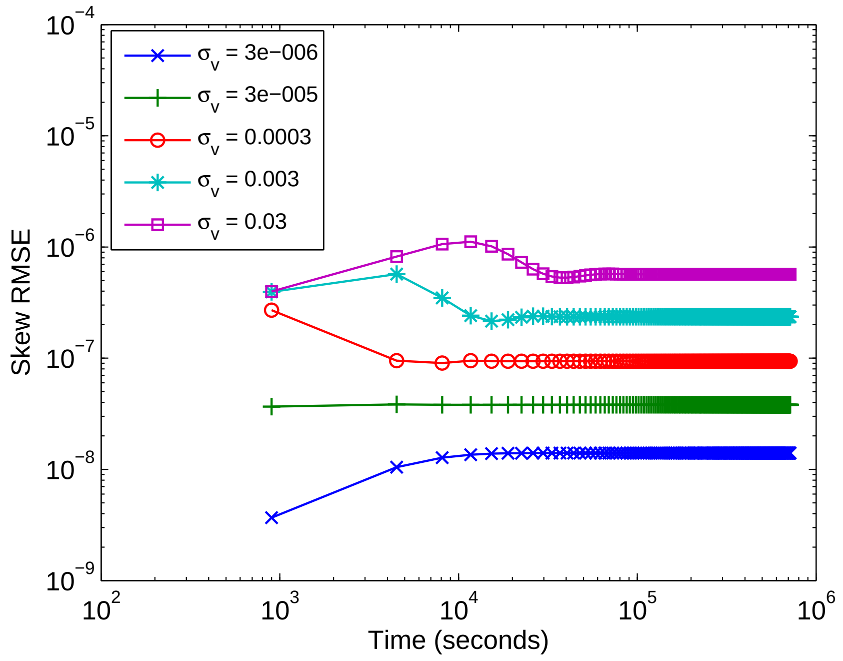
$$\begin{aligned} \text{RMSE}_{\text{offset}}[n] &= \sqrt{\frac{\sum_{i=1}^{n}(\hat{\theta}[i] - \theta[i])^2}{n}}, \\ \text{RMSE}_{\text{skew}}[n] &= \sqrt{\frac{\sum_{i=1}^{n}(\hat{\alpha}[i] - \alpha[i])^2}{n}}. \end{aligned} \tag{32}$$

For the comparison, an AR(1) model was derived as a clock skew model, while the recursive model in (13) was used as a clock offset model. As shown in Table 1, both the skew RMSE and the offset RMSE with the AR(1) skew model are higher than those with the AR(5) skew model. This shows that the clock skew model with the model order chosen by information criteria enhances the performance of the Kalman filter-based method.

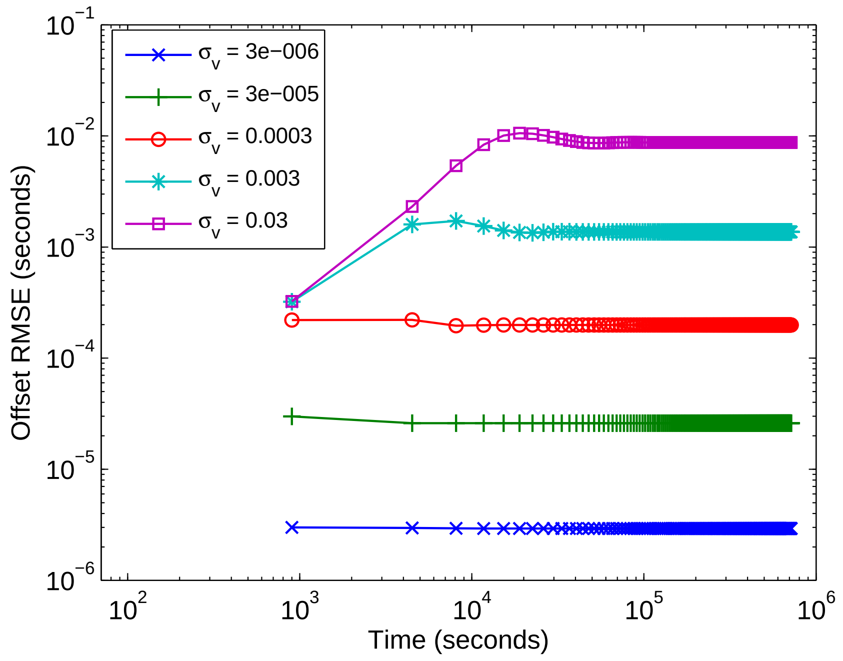
Table 1: The comparison of tracking errors for different model orders.

	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}

Another skew model for performance comparison is the constant skew model in (16). When the clock skew is constant, the clock offset represents a linear model.



(a)



(b)

Figure 10: RMSEs for various observation noise variances.

The RMSEs shown in Table 1 were calculated by using 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 highly affected and is 10^2 times worse than those of the AR(1) and AR(5) models. This shows the importance of using time-varying models for clock skew. Observation noise was not added for the simulations of a constant skew model while randomly generating the noise with $\sigma_v = 3 \times 10^{-4}$ seconds for the simulations of the AR(1) and AR(5). Therefore, it can be strongly inferred that a constant skew model will perform even worse if observation noise is added. Thus, the constant skew model is not suitable for real clocks with varying skew. Moreover, a higher-order skew model aids to better fit the clock measurements.

4.2 Clock Tracking over Unreliable Networks

In sensor networks, the links are unreliable and prone to interference. Packets containing timestamps may get lost or suffer from collisions. To handle the message losses, traditional wired networks simply 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 [96]. Additionally, retransmitted timestamps suffer from increased and variable delay. In this case, the received timestamps may not be uniform or may be impaired by network uncertainty. Here it is investigated how to track clocks when transmitted packets are lost or corrupted.

4.2.1 Tracking Clocks with Missing Data

The tracking performance was evaluated with three types of missing data: (i) uniform loss, (ii) consecutive loss, and (iii) random loss. Again a Kalman filter is adopted with the derived model. When an observation is missing, the Kalman filtering method does not update estimates of clock skew and clock offset. Instead, the predicted estimates take the absence of observations to find the new estimates of clock skew and offset.

Figure 11 presents the case that one out of every five observations is dropped and indicates the points of missing data by vertical lines. For missing data, the observed clock skew is not updated, however, the estimated clock skew closely tracks the true clock skew based on the AR model derived. This supports that the AR model captures the behavior of the clock skew well.

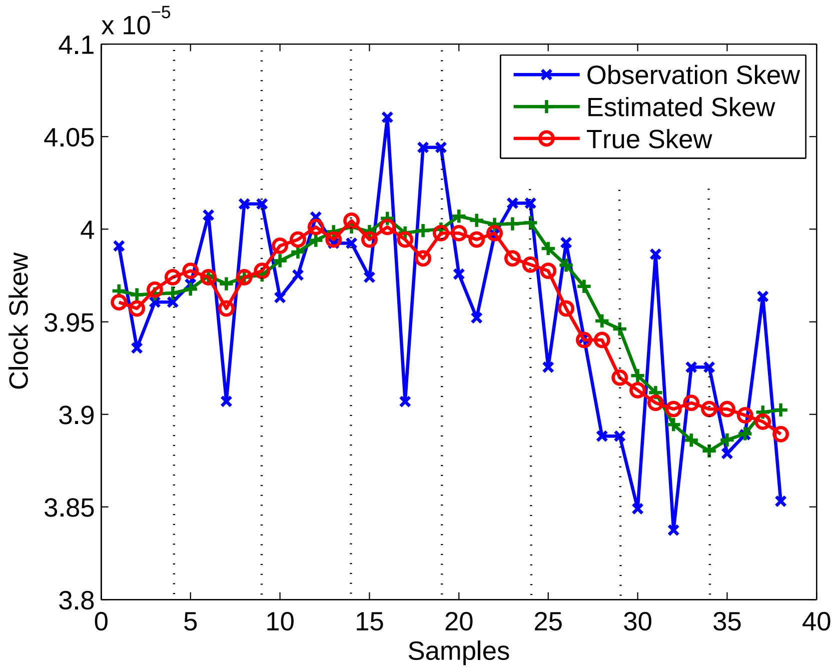


Figure 11: Uniform loss of data.

Next, let us suppose packets do not arrive at the destination node for more than a few synchronization periods because of mechanical problems, or channel congestion. Figure 12 shows an example of tracking clock skew when six consecutive measurements are missing. During each missing period, clock skew and offset are tracked only with estimates based on the clock models, which are already established. The estimated clock skew is close to the true clock skew even for consecutive missing observations.

However, the gap between the estimate of clock skew and the true clock skew becomes larger as more samples fail to be delivered. This means the approach tracks an unstable clock when a system drops clock information for a short while; however, if packets that include clock information are missed continuously for a longer period, the estimated clock deviates from the true clock. Because an AR model is a statistical forecasting model in which future values are computed only on the basis of past values of time series data, the clock-tracking method can not reach adequate estimates without valid previous data. Therefore, the tracking performance may decrease significantly when packet loss sequentially occurs for a long period.

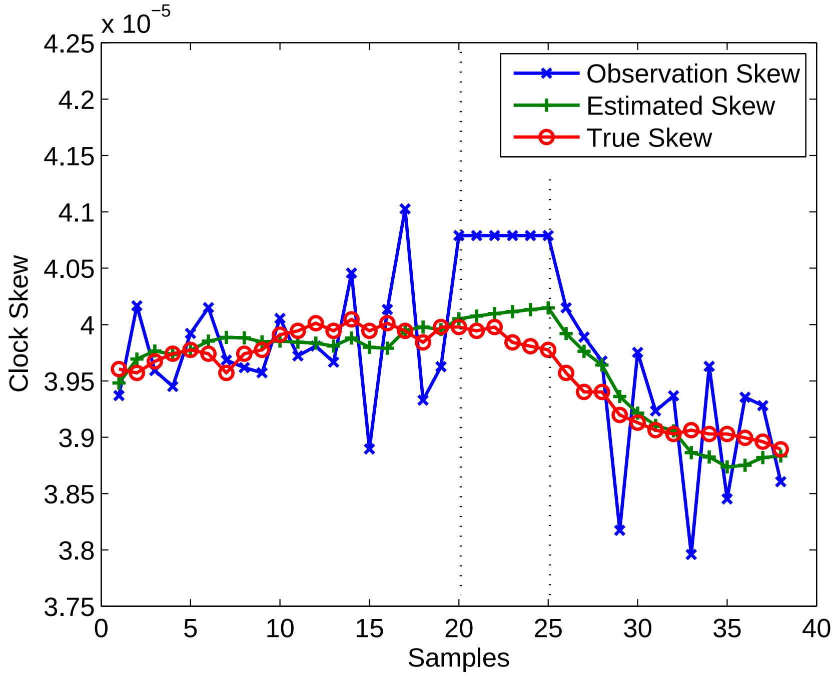


Figure 12: Consecutive loss of data.

For randomly missing data, the RMSEs in (32) for various missing rates are compared at each incoming sample. Here every packet is assumed to have the same probability of being 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 seconds and the standard deviation of observation noise of clock offset was set to 3×10^{-4} seconds. In the given conditions, two methods can be considered; the first method is a *with-tracking* method, which was applied in the two previous missing cases. This method estimates clock skew and offset using the clock models if observations are not available. The other one is a *without-tracking* method, which triggers the clock-tracking only when a new packet arrives.

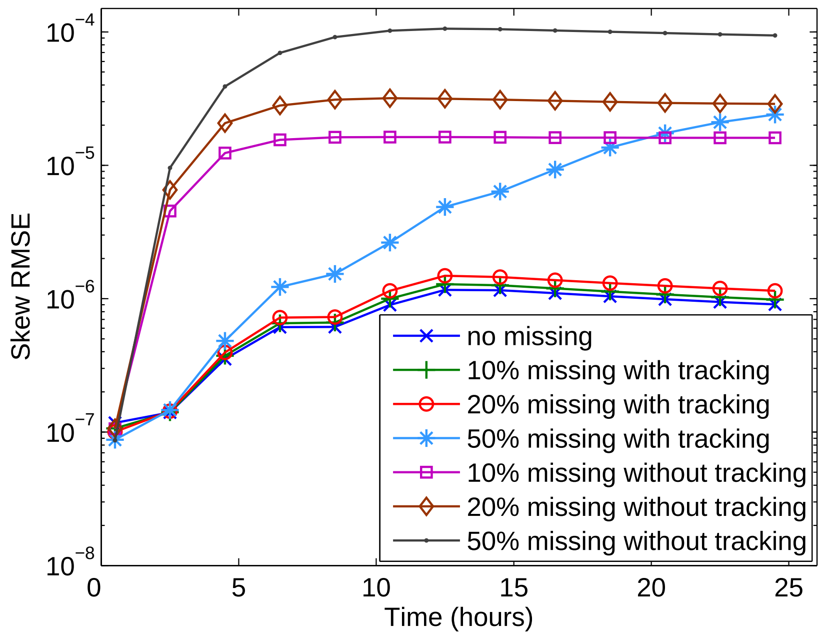
Figure 13 contrasts the RMSEs of the two methods with different packet loss rates (10%, 20%, and 50%). Both the methods encounter performance degradation when compared to the RMSE curves that do not have missing observations; however, the *with-tracking* method is more robust for handling missing data compared with the *without-tracking* method. The performance of the *without-tracking* method is significantly degraded. When the loss rate is 20%, both RMSEs of the *without-tracking* method are ten times higher than that of the *with-tracking* method for the given duration. When 50% of the observations are missing, the Kalman filter that uses the *with-tracking* method may not converge; thus, the performance is poor.

Two observations are observed: (i) that as packet loss rate increases, the tracking performance gets worse, but the clock-tracking method is fairly robust regarding missing data; and (ii) that the performance of the *with-tracking* always outperforms the *without-tracking*. The comparison of the two methods strongly supports that the *with-tracking* method is powerful in unstable networks.

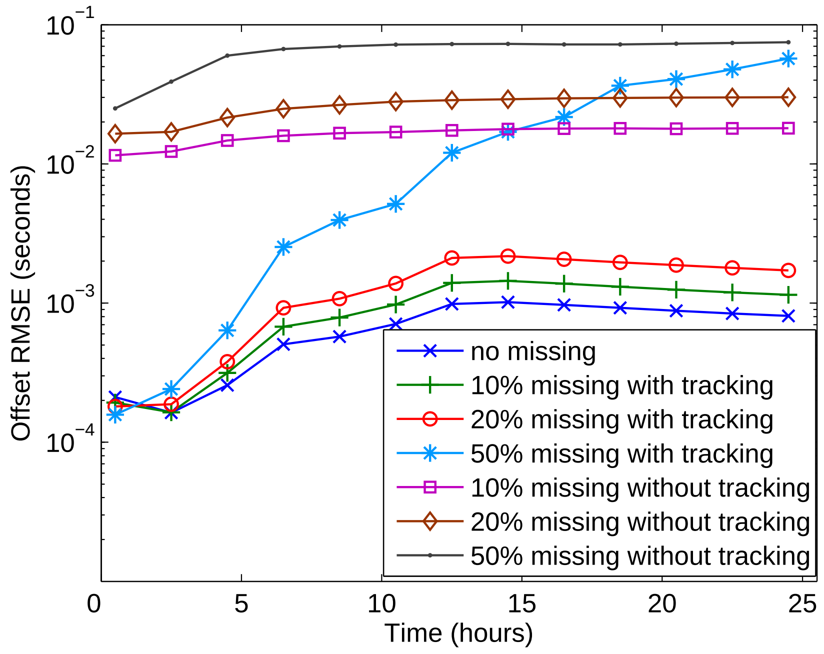
4.2.2 Tracking Clocks with Dirty Data

In wireless networks, received data can be corrupted by other data packets, channel noise, or jamming. To differentiate the case of the missing data, the corrupted data are called dirty data. In this case, the proposed Kalman filtering method alone can not perform effectively since it can not distinguish dirty data from normal data. However, dirty data detection methods (e.g. LASSO method) can be adopted and combined with Kalman filtering. Considering that the observed clock offset could include a dirty component, the observation equation is redefined as

$$\tilde{\theta}[n] = \theta[n] + v[n] + \xi[n], \tag{33}$$



(a)



(b)

Figure 13: RMSEs with random loss of data.

where $v[n]$ is Gaussian observation noise, which usually has little variance, and $\xi[n]$ is a dirty component that is zero for most samples, but can randomly become very large.

- **Threshold-based method**

  Dirty data can be detected by a threshold-based method. The removal of dirty data processes is as follows:

  (i) compute observation noise; $\hat{v}[n] = \tilde{\theta}[n] - \hat{\theta}[n]$

  (ii) compute $\frac{|\hat{v}[n]|}{\sigma_v}$ and compare this with a pre-defined threshold; an observation is determined to be dirty when $\frac{|\hat{v}[n]|}{\sigma_v}$ is greater than the threshold.

  (iii) replace $\tilde{\theta}[n]$ with $\hat{\theta}[n]$ when the observation is dirty.

  (iv) repeat (i)-(iii) every synchronization period.

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

- **LASSO (Least absolute shrinkage and selection operator)**

  To identify rarely occurring dirty data, the LASSO method can be employed as adding a regularization term to a standard least squares problem [98]. Note that to fit in the tracking algorithm and also keep the complexity low, a scalar LASSO algorithm is proposed for each observation instead of performing LASSO for vector observations. Assuming that $\psi[n]$ represents observation noise and a dirty part, i.e. $\psi[n] = v[n] + \xi[n]$, the problem is formulated as a constrained ℓ_0 norm minimization problem [110]:

$$\arg\min_{\psi[n]} \left(\left\| \hat{\theta}[n] + \psi[n] - \tilde{\theta}[n] \right\|_2^2 + \lambda \left\| \psi[n] \right\|_0 \right), \tag{34}$$

  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

$$\hat{\psi}[n] = \arg\min_{\psi[n]} \left(\left\| \hat{\theta}[n] + \psi[n] - \tilde{\theta}[n] \right\|_2^2 + \lambda \left\| \psi[n] \right\|_1 \right), \tag{35}$$

where $\|\cdot\|_1$ is the ℓ_1 norm.

Based on a scalar observation, the solution of (35) is expressed in a soft-threshold version of the least squares estimate, as explained in [26, 113]:

$$\hat{\psi}[n] = sign(\tilde{\theta}[n] - \hat{\theta}[n]) \left[\left| \tilde{\theta}[n] - \hat{\theta}[n] \right| - \frac{\lambda}{2} \right]_+, \tag{36}$$

where 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 (35) is a 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. Here λ is chosen based on some numerical tests. If the estimate $\hat{\psi}[n]$ by LASSO is not zero, this observation is assumed to have a non-zero dirty component and is categorized as dirty data. Once an observation is categorized as dirty, it is removed and treated as missing data.

LASSO can be adopted without any knowledge of the noise variance to detect dirty data. The detection process is applied to every observation; in this way, it is possible to determine if an incoming sample is corrupted for real-time synchronization. Combined with LASSO, the tracking method avoids large errors for estimating clock behavior and becomes more robust regarding network uncertainties.

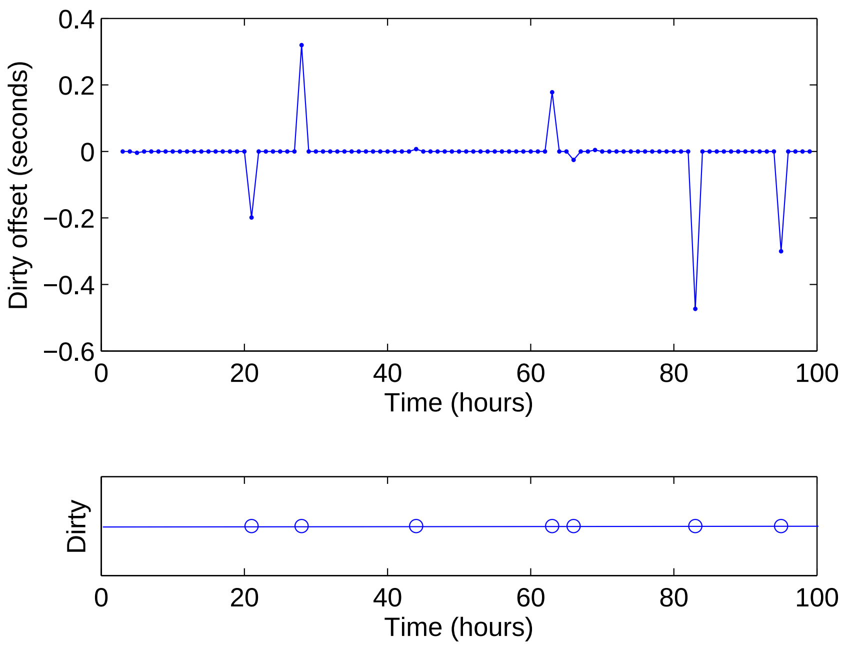
In the simulation, the dirty components were injected at random time instances. The amplitude of the dirty component was assumed to be Gaussian distributed with a mean of zero and a variance of one. The simulation was repeated over 500 runs for 100 synchronization periods, and the outputs were averaged when a synchronization period was 900 seconds, and σ_v was 3×10^{-4} seconds. When dirty data were generated, as in Figure 14a, the clock-tracking method that used LASSO detected the position of dirty data exactly (λ=0.01). The method discarded the possible dirty data and tracked the clock skew and offset by replacing them with estimates based on the clock models. Figure 14b demonstrates that the estimated skew tracked the true clock skew well in spite of the large values of the dirty components.

Figure 15 presents the tracking performance when every packet has a 5% or 1% probability of being dirty. The upper two curves are the RMSEs when dirty observations are not detected, the next two curves are those when dirty observations are detected by LASSO and removed, and the last curve represents the RMSEs when dirty observations do not exist as a reference. Without the removal of dirty data, the performance becomes much worse since the Kalman filter utilizes dirty data, which are incorrect clock information, as an observation. However, the proposed approach detects dirty data and prevents tracking performance from decreasing. This example shows that the clock-tracking method that uses LASSO is robust regarding dirty data.

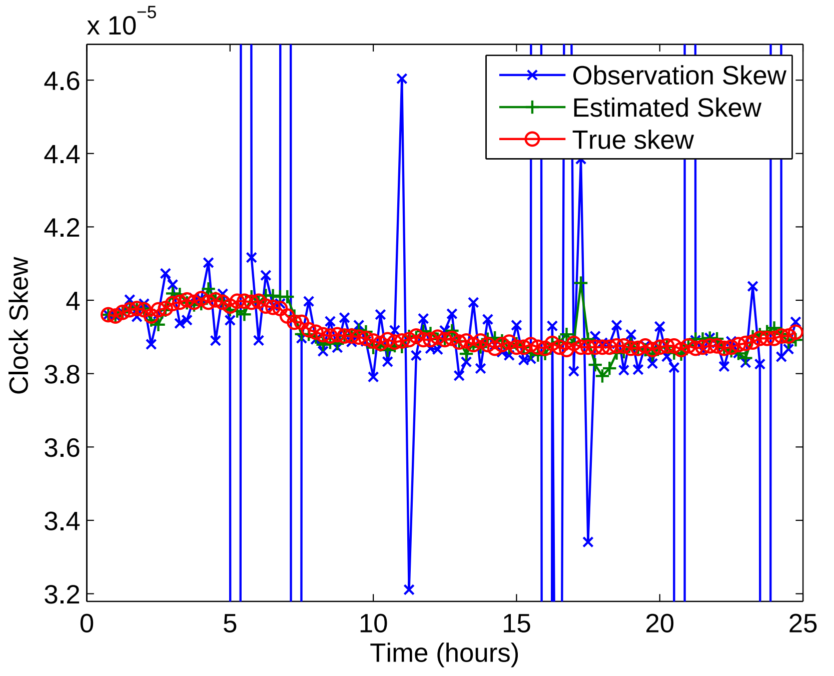
The clock model and the tracking method are evaluated by considering missing or corrupt observations, which may occur in unreliable links. Not only is the Kalman filter-based clock tracking method robust regarding missing data, but it also integrates well with least absolute shrinkage and selection operator (LASSO) to detect the corrupt data.

4.3 Clock Tracking with Periodic Training

Several simulations evaluated the effect of clock models on tracking performances. First, the length of training data was adjusted to 12 hours (48 samples), 24 hours (96 samples) or 72 hours (288 samples) and the tracking performance for each case was compared. The normalized MSEs (NMSEs) in (37) used as a performance metric reduce the influence of original data values. It prevents abnormal peaks from raising

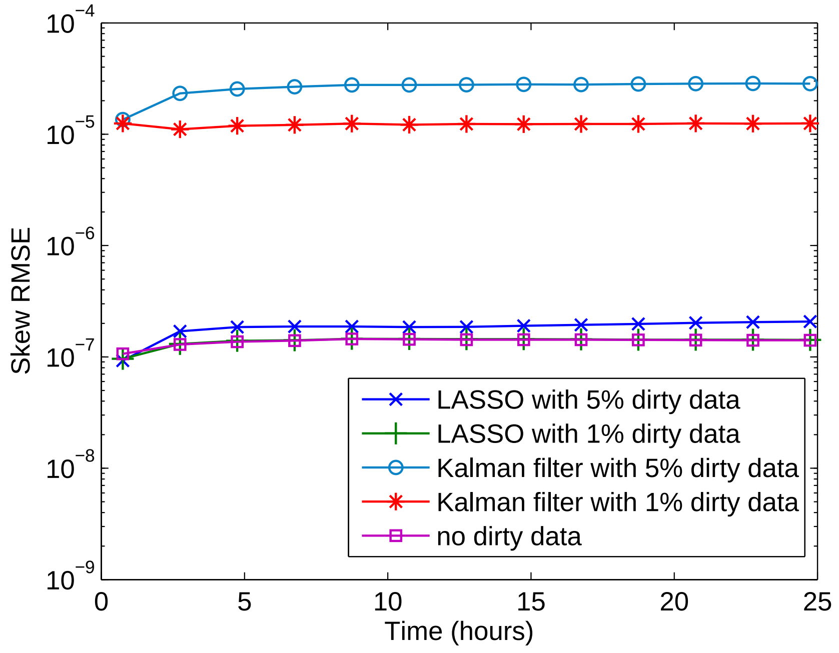


(a) Estimated dirty data position

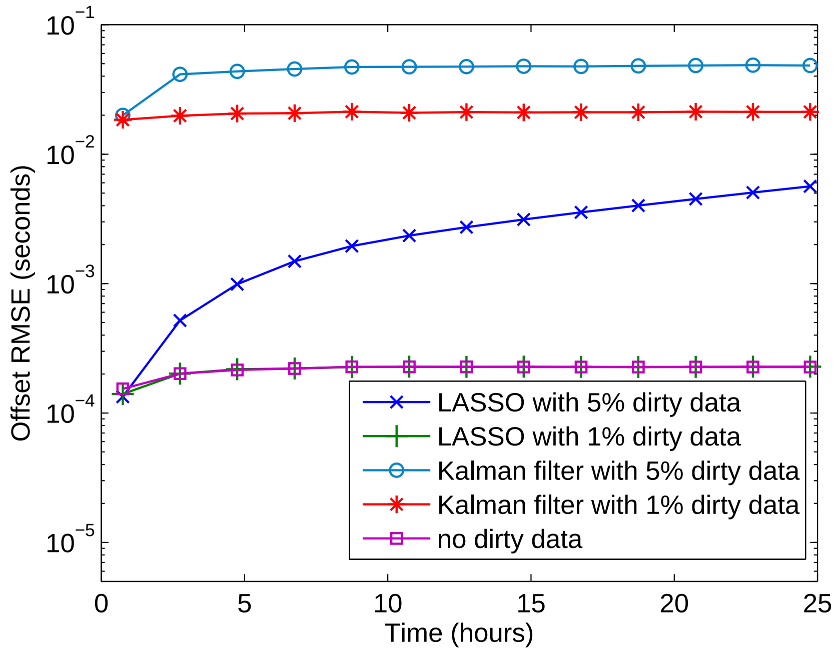


(b) Tracking of clock skew

Figure 14: Tracking clock skew when dirty observations exist.



(a)



(b)

Figure 15: The RMSEs of clock skew and offset when dirty data exist.

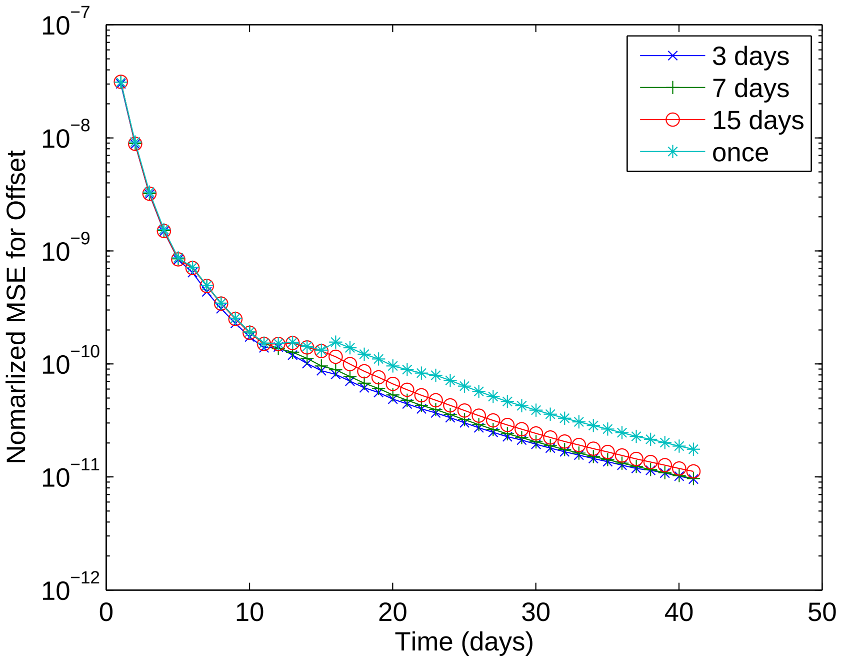
mean-square error extremely high. Through simulations, it is discovered that each training-length results in almost the same NMSE for both clock skew and offset; that is, the length of training data does not significantly affect tracking performance. The fairly short length of training data is enough to model time-varying clock skew.

$$\begin{aligned} \text{NMSE}_{offset}[n] &= \frac{1}{n}\sum_{i=1}^{n}(\hat{\theta}[i]-\theta[i])^2 / \frac{1}{n}\sum_{i=1}^{n}\theta[i]^2, \\ \text{NMSE}_{skew}[n] &= \frac{1}{n}\sum_{i=1}^{n}(\hat{\alpha}[i]-\alpha[i])^2 / \frac{1}{n}\sum_{i=1}^{n}\alpha[i]^2. \end{aligned} \tag{37}$$

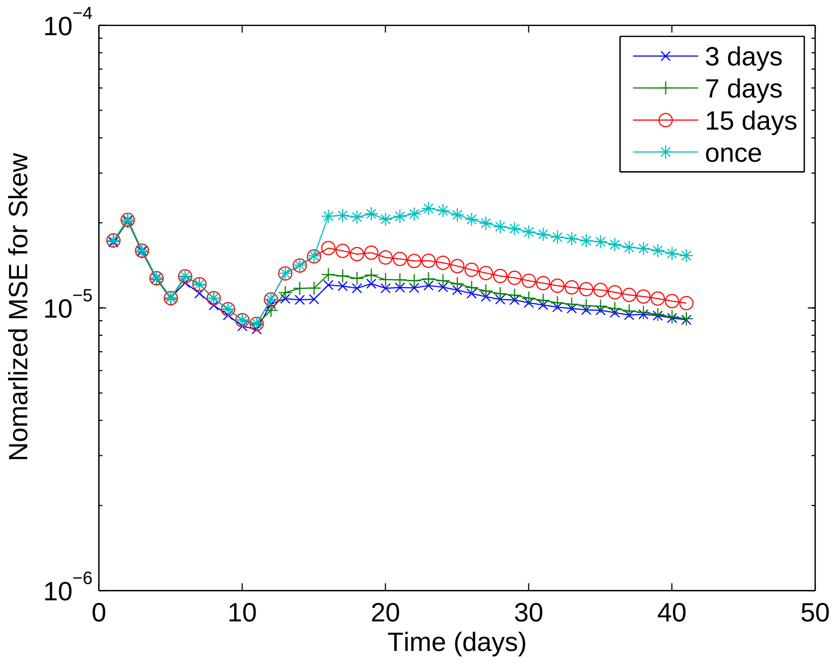
Next, the clock skew model was periodically updated through re-training of clock data. As time went on, clock behavior tended to change as a result of temperature, humidity variation, and lack of battery. If clock skew changed severely, the clock model could no longer follow the skew. Re-training clock data periodically provided an updated model of time-varying clock skew. Here, the re-training periods were every three days, seven days and 15 days. The simulation was iterated 50 times for 4000 synchronization periods (τ_0 = 900 seconds), and the iterations were averaged. For each run, random noise was added to the measurements. For every re-training, 24-hours data were used to obtain an undated clock model.

As shown in Figure 16, the frequent re-training raised the fitness of the model to real clock skew and resulted in better performance. Since the clock skew model drifted more around the 12th day, the current AR model did not fit the measurements. From the measurements, it is reasonable to keep updating the clock skew model every seven days since this helps to improve the tracking performance.

Suppose any congestion or interference severely distorts or loses clock information. In this case, instead of uniformly sampled data, one may have non-uniform and only incomplete observations. The proposed clock synchronization technique models the clock skews when every packet has a certain probability of being lost during training, and tracks clock offset and skew with these models. Table 2 presents the NMSEs with different packet loss rates (0%, 10%, and 30%). As the packet loss rate increases, the



(a)



(b)

Figure 16: Normalized MSE with Various Periodic Training

performance worsens marginally. This shows that the modeling method performs even when some packets are lost during a training period, and the tracking algorithm is quite robust regarding packet loss.

Table 2: The comparison of tracking performance based on clock models derived with missing observations

	Normalized Skew MSE	Normalized Offset MSE
No missing	7.8254×10^{-6}	7.2520×10^{-12}
10% missing	8.4727×10^{-6}	7.2930×10^{-12}
30% missing	9.8558×10^{-6}	8.0709×10^{-12}

CHAPTER V

TEMPERATURE EFFECTS ON CLOCK DISCREPANCY IN WIRELESS NETWORKS

Since a single oscillator is vulnerable to surroundings such as climate changes, and aging, it is reasonable to expect that clock inconsistency between nodes pertains to these changes. One challenge is that these environmental changes and other factors influence each oscillator in different ways. Some oscillators endure these changes while others are easily impaired to them. Therefore, employing clock synchronization techniques is indispensable for many applications where variations in time are not acceptable.

Another main source of errors in clock synchronization is uncertainty in wireless links between nodes. During transmission, packets may become corrupted or lost as a result of a collision with other packets, channel noise, or jamming. A packet, which conveys a timestamp, suffers from capricious delay, which consists of send and receive time, processing, and propagation delay. This chapter describes experiments which investigate how these delays distort network synchronization; it also details how much the surroundings degrade synchronization accuracy.

5.1 The Evaluation of Clock Error

5.1.1 Experimental Setup for a Wireless Connection

In the previous experiments (Chapter 3), since a sensor module communicated to a PC via a wired connection, processing time was very short and stable, negligible for each transmission. This chapter investigates clock error from wireless devices, which could suffer from a critical timing path [23]. The experiment consists of two nodes, and a wireless network is established over the nodes without any central units or a stable computer, emulating distributed sensor networks. The experiment employs Pandaboards Rev. A1 [78], which are used as low-power, low-cost, mobile development platforms. A Pandaboard features an OMAP4430 processor with a dual-core 1 GHz ARM Cortex-A9.

The exploration of autonomous oscillators in the platforms could reveal uncertainty caused by jitter, fluctuation in time domain, which is equivalent to phase noise affecting the oscillating frequency. Ubuntu was installed on two Pandaboards and wireless interfaces were enabled. Some modification after installation optimized the system for measuring local clocks. First, for both the devices, NTP [69] adjustments (both *ntpd* and *ntpdate*) were prohibited so that the inherent characteristics of the oscillators could be investigated. Another modification was to disable CPU frequency scaling in order to avoid an offset caused by changing a CPU frequency, which [72] evaluated in detail.

Once the nodes were prepared, they worked in the following way. The two closely-placed nodes establish an IEEE802.11g wireless network. One node becomes an access point (AP) by broadcasting a beacon frame every pre-defined period, which is 200 ms in this case. The AP node runs *hostapd* and *udhcpd*. The other node that listens to the beacon requests to join to the network. Each node opens a *datagram socket*, which does not guarantee successful transmissions. All packets containing timestamps are then transmitted via the sockets using user datagram protocol (UDP) to avoid the overhead of such processing at the network interface level. When the clock reading program is executed on each node, the program is allocated to one CPU by *taskset* to avoid offsets produced by switching CPUs.

After a network is built, one device (sender) disseminates its local time every synchronization period; the other device (receiver) detects the transmitted timestamps and keeps tracking the clock of the sender node. Local time is read by a *clock_gettime()* function. Since a sender does not have to be an AP node, the non-AP node becomes a sender in this experiment. A timestamp is loaded into a payload in the trial; however, it could be piggybacked into a header or a tail in a packet in a practical implementation. The receiver node executes the clock reading function whenever a timestamp arrives at the node.

These transmissions are based on a one-way clock synchronization, as shown in Figure 17. This kind of clock synchronization method can be extended to larger networks. In a larger network, each node that listens to a sender node internally keeps tracking the clock of the sender. As a result, the clock of the sender becomes a reference, and others keep being synchronized to the reference. A packet may suffer from unpredictable processing delay as well as varying propagation delay. Despite the weakness in transmission delay, one-way synchronization has an advantage in resource consumption since it halves the number of transmissions compared to those for two-way synchronization.

5.1.2 Time Difference

The two devices communicate to each other and to transmit timestamps for three weeks. Time differences are calculated by subtracting the collected timestamps as $C_{receiver}[n] - C_{sender}[n]$, where $C_i[n]$ is the n^{th} clock reading at node i. As shown in Figure 18, overall, the time differences seem to be increasing at a monotonic rate.

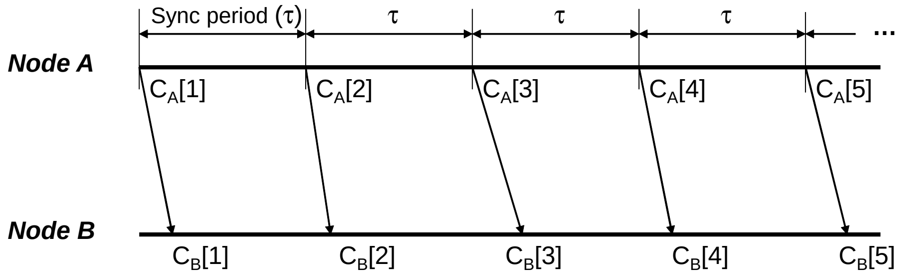


Figure 17: One-way timestamp dissemination. Each transmission suffers from a different, unpredictable delay.

Interestingly, the total increment is more than ten seconds during the period, which corresponds to about 0.5 seconds per day. The sign presents which node has a faster clock. Here, the node that sends timestamps (sender) has a faster time than the node that receives the timestamps (receiver). The initial difference is set to zero to highlight how much it changes during the observed period.

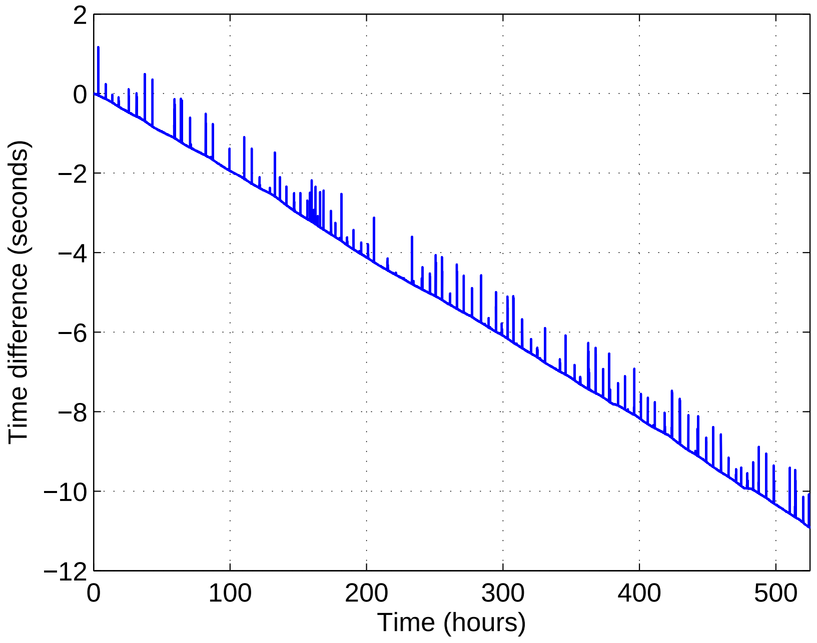


Figure 18: Time differences between two wireless nodes show lots of peaks as well as floating.

Considering that the devices consist of the same hardware components, the time deviation is much more significant than one might expect. It is obvious that two oscillators have different resonant rates. Moreover, the measured results show notably large peaks, which range from less than a microsecond to one second. This phenomenon is unlikely caused by the instability of the oscillators, since the devices were not exposed to severe changes. Instead, the peaks can be explained by unpredictable delay such as nodal processing time and receive/send time while transmitting. The late arrival of a packet because of unreliable channels causes its receiving time to deviate from its sending time. This observation seems to support that the network uncertainties significantly obstruct timestamp-based clock synchronization and that what is required is a new synchronization method that is free from large network uncertainties.

Allan deviation is a metric to gauge the frequency stability of clocks and oscillators by using noise processes, not systematic errors or imperfections of oscillators, as representing the variance of fractional frequencies during observation. A low Allan deviation indicates a clock with good frequency stability. Normally, Allan deviation becomes lower over a longer period since noise is averaged over the time period, but at some point, it increases because of the inherent inaccuracy of a clock. The fractional frequency ($\bar{y}[n]$) is represented as $\frac{x[n+1]-x[n]}{\tau[n]}$, where $x[n]$ is a timestamp. In this case, $x[n] = C_B[n]$, and $\tau[n] = C_A[n] - C_A[n-1]$. In Figure 19, the Allan deviation from the measurement indicates that a longer synchronization period results in better performance during the given period with raw timestamps, which is counter-intuitive. This is due to noise variance rather than oscillator instability. As a time period becomes longer, an absolute time gap (valid data value) increases, while measurement noise and transmission delay do not depend on the period.

Figure 18 and 19 illustrate that the clock captured by timestamps is neither accurate as a result of inherent frequency gap nor stable because of significant non-deterministic delay. It is implied that frequent resynchronization by timestamp exchanges may not be an optimal solution in some wireless networks. This observation leads to the proposal of a clock estimation technique to lower the obstruction of link anomalies in a timestamp-based synchronization.

The time error is not white, which implies that two nodes do not have the same frequency. Timestamp-based techniques in wireless networks can not avoid being disturbed by link delay. The time error, which is the difference between receiving time and sending time in a timestamp, can be separated into two components, an

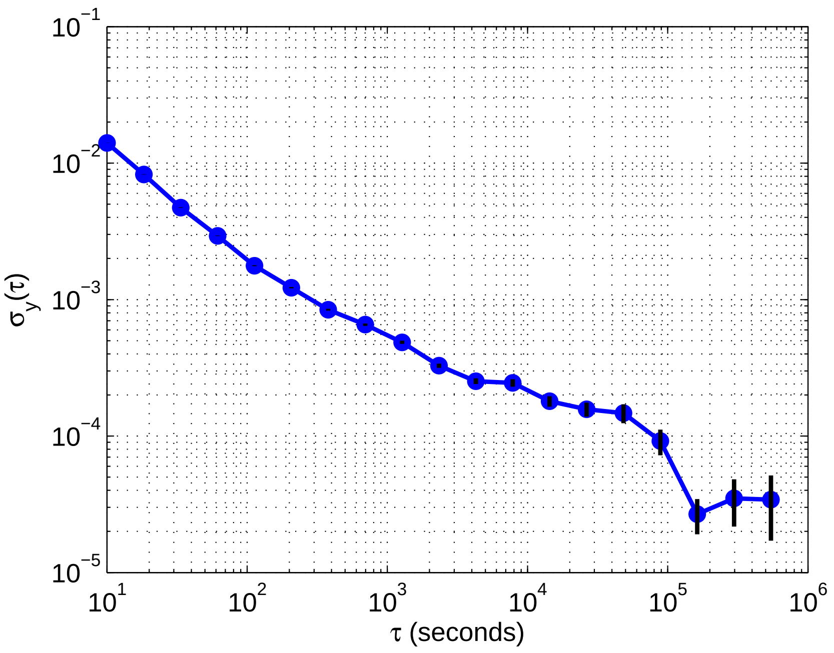


Figure 19: Allan deviation from a three-month experiment illustrates that clock difference between the two nodes was not stabilized even with long time intervals because of the uncertainties of timestamp transmissions.

offset and a link delay:

$$\begin{aligned} t_{error}[n] =& C_{receiver}[n] - C_{sender}[n] \\ =& \theta[n] + d[n]. \end{aligned} \tag{38}$$

$\theta[n]$ is the difference between the local time of one node and that of the other node, which can be either positive or negative. If the clock of the receiver indicates a faster local time than that of the sender, then $\theta[n]$ is positive. $d[n]$ is the sum of all delay that can occur during transmitting, including propagation delay, processing time in both the sender and receiver, and transmission delay. The delay is positive and generally assumed to be non-deterministic.

If it is assumed that each clock progresses at a fixed, but different frequency, clock offset ($\theta[n]$) will increase or decrease at a monotonic rate. Time error ($t_{error}[n]$) is fitted to a linear line by using a least squares approach. After being moved along the y-axis to make positive delay, the fitted line is $y[n] = aC_{receiver}[n] + b$, where a is

-0.0209, and b is -0.0095 in Figure 20. If oscillator frequencies are presumed to be fixed (a constant skew model), the line will be clock offset, and the slope, a, will be clock skew. Under the presumption of a constant skew, the link delay is calculated by $t_{error}[n] - y[n]$ and mostly distributed from zero to 200 ms as shown in Figure 21. The delay curve has a long tail in a positive direction that corresponds to the peaks shown in Figure 20. The long tail can be explained by processing delay caused by other systematic tasks or interruptions rather than by channel delay.

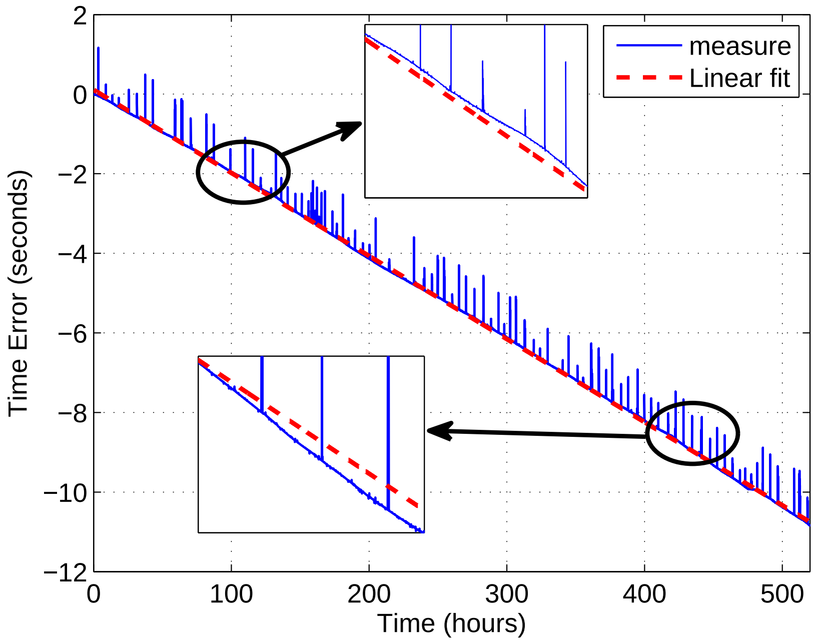


Figure 20: The best linear fit derived using a least squares approach does not well fit the time differences between two wireless nodes.

However, except for the long tail, the delay curve is still not Gaussian distributed even based on a long dataset. So that the curve would fit better, a fitting line is derived for a short period. From one-day data, the fitting line is $-0.0193C_{receiver}[n] - 0.0095$. As shown in Figure 21, the calculated delay is distributed mainly in a narrower range, from zero to about 50 ms with a long tail. Two strange peaks demonstrate that the delay contains not only link delay but also clock variations, which reflect offsets produced by skew varying. The mean is 0.1182 $seconds$, and the variance is

0.0065 for the entire set. For one day data, the mean and variance are 0.0360 and 0.0014, respectively. The insets in Figure 20 also indicate that the linear model does not fit the measured offsets. The linear fitting reveals that the offset is not linearly increasing or decreasing, but governed by non-constant clock skew.

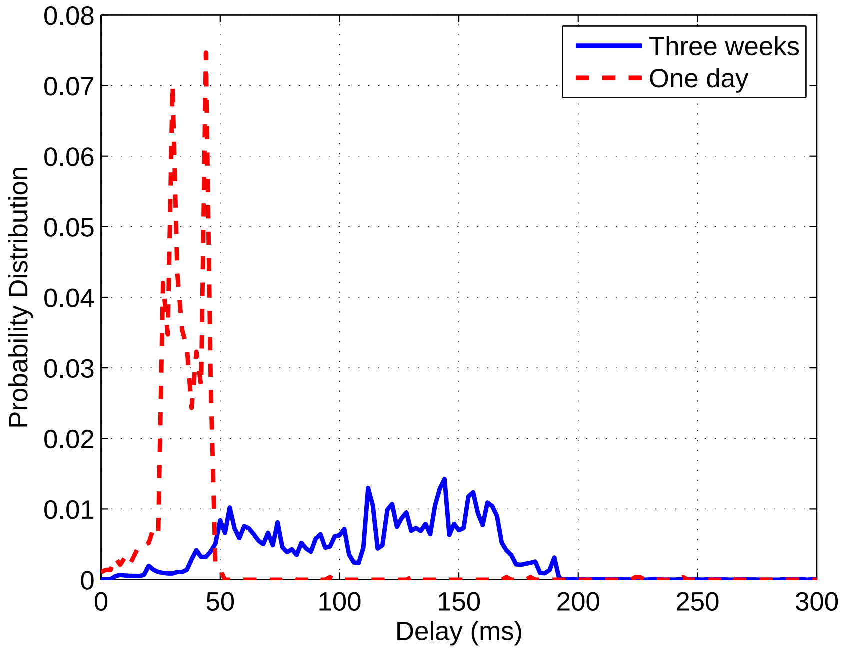


Figure 21: Delay estimates under an assumption that offset increases at a constant rate. $t_{error}[n] - y[n]$ contains time difference induced by skew variation as well as link delay.

The behavior of clock skew, in other words, the rate at which offset changes, is examined here. If one finds clock skew from an independent skew model in (17) with $\tau[n] = 60$ seconds, assuming random delay, then skew estimates are derived from time errors as

$$\begin{aligned}\hat{\alpha}[n] =& E\left\{\frac{t_{error}[n] - t_{error}[n-1]}{\tau[n]}\right\}\\ =& E\left\{\frac{\theta[n] + d[n] - \theta[n-1] - d[n-1]}{\tau[n]}\right\}\\ =& \frac{\theta[n] - \theta[n-1]}{\tau[n]} + \frac{E\left\{d[n] - d[n-1]\right\}}{\tau[n]},\end{aligned}$$

where $E\{d[n] - d[n-1]\}$ is zero. Figure 22 shows that the derived skew is distributed symmetrically around a certain non-zero mean. The non-zero, non-constant skew yields either faster or slower clock drifting. Since the distribution has heavy tails in both directions, the tails are exempt from fitting. About 70% data, which are close to their mean are fit to a Gaussian distribution. The heavy tails are likely to be produced by the peaks in link delays because of nodal processing delay in the sender or in the receiver.

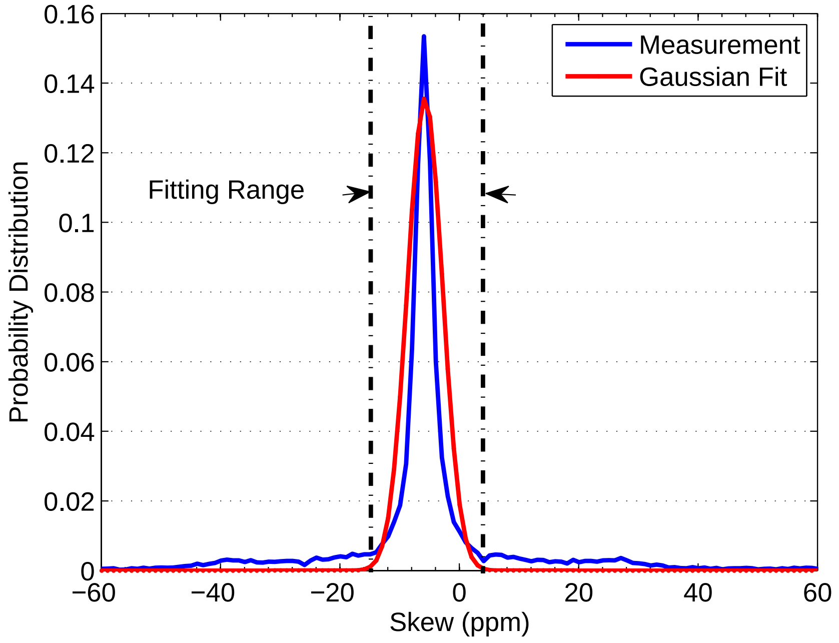


Figure 22: Clock skew is Gaussian-like distributed and has heavy tails. It is fitted to a Gaussian distribution whose probability density function (PDF) is $N(-5.7644, 2.9408)$.

Several sets of experiments were conducted with temperature measures to reveal the correlation with heat. One node was placed on a table, while another node was put into a paper box to produce a temperature gap. Table 3 shows an ambient temperature difference between the two nodes, a duration, and clock drifts for each set. The clock drifts are derived based on all of the data for each set. It is observed that a higher temperature gap causes clock offsets to become higher by shifting skew farther from zero. However, skew variance does not seem to be correlated to temperature.

Table 3: Temperature and clock drifts for each experiment.

	Low-Low	Mid-Low	High-Low
Duration	Three weeks	Five days	Six days
Temperature	–	$0.9 - 3.3°C$	$4.2 - 5.8°C$
Offset (*seconds*)	10.91	4.15	28.5
Mean(skew)	-5.7351×10^{-6}	-9.3933×10^{-6}	-5.1912×10^{-5}
Var(skew)	2.6084×10^{-6}	2.7276×10^{-6}	2.2799×10^{-6}

Figure 23 illustrates the skew distribution of each experiment. Here, the non-zero mean of clock skew indicates the inaccuracy of the crystal oscillator, and the variance represents its instability.

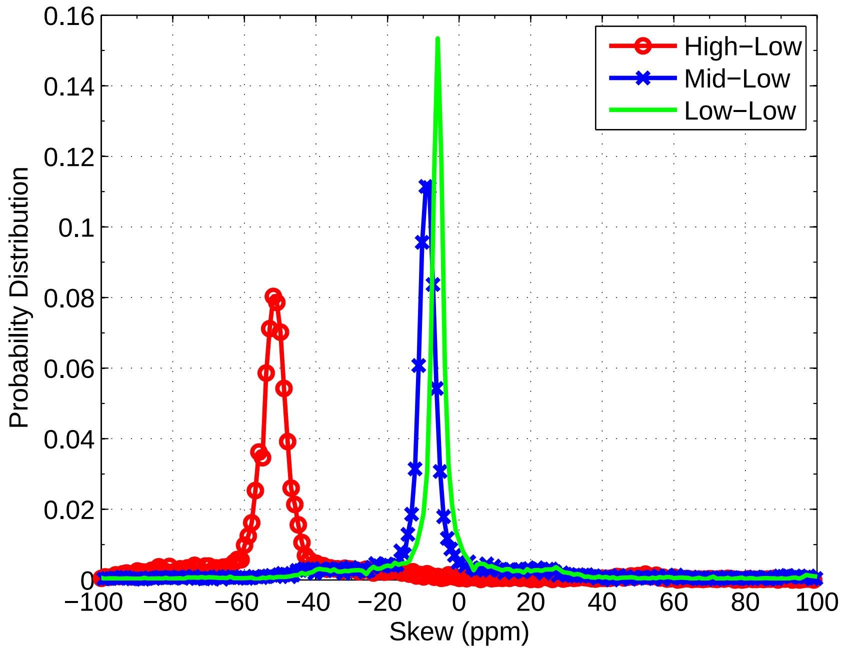


Figure 23: The distributions of clock skew estimates for different temperature conditions.

5.2 The Effects of Environmental Changes

The frequency characteristics of crystal oscillators over environmental changes have been described in some technical reports [2, 1]. This sub-chapter probes the effect of environmental changes on not a single oscillator, but on clock discrepancy among wireless nodes.

5.2.1 Observed Temperature and Humidity

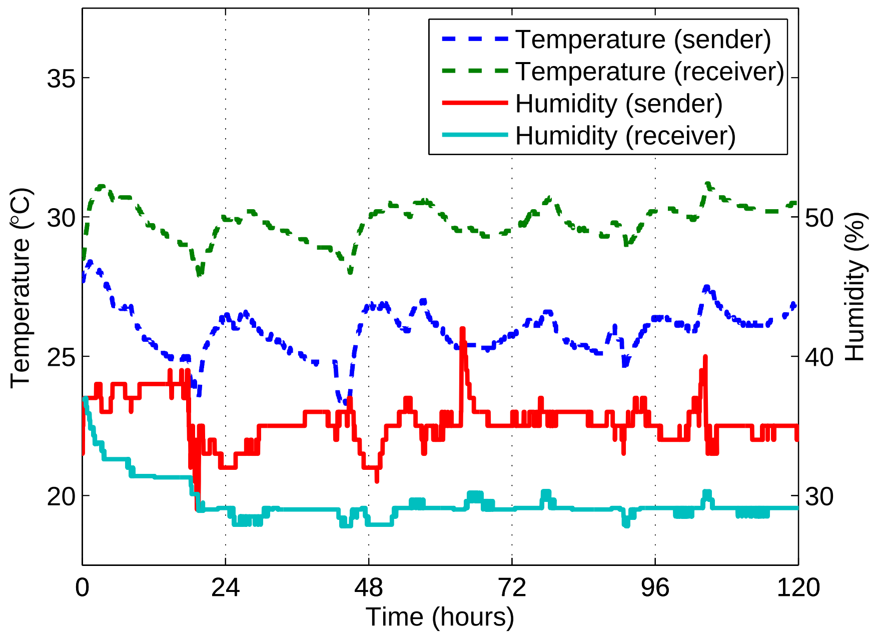


Figure 24: Varying environmental factors surrounding two nodes for five days.

The frequency error is noted for a third-polynomial equation of temperature. However, around $10°C - 40°C$, the error almost linearly depends on temperature. Since it is conjectured that one simple and apparent cause of the time difference between two clocks is heat, the experiment includes measuring temperature with local clock readings. The dependency of clock drifting on environmental changes can be explained by formulating the relationship. Temperature shifts relatively slowly, and its measurements are not impaired by uncertain delays since temperature itself is disassociated with the current time. The fact that temperature directly acts on clock drifts is one reason to attempt to achieve an accurate and robust estimation of clock error among nodes by exploiting temperature rather than timestamps.

Temperature and humidity, which are considered the main factors that perturb oscillators [103, 40] as well as timestamps, were measured for five days. During this period, temperature was not controlled dynamically. Instead, one Pandaboard was placed into a closed box with a temperature-humidity logger [1] while the other one was placed on a table. A temperature-humidity logger sensed temperature and humidity around each platform and recorded them every minute. Figure 24 indicates that temperature curves have roughly daily patterns, while humidity curves do not show clear periods. The receiver in the box was hotter and drier than the sender, and the temperature gap was about $4°C$ during the period. The temperature of the receiver ranged from $27.7°C$ to $31.2°C$ while that of the sender ranged from $23.1°C$ to $27.5°C$ after it was stabilized. Note that humidity is not correlated with temperature in this experiment.

5.2.2 Time-varying Clock Skew

By looking at the recorded time measurements, one can spot the correlation between temperature and a clock difference between the nodes. Over the period, the measured time difference increased, and the clock skew, the solid line in Figure 25, fluctuated around -15 *ppm*. Since the measured skew is highly disturbed by random noise, it is smoothed through pre-filtering to obtain a better estimate for clock skew. To de-noise, a moving average method can be adopted. However, a moving average spreads error to many points, or consequently contaminates multiple clock estimates. One may adopt a least squares approach. However, it is not only expensive in computation, but also inappropriate to track small variations.

The way used here is to use a median filter whose main idea is to run through the signal entry by entry, replacing each entry with the median of the neighboring entries. Median filtering is not used for real-time clock estimation, but it helps to comprehend clock behavior. In Figure 25, the dashed line represents the skew smoothed by a median filter (filter length = 60 seconds) that traces the tendency of the measured

[1]http://www.imagintronix.com/#!_download

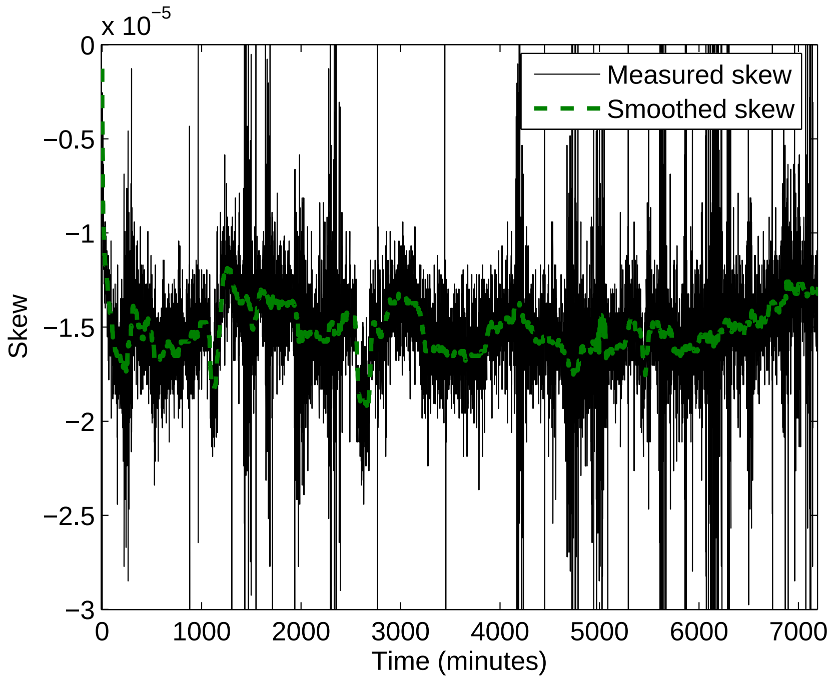


Figure 25: Instantaneous clock skew from timestamps (measured skew) is affected by unpredictable noise, and the smoothed skew follows the trend of the instantaneous skew.

one. The smoothed skew, which is neither constant nor consistently deviating, implies that it reflects the effect of the environmental changes.

Clock skew does not follow the temperature or humidity of any one of nodes in Figure 24; however, it seems to be correlated with a temperature gap between the two nodes. As shown in Figure 26, the temperature gap between the nodes corresponds to the measured skew in a negative way, especially at the peaks at around 1120 and 2660 minutes. When the analogy by Pearson's correlation coefficient is assessed, the correlation coefficient between the temperature gap and the smoothed skew is -0.5313 every minute, and -0.6066 every ten minutes. The coefficient is not as high as expected since many other factors can disrupt concurrence among clocks. The correlation coefficients could confirm that clock drift is correlated to a temperature difference between nodes, not the temperature in a single node. This makes it necessary to consider both the temperature of a receiver and that of a sender, which differs from approaches used in other references [88, 107].

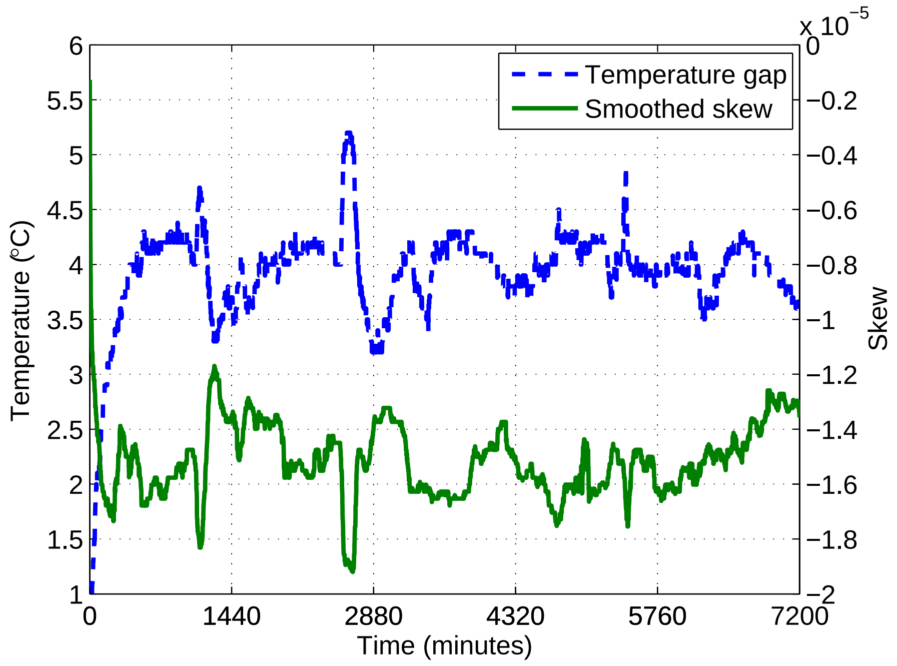


Figure 26: Clock drifting and temperature gap are correlated in a negative way.

5.3 Temperature-correlated Clock Skew

5.3.1 The Effect of Temperature on Clock Skew

The experiment is repeated under a temperature-controlled environment during a short period to restrict other impacts from hindering to the disclosure of the relationship between temperatures and the frequency drifts of oscillators. Sensors are likely to be exposed to severe environmental changes during their applications. A temperature-controlled environment may mimic these changes and make it possible to observe clock behavior in the devices according to the changes. So that the effect of temperature, a dominant influencer to clock error, can be maximized, a heater and a cooler are used for one node in a closed room. An external logger recorded temperature and humidity every ten seconds while a timestamp was exchanged every second.

The experiment was completed in one hour, a relatively short period, during which the sender stayed at around 27.8°C. For the receiver, the heater and cooler worked alternatively, as shown in Figure 27. When the heater turned on, the temperature around the receiver went up to 36.7°C. When the heater turned off, the temperature moved down, and when the cooling device turned on, the temperature reached to 27.7°C. The temperature around the receiver changed 9°C during the observation. Note that temperature and humidity measured in the given time period seem to correlate. However, this phenomenon may be explained by the heater and cooler acting on both factors. This is not a general case (see Figure 24).

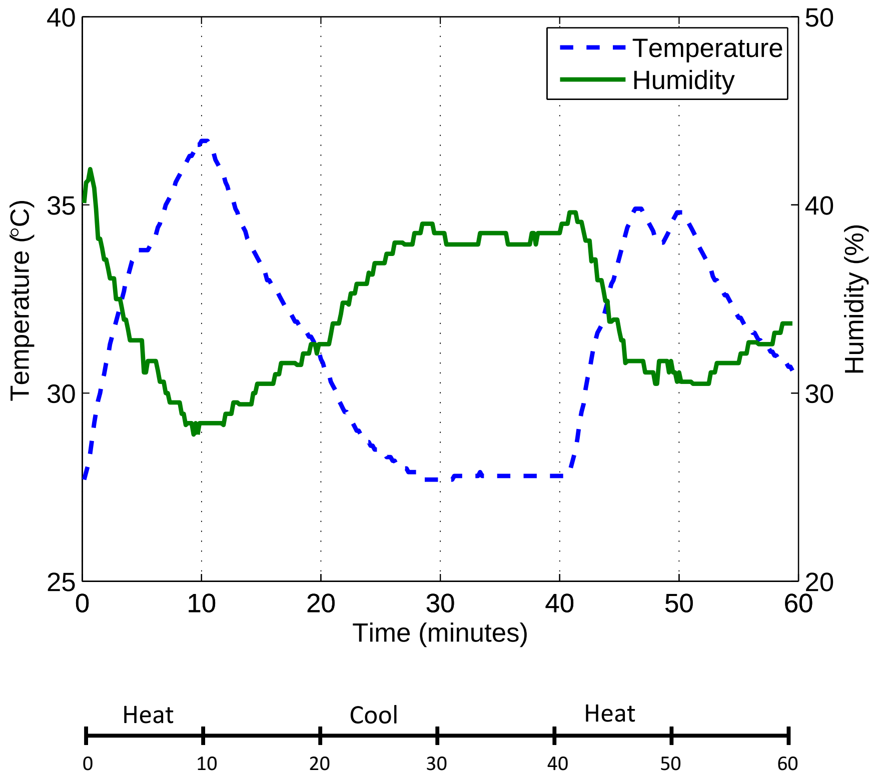
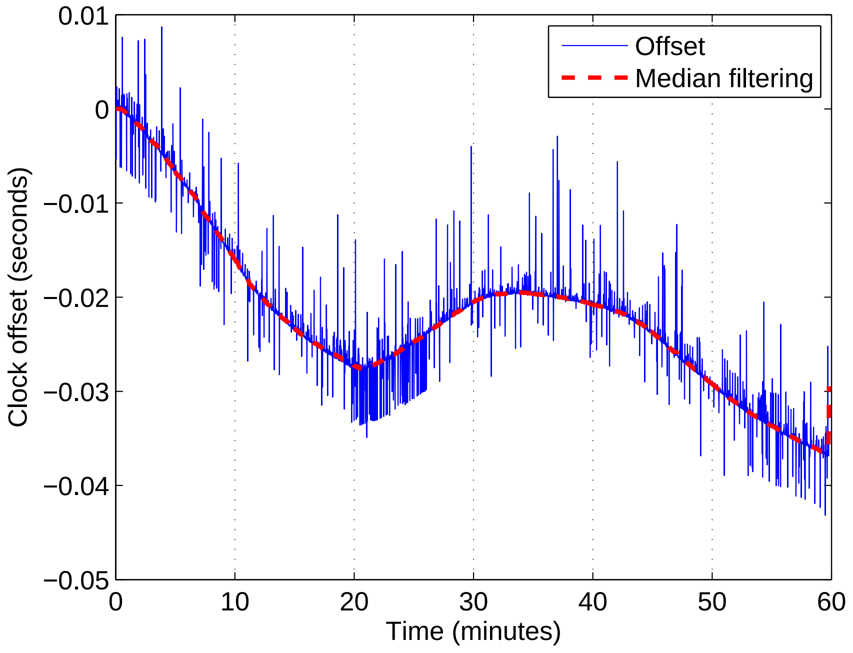
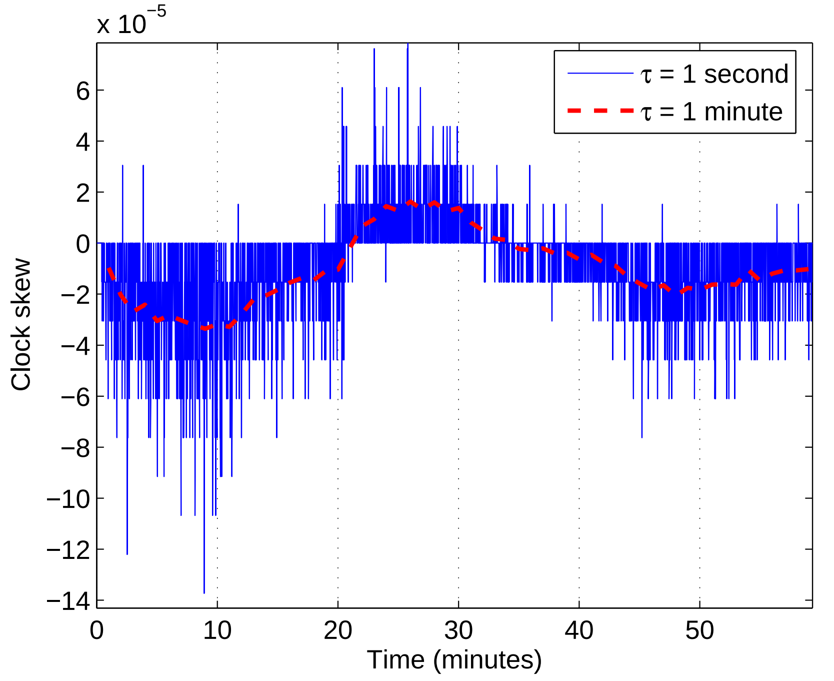


Figure 27: A heater and a cooler controlled temperature and humidity for an hour.

In Figure 28a, the time difference of the two nodes, clock offset, increased by 37 ms in an hour. Unlike the previous result, clock offset here did not drift at a monotonic rate. This is because temperature shift acted as a dominant factor on clock behavior in this experiment. The abnormal peaks were eliminated by a median filter,



(a) Clock offset suffering from random noise.



(b) Clock skew by different intervals (τ).

Figure 28: Clock drifting affected by varying environmental factors lasting an hour.

which is normally used to reduce noise spatially; at the same time, the median filter preserved useful details. After median-filtering with the window size 60, the curve of the clock offset became smooth (the dashed line). The offset decreased when the heater was working, while it increased when the cooler was working. However, when both temperature controllers were not operating, the offset is not clearly explained.

Clock skew that is an instant rate of the smoothed offset varies from -140 to 70 *ppm* (the solid line in Figure 28b). Since the clock-drifting rate does not change extremely fast, it is inferred that link delay dominantly causes the high fluctuation of clock skew estimates. However, defineing the skew as the slope of the smoothed offset every minute ($\tau[n] = 60$) reveals the moving tendency and shows the skew to be correlated with temperature, as shown in Figure 27. There are two notable observations. First, skew moving is coincident with the controlled temperature changes, demonstrating the thermal effect on a clock oscillator. Secondly, a short synchronization period does not improve its performance because of the disturbance of noise.

Median filtering can be an off-line approach to extract clock drifts from timestamp exchanges. However, this approach requires the awareness of prior samples, and moreover, a proper window size depends on the measured data patterns. Sometimes valuable phenomena of clock drifting may be lost as a result of smoothing without the selection of an appropriate window size. Therefore, median filtering can not be a real-time solution for synchronization, but it can be used as a benchmark for performance analysis of estimating techniques.

5.3.2 Correlation Analysis

Clock skew is not stationary, but the relationship between clock skew and temperature is stable. However, since temperature is one of the major determinants, but not the only influence, it is not expected that clock drift is a one-to-one function of temperature as there are many other factors. Figure 29 explicitly represents the relationship between a temperature gap among nodes and clock skew, which is smoothed by a median filter. Here, since the temperature of the sender was constant ($27.8°C$), its impact can be ignored. The linear line derived by a least squares approach is $y = -4.356\text{x}10^{-6}x + 5.601\text{x}10^{-6}$.

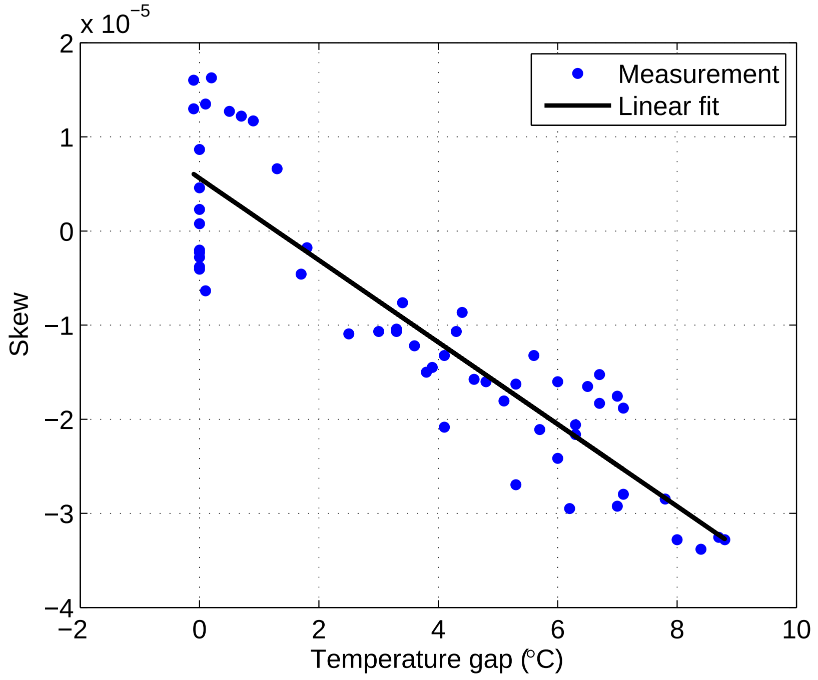


Figure 29: Linear relationship between a temperature gap and clock skew. The error variance is 3.5306×10^{-11}.

The experimental findings so far support that the correlation coefficient with temperature is very strong (-0.9051) in any circumstance that limits other factors over a short period. In this measurement, since humidity is highly analogous to temperature (correlation coefficient = -0.9299), it also strongly correlates with frequency error (0.7907). However, this is because humidity change was a by-product of temperature control caused by fans in the measurement. It is hard to tell which one is the main influencer of clock error; however, from many measurements, temperature was a stronger effect than humidity, as was proven in the previous section.

The correlation between temperature and clock skew is clarified when an internal sensor for each platform detects CPU temperature. From the user perspective, a Pandaboard provides access to reading a temperature sensor. Figure 30 shows CPU temperature, which was measured inside the processor. CPU temperature is much higher than air temperature sensed by an external logger while showing analogous patterns. During the observation, the CPU temperature of the receiver (temperature-controlled node) shifted $23°C$, while that of the sender varied $4°C$.

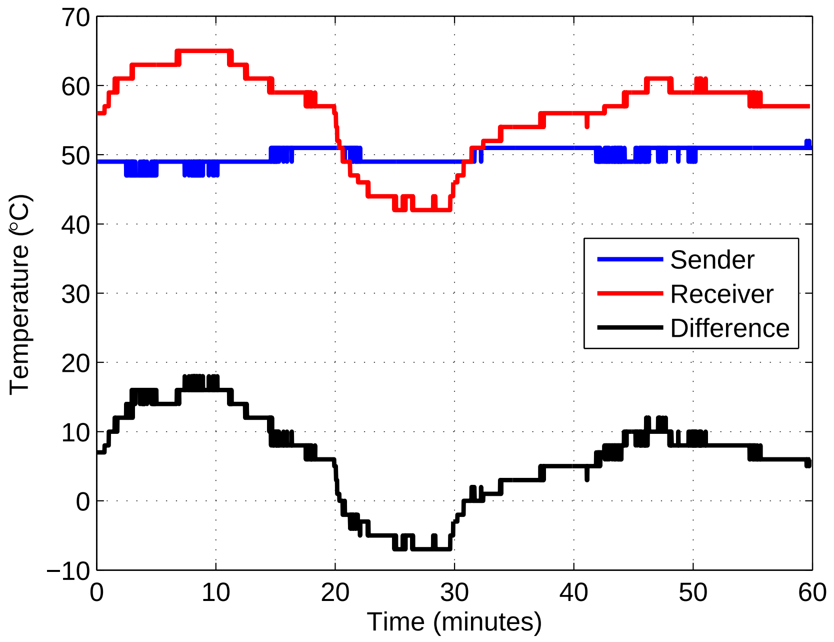


Figure 30: CPU temperature by a temperature sensor inside the OMAP chip on a Pandaboard.

As shown in Figures 31, clock skew has a linear dependency on temperature when temperature was controlled. The linear line is $y = -2.097\text{x}10^{-6}x + 2.582\text{x}10^{-6}$. As explained in the previous section, clock skew (frequency error) changes almost linearly around room temperature. The shown linearity corroborates that temperature was a dominant factor in the measurement. CPU temperature tends to correlate more strongly with clock skew (correlation coefficient = -0.9836) than with air temperature, which was measured outside the platform. This is because of the proximity of the temperature sensor on the platform to the oscillator. However, an internal temperature sensor is not always available for all devices, and sometimes its resolution is coarse. In the current experiment, temperature was increased by $0.25°C - 2°C$, depending on the temperature ranges.

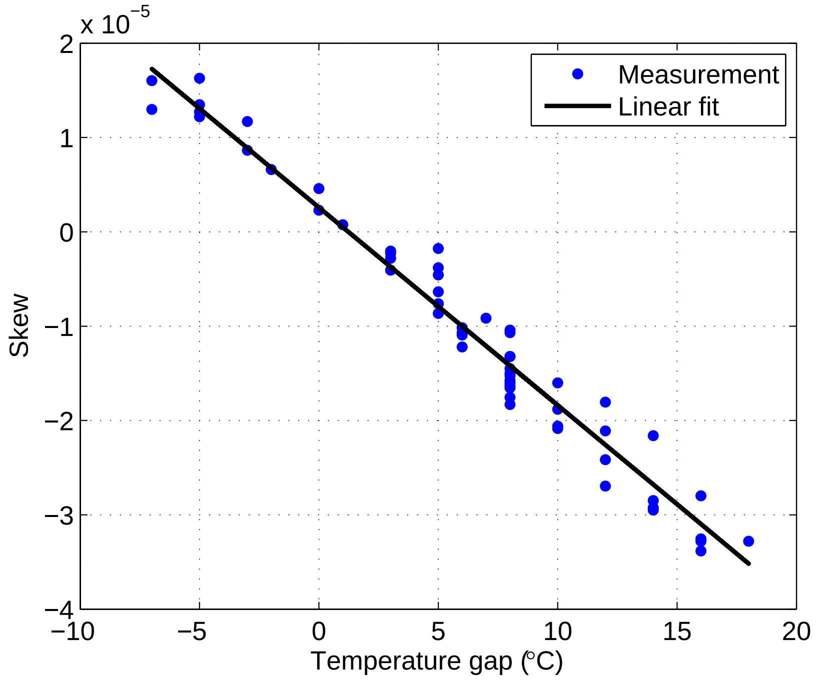


Figure 31: Linear relationship between a CPU temperature gap and clock skew. The error variance is 6.1734×10^{-12}.

5.4 Maximum-likelihood-type Clock Estimator (M-estimator)

5.4.1 Skew-Temperature Modeling

Here, the solution of clock synchronization is sought not in timestamps but in temperature. Since the previous sections have already confirmed that temperature is a dominant influence on clock error, a formula of clock characteristics on temperature can now be developed. This sub-chapter introduces the derivation of a skew-temperature model (STM) from noisy timestamps.

Since each clock oscillator behaves differently depending on its influencers, the model derivation takes into account both the temperature measurements of the sender and those of the receiver. According to some technical reports [2, 1], a crystal oscillator is accurate at $T = 25°C$, which means that the oscillator swings at the ideal frequency. Therefore clock skew to the absolute time is assumed as zero at $T = 25°C$, and an observed clock skew is defined as

$$\alpha_j[i](ppm) = \kappa_j(T_j[i] - T_0) + \varepsilon_j[i], \tag{39}$$

where κ_j is a temperature coefficient of node j, and $T_j[i]$ is the i^{th} temperature measurement at node j. $T_0 = 25°C$, and $\varepsilon_j[i]$ is modeling and measurement error. Based on (39), the relative clock skew of a receiver from a sender (α_R^S) is expressed as

$$\begin{aligned}\alpha_R^S[i] &= \alpha_R[i] - \alpha_S[i] \\ &= \kappa_R(T_R[i] - T_0) + \varepsilon_R[i] - \kappa_S(T_S[i] - T_0) - \varepsilon_S[i].\end{aligned} \tag{40}$$

This problem is solved as a least squares approach. Stacking up valid observations, one can write the relationship as a matrix formula:

$$\begin{bmatrix}\alpha_R^S[1] \\ \alpha_R^S[2] \\ \vdots \\ \alpha_R^S[n]\end{bmatrix} = \begin{bmatrix}T_R[1] - T_0 & T_S[1] - T_0 \\ T_R[2] - T_0 & T_S[2] - T_0 \\ \vdots & \vdots \\ T_R[n] - T_0 & T_S[n] - T_0\end{bmatrix} \begin{bmatrix}\kappa_R \\ -\kappa_S\end{bmatrix} + \begin{bmatrix}\Upsilon[1] \\ \Upsilon[2] \\ \vdots \\ \Upsilon[n]\end{bmatrix}, \tag{41}$$

where $\Upsilon[i] = \varepsilon_R[i] - \varepsilon_S[i]$. Simply, (41) can be represented as

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{p} + \boldsymbol{\Upsilon}, \tag{42}$$

where $\boldsymbol{z}$ is clock skew calculated by timestamps, $\boldsymbol{H}$ is a temperature measurement matrix, $\boldsymbol{p}$ is a temperature coefficient vector, and $\boldsymbol{\Upsilon}$ is a total error vector. The temperature coefficient vector ($\boldsymbol{p}$) can be solved by a pseudo inverse:

$$\hat{\boldsymbol{p}} = \boldsymbol{H}^{\dagger}\boldsymbol{z}, \tag{43}$$

where $\boldsymbol{H}^{\dagger} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T$.

In this case, $\hat{\boldsymbol{p}}$, derived based on ten-minute measurements over one hour, is $[2.197, -2.099]^T$ by CPU temperature, and $\hat{\boldsymbol{p}} = [5.559, -3.780]^T$ by air temperature. Thus, as an example, the STM derived by ambient temperature between the two nodes is $\hat{\alpha}_R^S[i](ppm) = 5.559T_R[i] - 3.780T_S[i] - 44.475$. If a sender transmits temperature measurements, which shift relatively slowly, instead of timestamps, a receiver can estimate the skew of its local clock from that of the sender based on the established STM.

This sub-chapter proposes introducing robust regression techniques, called M-estimators to derive STMs [2]:

$$min \sum \rho(r_i), \tag{44}$$

where $r_i = \alpha[i] - \hat{\alpha}[i]$. The standard least squares approach adopts $\rho(r_i) = r_i^2$ and minimizes $\sum r_i^2$. However, this approach is sometimes unstable if outliers exist in the dataset. As shown in Figure 18, timestamp-transmission algorithms are influenced by various types of noise and result in outliers in clock estimates. Employing a weight function, (44) can be written as

$$min \sum w(r_i^{k-1}) r_i^2, \tag{45}$$

where $w(x)$ is a weight function. For each observation, an error is calculated by being iteratively re-weighted according to a pre-defined weight function.

Here the M-estimator adopts a Tukey (bisquare) weight function that assigns heavier penalties to anomalies and excludes huge delays from the estimation. The temperature coefficients derived by the M-estimator are $[5.411, -3.676]^T$ with a temperature logger, and $[2.340, -2.136]^T$ with a CPU sensor. This derivation was accomplished by training ten-minute timestamps with temperature measurements over a

[2]http://research.microsoft.com/en-us/um/people/zhang/INRIA/Publis/Tutorial-Estim/node24.html

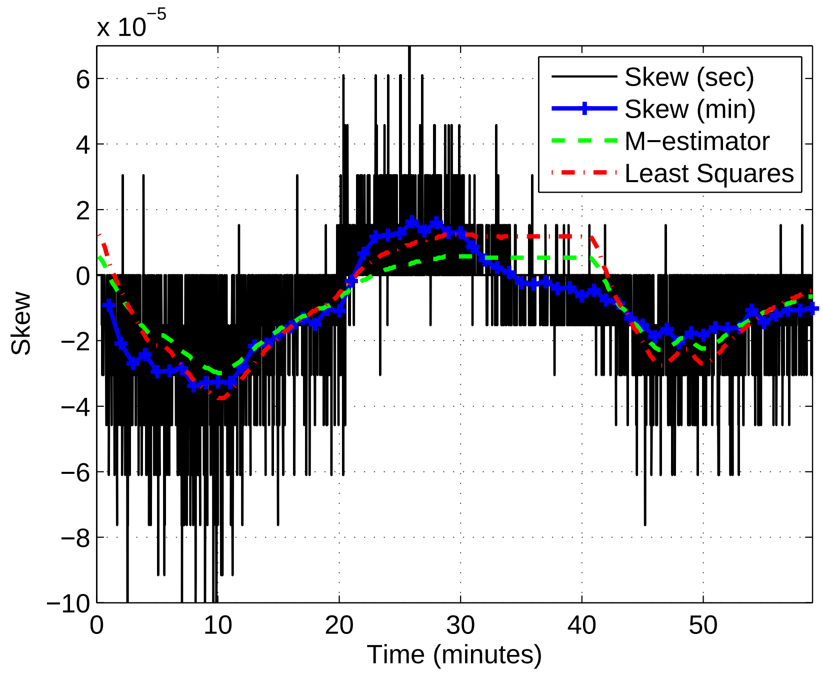
one-hour observation. This derivation did not require any pre-filtering or knowledge about measurement data.

5.4.2 Clock Error Estimation

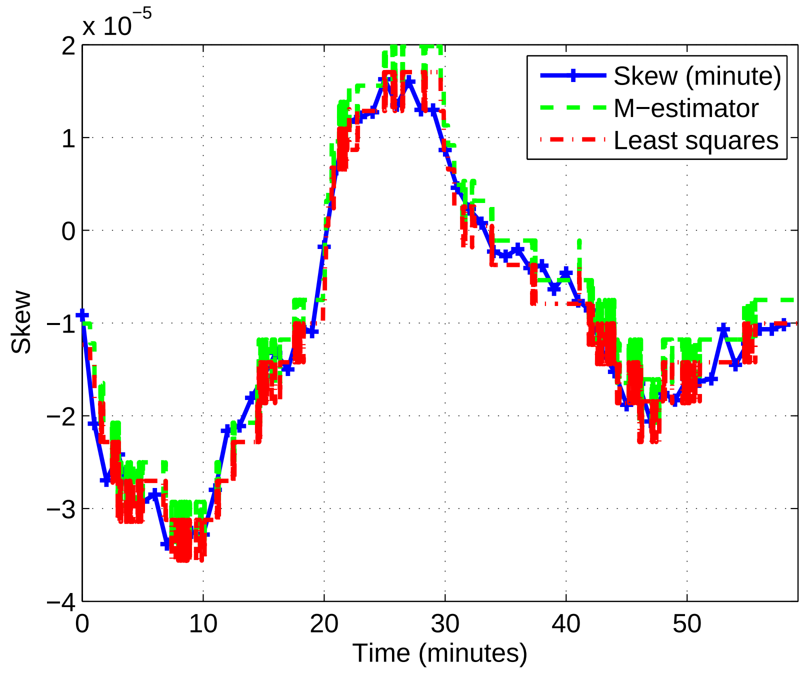
Clock skew can be estimated by the derived STM with temperature measurements. Figure 32a shows both skew estimates using a least squares approach and skew estimates using an M-estimator based on ambient temperature measurements. Surprisingly, the STMs derived from the noisy (non-filtered) timestamps and ambient temperature closely track the skew smoothed by a median filter while eliminating peaks considerably. If the filtered skew is assumed to present true clock behavior, the mean squared error (MSE) is 1.0041×10^{-10} for the least squares approach and 7.7451×10^{-11} for the M-estimator.

Due to the proximity of a heat sensor, the estimates by means of CPU thermal information are more accurate as shown in Figure 32b. The MSEs of a least squares and M-estimator are 1.0342×10^{-11} and 1.6006×10^{-11}. The reason the skews by both the least squares approach and the M-estimator seem similar is that the noise on the observed skews is huge but distributed evenly. However, since CPU temperature is measured in coarse resolutions, the estimation curves show stepwise patterns. Besides, a CPU temperature sensor may not be available for some hardware.

While building a mapping function of temperature to clock drift, the proposed technique seeks to deliver both a temperature measurement and a timestamp. However, after establishing a temperature coefficient vector ($\boldsymbol{p}$), the estimation method presented here does not require timestamps, which can be severely disturbed by random noise. Clock skew is estimated only with temperature, not with a timestamp. Based on the experimental results, temperature is less noisy and changes more slowly than a timestamp. Furthermore, temperature requires fewer bits to be stored in a transmission packet, which means that it consumes less energy and bandwidth.



(a) Clock estimation based on ambient temperature.



(b) Clock estimation based on CPU temperature.

Figure 32: Comparison between clock skew estimates by timestamp and by temperature.

Table 4: The parameters and accuracy of the derived STMs.

Temperature	Estimator	Parameters ($\hat{\boldsymbol{p}}$)	MSE
External logger	LS	5.559, −3.780	1.004×10^{-10}
	Tukey	5.411, −3.67	7.745×10^{-11}
CPU sensor	LS	2.197, −2.099	1.034×10^{-11}
	Tukey	2.340, −2.136	1.601×10^{-11}

Therefore, the proposed estimation is suitable for resource-constraint applications. The proposed estimator is independent of unpredictable network noise and tracks true clock drifts. If the estimator is updated by periodic training, the accuracy of clock estimation will be improved.

Table 4 summarizes the temperature parameters derived by using a CPU temperature sensor and by using an external sensor. This table also compare the MSEs for the two estimators. The estimation by using CPU performs better for any estimator, which means that precise temperature information is necessary to improve synchronization accuracy.

CHAPTER VI

CLOCK TRACKING IN TEMPERATURE-VARYING ENVIRONMENTS

6.1 Kalman Filter Representation

The STMs introduced in the previous chapter can estimate clock skew from temperature measurements. This chapter proposes an enhanced clock-tracking technique that uses Kalman filtering combined with the STM derived based on the findings from the experiments. A Kalman filter is often used to solve tracking problems since it brings an optimal solution for a linear system. The proposed clock-tracking algorithm actively tracks small variations while compensating clock frequency errors induced by temperature shifts. The accuracy of the clock-tracking technique that this chapter introduces is shown and compared to that of the technique that was proposed in Chapter 4 later.

A Kalman filter is designed to estimate the internal state,

$$\boldsymbol{x}[n] = \begin{bmatrix} \theta[n] & \alpha[n] & \dots & \alpha[n-P+1] \end{bmatrix}^T,$$

given a sequence of noisy observations. Some matrices are specified: $\boldsymbol{A}$ is a transition matrix; $\boldsymbol{B}$ is a control-input model; $\boldsymbol{C}_w$ is the covariance matrix of process noise. $\boldsymbol{u}[n]$ is a control vector and sets to $\begin{bmatrix} \Delta T_S[n] & \Delta T_R[n] \end{bmatrix}^T$. $T_S[n]$ is the temperature of a sender, and $T_R[n]$ is the temperature of a receiver. $\Delta T_j[n] = T_j[n] - T_j[n-1]$ is a temperature shift during an observation period. As a designed Kalman filter takes thermal information into the control vector, the filter compensates for frequency shifts. Here, ambient temperature measurements were utilized because the external logger recorded temperature with a better resolution than the internal sensor in the experiments (see Figure 27 and Figure 30).

The state equation is defined as

$$\boldsymbol{x}[n] = \boldsymbol{A}\boldsymbol{x}[n-1] + \boldsymbol{B}\boldsymbol{u}[n-1] + \boldsymbol{w}[n], \tag{46}$$

where $\boldsymbol{w}[n]$ is a process noise vector. The derived skew model is incorporated into the transition matrix, and the skew-temperature model is used as an input-control model:

$$\boldsymbol{A} = \begin{bmatrix} 1 & \tau_0 & 0 & \dots & 0 \\ 0 & \hat{c}_1 & \hat{c}_2 & \dots & \hat{c}_P \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ \kappa_A & -\kappa_B \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix},$$

where κ_A and κ_B are the temperature parameters of a receiver and a sender, respectively, which were derived by using the proposed M-estimation based on the measurements. The observation equation is written as

$$\theta[n] = \boldsymbol{b}^T\boldsymbol{x}[n] + v[n], \tag{47}$$

where $\boldsymbol{b}^T = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}$, and $v[n]$ is observation noise.

The n^{th} *a posteriori* state estimate given observations up to and including the n^{th} sample is

$$\begin{aligned} \hat{\boldsymbol{x}}[n] &= \boldsymbol{A}\hat{\boldsymbol{x}}[n-1] + \boldsymbol{B}\boldsymbol{u}[n-1] \\ &\quad + \boldsymbol{G}[n](\theta[n] - \boldsymbol{b}^T(\boldsymbol{A}\hat{\boldsymbol{x}}[n-1] + \boldsymbol{B}\boldsymbol{u}[n-1])), \end{aligned} \tag{48}$$

where $\hat{\boldsymbol{x}}[n]$ is the estimate of the offset and skew state at the n^{th} sample, and $\boldsymbol{G}[n]$ is a Kalman gain. The predicted (*a priori*) estimate covariance is

$$\boldsymbol{\Sigma}[n] = \boldsymbol{A}\boldsymbol{M}[n-1]\boldsymbol{A}^T + \boldsymbol{C}_w. \tag{49}$$

The minimum mean squared error (MMSE) of the estimate, which is a posteriori, is

$$\boldsymbol{M}[n] = (\boldsymbol{I} - \boldsymbol{G}[n]\boldsymbol{b}^T)\boldsymbol{\Sigma}[n]. \tag{50}$$

These mean squared errors (MSEs) are the performance metrics for the designed Kalman filter itself without considering observations.

The optimal Kalman gain, which determines performance of a Kalman filter, is

$$\boldsymbol{G}[n] = \boldsymbol{\Sigma}[n]\boldsymbol{b}\left(\sigma_v^2 + \boldsymbol{b}^T\boldsymbol{\Sigma}[n]\boldsymbol{b}\right)^{-1}, \tag{51}$$

where σ_v^2 is the observation noise variance. On every observation, first, a Kalman filter predicts a state estimate based on a previous estimate and a control input. Next, the Kalman filter updates the state estimate by using a current observation. Since Kalman gain, $\boldsymbol{G}[n]$, assigns a weight on the current observation, finding an accurate $\boldsymbol{G}[n]$ is crucial for the performance. In (51), it is seen that σ_v^2 and $\Sigma[n]$ determine a Kalman gain. $\Sigma[n]$ is a prediction MSE, which is calculated by a previous MMSE and $\boldsymbol{C}_w$.

Figure 33 presents a graphical expression of the designed filter, which is called a temperature-compensated Kalman filter (TCKF). For the clock synchronization task, it is required to append more parts including clock modeling and the calculation of clock drift. The calculation of a Kalman filter is complicated and requires a great deal of computation. Moreover, the proposed filter contains matrix operations, and the size of the matrices depends on the order of the AR process model. Therefore, it is required that some implementation issues be solved in order to realize the proposed algorithm. Once the filter is built, it is static in time when temperature is either changing or constant, different from [107].

6.2 Performance Analysis

Determining covariances is an important step for Kalman filtering. Here, noise variances are statistically found from observations, and then, based on the variances, the Kalman filter tracks true clock drifts. However, accurate noise variances are not always available in real-time tracking applications. The tracking performance

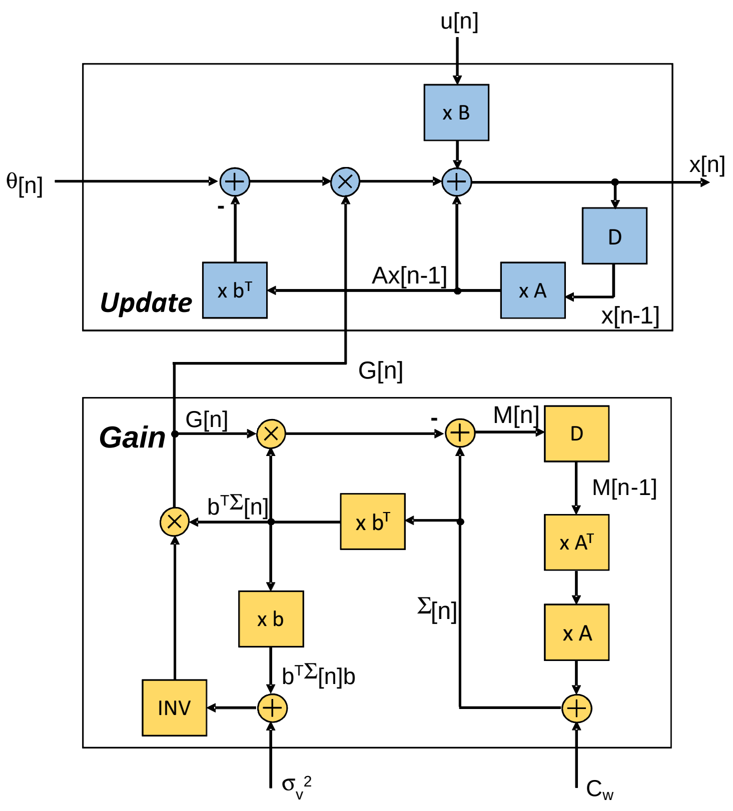


Figure 33: Signal flow for TCKF.

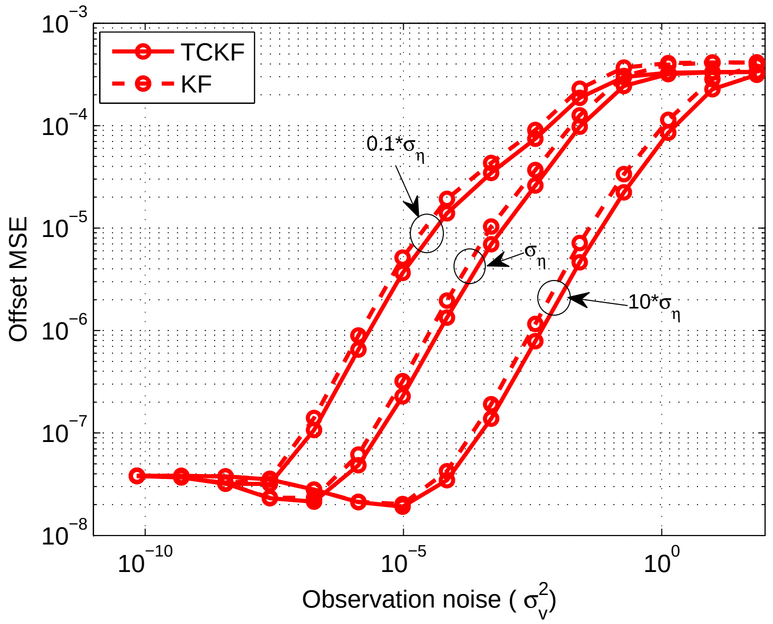
is predicted regarding different noise variances for which there is no a priori knowledge about data. Generally, the performance is determined based on the interaction between observation and process noise variances.

Simulations based on the measurements were used to evaluate the proposed TCKF. First, ten-minute measurements were trained to obtain an AR model for clock skew. The optimal order (P) was determined to be three by the result of information criteria, and the model coefficients were $\hat{\boldsymbol{c}}$=[0.4397, 0.3106, 0.1874], which were components for the transition matrix ($\boldsymbol{A}$). The derived temperature coefficient vector ($\hat{\boldsymbol{p}}$) by air temperature was integrated into the control input model ($\boldsymbol{B}$). $\boldsymbol{C}_w$=[$\tau_0^2\hat{\sigma}_\alpha^2$ 0 0 0; 0 $\hat{\sigma}_\eta^2$ 0 0; 0 0 0 0; 0 0 0 0], where $\hat{\sigma}_\alpha$, $\hat{\sigma}_\eta$, and $\hat{\sigma}_v$ were set to the standard derivations of true skew, the fitting error of the derived AR model, and observed offset, respectively, with $\tau_0 = 10$ seconds. With the given settings, the MSEs of the measured offset and skew were calculated from the true offset and skew that had been smoothed by a median filter.

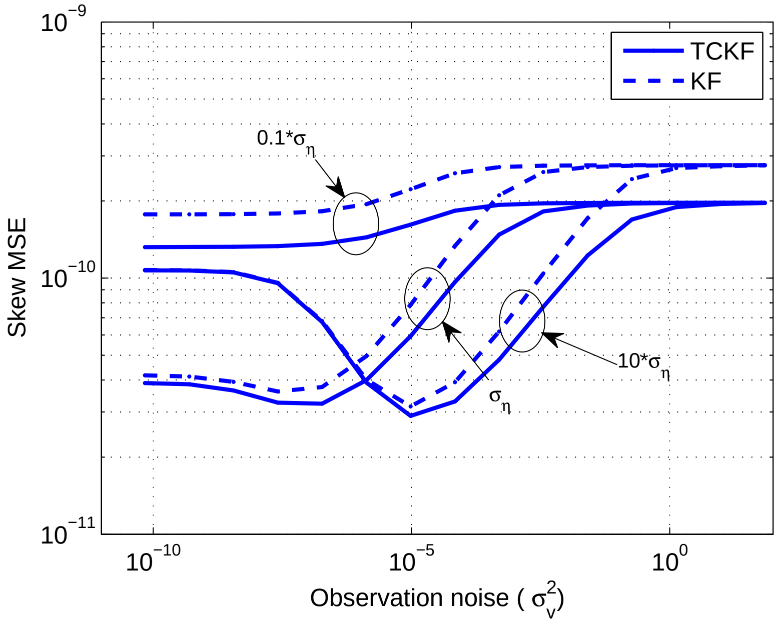
In (51), σ_v^2 is in inverse proportion to $\boldsymbol{G}[n]$. When σ_v^2 increases, $\boldsymbol{G}[n]$ decreases, and estimates slowly converge regarding for some changes. However, for a low σ_v^2, estimates well follow dynamics; meanwhile, anomalies may not be filtered out. In other words, the Kalman filter becomes unstable, and as a result, may produce high ripples on its estimates. Figure 34 shows offset and skew MSEs in terms of various observation noise variances. Around the variance estimate based on observations ($\hat{\sigma}_v^2 = 6.9 \times 10^{-5}$), is an optimal point (lowest MSE). If the observation noise variance is lower, the filter can not track small variations. If the observation variance is higher than the optimal value, the outputs may be distorted by a few abnormal measurements. A low variance limits the freedom of the designed filter, while a high variance makes it unstable.

Oppositely, $\boldsymbol{C}_w$ is proportional to the gain, which means that a high $\boldsymbol{C}_w$ allows the Kalman filter to follow observations closely. Figure 35 presents offset and skew MSEs regarding process noise variances. The performance of a high process noise variance worsens as the variance escalates the uncertainty of the linear model. Not only the optimal process noise but also the stabilized MSEs are influenced by observation noise variance. The performance of a Kalman filter depends on the selection of noise variances.

Figures 34 and 35 also show a comparison between the performance of the TCKF proposed here and that of a Kalman filter (KF) without temperature inputs [46, 47], which was introduced in Chapter 4. If observation noise variance (σ_v^2) increases, the MSEs of both the TCKF and the KF become higher and finally converge, but the TCKF performs better than the KF. Each filter is also affected by process noise variance (σ_η^2). In contrast to the previous result, when process noise variance increases, the MSEs for both the TCKF and the KF perform evenly. If the variance decreases,

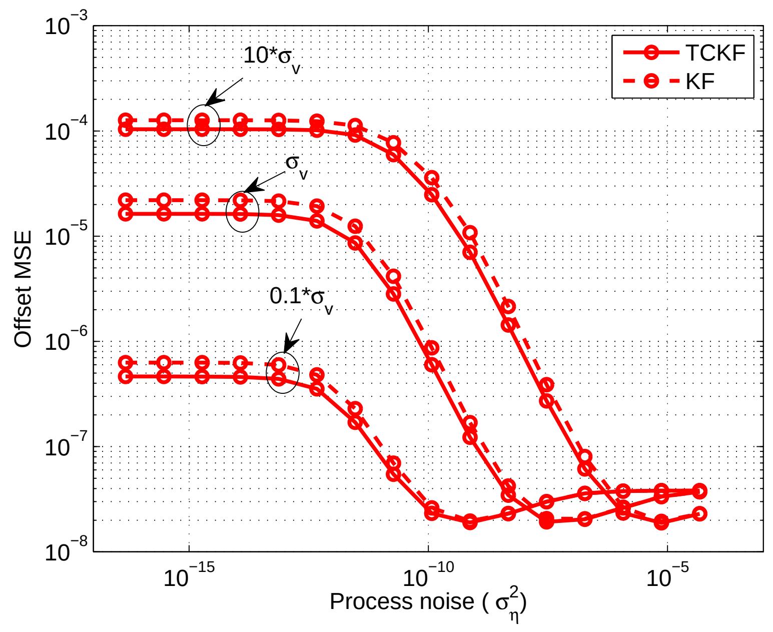


(a)

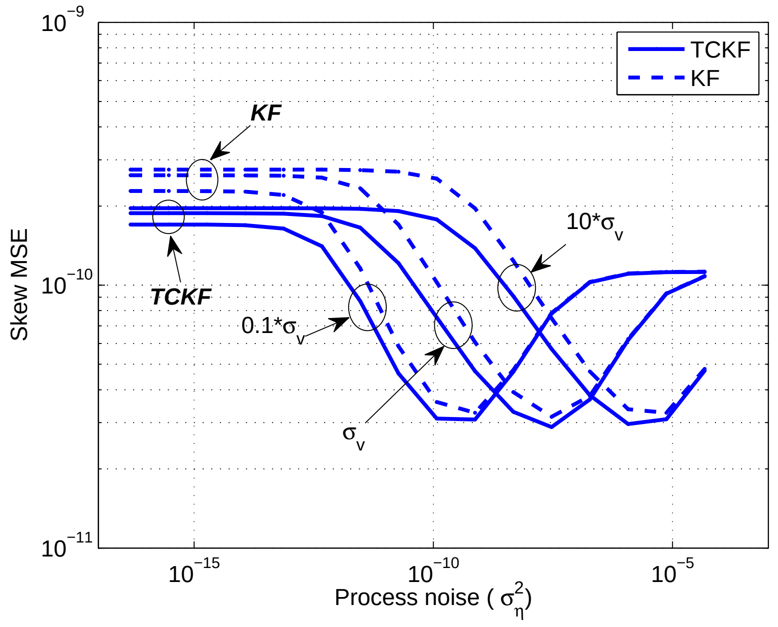


(b)

Figure 34: Tracking performance regarding observation noise variance when σ_n=6.8578 $\times$ 10^{-6}. The designed Kalman filter is combined with an optimal skew model (AR(3)) and the derived STM.



(a)



(b)

Figure 35: Tracking performance regarding measurement noise variance when σ_v=0.833×10^{-2}.

the TCKF outperforms the KF; however, the performance is limited by observation noise variances. Because the nodal temperature is leveraged, the TCKF is able not only to track clock inaccuracy, but also, interestingly, to perform better than the KF regardless of noise variances. This is because thermal measurements help estimate frequency variation. The TCKF supplements the inaccuracy of noise variances in the designed filter.

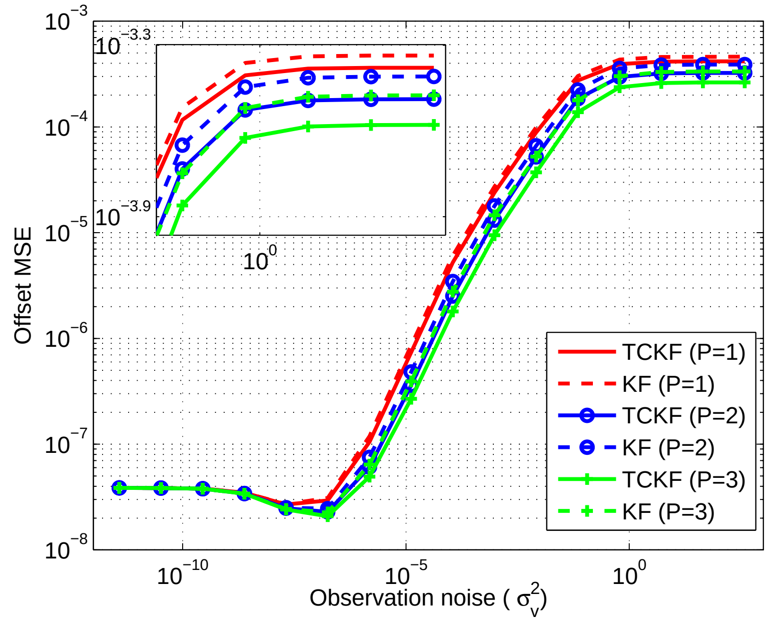
The performance of a TCKF is also explored regarding skew models. Normally, as the order of an AR model increases, the model better fits an original dataset. However, the derived model becomes more complex and raises modeling costs, which is not desirable for many applications. Figures 36 and 37 show performance regarding AR(P), where P=1,2,3. AR(1) and AR(2) clock-skew models based on the same dataset are compared to AR(3), which is the optimal order by information criteria. The TCKF as well as the KF boost their efficacy with a higher-order AR model. It is proven that the tracking performance can be improved by generating more accurate clock models.

Each AR model was found by training the same length of data (one day). The coefficients for the derived AR(1) and AR(2) are $\hat{\boldsymbol{c}}$=[0.8708] and $\hat{\boldsymbol{c}}$=[0.5161, 0.4073], respectively. Since AR(1) and AR(2) demand smaller sizes of matrices and vectors than AR(3), they are implemented by using less hardware registers and less computing elements. However, as shown in the figures, AR(1) and AR(2) achieve lower accuracy for both the TCKF and the KT. Therefore, the synchronization accuracy is traded with required resources.

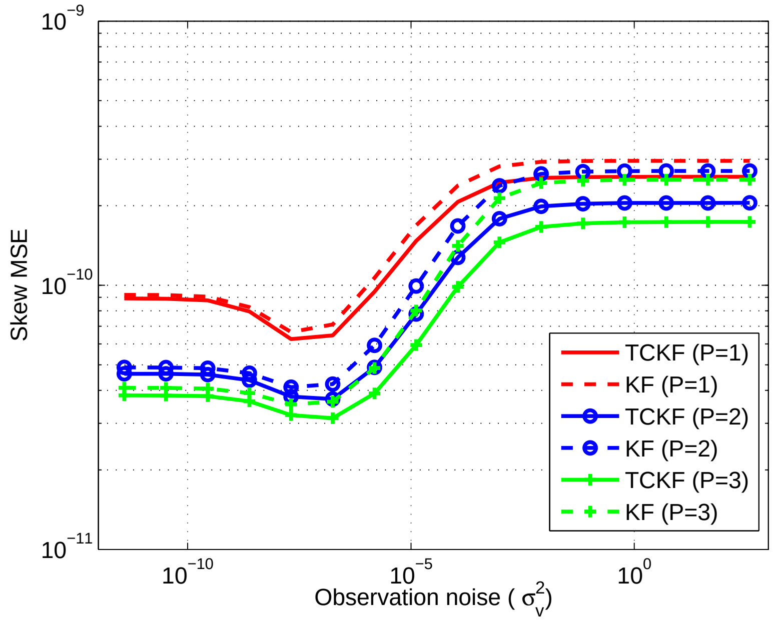
For various skew models, a TCKF also outperforms a KF except when the observation noise variance is set to be lower or the process noise variance is set to be higher than optimal variances. If the variances for the filter are adequately granted, a TCKF is able to enhance synchronization accuracy. The use of temperature measurements allows the proposed TCKF to better track varying clock offset and skew while avoiding increasing the order of the AR process.

This synchronization problem is approached by investigating the causes of clock inaccuracy and, at the same time, by looking into the statistical property of clock drifting. Based on the clock measurements of the mobile devices in a noisy link, this research discovers that clock skew is correlated with temperature drifting. It further proposes a Kalman filtering algorithm that compensates with temperature changes and estimates clock offset and skew in real time. The proposed synchronization technique is more profitable when temperature considerably changes. Temperature information is available for many devices, especially monitoring sensors. The proposed technique can quickly follow varying clock offset and skew by applying frequency shifts caused by temperature changes.

Kalman filtering is a widely used technique to solve tracking problems. However, since Kalman filtering requires a great deal of computation for matrix multiplications as shown in Figure 33, it may need to be simplified or optimized for some resource-constrained applications. One possible way is to update the demanding calculation part, which is Kalman gain, periodically instead of on every observation. The periodic update will save computing resources for the Kalman filter-based approach. Figure 38 presents the proposed architecture for energy-efficient and resource-efficient clock synchronization.

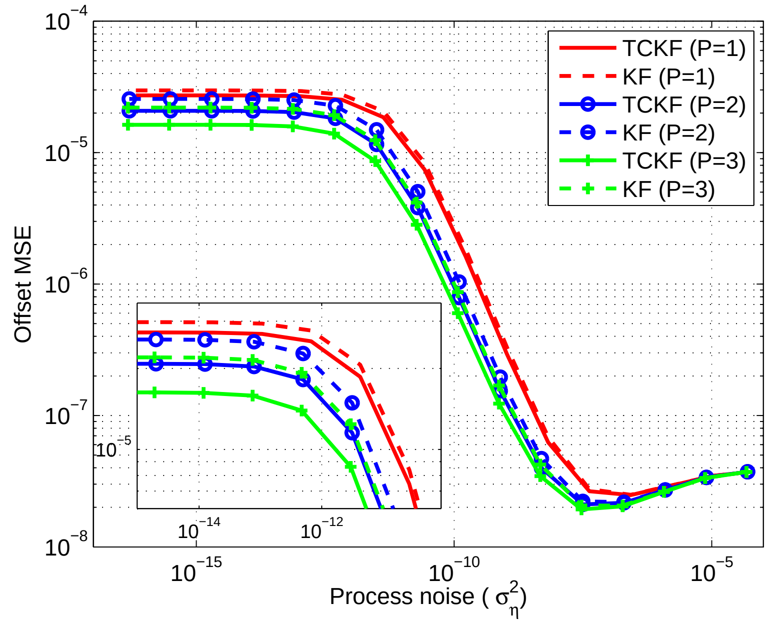


(a)

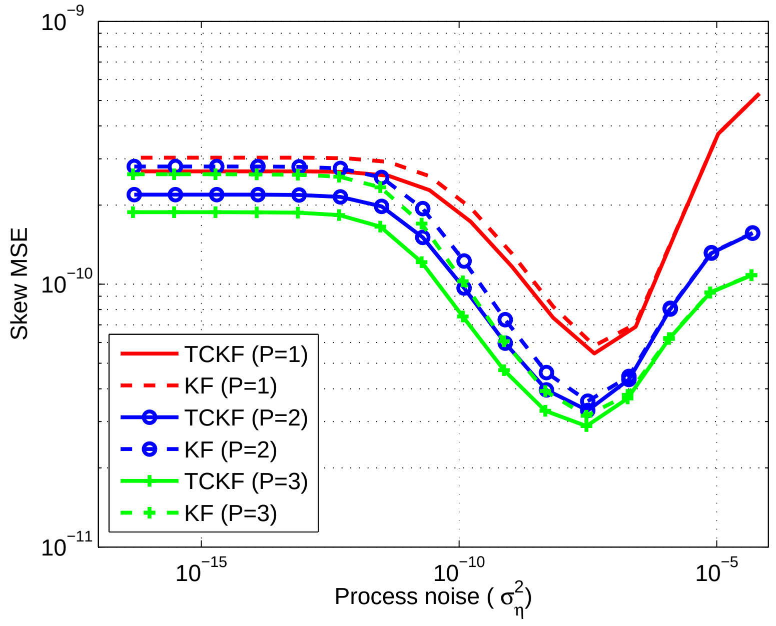


(b)

Figure 36: The MSEs regarding observation noise variances for different clock model orders when σ_n=6.8578 $\times$ 10^{-6}. An accurate clock model can improve tracking performance.



(a)



(b)

Figure 37: The MSEs regarding process noise variances for different clock model orders when σ_v=0.833 × 10^{-2}.

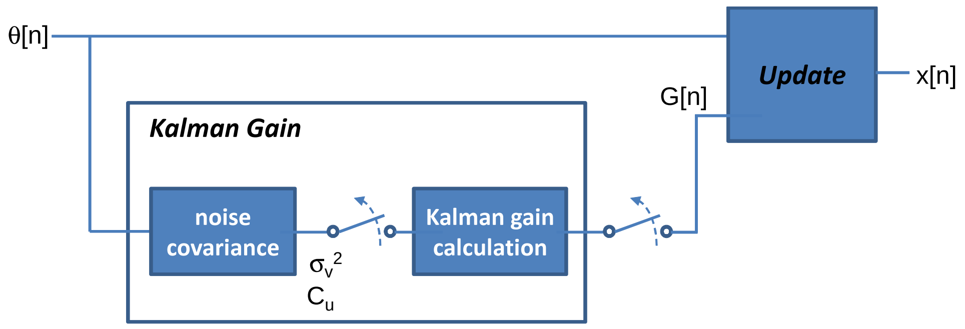


Figure 38: The proposed Kalman filter architecture.

CHAPTER VII

IMPLEMENTING KALMAN FILTER-BASED CLOCK SYNCHRONIZATION

This chapter explains the implementation of the Kalman filter-based clock tracking technique introduced in Chapter 4. The tracking algorithm is developed based on a Kalman filter integrated with the clock models, which can be generally applied and accurately follow the instability of clock offset and skew. This approach produces statistical estimates of a drifting clock from noisy time observations, and achieves synchronization among nodes under non-deterministic network uncertainties. However, to run an algorithm on a processing unit of a sensor device, one may encounter some restrictions such as a processing speed and memory spaces. A desirable algorithm should be able to perform on the limited resources of target devices.

The developed Kalman filter-based clock synchronization algorithm was realized by utilizing restricted resources in a platform, which might degrade synchronization accuracy. The purpose of this realization was to validate the real-time operation of the synchronization algorithm. This chapter demonstrates that the algorithm realized in C runs on Pandaboards [78] and tracks clock offset and skew over a wireless connection in real time.

7.1 Implementation

The clock synchronization was implemented in C. Figure 39 shows the functional flow, where each block represents a function in the program. When a packet conveying a time record of a reference arrives, *calculate_clock()* determines an observed clock offset and skew based on send and receive timestamps. The observed clock, which is an output of *calculate_clock()*, is an input of the designed Kalman filter. The filter outputs an estimated clock offset and skew from the observed clock. *Kalman_predict()* predicts clock offset and skew based on a previous estimate and the clock models. In *Kalman_update()*, the predictions from *Kalman_predict()* are updated by the observed offset calculated in *calculate_clock()*. This process is triggered by receiving a new timestamp on a receiver.

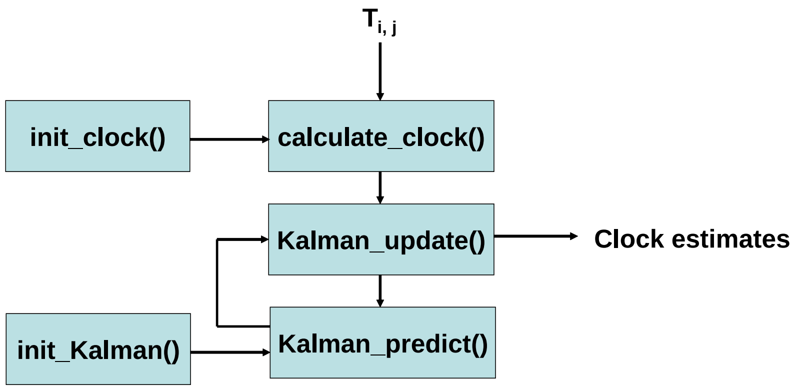


Figure 39: The functional flow of Kalman filter-based clock synchronization.

For each variable, a data type needs to be properly declared. A variable should be assigned to enough bits to avoid overflow; however, it should not waste resources and raise implementation costs. A timestamp (time record), which was recorded in μs was allocated to a 64-bit unsigned integer. Clock offset was declared as a 32-bit signed integer, which could range from -2147.483648 to 2147.483647 seconds. Clock skew was assigned to a 32-bit float, whose representable positive value was from 1.18×10^{-38} ppm to 3.4×10^{38} ppm.

For the implementation, an AR skew model and variances were assumed known. The Kalman filter was designed with a fixed AR length ($P = 3$) for simplicity. The coefficients of the AR skew model were employed from the previous measurements. $\hat{\boldsymbol{c}} = [0.4397, 0.3106, 0.1874]$, each of which was declared as a float. The variances of the Kalman filter, which were estimated based on previous measurements, were also declared as a float data type. Except timestamps, all variables and parameters were assigned to a 32-bit integer or float.

The clock synchronization technique consists of many matrix arithmetic operations especially for the calculation of Kalman gain. However, since C language does not support matrix multiplications, matrices should be broken down into components. Each component can be operated for multiplication separately and sequentially.

7.2 System and Network Setup

The system and network were set up as same as those in Chapter 5. A wireless network, conforming to IEEE802.11g was established over two Pandaboards that Ubuntu was installed on. For both the devices, NTP adjustments (both *ntpd* and *ntpdate*) and CPU frequency scaling were disabled to explore the inherent characteristics of the oscillators. Therefore, this experiment might verify the implemented algorithm under not only oscillator instability but also wireless link uncertainty.

The two nodes in the wireless network communicated with each other. One node periodically disseminated timestamps, which were the records of the local time, every minute. The other node tracked the local clock of the sender by running the proposed algorithm on its processing unit at the moment that a timestamp arrived. The clock synchronization may be deferred by pre-allocated system tasks, which have a high priority. However, by showing that the synchronization task was completed in a fairly short time, the real-time operation could be verified.

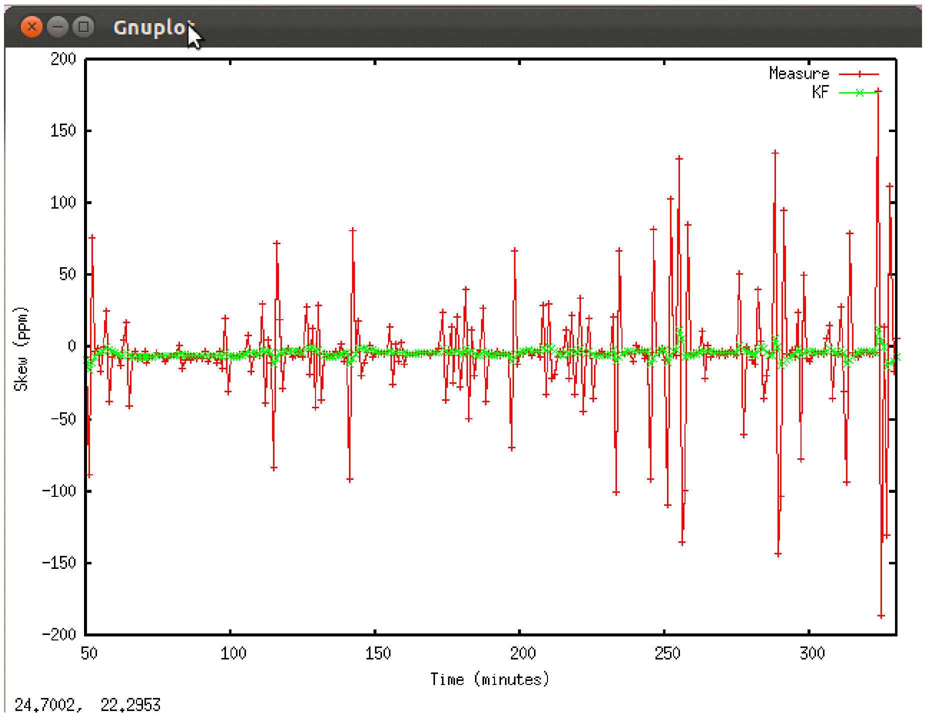
During the observation, the two nodes were placed in close proximity to eliminate interferences and be free from other access points. The two nodes were exposed to a moderate environment by being placed in a room without any temperature-controlling system. Since the aim of the implementation is to verify the real-time operation of the proposed Kalman filter-based clock synchronization, temperature measure was exempted for this experiment.

7.3 Real-time Clock Tracking

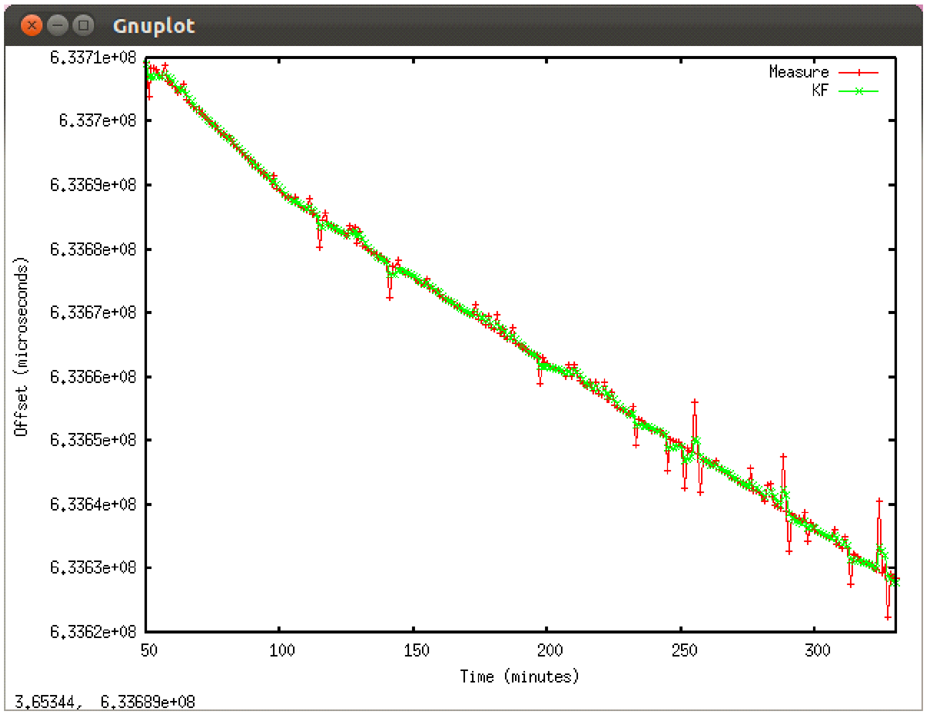
The synchronization algorithm was realized by using the hardware resources in the platform. While the algorithm was running on a receiver, the node could monitor both a time difference between the two nodes and a synchronized clock. The performance figures were plotted on the receiver by GNUPLOT, which was a command-line driven graph utility for Linux, while synchronizing the nodes.

Figure 40 shows clock drifts between the local clock of the receiver and that of the sender for each observation point. The red lines indicate clock skew and offset derived by using transmitted timestamps, which were disturbed by measurement noise and network uncertainties. The measured skew and offset shows many peaks that explains the disturbance. The total offset during five hours increased about 80 *ms* in a negative direction, which means that the average drifting rate was about -4.7 ppm. Meanwhile, the skew was not constant and ranged from -190 ppm to 175 ppm. The filtered clock skew and offset (green lines) represent the synchronized clock every minute, in other words, the estimated drift of the local clock from the sender. The execution for synchronization was completed in three to seven cycles in a 32.768kHz input clock, whose average time is 122 μs. Since the execution time is much less than a beacon period (200 ms) and a synchronization period (one minute), this clock tracking can be considered to be accomplished in real time.

The real-time tracking was compared to computer simulations in terms of synchronization performance. Similarly, one node sent its local clock reading periodically. While the receiver was estimating clock offset and skew, it also collected send, receive timestamps and clock estimates, and saved them into a file. For fair comparison, MATLAB simulations were conducted by using the collected timestamps, which were used for the implemented synchronization. The variances of the Kalman filter were set to the estimates derived by multiple experiments for both the simulation and the C implementation.



(a)

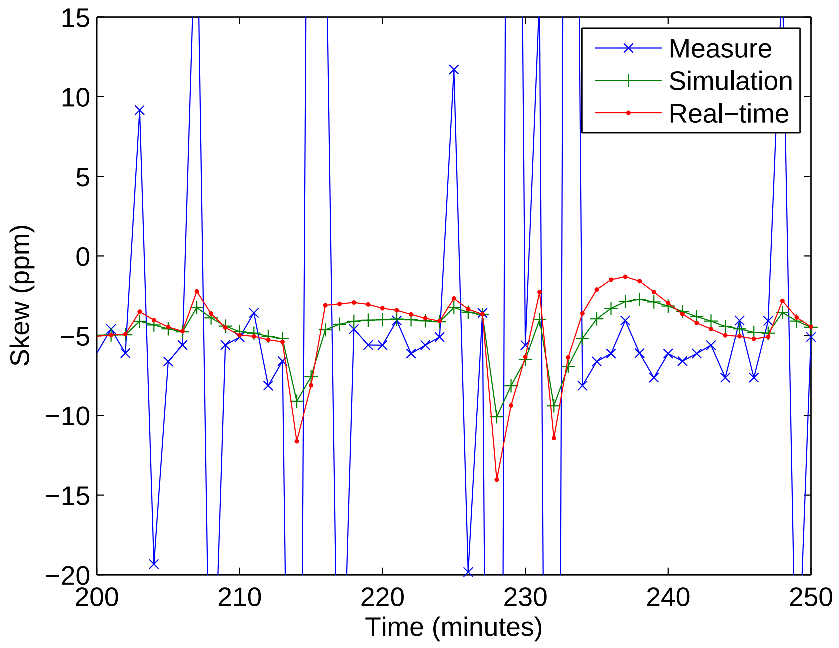


(b)

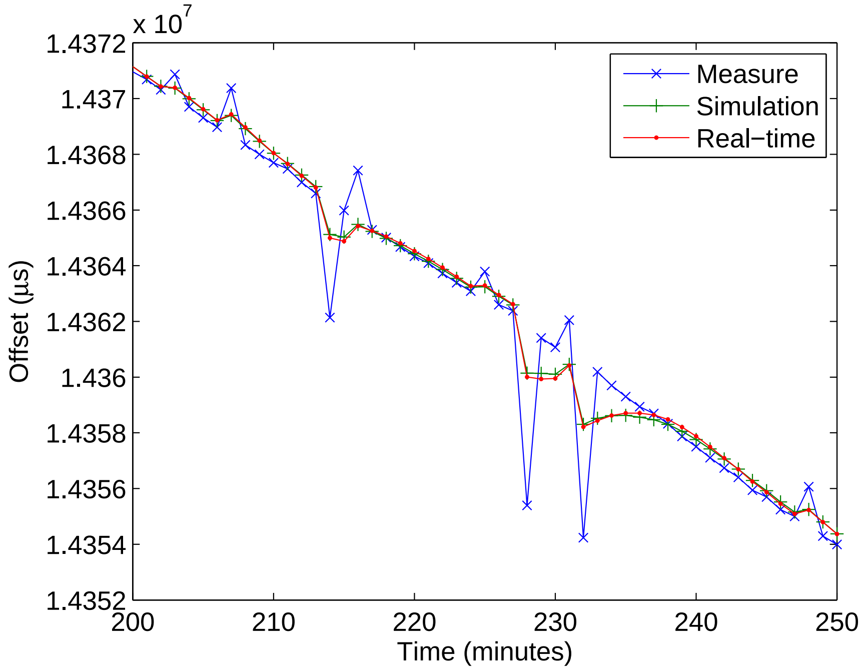
Figure 40: Real-time clock synchronization.

Figure 41 shows tracking by the C program in real time as well as tracking in a MATLAB simulation when the synchronization period is one minute. Both skew and offset estimated by on-board, real-time tracking are very close to those estimated by the simulations, which prove the operation of the implementation. This is because the platform used for this implementation, Pandaboard, supports floating-point arithmetic. In other words, the platform has sufficient computational capability for the proposed clock synchronization. However, the performance may be limited because each variable is assigned to a fixed number of bits. While MATLAB assigns a double-precision data type (64 bits) by default, all data, except timestamps, are declared to 32-bit integer or float types in C. It can be inferred that arithmetic underflow for some operations occurs during the real-time tracking.

In this work, the Kalman filter-based clock synchronization technique was implemented on a wireless platform. This experiment verified the real-time synchronization between two mobile nodes and calibrated the performance under the oscillators and link uncertainties as well as hardware limitations. It was observed that the implemented technique could run on a processor by assigning a fixed number of bits for each variable and perform in fairly short latency. The experimental results presented a possibility that the proposed synchronization technique could be used for practical applications.



(a)



(b)

Figure 41: Comparison between MATLAB simulation and C implementation at $\tau_0 = 1$ minutes, $\sigma_v^2 = 1.161635 \times 10^{10}$, and $\sigma_\eta^2 = 7.421050 \times 10^5$.

CHAPTER VIII

FIXED-POINT CLOCK SYNCHRONIZATION

Many digital signal processing algorithms are realized by using floating-point arithmetic in high-level languages for functional verification. However, to run on a fixed-point digital signal processor (DSP), the algorithms need to be implemented by using fixed-point numbers and arithmetic operations. Fixed-point implementation may lower the performance of the algorithm; however, the algorithm runs on low-cost, low-power DSPs, which give the benefits of fast execution time and low energy consumption [85, 91].

This chapter presents the conversion to fixed-point realization of the clock synchronization technique introduced in Chapter 4. Similarly, IEEE 1588 synchronization protocol was prototyped on a fixed-point DSP [41], and some Kalman filter-based algorithms were implemented by using fixed-point arithmetic operations, which allowed the algorithms to run on a fixed-point DSP [19, 53, 104].

8.1 Fixed Point Representation

Figure 42 shows the process of converting from floating-point to fixed-point algorithms. The conversion replaces floating-point with fixed-point representation for each variable. A single-precision floating-point number consists of 1 sign bit, 23 fraction bits and 8 exponent bits, and ranges to 3.40282×10^{38}. However, a 32-bit fixed-point unsigned integer varies from 0 only up to $2^{32} - 1$. Therefore, an appropriate wordlength (WL) for fixed-point representation needs to be selected not only to represent a variable without overflow but also to avoid a significant quantization error.

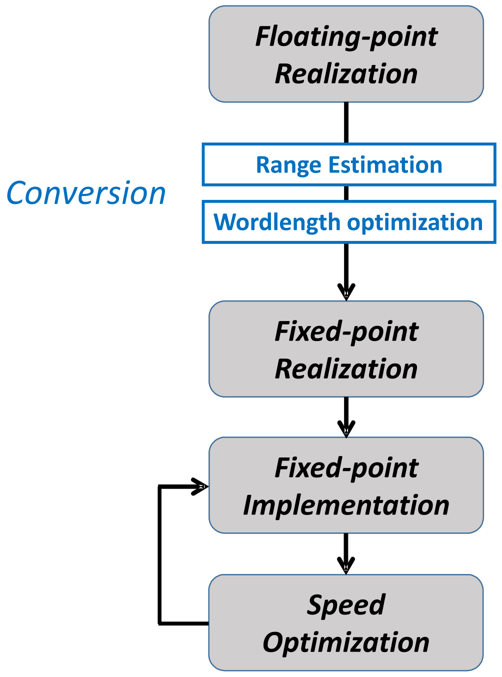


Figure 42: The conversion from floating-point to fixed-point programs.

Since the Kalman filtering is a recursive method, an output of a fixed-point arithmetic logic returns to an input of the logic through a feedback loop. In other words, the WL of the output will increase every iteration. Therefore, to achieve a stable system, it is necessary to limit wordlengths for a feedback path. However, this may cause the output to suffer from a quantization error.

The wordlength (WL) of a variable is the sum of an integer wordlength (IWL), a fraction length (FL), and a sign bit only if the variable is signed. Suppose a is a floating-point number. A fixed-point number for a can be found by $b = a \times 2^{FL}$ for a certain FL. From bit_0 to bit_{FL-1} in b are selected as a fraction part of the fixed-point number. From bit_0 to bit_{WL-1} are valid bits. A WL could be equal or larger than the number of bits of b to prevent the fixed-point number from producing a quantization error.

Finding a proper FL and WL is critical to determine the accuracy of a fixed-point implementation. An IWL is decided by $log_2(max(abs(a)))$ to avoid overflow. In other words, a different IWL can be allocated to each variable, depending on its maximum value. An FL determines the level of precision, that is, a minimum representable value. By assigning a reasonable FL, the wordlength is determined as WL = IWL + FL + S.

8.2 Range Estimation and Wordlength Decision

Determining a wordlength usually consists of two phases, a range estimation and a wordlength optimization. First, the range estimation phase searches the ranges of each variable to avoid overflow or underflow. Next, the wordlength optimization phase finds an optimum wordlength for a variable based on a searching method such as exhaustive search [97], sequential search [32], or local search [20].

For range estimation, analytical approaches calculate a range by using the propagation of variable ranges [102, 93, 75, 22]. On the other hand, statistical approaches find a minimum and maximum through simulations [21, 50, 52]. In this clock synchronization technique, a simulation based on the given coefficients provides a minimum and a maximum for each variable. As explained, since Kalman filtering is a recursive method, a simulation-based approach is appropriate.

Table 5 shows the ranges of some variables obtained by simulations. Other than several variables that represent clock offset, many variables have only fractional parts. The fixed-point representation for small fractional numbers is set to FL $>$ WL. For example, if $a = \boldsymbol{G}[2][n]$, where $\boldsymbol{G}[2][n]$ is the second component of a vector $\boldsymbol{G}$ (Kalman gain) at n^{th} iteration, then $b = a \times 2^{FL}$. If FL = 20, b ranges from 154.141 to 419.430, whose integer parts can be represented by using eight bits. Therefore, if one sign bit is added, the wordlength of $\boldsymbol{G}[2][n]$ can be decided as nine.

The accuracy of a fixed-point algorithm depends on the wordlengths used. A short WL causes distortion of a result, while a long WL can increase implementation cost. Bit widths for integer and fraction can be decided as

1. Find a range of each variable (a).

Table 5: One example of fixed point representation for each variable. $a_{max} = max(abs(a))$.

Variable	Max	Min	a_{max}	$log_2 a_{max}$	IWL	WL	FL
$\theta[n]$	152.3181	0	152.3181	7.250943578	7	13	5
$\hat{x}[1][n]$	152.3182	0	152.3182	7.250944525	7	13	5
$\hat{x}[2][n]$	0.0023	-0.0096	0.0096	-6.702749879	-7	9	15
$\boldsymbol{G}[1][n]$	0.684	0.4392	0.684	-0.54793177	-1	9	9
$\boldsymbol{G}[2][n]$	0.0004	0.000147	0.0004	-11.28771238	-12	9	20
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

2. Take $floor(log_2(abs(a_{max})))$ as a IWL.

3. Find an optimum WL.

4. Assign FL = WL - IWL - 1.

Following the process, every variable used for the clock synchronization was redefined as a fixed-point number by defining a proper WL.

8.3 Accuracy Evaluation

The *fi*, one of the functions in the Fixed-Point Toolbox in MATLAB, is used for defining a fixed-point number. The WL of an output of an arithmetic operation is determined by the WLs of its operands. However, since this is a recursive method, in other words, an output has a feedback loop to an input, intermediate variables as well as inputs and outputs are confined to limited bit widths. Moreover, since each component in a matrix has a different range, each is separately considered regarding a wordlength decision. Finding an optimum WL and FL for each variable will enhance the accuracy of fixed-point clock synchronization; at the same time, it will spare the unnecessary implementation costs. One way to optimize a WL is to probe the performance of the clock synchronization by changing a WL for each variable.

In this analysis, the dependency of bit lengths on system quality was checked by assigning the same WL for every fraction-only variable. All variables were converted into fixed point representation. An IWL for each variable was determined according to the range estimation. For the variables that had non-zero integer parts, only an FL was limited, which might have a quantization error in a fixed-point number.

Figure 43 illustrates tracking performance of fixed-point clock synchronization. The 16-bit Kalman filter reveals an underflow problem, while the 32-bit Kalman filter performs comparably to the floating-point Kalman filter. This means that the fixed-point implementation restricts the performance of clock synchronization, however, the synchronization techniques that are realized by using more numbers of bits may be able to satisfy required accuracy for certain applications.

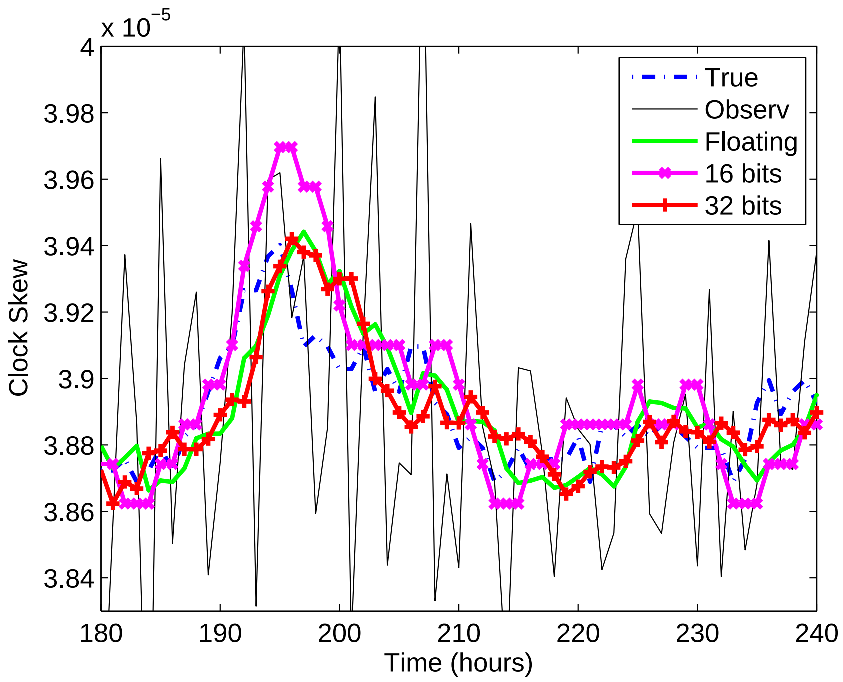
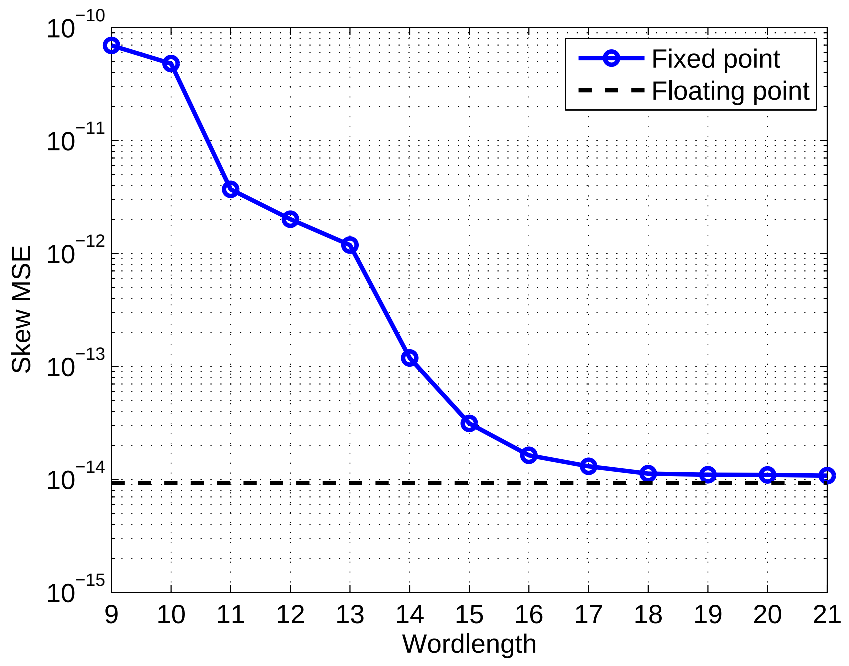


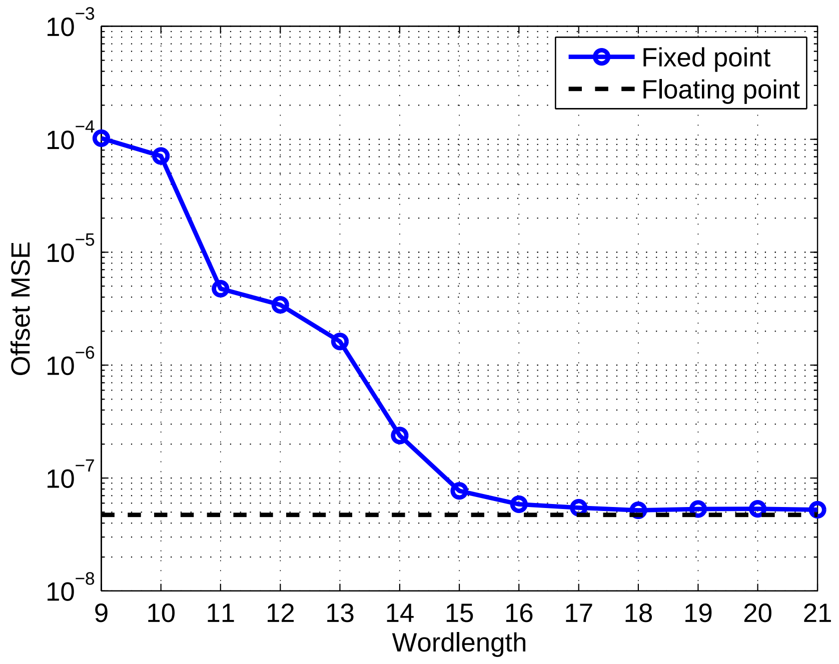
Figure 43: Tracking performance by using the fixed-point Kalman filter.

Figure 44 shows the accuracy of the clock synchronization technique that was implemented in fixed-point arithmetic. When a certain WL was given, an MSE was derived based on 1000 iterations for 100 synchronization periods. By default, a variable was declared as a double-precision data type (64 bits) in MATLAB. The floating-point clock synchronization performed the most accurately for both clock skew and offset. However, as WLs in fixed-point numbers increased, the error decreased, which illustrated that bit lengths restricted performance. When WL = 9, the tracking performance degraded more than 1000 times. However, when a WL was higher than 18, the MSEs for both clock skew and offset increased only less than 15%.

The fixed-point implementation of the proposed clock synchronization algorithm depicts that it achieves comparable quality to the floating-point algorithm and yields only 15% performance degradation by using bit widths which are much less than the single-precision data length (32 bits). Moreover, this conversion to the fixed-point synchronization may help hardware implementation since it evaluate total required bits by finding an optimum WL for the algorithm. Optimizing a WL for each variable would improve the performance of this fixed-point algorithm.



(a)



(b)

Figure 44: The MSEs for different wordlengths in fixed-point numbers.

CHAPTER IX

CONCLUDING REMARKS AND FUTURE DIRECTIONS

The objective of this research is to develop clock synchronization techniques, which are efficient for wireless networks under varying environments. First, this research explored the characteristics of clock oscillators based on time measurements and derived statistical models for clock uncertainties. Combined with the derived models, a Kalman filter was designed to track time-varying clock drifts and was verified under many unreliable circumstances through simulations. The other approach to achieve clock synchronization was to utilize environmental factors to cause an oscillator frequency to float. The experiments exposed the correlation of temperature, which was one main factor, with clock skew between two nodes in wireless links. A clock-tracking technique, which was improved by using the temperature characteristics, compensated clock drifts and enhanced tracking performance. For practical applications, the proposed technique, was implemented on mobile devices. This implementation demonstrated that one node in a network, through the technique, could estimate clocks of other nodes in real time. Moreover, the fixed-point Kalman filter, which was realized by using fixed-point arithmetic operations, presented the applicability of the technique.

9.1 Contributions

The primary contributions of this dissertation are summarized below:

- Based on the clock measurements of low-cost oscillators, a clock is modeled as an AR process whose order is determined by using information criteria.
- The derived clock models closely follow the real clock drift.

- A Kalman filter is designed to track clock uncertainties based on the clock models.
- The model and the tracking technique are evaluated by considering missing or corrupt observations which can occur in unreliable links.
- Through experiments on wireless devices, it is shown that timestamp transmissions are notably disturbed by unpredictable delays.
- Temperature is proven as an important factor to cause oscillators erroneous from real measurements.
- An M-estimator is proposed to estimate clock error induced by temperature changes.
- Temperature-compensating Kalman filter (TCKF) tracks the variation of the clock drift and thus enhances the synchronization performance.
- The implementation on devices confirms the real-time operation of the proposed techniques.
- The fixed-point realization of the Kalman filter-based technique evaluates the applicability in low-cost processors.

9.2 Future Directions

The following is a list of interesting research topics that can be pursued as extensions of this dissertation:

- Automatic or periodic updates of noise variances for the designed Kalman filter.
- Improve the resolution of CPU temperature measure to employ for the Temperature-compensating Kalman filter (TCKF).

- Hardware implementation of the proposed Kalman filter-based clock synchronization.
- Analyze required resources and timing/area constraints.
- Multi-hop, network-level simulation.
- Fast converging synchronization for resource-constraint networks.

REFERENCES

[1] "Clock oscillator stability." http://www.cardinalxtal.com/docs/notes/cardinal_clock_stability.pdf. Cardinal Components Inc. Applications Brief No. A.N. 1006.

[2] "Fundamentals of quartz oscillators," tech. rep., Hewlett Packard, May 1997.

[3] Akhlaq, M. and Sheltami, T. R., "RTSP: An accurate and energy-efcient protocol for clock synchronization in wsns," *IEEE Trans. Instrum. Meas.*, vol. 62, pp. 578–589, Mar. 2013.

[4] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E., "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, pp. 393–422, Mar. 2002.

[5] Arceo-Miquel, L., Shmaliy, Y., and Ibarra-Manzano, O., "Optimal synchronization of local clocks by gps 1pps signals using predictive fir filters," *IEEE Trans. Instrum. Meas.*, vol. 58, pp. 1833–1840, June 2009.

[6] Auler, L. and d'Amore, R., "Adaptive Kalman filter for time synchronization over packet-switched networks (an heuristic approach)," in *Proc. IEEE COMSWARE*, pp. 1–7, Jan. 2007.

[7] Barford, L. and Burch, J., "Evaluation of Kalman filtering for network time keeping," *IEEE Trans. Instrum. Meas.*, vol. 56, pp. 1601–1604, Oct. 2007.

[8] Barron, A., Rissanen, J., and Yu, B., "The minimum description length principle in coding and modeling," *IEEE Signal Processing Magazine*, vol. 44, pp. 2743–2760, Oct. 1998.

[9] Ben-El-Kezadri, R. and Pau, G., "TimeRemap: Stable and accurate time in vehicular networks," *IEEE Commun. Mag.*, vol. 48, pp. 52–57, Dec. 2010.

[10] Ben-El-Kezadri, R., Pau, G., and Claveirole, T., "TurboSync: Clock synchronization for shared media networks via principal component analysis with missing data," in *Proc. IEEE INFOCOM*, pp. 1170–1178, Apr. 2011.

[11] Bletsas, A., "Evaluation of Kalman filtering for network time keeping," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 52, pp. 1452–1460, Sep. 2005.

[12] Bovy, C., Mertodimedjo, H., Hooghiemstra, G., Uijterwaal, H., and Mieghem, P., "Analysis of end-to-end delay measurements in internet," in *Proc. Passive and Active Measurements Workshop*, pp. 26–33, Mar. 2002.

[13] Cao, X., Yang, F., Gan, X., ans Liang Qian, J. L., Tian, X., and Wang, X., “Joint estimation of clock skew and offset in pairwise broadcast synchronization mechanism,” *IEEE Trans. Commun.*, vol. 61, pp. 2508–2521, June 2013.

[14] Chaudhari, Q., Serpedin, E., and Qaraqe, K., “On maximum likelihood estimation of clock offset and skew in networks with exponential delays,” *IEEE Trans. Signal Process.*, vol. 56, pp. 1685–1697, Apr. 2008.

[15] Chaudhari, Q., Serpedin, E., and Qaraqe, K., “On minimum variance unbiased estimation of clock offset in distributed networks,” *IEEE Trans. Inf. Theory*, vol. 56, pp. 2893–2904, June 2010.

[16] Chaudhari, Q., Serpedin, E., and Wu, Y., “Improved estimation of clock offset in sensor networks,” in *Proc. IEEE ICC*, pp. 1–4, June 2009.

[17] Chen, J., Yu, Q., Zhang, Y., Chen, H.-H., and Sun, Y., “Feedback-based clock synchronization in wireless sensor networks: A control theoretic approach,” *IEEE Trans. Veh. Technol.*, vol. 59, pp. 2963–2963, July 2010.

[18] Choi, B. J., Liang, H., Shen, X. S., and Zhuang, W., “DCS: Distributed asynchronous clock synchronization in delay tolerant networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, pp. 491–504, Mar. 2012.

[19] Choi, D. and Roy, B. V., “A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning,” *Discrete Event Dynamic Systems*, vol. 16, pp. 207–239, Apr. 2006.

[20] Choi, H. and Burleson, W. P., “Search-based wordlength optimization for vlsi/dsp synthesis,” in *Proc. IEEE Workshop on VLSI Signal Processing*, pp. 198–207, Oct. 1994.

[21] Cmar, R., Rijnders, L., Schaumont, P., Vernalde, S., and Bolsens, I., “A methodology and design environment for dsp asic fixed point refinement,” in *Proc. IEEE Design, Automation and Test in Europe Conference*, pp. 271–276, Mar. 1999.

[22] Constantinides, G. A., Cheung, P. Y., and Luk, W., “wordlength optimization for linear digital signal processing,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, pp. 1432–1442, Oct. 2003.

[23] Elson, J., Girod, L., and Estrin, D., “Fine-grained network time synchronization using reference broadcasts,” *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.

[24] Exel, R., Gaderer, G., and Kero, N., “Physical layer ethernet clock synchronization,” in *42nd Annual Precise Time and Time Interval (PTTI) Meeting*, pp. 1–5, Dec. 2008.

[25] Ferrant, J.-L. and Fuffini, S., "Evolution of the standards for packet network synchronization," *IEEE Commun. Mag.*, vol. 49, pp. 132–138, Feb. 2011.

[26] Friedman, J., Hastie, T., Hofling, H., and Tibshirani, R., "Pathwise coordinate optimization," *Annals of applied statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[27] Ganeriwal, S., Kumar, R., and Srivastava, M., "Timing-sync protocol in sensor networks," in *Proc. ACM SENSYS*, pp. 1266–1273, Nov. 2003.

[28] Giorgi, G. and Narduzzi, C., "Performance analysis of Kalman-filter-based clock synchronization in IEEE 1588 networks," *IEEE Trans. Instrum. Meas.*, vol. 60, pp. 2902–2909, Oct. 2011.

[29] Giridhar, A. and Kumar, P., "Distributed clock synchronization over wireless networks: algorithms and analysis," in *Proc. IEEE CDC*, pp. 4915–4920, Dec. 2006.

[30] Greunen, J. V. and Rabaey, J., "Lightweight time synchronization for sensor networks," in *Proc. ACM Int. Conf. Wireless Sensor Networks and Applications*, pp. 11–19, Mar. 2003.

[31] Hamilton, B. R., Ma, X., Zhao, Q., and Xu, J., "ACES: adaptive clock estimation and synchronization using Kalman filtering," in *Proc. ACM MobiCom*, pp. 152–162, Sept. 2008.

[32] Han, K., I. Eo, K. K., and Cho, H., "Numerical word-length optimization for cdma demodulator," in *Proc. IEEE Int. Sym. on Circuits and Systems*, pp. 290–293, May 2001.

[33] Hayes, M. H., *Statistical Digital Signal Processing and Modeling.* Wiley, first ed., 1996.

[34] Huang, H., Yun, J., Zhong, Z., Kim, S., and He, T., "PSR: Practical synchronous rendezvous in low-duty-cycle wireless networks," in *Proc. IEEE INFOCOM*, pp. 2661–2669, Apr. 2013.

[35] Huang, P., Desai, M., Qiu, X., and Krishnamachari, B., "On the multihop performance of synchronization mechanisms in high propagation delay networks," *IEEE Trans. Comput.*, vol. 58, pp. 577–590, May 2009.

[36] Huang, W., Jin, Y., Wang, W., Shi, Y., and Zhou, Y., "Hardware-based solution of precise time synchronization for networked control system," in *Proc. ICECC*, pp. 4324–4328, Sept. 2011.

[37] "IEEE std. 1588 - 2002 IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2002*, 2002.

[38] Jeske, D., "On the maximum likelihood estimation of clock offset," *IEEE Trans. Commun.*, vol. 53, pp. 53–54, Jan. 2005.

[39] Jones, R. H. and Boadi-Boateng, F., "Unequally spaced longitudinal data with ar(1) serial correlation," *Biometrics*, vol. 47, no. 1, pp. 161–175, 1991.

[40] Joshi, S. G. and Brace, J. G., "Measurement of humidity using surface acoustic waves," in *IEEE Ultrasonics symposium*, pp. 600–603, Oct. 1985.

[41] Kannisto, J., Vanhatupa, T., Hnnikinen, M., and Hmlinen, T. D., "Software and hardware prototypes of the IEEE 1588 precision time protocol on wireless lan," in *IEEE Workshop on Local and Metropolitan Area Networks*, pp. 1–6, Sep. 2005.

[42] Karp, R., Elson, J., Papadimitriou, C., and Shenker, S., "Global synchronization in sensornets," in *Proc. Latin American Symp. Theoretical Informatics*, pp. 609–624, Apr. 2004.

[43] Kay, S., "Conditional model order estimation," *IEEE Trans. Signal Process.*, vol. 49, pp. 1910–1917, Sep. 2001.

[44] Kay, S., "Exponentially embedded families-new approaches to model order estimation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, pp. 333–345, Jan. 2005.

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

[46] Kim, H., Ma, X., and Hamilton, B., "Modeling and tracking clocks with time-varying skews using information criteria," in *Proc. IEEE MILCOM*, pp. 1747–1752, Nov. 2010.

[47] Kim, H., Ma, X., and Hamilton, B. R., "Tracking low-precision clocks with time-varying drifts using Kalman filtering," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 257–270, Feb. 2012.

[48] Kim, J.-S., Lee, J., Serpedin, E., and Qaraqe, K., "A robust clock synchronization algorithm for wireless sensor networks," in *Proc. IEEE ICASSP*, pp. 3512–3515, May 2011.

[49] Kim, K. and Lee, B., "KALP: A Kalman filter-based adaptive clock method with low-pass prefiltering for packet networks use," *IEEE Trans. Commun.*, vol. 48, July 2000.

[50] Kim, S., Kum, K., and Sung, W., "Fixed-point optimization utility for c and c++ based digital signal processing programs," *IEEE Trans. Circuits Syst.*, vol. 45, pp. 1455–1464, Nov. 1998.

[51] Kitagawa, G. and Gersch, W., *Smoothness priors analysis of time series.* Springer, first ed., 1996.

[52] Kum, K. and Sung, W., "Combined word-length optimization and high-level sysnthesis of digital signal processing systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, pp. 921–930, Aug. 2001.

[53] Labbate, M., Petrella, R., and Tursini, M., "Fixed point implementation of Kalman filtering or AC drives: a case study using TMS320F24x DSP," in *Proc. European DSP Education and Research Conference*, pp. 20–24, Sep. 2000.

[54] Lemmon, M., Ganguly, J., and Xia, L., "Model-based clock synchronization in networks with drifting clocks," in *Proc. Pacific Rim Int. Symp. Dependable Computing*, pp. 177–184, Dec. 2000.

[55] Leng, M. and Wu, Y., "On clock synchronization algorithms for wireless sensor networks under unknown delay," *IEEE Trans. Veh. Technol.*, vol. 59, pp. 182–190, Jan. 2010.

[56] Leng, M. and Wu, Y.-C., "Distributed clock synchronization for wireless sensor networks using belief propagation," *IEEE Trans. Signal Process.*, vol. 59, pp. 5404–5414, Nov. 2011.

[57] Leng, M. and Wu, Y.-C., "Low complexity maximum likelihood estimators for clock synchronization of wireless sensor nodes under exponential delays," *IEEE Trans. Signal Process.*, vol. 59, pp. 4860–4870, Oct. 2011.

[58] Lenzen, C., Sommer, P., and Wattenhofer, R., "Optimal clock synchronization in networks," in *Proc. ACM SenSys*, pp. 225–238, Nov. 2009.

[59] Li, Q. and Rus, D., "Global clock synchronization in sensor networks," in *Proc. IEEE INFOCOM*, pp. 564–574, Mar. 2004.

[60] Liao, C., Martonosi, M., and Clark, D. W., "Experience with an adaptive globally-synchronizing clock algorithm," in *Proc. ACM Symposium on Parallelism Algorithms and Architectures*, pp. 106–114, June 1999.

[61] Liu, J., Wang, Z., Zuba, M., Peng, Z., Cui, J.-H., and Zhou, S., "JSL: Joint time synchronization and localization design with stratication compensation in mobile underwater sensor networks," in *Proc. IEEE SECON*, pp. 317–325, June 2012.

[62] Loschmidt, P., Exel, R., Nagy, A., and Gaderer, G., "Limits of synchronization accuracy using hardware support in IEEE 1588," in *IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pp. 12–16, Sep. 2008.

[63] Luo, B. and Wu, Y. C., "Distributed clock parameters tracking in wireless sensor network," *IEEE Trans. Wireless Commun.*, vol. 12, pp. 6464–6475, Dec. 2013.

[64] Magee, A., "Synchronization in next-generation mobile backhaul networks," *IEEE Commun. Mag.*, vol. 48, pp. 110–116, Oct. 2010.

[65] Maggs, M. K., OKeefe, S. G., and Thiel, D. V., "Consensus clock synchronization for wireless sensor networks," *IEEE SENSORS JOURNAL*, vol. 12, pp. 2269–2277, June 2012.

[66] Maròti, M., B. Kusy, G. S., and Lèdeczi, A., "The flooding time synchronization protocol," in *Proc. ACM SENSYS*, pp. 39–49, Nov. 2004.

[67] Meier, L., Blum, P., and Thiele, L., "Internal synchronization of drift-constrained clocks in ad-hoc sensor networks," in *Proc. ACM MobiHoc*, pp. 90–97, May 2004.

[68] MEMSIC, I., "Mica2 datasheet." http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html.

[69] Mills, D. L., "Internet time synchronization: the network time protocol," *IEEE Trans. Commun.*, vol. 39, pp. 1482–1493, Oct. 1991.

[70] Mills, D. L., "Improved algorithms for synchronizing computer network clocks," *IEEE/ACM Trans. Netw.*, vol. 3, pp. 245–254, June 1995.

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

[72] Moinzadeh, P., Mechitov, K., Shiftehfar, R., Abdelzaher, T., Agha, G., and Spencer, B., "The time-keeping anomaly of energy-saving sensors: Manifestation, solution, and a structural monitoring case study," in *Proc. IEEE SECON*, pp. 308–388, June 2012.

[73] Moon, S. B., Skelly, P., and Towsley, D., "Estimation and removal of clock skew from network delay measurements," in *Proc. IEEE INFOCOM*, vol. 1, pp. 227–234, Mar. 1999.

[74] Murdoch, S. J., "Hot or not: Revealing hidden services by their clock skew," in *Proc. ACM CCS*, pp. 27–36, Nov. 2006.

[75] Nayak, A., Haldar, M., Choudhary, A., and Banerjee, P., "Precision and error analysis of matlab applications during automated hardware synthesis for fpgas," in *Proc. IEEE Design, Automation and Test in Europe Conference*, pp. 722–728, Mar. 2001.

[76] Noh, K., Chaudhari, Q. M., Serpedic, E., and Suter, B. W., "Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks," *IEEE Trans. Commun.*, vol. 55, pp. 766–777, Apr. 2007.

[77] PalChaudhuri, S., Saha, A., and Johnson, D., "Adaptive clock synchronization in sensor networks," in *Proc. IEEE IPSN*, pp. 340–348, Apr. 2004.

[78] Pandaboard.org, "User mauals." http://pandaboard.org/sites/default/files/board_reference/pandaboard-a/panda-a-manual.pdf, Nov. 2010.

[79] Pàsztor, A. and Veitch, D., "PC based precision timing without GPS," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, pp. 1–10, June 2002.

[80] Phillips, J. and Kundert, K., "Noise in mixers, oscillators, samplers, and logic: an introduction to cyclostationary noise," in *Proc. IEEE CICC*, pp. 431–438, May 2000.

[81] Pottie, G. and Kaiser, W., "Wireless integrated network sensors," *Communications of ACM*, vol. 43, pp. 51–58, May 2000.

[82] Ribeiro, A., Giannakis, G. B., and Roumeliotis, S. I., "SOI-KF: Distributed Kalman filtering with low-cost communications using the sign of innovations," *IEEE Trans. Signal Process.*, vol. 54, pp. 4782–4795, Dec. 2006.

[83] Ridder, F. D., Pintelon, R., Schoukens, J., and Gillikin, D. P., "Modified aic and mdl model selection criteria for short data records," *IEEE Trans. Instrum. Meas.*, vol. 54, pp. 144–150, Feb. 2005.

[84] Romer, K., "Time synchronization in ad hoc networks," in *Proc. ACM MobiHoc*, pp. 173–182, Oct. 2001.

[85] Roy, S. and Banerjee, P., "An algorithm for converting floating-point computations to fixed-point in matlab based fpga design," in *Proc. IEEE Design Automation Conference*, pp. 484–487, June 2004.

[86] Sadler, B. M., "Fundamentals of energy-constrained sensor network systems," *IEEE AESS magazine*, vol. 20, pp. 17–35, Aug. 2005.

[87] Schmid, T., Charbiwala, Z., Friedman, J., Cho, Y. H., and Srivastava, M. B., "Exploiting manufacturing variations for compensating environment-induced clock drift in time synchronization," in *Proc. ACM Sigmetrics*, pp. 97–108, June 2008.

[88] Schmid, T., Charbiwala, Z., Shea, R., and Srivastava, M. B., "Temperature compensated time synchronization," *IEEE Embedded Systems Letters*, vol. 1, pp. 37–41, Aug. 2009.

[89] Scopigno, R. and Cozzetti, H. A., "GNSS synchronization in vanets," in *Proc. NTMS*, pp. 1–5, Dec. 2009.

[90] Sichitiu, M. and Veerarittiphan, C., "Simple, accurate time synchronization for wireless sensor networks," in *Proc. IEEE WCNC*, pp. 1266–1273, Mar. 2003.

[91] Singh, C. K., Prasad, S. H., and Balsara, P. T., "A fixed-point implementation for qr decomposition," in *IEEE Dallas/CAS Workshop on Design, Applications, Integration and Software*, pp. 75–78, Oct. 2006.

[92] Solis, R., Borkar, V., and Kumar, P. R., "A new distributed time synchronization protocol for multihop wireless networks," in *Proc. IEEE CDC*, pp. 2734–2739, Dec. 2006.

[93] Stephenson, M., Babb, J., and Amarasinghe, S., "Bitwidth analysis with application to silicon compilation," in *Proc. SIGPLAN Program. Lang. Design Implementation*, pp. 108–120, June 2000.

[94] Stoica, P. and Selen, Y., "Model-order selection: a review of information criterion rules," *IEEE Signal Processing Magazine*, vol. 21, pp. 36–47, July 2004.

[95] Su, W. and Akyildiz, I., "Time-diffusion synchronization protocols for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 13, pp. 384–397, Apr. 2005.

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

[97] Sung, W. and Kum, K., "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Trans. Signal Process.*, vol. 43, pp. 3087–3090, Dec. 1995.

[98] Tibshirani, R., "Regression shrinkage and selection via the lasso," *J. R. Statist. Soc. B*, vol. 58, no. 1, pp. 267–288, 1994.

[99] Tulone, D., "Resource-efficient time estimation for wireless sensor networks," in *Proc. ACM DIALM-POMC*, pp. 52–59, Oct. 2004.

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

[101] Vig, J. R., "Introduction to quartz frequency standards," Tech. Rep. SLCET-TR-92-1 (Rev. 1), Army Research Laboratory, Oct. 1992.

[102] Wadekar, S. A. and Parker, A., "Accuracy sensitive sensitive word-length selection for algorithm optimization," in *Proc. ICCD*, pp. 54–61, Oct. 1998.

[103] Walls, F. B. and jacques Gagnepain, L., "Environmental sensitivities of quartz oscillators," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 39, pp. 241–249, Mar. 1992.

[104] Wang, J., "Ballbot: A low-cost robot for tennis ball retrieval," Tech. Rep. No. UCB/EECS-2012-157, University of California at Berkeley, Electrical Engineering and Computer Sciences, June 2012. http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-157.html.

[105] Wu, Y., Chaudhari, Q., and Serpedin, E., "Clock synchronization of wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 28, pp. 124–138, Jan. 2011.

[106] Xueqiao, L., Yuan, C., Shuang, L., and Yaxing, D., "Implementation and research of hardware time stamping techniques based on IEEE1588," in *Proc. ICCSN*, pp. 6–9, May 2011.

[107] Yang, Z., Cai, L., Liu, Y., and Pan, J., "Environment-aware clock skew estimation and synchronization for wireless sensor networks," in *Proc. IEEE INFOCOM*, pp. 1017–1025, Mar. 2012.

[108] Zhang, L., Liu, Z., and Xia, C. H., "Clock synchronization algorithms for network measurements," in *Proc. IEEE INFOCOM*, vol. 1, pp. 160–169, Mar. 2002.

[109] Zhang, W., Branicky, M., and Phillips, S., "Stability of networked control systems," *IEEE Trans. Control Syst. Technol.*, vol. 21, pp. 84–99, Feb. 2001.

[110] Zhang, Y., Ge, Z., Greenberg, A., and Roughan, M., "Network anomography," in *Proc. ACM SIGCOMM IMC*, pp. 30–30, Oct. 2005.

[111] Zhou, D. and Lai, T., "A scalable and adaptive clock synchronization protocol for IEEE 802.11-based multihop ad hoc networks," in *Proc. IEEE MASS*, pp. 551–558, Nov. 2005.

[112] Zhou, H., Nicholls, C., Kunz, T., and Schwartz, H., "Frequency accuracy & stability dependencies of crystal oscillators," Tech. Rep. SCE-08-12, Carleton University, Systems and Computer Engineering, Nov. 2008.

[113] Zhu, H., Giannakis, G. B., and Leus, G., "Weighted and structured sparse total least squares for perturbed compressive sampling," in *Proc. IEEE ICASSP*, pp. 3792–3795, May 2011.

VITA

Ha Yang Kim was born in South Korea on May 19, 1980. She received the B.S. degree in Electronics Engineering in 2003, and the M.S. degree in Information Electronics in 2005 from Ewha Womans University, Seoul, South Korea. From 2005 to 2008, she had worked for LG Electronics, Seoul, South Korea, as a design engineer. She is now working towards the Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, Georgia, USA.