# FLIGHT: Clock Calibration Using Fluorescent Lighting

Zhenjiang Li[1,4], Wenwei Chen[1], Cheng Li[1], Mo Li[1], Xiang-yang Li[2], Yunhao Liu[3,4]
[1]Nanyang Technological University, Singapore
[2]Illinois Institute of Technology, USA
[3]Tsinghua University, China
[4]Hong Kong University of Science and Technology, Hong Kong
{lzjiang, chen0746, cli6, limo}@ntu.edu.sg, xli@cs.iit.edu, yunhao@greenorbs.com

## ABSTRACT

In this paper, we propose a novel clock calibration approach called FLIGHT, which leverages the fact that the fluorescent light intensity changes with a stable period that equals half of the alternating current's. By tuning to the light emitted from indoor fluorescent lamps, FLIGHT can intelligently extract the light period information and achieve network wide time calibration by referring to such a common time reference. We address a series of practical challenges and implement FLIGHT in TelosB motes. We conduct comprehensive experiments using a 12-node test-bed in both static and mobile environments. Over one-week measurement suggests that compared with existing technologies, FLIGHT can achieve tightly synchronized time with low energy consumption.

## Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols; C.2.4 [**Computer Communication Networks**]: Distributed Systems

## General Terms

Design, Algorithms, Performance

## Keywords

Clock calibration, Fluorescent lighting, Energy efficiency

## 1. INTRODUCTION

Maintaining a common notion of time is one fundamental service in many distributed networking systems. A variety of applications depend on the availability of tightly synchronized time cross network nodes. For instance, the emerging High-speed Locational, Phone-to-Phone (HLPP) gaming [1] with a real-time requirement would benefit from such a service. A series of network-wide events must follow a strict common time order on different phones. In body area networks and healthcare monitoring networks [2–4], sensory data from multiple devices are usually processed to cooperatively analyze certain body movements or events

of interest. Consistent time clocks among different devices are necessary for correct time alignment and data processing. Other typical application examples include wireless sensor networks, mobile ad hoc networks, etc.

Retaining a common time in the network, however, faces substantial challenges. Due to the low-cost design, CMOS crystal oscillators serve as the most privileging signal source to generate on-board clocks. The frequency of an oscillator is not stable and it fluctuates with the surrounding environment, affected by various factors like temperature, humidity, voltage, pressure, etc. For example, the $32KHz$ oscillator adopted by popular sensor platforms has a clock drift rate of 30-50 ppm [5]. The clock drift in smart phones normally ranges from 5-100 ppm [6, 7]. Thus, clocks on different nodes need to be precisely calibrated for correcting above errors such that consistent time can be maintained. Due to the inherent uncertainty, no matter how accurately clocks are initially calibrated, they will ultimately tick towards divergency. Hence, frequent clock calibrations are necessary in most network systems. Existing networks may typically comprise hundreds or even more individual nodes, and those nodes are usually wireless interconnected in a multi-hop manner. As a direct consequence, frequently calibrating clocks prohibitively incurs high overhead and significant energy consumption for coordinating the entire network.

Great efforts have been made in the past decade to address above issues. Proposed solutions mainly rely on heavy intercommunications between nodes to exchange their local time references, including RBS [8], TPSN [9], FTSP [10], etc. The calibration error, however, gets accumulated hop by hop exponentially as the clock calibration spreads to the entire network [11]. In addition, those solutions lead to excessively high communication overhead and significant power drain in the system. Such facts largely prohibit the scalability of those solutions in practice. Against the problems, an emerging type of solutions have been recently proposed, which make use of certain external signal sources as common time references. Typical examples are using power lines [12], FM radio [13], radio stations [14], Wi-Fi [15]. Although such newly emerging solutions can dramatically reduce the communication cost, they also utilize radios for the clock calibration at the expense of higher power consumption. Even if clock calibration in [12–15] can be performed less frequently, the energy consumption of *each* calibration could remain high. Besides, most of those solutions need specific hardware components to generate or capture periodical signals, which introduce extra cost and design complexity.

In this paper, we propose a new clock synchronization approach that allows network nodes to receive a globally available clock reference by tuning to the fluorescent lighting. Our approach is based on the following observations and facts. Alternating Current (AC) is a periodical signal with a frequency of 50 or $60Hz$ and the fre-

quency has adequate stability of $5 \cdot 10^{-5}$ measured by the Allan variance [12, 16]. In order for power to be delivered efficiently across the country, the phase difference between any two points remain highly constant [12, 16]. Powered by AC, the fluorescent light intensity thus changes with a stable period that equals half of the alternating current's. Commonly available in most indoor environments like universities, airports, hospitals, and supermarkets, the fluorescent light provides a universal period reference. On the other hand, the light sensor or camera is widely embedded in commodity wireless platforms, e.g., sensor nodes, and smart phones. This offers a great opportunity for a plenty of indoor applications to maintain common time by referring to the light emitted from fluorescent lamps.

The proposed solution has several key advantages. First, it does not require any extra hardware support. By sampling the light sensor or camera with a sufficiently high rate, accurate period information can be detected for clock calibration. Hence, our solution is widely available on most existing commodity platforms. Second, taking advantage of the stability of the AC frequency, the detected light intensity, even from different lamps, exhibits a consistent and stable period, which ensures high synchronization accuracy. Third, compared with prior radio operation based solutions, sampling the light sensor consumes substantially less energy, which provides us a lightweight solution. Finally, since our approach is independent to the network message exchange, time synchronization can be retained even when the network is temporarily disconnected. When individual devices are moving, they can still obtain desired periodical patterns for clock calibration. Such characteristics particularly suit various mobility-enabled scenarios.

Successfully implementing our idea is non-trivial. We first empirically study the feasibility of detecting periods from the fluorescent lighting. Starting from static deployment in the laboratory, our measurements validate that the fluorescent lighting is able to serve as a viable reference for time calibration. We then propose the Fluorescent LIGHTing (FLIGHT) approach for achieving a low-power and accurate clock calibration. FLIGHT can intelligently extract the period information from the fluorescent light in both static and dynamic environments. In addition, to achieve a lightweight FLIGHT implementation, we address a sequence of practical issues including responsive light sensor sampling, period detection with interference from other light sources, etc. To evaluate our design, we implement FLIGHT in TelosB motes and conduct extensive experiments using a 12-node test-bed in the laboratory and further with mobility across the university main academic building of over $20,000m^2$. The results demonstrate that FLIGHT can achieve tightly synchronized time with low energy consumption.

The rest of this paper is organized as follows: In Section 2, we introduce the preliminary information and perform a measurement study. The architecture and the design detail of FLIGHT are given in Sections 3 and 4, respectively. In Section 5, we evaluate FLIGHT's performance through extensive experiments. In Section 6, we introduce the limitations and future works of this paper. Related works are reviewed in Section 7 and we conclude this paper in Section 8.

## 2. PRELIMINARY

In this section, we introduce preliminary concepts of this paper. We first discuss the clock calibration problem that this paper studies and compare it with the traditional clock synchronization problem. We then detail the target applications for clock calibration. We illustrate the principle of fluorescent lighting and present our initial empirical measurement study that motivates this work. To facilitate the discussions, we tabulate key notations in Table 1.

**Table 1: Notation**

| | |
|---|---|
| $c_n(t)$ | Native clock at time $t$ |
| $c_g(t)$ | Global reference clock at time $t$ |
| $c_l(t)$ | Logic clock at time $t$ |
| $f_n(t)$ | Frequency of the native clock at time $t$ |
| $f_g$ | Frequency of the fluorescent light |
| $\alpha(t)$ | Frequency ratio: $f_n(t)/f_g$ |

### 2.1 Clock calibration v.s. synchronization

A *native clock* is the clock driven by a node's built-in crystal oscillator directly. We denote the native clock as $c_n(t_0 + t)$, referring to the measured time duration from an initial time $t_0$ until $t$ by the native clock. Without loss of generality, we assume $t_0$ to be zero; hence the native clock can be written as $c_n(t)$ for short. The periodical pattern of the fluorescent light intensity can be treated as a *global reference clock*, which is denoted as $c_g(t)$. In our system, the AC frequency is $50Hz$; thus, the frequency of the global reference clock $f_g$ is $100Hz$ and the time unit of the global reference clock is $1/(100Hz) = 10ms$. In addition, each node maintains a *logic clock*, written as $c_l(t)$. The logic clock is used by upper-layer applications and it advances as follows:

$$c_l(t) = c_l(0) + \int_0^t r(\tau)d\tau, \tag{1}$$

where $r(\tau)$ is the instant rate of the native clock at time $\tau$. The goal of FLIGHT is to ensure **logic clocks** consistent among different nodes.

*Clock synchronization* in prior literatures [8–10] ensures the absolute clock values of different nodes to be consistent, while in this work we primarily focus on *clock calibration*, which mainly ensures that different clocks advance with a same speed. Referring to Eq. (1), clock synchronization essentially first compensates the difference of the starting time $c_l(0)$ of each node and further maintains consistent clock ticks for $\int_0^t r(\tau)d\tau$. On the contrary, clock calibration only guarantees a consistent clock rate for $\int_0^t r(\tau)d\tau$ despite of different node starting times. Therefore, with accurate clock calibration, clock synchronization can be easily achieved by compensating the initial time differences of nodes, while sole clock calibration already suffices to support many practical applications. In this paper, we first propose FLIGHT to achieve an accurate clock calibration, and then we discuss how FLIGHT can be integrated with existing mechanisms (e.g., FTSP [10]) to achieve precise clock synchronization with much reduced communication and energy overhead.

### 2.2 Target applications

Nowadays, the fluorescent lighting resource is commonly available in most indoor environments. Due to the stability of AC, fluorescent lighting can provide a universal period reference in such environments. Meanwhile, the light sensor or camera has been widely embedded in commodity wireless platforms. This offers a great opportunity for a plenty of indoor applications to maintain common time by referring to the light emitted from fluorescent lamps. The primary applications of FLIGHT are those, in which a precise timing service across indoor environments is desired with minimal communications. For instance, body area networks and healthcare monitoring networks need to detect and report a series of gestures, movements, and accidents (e.g., falling down of a patient). Clock calibration could be adopted in such applications since event detections rely on an accurate data timing sequence.
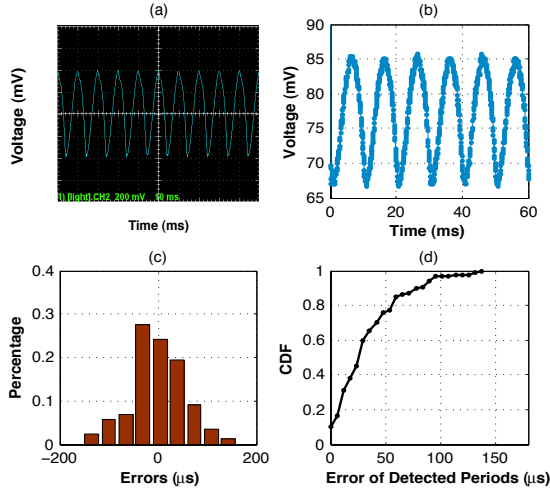
**Figure 1: Single-lamp experiment in the laboratory**



**Figure 2: Multi-lamp experiment in the laboratory**

Absolute time values of sensory data, however, are not necessarily needed. In such a scenario, nodes are attached to human bodies. The miniature requirement on the node size prevents prior external signal based approaches, e.g., [12–14], from being applied due to the extra hardware requirement. On the other hand, the node density in such applications could be high. Frequent message exchanges of radio-based approaches, like [8–10], can influence the regular event detection and report. Against such issues, body area and healthcare monitoring networks would greatly benefit from the design of FLIGHT.

In wireless sensor networks, clock calibration can support many services as well, like the target tracking, the topology control, and the event logging for diagnosis where the order of events rather than their precise time matters [17–19]. Sensor networks normally consist of hundreds, or even thousands of nodes [20]. Extra hardware components [12–14] may dramatically increase the deployment and maintenance costs. On the other hand, the useable bandwidth resources of sensor nodes are usually limited (e.g., due to low data rates [21] and duty-cycled operation mode [22]). FLIGHT is parallel to the wireless communication, which saves the bandwidth resource and simplifies the MAC protocol design.

Emerging HLPP gaming can also benefit from FLIGHT, such as multi-player games by using phones as props (e.g., playing table tennis [1]). With the help of clock calibration, the captured movements on different devices can be ordered to precisely infer gestures. Sensory data from players can also be analyzed based on such time sequences to make accurate judgement. Different players may be in different rooms or buildings, and radio-based approaches thus fail to work in such applications. Due to the inconvenience brought by the extra hardware in the game, solutions like [12–14] may not be suitable candidates either. FLIGHT, on the other hand, can overcome both barriers in this type of applications.

There is no strict requirement on the space and time coverage of fluorescent lamps in FLIGHT. As we will show later, the period information can be extracted even when the detected light intensity is very low. One fluorescent lamp, thus, can coordinate plenty of surrounding nodes. Fluorescent lighting from different lamps can exhibit a stable and consistent period. Hence, lamps can be sparingly used in the field. In our experiment, only a small number of lamps can coordinate all the nodes in our laboratory of $600m^2$. Moreover, lamps can also coordinate nodes in different rooms even different
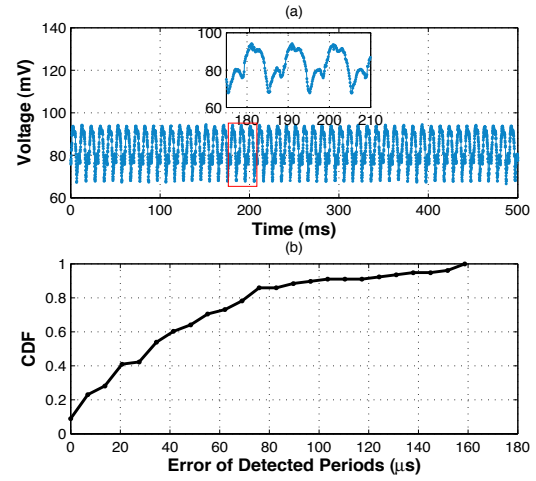
buildings. Therefore, in order for the proper working of FLIGHT, the only requirement is to turn on several fluorescent lamps, which is always available in many places, like universities, hospitals, airports, supermarkets, etc.

## 2.3 Empirical measurement study

The fluorescent lamp is a gas discharge lamp using electricity to excite mercury vapor. When the lamp is turned on, the electric power heats up the cathode to emit electrons. Electrons quickly jump from a higher energy level to a lower level and photons will be emitted. Emitted photons are absorbed by electrons in the atoms of the interior fluorescent coating of lamp, leading to physical reactions with emission of visible light. After being heated up, the gas conductivity inside the lamp rapidly rises, allowing the alternating current to flow through and continuously emit light. From the principle of fluorescent lighting, the emitted light is expected to exhibit a periodical pattern since its AC power source is a periodical signal. As the period of AC is adequately stable and phase-coherent, we expect that the light generated by fluorescent lamps can be used as a periodic reference for clock calibration.

We focus on evaluating following two properties of the fluorescent lighting strategy: *stability* and *accuracy* of detected light periods. In the rest of this paper, we mainly take standard TelosB motes as a vehicle to present the measurement results and the design principle. Due to the limited computation capacity and buffer size, design challenges of FLIGHT for TelosB motes are more general. Dealing with the performance optimization in other high-end devices, like smart phones and laptops, can further serve as a promising future work of this paper. In the experiments, 12 TelosB motes are deployed in the laboratory and the longest distance between any two pairs of sensor nodes is up to $15m$. We report the results based on one week measurement. Among those 12 deployed TelosB nodes, the minimum distance to a light lamp is around $20cm$ and the maximum distance is approximately $7m$. The detected light intensity is measured by the voltage value of two pins of the light sensor.

We start from examining a simple case, in which only one lamp is open. As a benchmark, we plot the instant voltage value from two frontend pins of the light sensor using an oscilloscope. Through one week measurement, we observe that the reading from oscilloscope is quite stable and exhibits a regular fluctuation with a period
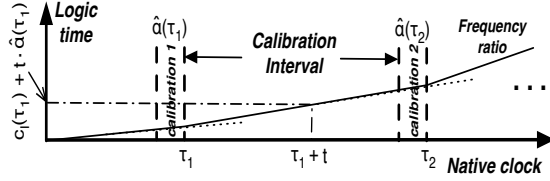
**Figure 3: Illustration of periodical calibrations**

of 10*ms* (depicted in Figure 1(a)). In practice, sensor nodes need to sample the light sensors. After sampling, we see that the sampling results exhibit the same periodical property as depicted in Figure 1(b), which suggests that the obtained light intensity pattern can be used for clock calibration. Figure 1(c) summarizes the statistics of the detected period lengths from the trace. We observe that the detected period lengths are highly concentrated within the range centered at 10*ms*. The error of detection is less than 50*μs* for most cases and it is bounded by 145*μs* as depicted in Figure 1(d).

We further examine the case where there are more than one lamps. During the measurement, 200 fluorescent lamps in our laboratory are all turned on. Nodes may extract period patterns from different lamps. Similar to the single-lamp case, readings from different nodes all exhibit good periodical patterns of 10*ms* duration. Different distances to a lamp only incur different amplitudes of the detected patterns. The periodical property always exists. In Figure 2(a), we plot a portion of the trace from the sensor node with the largest distance to its closest lamp. Note that in one period, three local maximum points do not mean the light pattern in Figure 2(a) is from exact three different lamps. They only represent phase differences. Each local maximum point of the light intensity might be composed of the light from multiple lamps with the same phase. In Figure 2(b), we further depict the statistical result of detected period lengths of all the nodes with respect to different lamps. Based on the trace, the error is normally smaller than 65*μs*.

In summary, our initial empirical studies validate the stability and accuracy of fluorescent lighting in practice. Later we will make use of such lighting period information for clock calibration in FLIGHT.

# 3. SYSTEM OVERVIEW

In this section, we first introduce the basic idea of the fluorescent lighting based clock calibration scheme. We then give an overview of the design challenges as well as the FLIGHT architecture and implementation.

## 3.1 Principle of clock calibration

The objective of FLIGHT is to ensure logic clocks consistent among different nodes. In Section 2.1, Eq. (1) has stated that the logic clock evolves as $c_l(t) = c_l(0) + \int_0^t r(\tau)d\tau$, where $r(\tau)$ is the instant rate of the native clock at time $\tau$. Ideally, we have $\int_0^t r(\tau)d\tau = c_n(t)/f_n$, where $f_n$ represents the claimed frequency of the built-in crystal oscillator. In practice, however, $f_n$ is not stable and it fluctuates due to the variances of surrounding environmental parameters as mentioned in Section 1. In other words, $f_n$ is not constant in real systems and keeps varying. Such uncertainty is also known as **skew** [6,23]. We utilize $f_n(t)$ to capture the time-varying nature of $f_n$. By defining the **frequency ratio** as $\alpha(t) = f_n(t)/f_g$,
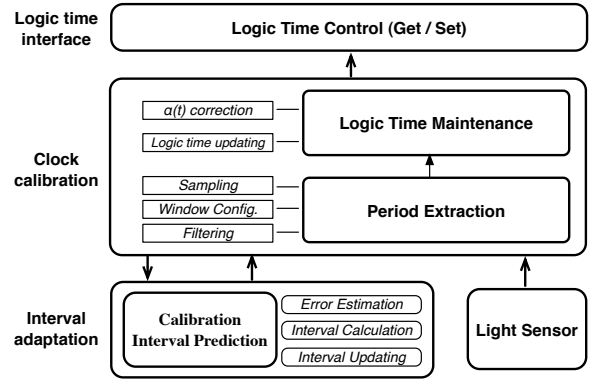


**Figure 4: Illustration of the FLIGHT architecture**

Eq. (1) can be rephrased as follows:

$$c_l(t) = c_l(0) + c_n(t)/(\frac{1}{t}\int_0^t(\alpha(\tau)\cdot f_g)d\tau),$$
$$= c_l(0) + c_n(t)/(\overline{\alpha}(t)\cdot f_g). \quad (2)$$

where $\overline{\alpha}(t)$ is the average value of $\alpha(t)$ from time 0 to $t$. As unveiled in existing works [6,11,23], clock skew can accumulate with time. Thus, FLIGHT should calibrate clocks to minimize its negative impact by controlling the differences of each $c_n(t)/(\overline{\alpha}(t)\cdot f_g)$ among different nodes; otherwise, logic times will rapidly exhibit heterogeneous advancing rates respect to the global reference and among themselves. As a convention, the time difference caused by the skew accumulation is also referred to as **drift**.

Eq. (2) essentially indicates that by continuously estimating the frequency ratio, the logic time can be precisely maintained. As a matter of fact, the variance of the logic time is limited within a short period of time and most applications can tolerate certain amount of such errors [24, 25]. Hence, instead of continuously measuring the frequency ratio, we predict the logic time by using the native clock and the estimated frequency ratio. To further guarantee the accuracy of such a prediction, we configure the clock calibration in a periodical fashion, as depicted in Figure 3.

In Figure 3, we illustrate two calibration instances. During *calibration 1*, the node estimates the real frequency ratio $\overline{\alpha}(\tau_1)$ as $\hat{\alpha}(\tau_1)$. The slope of the straight line between *calibrations 1* and *2* can be viewed as $\hat{\alpha}(\tau_1)$. After the first calibration, the node relies on $\hat{\alpha}(\tau_1)$ to calculate its logic time until *calibration 2*. For example, $t$ native clock ticks (units) after time $\tau_1$ (before $\tau_2$) will be mapped to $t \cdot \hat{\alpha}(\tau_1)$ time units in the logic time domain. If the estimated frequency ratio can well compensate the native clock uncertainty, the mapped time stamp in the logic time domain will be consistent in
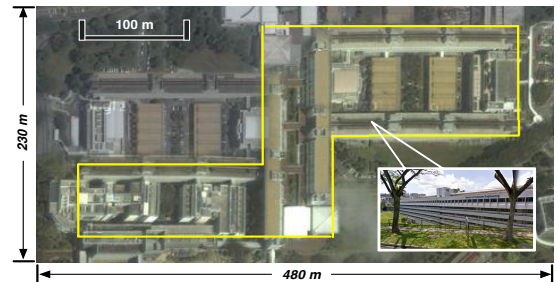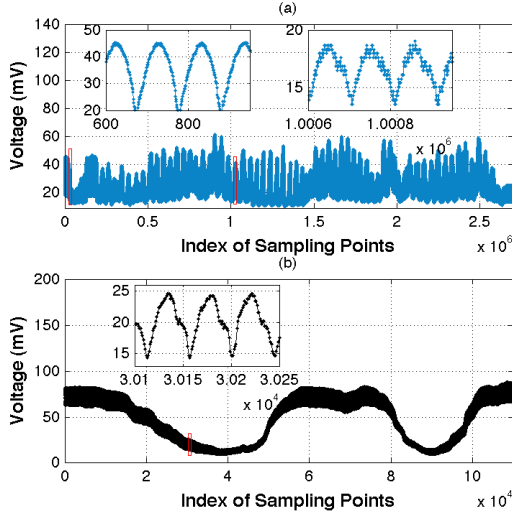


**Figure 5: Main academic building**

**Figure 6: Experiments with mobility and sparsely deployed lamps**



**Figure 7: Frequency domain property of the harvested trace**



**Figure 8: CDF of detected period errors in different scenarios**

each node. However, in practice skew cannot be fully compensated and it is inevitable for drift to accumulate gradually. If such a drift can be (even partially) compensated in each calibration, the differences of $\int_0^t r(\tau)d\tau$ in the entire system will be largely reduced and the consistent logic time with a longer time span can be maintained by clock calibration. Hence, clock calibration in FLIGHT needs to precisely compensate the clock frequency and properly eliminate drift.

## 3.2 System architecture and implementation

Figure 4 depicts the system architecture. There exist three major components in FLIGHT: clock calibration, logic time interface, and interval adaptation. Based on the readings from the light sensor, the period generation module in the calibration component is launched to produce the global reference clock for later calibration. The output of the period generation module acts as the input of the logic time maintenance module, which is responsible for updating the frequency ratio $\alpha(t)$ and correcting the accumulated drift between two calibrations. The newly obtained logic time can be used by upper layer applications. Meanwhile, the output from the logic time maintenance module can be adopted to determine the interval length for the next calibration.

We implement FLIGHT in TinyOS 2.1x on the TelosB platform. The native clock is driven by the built-in crystal oscillator with a $32KHz$ frequency. The AC frequency in our experiments is $50Hz$. Thus, the frequency of the global reference clock is $100Hz$. The kernel of FLIGHT includes 600 lines of NesC code that was complied to 2242 bytes of RAM and 16114 byes of ROM. As a result, FLIGHT only occupies 23% of the total buffer and the remaining space is available for other applications. We use the built-in light sensor on the TelosB mote in our experiments. The light sensor utilizes an ADC module to measure the light intensity and records it as a corresponding voltage value in the register. The standard TinyOS packages the ADC module and relies on the MSP430 Timer to control the ADC sampling. The maximum sampling rate of the MSP430 Timer is around $1KHz$, which is not fast enough for FLIGHT. To overcome this issue, we write a driver to directly access the register of the ADC module. By doing so, the effective sampling rate can break the $1KHz$ barrier and achieves up to
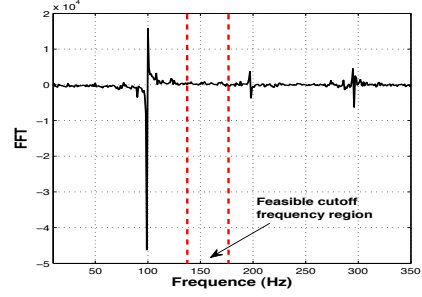
$83KHz$ in the implementation. There are some system parameters that we have not touched yet and we will later discuss each of them in corresponding sections.

## 4. SYSTEM DESIGN

In this section, we introduce the design details of FLIGHT.

### 4.1 Period generation

After sampling the light sensor, nodes need to extract the light period and further generate a series of signals with the same period to serve as the common time reference. In this subsection, we discuss how to efficiently and precisely generate such global reference signals.

#### 4.1.1 Challenging issues

Assume the light sampling rate to be $10KHz$. Ideally, $100 = 10ms \times 10KHz$ samples are detected within one light period. A greedy searching solution for counting the maximum (or minimum) points cannot work here, since there exist multiple local maximum points in one light period for the general multi-lamp scenario. A more sophisticated method may record samples of one period and then apply the greedy searching. When one local maximum point is detected, if its value is close to the recorded maximal value, such a point can be viewed as a period delimiter. Nevertheless, as we will show next in the mobile case, although the light period is still stable, the detected light intensity can fluctuate significantly.

We perform experiments to verify such a point in the mobile environment. In the experiments, three persons roam in the main academic building of an University and each of them holds one TelosB mote and one laptop computer. Due to the limited buffer size, motes are connected to computers through a USB 2.0 port for storing sampled data. Particularly, they move cross different floors. During the movement, sensor nodes might be rotated, shaken, or even temporarily blocked (not sight-in-line to lamps). Figure 5 depicts the
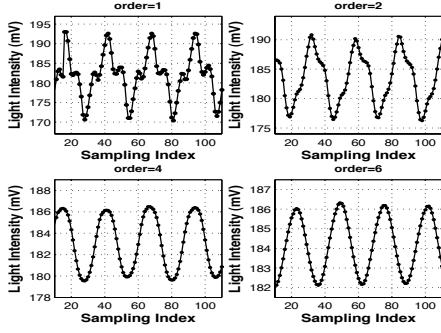
**Figure 9: Filtering results with different orders**

academic building where the whole area counting different floors is over $20,000m^2$. The moving duration of each person lasts around 120 minutes. During the movement, many places that we passed by are only sparsely covered by lamps, i.e., some places are fully covered by the light while some places are relatively dark. To clearly present our harvested trace, we depict one representative clip in Figure 6(a), which contains the portions collected in both **light** and **dark** places. From the zoom in sub-figures in Figure 6(a), however, we can observe that the detected pattern always demonstrates a good periodical property in both light and dark scenarios. The only difference is the observed light intensity.

To further exploit the sensitivity of the detected light intensity, we record the detected pattern of one node when it is rotated. We hold the sensor node in the multi-lamp laboratory and its distance to the closest lamp is around $4m$. Initially, the light sensor faces to the lamp. After a 180-degree rotation, the light sensor almost faces to the ground and the incident light is mainly the scattering light. Figure 6(b) summarizes the result during a 540-degree rotation. In the zoom in sub-figure, we can see that even when the light sensor is opposite to the lamp and the detected light intensity is low, the detected pattern can exhibit a periodical property as well. As we will show later, one lamp can cover up to around $100m^2$ floor size such that fluorescent lamps can be sparingly used in FLIGHT. Now the question is how we can capture such period information under such intense light fluctuation.

One possible way for the period generation is to use existing DSP techniques [26–28], e.g., self-correlation, folding, etc. In principle, they are effective for extracting periodic signals in both static and mobile environments. In practice, however, the extensive computation burdens involved in those techniques will occupy the MCU of nodes for a long time, which may disturb the processing of other important tasks, and inevitably consume more energy.

### 4.1.2 Filtering solution

To tackle such challenges, we propose to utilize a lightweight filtering method to extract periods. In Figure 7, we perform the Fast Fourier Transformation (FFT) for the trace depicted in Figure 6(a) and find that the trace demonstrates a dominant frequency response around $100Hz$. Such a result indicates that the light intensity indeed changes with a stable period of $10ms$. However, due to noises, the light samples are also mixed with other frequency band components. Figure 7 implies if the trace is processed by a low-pass filter with a cutoff frequency between $135Hz$ and $175Hz$, the trace can be used to generate stable periodical signals.

Denote $x[i]$ to be input function and $y[i]$ to be the output function, where $i$ is the sample index. In general, the first-order filter can be
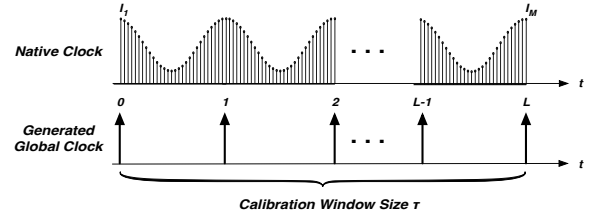


**Figure 10: Illustration of the clock calibration**

expressed as:

$$y[i] \quad = \quad a \cdot x[i] + b \cdot y[i-1], \qquad (3)$$

where $a$ and $b$ are filter parameters and $a + b = 1$. Based on Eq. (3), a higher order filter can be further constructed. For instance, the second-order filter is $y[i] = a \cdot x[i] + b \cdot (a \cdot x[i-1] + b \cdot y[i-2])$. A general expression for any $m$-order filter, where $m \geq 1$, can be obtained iteratively. The settings of $a$ and $b$ have been discussed in [29]. We conduct an experiment to evaluate performances of the period extraction with filters of different orders. Figure 9 shows that if the filter order is small, the trace still exhibits multiple maximum points. When the filter order is sufficiently high, it becomes viable to apply a simple greedy searching scheme to accurately extract periods. In our experiments, we find that a 6-order filter is adequate in FLIGHT (as shown by Figure 9 and Section 5). In such a case, a node only needs to buffer 6 latest samples for the filtering operation. The involved computation burden contains merely 12 multiply operations, 6 plus operations and the greedy searching, which is particularly beneficial for resource constrained mobile devices.

In Figure 8, we summarize the period detection accuracy of the filtering method. We have shown the detection errors in static scenarios in Section 2. For the mobile scenario in Figure 6(a), Figure 8 shows that the error is bounded by $165\mu s$ and smaller than $100\mu s$ most of time. The detection errors in our rotation experiments are also within $165\mu s$ and less than $65\mu s$ for majority cases.

## 4.2 Logic time maintenance

To maintain the logic time, frequency ratio must be updated based on the extracted light period in each calibration.

### 4.2.1 Frequency ratio calibration

According to the output from the period generation component, the generated signal can be viewed as a common time reference. Suppose the start time of the calibration and the calibration window size are $t_c$ and $\tau$, respectively. For the presentation simplicity, we assume $t_c = 0$ without loss of generality. The basic principle of clock calibration is illustrated in Figure 10. In Figure 10, there are totally $M$ samples between the first and the last generated periods. Each $I_j$ represents the sample index where $1 \leq j \leq M$. We denote the native clock frequency in $\tau$ as $\overline{f}_n(\tau)$. Based on the generated global reference, $\overline{f}_n(\tau)$ can be measured by $\hat{f}_n(\tau)$ as follows:

$$\frac{L}{f_g} \quad = \quad \frac{I_M - I_1}{\hat{f}_n(\tau)}, \qquad (4)$$

where $L$ is the number of light periods detected in the calibration window $\tau$. Based on the definition, the frequency ratio can be calibrated by the following equation:

$$\hat{\alpha}(\tau) \quad = \quad \frac{\hat{f}_n(\tau)}{f_g} = \frac{I_M - I_1}{L}. \qquad (5)$$

The rationale behind the calibration is that native clocks of different nodes run with different speeds. By referring to the global reference, frequency ratios are calculated to compensate those differences. If the measured native clock $\hat{f}_n(\tau)$ in Eq. (4) is accurate, the frequency ratio in Eq. (5) can precisely capture the real situation when predicting the logic time after the calibration.

### 4.2.2 Calibration window configuration

In Figure 10, $M$ native clock ticks have been elapsed during the calibration window $\tau$. Due to the instability of the native clock, $M/\overline{f}_n(\tau)$ may not exactly equal to $M/f_n$, where $f_n$ is the claimed clock frequency and $\overline{f}_n(\tau)$ is the average value of the native clock within $\tau$. We denote this offset in the calibration window as $o(\tau) = M/\overline{f}_n(\tau) - M/f_n$ and it can be viewed as a relative drift of the native clock with respect to the global reference since both $f_n$ and $f_g$ are constant. The relative skew, thus, can be expressed as $s(\tau) = o(\tau)/\tau$.

On the other hand, due to jitters, each sampling point (except the first one $I_1$ since we artificially fix it and view it as the starting point) actually further exhibits a tiny shift, $h$, compared with the ideally stable sampling point. Prior studies have shown that such a shift follows $h \sim \mathcal{N}(0, \sigma_h^2)$. Therefore, the really observed offset in the calibration window $\tau$ can be expressed as:

$$\begin{aligned} \hat{o}(\tau) &= \sum_{j=2}^{M}\left[\left(\frac{I_j}{\overline{f}_n(\tau)} + h_j\right) - \left(\frac{I_{j-1}}{\overline{f}_n(\tau)} + h_{j-1}\right)\right] - \frac{L}{f_g}, \\ &= o(\tau) + h_M, \end{aligned} \quad (6)$$

where each $h_j$ indicates the shift for the sampling point $I_j$ for $2 \le j \le M$. We can show that if $\hat{s}(\tau) = \hat{o}(\tau)/\tau$ is an accurate estimation of $s(\tau)$, $\hat{f}_n(\tau)$ will precisely estimate its true value $\overline{f}_n(\tau)$. In other words, the frequency ratio $\overline{\alpha}(\tau)$ can be well measured by the observed ratio $\hat{\alpha}(\tau)$ in Eq. (5), yielding a good calibration result. We utilize $Var[(\hat{s}(\tau) - s(\tau))]$ to quantify the estimation accuracy. According to the definition of skew, we have:

$$Var[(\hat{s}(\tau) - s(\tau))] = Var[\frac{(\hat{o}(\tau) - o(\tau))}{\tau}] = \sigma_h^2/\tau^2. \quad (7)$$

Eq. (7) suggests that it is beneficial to set the calibration window to be large. However, we argue that in practice, the solution is beyond such a simple strategy and there exist practical limitations. For example, the buffer size limitation may prohibit to obtain enough data during the calibration. Devices usually have buffers with limited capacity for multiple applications, which may not be able to hold enough samples during the calibration window. For instance, the buffer size of a TelosB mote is $10kB$, in which $8kB$ can be allocated to store sampled data at the most. If each sample occupies two bytes, the buffer can hold up to 4000 samples. If the sampling rate is $10KHz$, only 40 periods can be collected, which may not be enough for an accurate calibration.

To tackle such an issue in FLIGHT, we propose to parallel the sampling and period generation operations. After initial raw data have been filtered, we can drop those data and only store 1) the number of periods that have been detected, and 2) the total amount of samples within those periods. By doing so, the buffer size is virtually increased so as to accommodate adequate data for the calibration. On the other hand, the real buffer occupancy is small and it will not conflict with other applications. To implement such an idea in FLIGHT, we insert the computation process of filtering between the interval of two consecutive samplings. Whenever a new sample has been obtained, it will be processed immediately. We claim that the selection of the sampling rate and the order of the filter should guarantee that the filtering computation must be completed in between two consecutive samplings; otherwise, the data
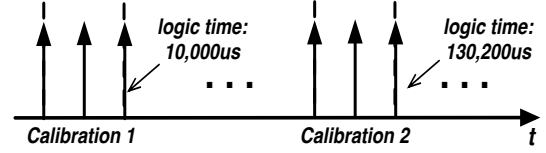


**Figure 11: Example of the logic time error**

loss will lead to errors in the generated global reference signals. In Section 5, we will verify such a statement.

### 4.2.3 Logic time updating

After the frequency ratio has been updated as $\hat{\alpha}(\tau)$, the logic time afterward can be predicted by using such a new frequency ratio and the native clock till the next calibration window. To maintain an accurate logic time, in addition to the calibration, we also need to correct the drift accumulated between two consecutive calibrations. We utilize Figure 11 to illustrate such a point. In the first calibration interval, we set the logic time to $10,000\mu s$ at the end of the last detected period. Since the period length of light is $10ms = 10,000\mu s$, the logic time should increase $10,000\mu s$ whenever one period has elapsed. After the first calibration, we assume that after 10 light periods, the node performs the next calibration. If the clock is stable, the logic time at the last detected period in the second calibration should be $130,000\mu s$. However, the real logic time might be $130,200\mu s$ due to the clock uncertainty. Such an error cannot be corrected by the frequency ratio updating, while it will naturally deteriorate the time consistency among different nodes.

The insight obtained from Figure 11 is that if the accumulated error can be controlled within a certain range, the last several digits contains the information about the drift. In Figure 11, if the accumulated drift is smaller than $10,000\mu s$, in principle, the logic time can be corrected by resetting the last four digits to zero, i.e., $130,200\mu s$ will be changed to $130,000\mu s$. If we denote $t_{c_i}$, $L_i$, and $n_i$ to be the finish time of the $i$th calibration, the number of light periods detected in the $i$th calibration, and the number of light periods between the $i$th and $i + 1$th calibration respectively, we can essentially adjust the logic time at the last detected period to $c_l(t_{i+1}) = c_l(t_i) + (n_i + L_i)/f_g$ to eliminate the drift. Now we formally derive the logic time in FLIGHT. If $t = t_{c_i}$, then

$$c_l(t) = \sum_{j=1}^{i}(L_j + n_j)/f_g. \quad (8)$$

As a matter of fact, each $n_j$ is unknown. The operation in Eq. (8) can be achieved by resetting the last several digits of the logic time to zero (the exact digit number is related to the required error bound), or $n_j$ can be estimated via $\hat{\alpha}(t_{c_{j-1}}) \times n_{j-1}$. On the other hand, if $t_{c_i} < t < t_{c_{i+1}}$, we have:

$$c_l(t) = c_l(t_{c_i}) + (c_n(t) - c_n(t_{c_i}))/(\hat{\alpha}(t_{c_i}) \cdot f_g). \quad (9)$$

Now, by directly tuning to the light period at the end of each calibration, the logic time can be precisely aligned to the global reference and the drift accumulated in the previous calibration interval is thus eliminated.

## 4.3 Calibration intervals

In this section, we describe how we can further utilize the calibration interval to reduce the number of samplings. We also discuss how to determine the interval length between two calibrations.

In Figure 11, if a proper calibration window size should at least contain 13 light periods, a naive method is to extract 13 light periods for each calibration window. However, Figure 11 implies that
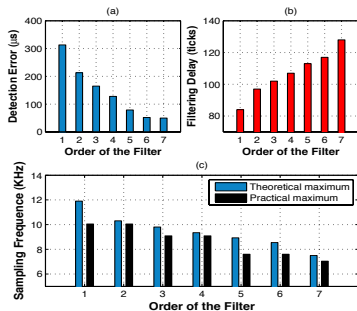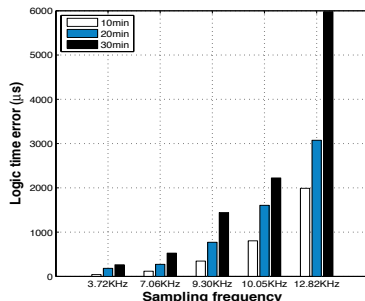
**Figure 12: Filtering configurations**



**Figure 13: Logic time error v.s. sampling frequency**
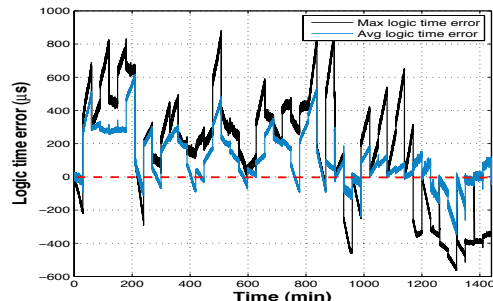


**Figure 14: Maximum and average logic time error in the stable environment**

if we know that 10 light periods will elapse between two consecutive calibration windows, we only need to sample three additional periods in the second calibration window (*Calibration 2*). By using the similar method in Eq. (8), the node can estimate how many light periods have been elapsed between *Calibrations 1* and *2*. In addition to the number of the light periods have been detected in *Calibration 2*, the total number of light periods between two calibration windows can be obtained. On the other hand, the number of the native clock ticks are always recorded (to calculate any instant logic time). As a result, the frequency ratio can still be calibrated using Eq. (5). Benefited from such a method, although only three light periods have been extracted in *Calibration 2*, the frequency ratio can be accurately calibrated based on 13 light periods. In practice, the calibration interval normally spans more than 20 minutes. Hence, nodes are able to sample a small number of periods while still achieve high accuracy in each calibration.

In Section 5, we examine the calibration interval selection to balance the calibration accuracy and cost. In this study, we also propose a dynamic interval adjustment scheme. We introduce a drift controlling factor ($u$) to restrict the total amount of logic time errors accumulated between two calibrations. Suppose the drift detected at the last period in the $i$th calibration is $e_i$ and the calibration interval length between the $i$th and the $i + 1$th calibrations is $d_i$. The next calibration interval $d_{i+1}$ in FLIGHT can be predicted as:

$$d_{i+1} = d_i \times \frac{u}{|e_{i+1}|}. \tag{10}$$

Initially, $d_0$ is set to be $20min$. Eq. (10) states that if the drift rapidly accumulates, the calibration interval should be short such that the drift can be corrected in time; vise versa. In Section 5, we will investigate the impact of the factor $u$.

### 4.4 Upgraded to time synchronization

Clock calibration only guarantees a similar increasing rate of the maintained logic clock despite of different starting times. As mentioned in Section 2.2 and [12, 13, 15], clock calibration itself serves for a variety of useful applications. Sometimes, people may want to upgrade to time synchronization for a finer understanding of the system performance in the time domain. For instance, nurses in the healthcare networks may need to understand timely situations of monitored patients. System operators in wireless sensor networks may need to record the exact time of certain events. FLIGHT can be easily used to facilitate such an upgrading. To eliminate the clock offset globally, we may simply adopt the FTSP protocol [10]. With the FLIGHT calibration, we do not need to run the full synchronization protocol with heavy intercommunications between nodes. Instead, we only need to cancel the initial time difference with one

communication round. We note that the communication overhead and energy consumption to maintain an absolute clock value can be dramatically reduced when clocks of nodes are already precisely calibrated with respect to a common time reference.

## 5. PERFORMANCE EVALUATION

In this section, we evaluate the system performance of FLIGHT through extensive experiments.

### 5.1 Filter configuration

In Figure 12, we evaluate the impact of the order of the filter by varying it from 1 to 7. We output a series of samples to computers (due to the limited buffer, we do not store such information in sensor motes) and analyze the error of extracted periods. Figure 12(a) shows that when $m = 1$, the period generation component leads to the largest detection error. From statistics, we observe that the average error reaches as high as $323\mu s$. As we increase $m$, errors of generated periods reduce dramatically. From Figure 12(a), we can also notice that if we increase $m$ after 5, the error reduction is slight, implying that the order of the filter can be kept relatively low without compromising the achieved period accuracy. In Figure 12(b), we further examine the processing delay to filter one sample respect to different filter orders. As what we expect, the filtering delay becomes linearly increased when $m$ increases as depicted in Figure 12(b). Such a processing delay essentially limits the maximum sampling rate that can be used for each specific $m$; otherwise, the filtering and sampling operations cannot be parallelized.

To facilitate the sampling rate selection for the practical development, we plot the maximum sampling rates allowed for each $m$ in principle (i.e., theoretical maximum in Figure 12(c)). Since feasible frequencies in practice are not continuous, in Figure 12(c), we plot the maximum frequency that can be adopted for each $m$ in the TelosB platform as an example. Based on the results in the figure, we find that the pair of $m = 6$ and $f_s = 7.06KHz$ can produce both a small period detection error and a sufficiently high sampling rate.

In Section 4.2.2, we claim that the filtering computation should be completed within two consecutive samplings; otherwise, the inaccurately detected periods may incur serious logic time errors. To validate such an argument, we conduct an experiment using two TelosB motes. In this experiment, $m = 6$ and the sampling rate changes from $3.72KHz$ to $12.42KHz$. After the initial calibration, we examine their logic time error in 10, 20, and 30 minutes. From Figure 13, we can easily observe that when $f_s$ is greater than $7.06KHz$, the error increases dramatically. On the other hand, due to the extra computation burden from the period delimiter searching and the interruptions of the MCP to other tasks, the $7.06KHz$ sam-
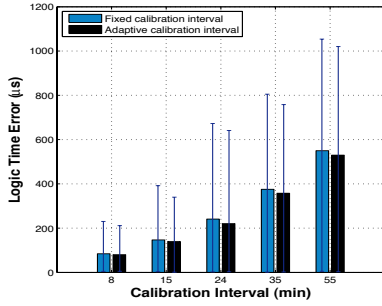
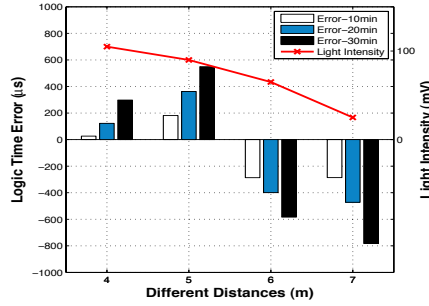**Figure 15: Logic time error v.s. calibration intervals**



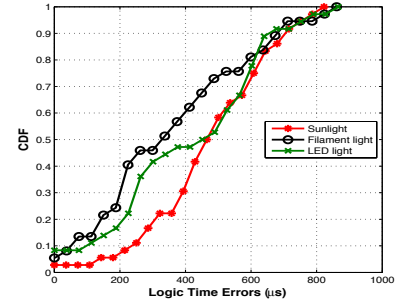**Figure 16: Logic time error v.s. light intensity**



**Figure 17: CDF of logic time errors mixed with other types of lights**

pling rate may not always provide sufficient safe guarding region to parallel the sampling and computation. As a result, the synchronization accuracy might be impacted. In our experiment, we use a smaller frequency value $3.72KHz$ that offers adequate safe guarding region and reliably provides a small logic time error. We use the $3.72KHz$ sampling rate in the following experiments.

## 5.2 Logic time accuracy

In this section, we examine the logic time differences among different nodes in both static and dynamic environments.

### 5.2.1 Static environment

In the static environment, 12 sensor nodes are uniformly distributed in our laboratory. One beacon node is placed in the middle of the laboratory and broadcasts beacon messages to trigger each node logging its current logic time. The experiment has been conducted for one week and we summarize the performance from Figures 14 to 17.

Figure 14 depicts the max and average logic time errors with respect to a randomly selected reference node. We plot one 24-hour trace clip for a clear presentation. Overall, the trace evolves following a zigzag pattern. The sudden drops of two curves are due to the logic time updating in each calibration window. From Figure 14, we can observe that on average, different clocks have been tightly synchronized. From statistics, we find that the average error keeps less than $600\mu s$ all the time and it is smaller than $200\mu s$ for over 80% of time. The maximum logic time error is also well controlled in the experiment. Statistics show that the maximum error is always smaller than $950\mu s$ and it is less than $350\mu s$ most time. Figure 14 indicates that different clocks can be highly calibrated by FLIGHT.

In Figure 15, we evaluate the adaptive calibration interval scheme compared with the deterministic interval setting. We choose five different error controlling factors to be 5, 10, 15, 25, and 30 native clock ticks and each native clock tick corresponds to $30.515\mu s$. For each factor, the experiment lasts 5 hours. When the experiment terminates, we calculate the average length of the calibration interval to be 8, 15, 24, 35 and $52min$, respectively. Then, we repeat the experiment for the same duration by using the deterministic calibration interval policy with the calculated interval length. In Figure 15, we plot the average, maximum, and minimum logic time errors observed in the experiment. From the figure, we find that the deterministic policy generally incur a larger error than the adaptive policy. In order to control the average error below $500\mu s$ as [13,15], we choose 25 as the default value of $u$ in FLIGHT since such a setting can achieve the accuracy requirement and reduce the number of calibrations at the same time.

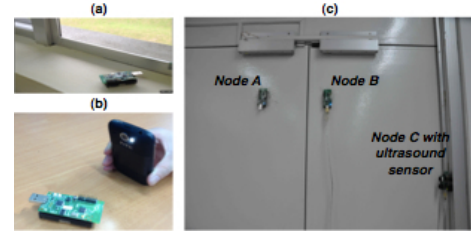In Figure 16, we examine the impact of the light intensity on



**Figure 18: Illustration of several experimental settings**

the logic time consistence. In this experiment, we open one lamp array in the rear of our laboratory and switch off all other lamps. Then, we place sensor nodes in a line covering both light region and dark region. In Figure 16, the x-axis represents the approximate distance of one node to the rear of the laboratory. We first plot the received light intensity in Figure 16. As the node becomes far away from the rear region, the detected light intensity drops. From Figure 16, we can observe that when the light intensity is higher than $25mV$, logic times are accurately maintained, e.g., the logic time errors are smaller than $600\mu s$ after 30 minutes. Only after the light intensity drops below $25mV$, the logic time error reaches $780\mu s$. However, such a performance is still acceptable in many systems with the $1ms$ error bound [15]. As a matter of fact, in most indoor environments, the node can normally observe $50mV$ above light intensity. Therefore, one lamp can cover a large-scale field such that fluorescent lamps can be sparingly used in FLIGHT.

Finally, we examine the impact of the interference from other light sources. We first mix the fluorescent light with the sunlight, which commonly occurs when the node is deployed close to the window (Figure 18(a)). The sunlight is a direct signal. We observe that the combined reading of the node consists of a $800mV$ direct component and a $90mV$ alternating component, and find that the periodical pattern from fluorescent lighting can still be accurately detected. The result shows that up to 80% of time, the logic time error is less than $600\mu s$. In addition, we also mix the fluorescent light with the LED signal from a smart phone (Figure 18(b)), which is the most widely existing noise for FLIGHT in the indoor environment. Although the generated light from LED is relative stable, due to the slight shaking of the smart phone, the amplitude of the combined signal varies, and the amplitude of the LED signal is greater than that of the fluorescent light. Similar to Figure 6, FLIGHT can still capture the frequency component close to $100Hz$ and accurately detect the lighting period delimiters. From statistics, we observe that the logic time is still accurate. Figure 17 shows that the
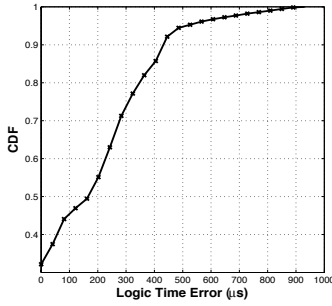
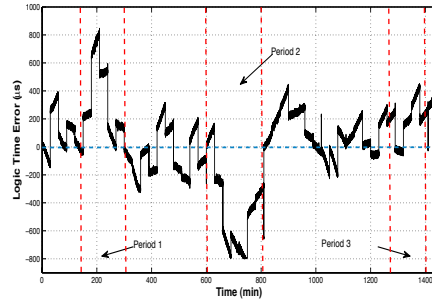**Figure 19: Time error with controlled mobility**



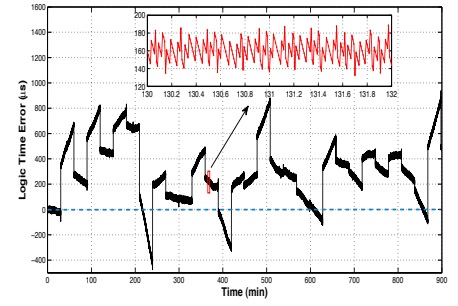**Figure 20: Nodes' logic time error during the movement**



**Figure 21: Nodes' logic time error in different buildings**

error is controlled within $900\mu s$ and in most cases the error is less than $610\mu s$. Finally, we mix the fluorescent light with the filament light that is another common noise for FLIGHT. In this scenario, the logic time is also accurate according to Figure 17. Therefore, we believe that FLIGHT is robust to external interferences from other light sources during clock calibration.

### 5.2.2 Dynamic environment

In this subsection, we conduct three trails of experiments to evaluate FLIGHT in dynamic environment. In the first experiment, two sensor nodes are attached to the gates of the laboratory as shown in Figure 18(c). Notice that only the gate with sensor node A can be open. Both of the two nodes can synchronize their clocks with the same lamp. To avoid disturbing the entrance of students to the laboratory, node A transmits its recorded logic times through wireless communications. This experiment lasts 6 hours and we also deploy a node (node C) with the ultrasonic wave sensor to count the number of people passing through the gate to approximate the times of gate opening. During the experiment, the final value of the counter is 158 and we estimate the gate has been opened more than 70 times during the experiment. We summarize the system performance in Figure 19. We find that the average clock difference between node A and B is smaller than $1,000\mu s$ all the time and it is less than $400\mu s$ most of time. Compared with the static environment in Figure 13, the average error increasing is slight and accounts for 12% performance degradation.

In the second trail of experiment, three persons take turns to roam in the academic building. In particular, during periods 1, 2, and 3 in Figure 20, they roam in the office area, the class room area, and the laboratory, respectively. In each period, each person holds three TelosB motes and roams for around one hour. Two sensor nodes are used to be calibrated and the third one periodically sends out beacon messages to trigger the logic time recording. To store recorded logic times, the to-be-synchronized motes are connected to a laptop inside a bag carried by the person. Except the three highlighted periods, sensor nodes are placed in the laboratory stilly. From Figure 20, we can observe that when the motes roam outside the laboratory (i.e., periods 1 and 2), the logic time error is mainly distributed within $[600, 800]\mu s$. The increasing of the logic time error in the two periods is mainly due to the noises introduced by the mobility and surrounding environments. In addition, some places during the roaming are not well covered by light, which also contributes to the performance deterioration. Differently, logic times in period 3 suffers from a much less error during the movement. Compared with the static performance, the logic error in period 3 only slightly increases.

In the third experiment, we place two sensor nodes in two dif-

ferent buildings that are around $150m$ away from each other. We place one beacon node in between. The beacon node periodically sends out a beacon message with the maximum power to trigger the two experimenting nodes to record their instant logic times. During $[350, 550]min$, two nodes roam within their local regions (constrained by the transmission range of the beacon node) separately. We plot their logic time differences in Figure 21. Figure 21 shows that the average error is around $400\mu s$. When the mobility is enabled, the error is limited within $1,000\mu s$ all the time. The overall performance crossing different buildings is similar to the results obtained in previous two experiments.

## 5.3 Energy consumption

To understand the energy efficiency of FLIGHT, we first evaluate FLIGHT under different calibration intervals and then compare it with some recent approaches including ROCS, WizSync, and FTSP [10, 13, 15]. We directly measure the energy consumption of FLIGHT from the hardware and report the result in Figure 22. The working power of FLIGHT is around $5.394mW$ and each calibration in FLIGHT can be finished within $100ms$. The energy consumption of FLIGHT with different intervals is plotted in Figure 22. We see that the energy consumption of FLIGHT is usually less than $5\mu W$, which is highly desired for power-constrained mobile devices. For the comparison study, we directly measure the energy consumption of FTSP as well. For ROCS and WizSync, on the other hand, since bot two methods need specific-designed hardware and algorithms, we refer to the results reported in their corresponding literatures. We configure all four schemes with consistent settings and report the results in Figure 23. We evaluate the energy consumptions of FLIGHT, ROCS, and WizSync, by varying the calibration interval from $10min$ to $30min$. From Figure 23, we can see that the communication-based protocol indeed incurs a high-energy consumption. Compared to FLIGHT, ROCS, and WizSync, the power consumption of FTSP is much higher even when we set a much larger calibration interval. As we can see from Figure 23, the energy consumption of ROCS and WizSync is similar with the same calibration interval setting. Due to the uncertainty of the Wi-Fi beacon period, the calibration interval of WizSync is usually configured to be $10min$ in practice and the corresponding power consumption is around $50\mu W$. The calibration interval of FLIGHT and ROCS can be set around $30min$, so their energy efficiencies are around 40 times and 6 times higher than WizSync, respectively. Since sampling the light sensor in FLIGHT consumes much less energy than operating the FM receiver module in ROCS, FLIGHT can achieve higher energy efficiency.
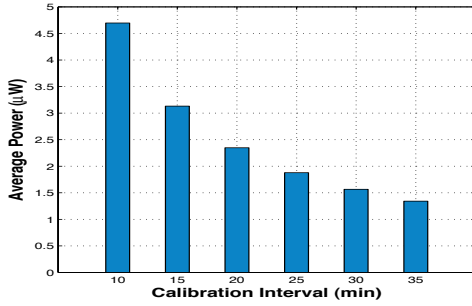
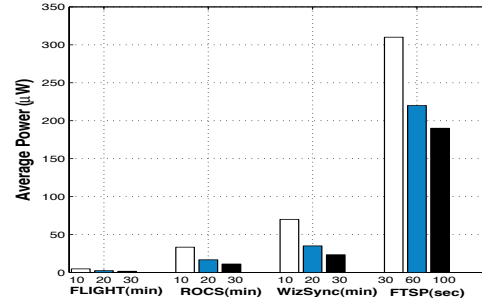**Figure 22: Energy consumption of FLIGHT with different calibration intervals**



**Figure 23: Energy consumption comparison of different schemes**

## 6. LIMITATIONS AND FUTURE WORKS

- **Lighting availability**: FLIGHT requires the availability of fluorescent lighting. As we have shown in Section 5, the period information can be extracted even when the detected light intensity is very low. One fluorescent lamp, thus, can coordinate plenty of surrounding nodes. Fluorescent lighting from different lamps can exhibit a stable and consistent period. Hence, lamps can be sparingly utilized in the field, which is available across almost all indoor places, e.g., universities, hospitals, airports, supermarkets, etc. FLIGHT might generate extra cost while the cost is generally slight.

- **Sensor availability**: In FLIGHT, light sensors should be exposed to the environment during the calibration. It can be achieved for the majority of the applications that have been introduced in Section 2, such as the emerging HLPP gaming, the event detection and diagnosis in sensor networks and so on. In some other applications, like the body area networks, if certain nodes are permanently placed inside pockets, those nodes have to synchronize their clocks with their neighbors via communication based protocols, e.g., FTSP, while those exposed nodes can still use FLIGHT to correct clocks. Nevertheless, benefiting from the existing clock calibration with FLIGHT, further communication is minimized and synchronization overhead can be dramatically reduced.

- **Enhancing robustness**: In the current filter design for time period generation in FLIGHT, we focus on filtering the noises outside the $100Hz$ frequency band. It is mainly because the $100Hz$ component in the frequency domain serves as an indication for the fluorescent lighting in most indoor environments. Such a design, however, could be vulnerable to noise near $100Hz$. Some advanced digital techniques, e.g., PLL, can be applied to further improve the reliability of FLIGHT.

## 7. RELATED WORK

Great efforts have been made for achieving common time notion across the networking systems over the past decade. Most existing works rely on wireless communications and mainly focus on improving the synchronization accuracy. RBS [8] eliminated the sender-side delay for synchronization. TPSN [9] further canceled out the propagation delay. By using the MAC-layer stamping technique, FTSP [10] largely increases the synchronization accuracy. However, the authors in [11] find that errors among different clocks exponentially increase with the network diameter. [30] proposes a fast flooding scheme and [24] studies the synchronization

accuracy in low-duty-cycle sensor networks. [22] estimates clock uncertainty for duty-cycled sensor networks. On the other hand, some other works also focus on addressing the clock uncertainty and reducing the time synchronization cost. ACES [23] suggested to track skew by using Kalman filter and ODS [6] introduces the on-demand accuracy synchronization. Both [31] and [32] explore the temperature compensated scheme to mitigate the clock skew. In [24] and [25], authors introduce to maintain a common time by using two clocks on each individual node with different accuracies.

Recently, several emerging studies exploit the external signal source with a stable period for clock calibration and synchronization. Communication overhead and energy consumption can be reduced compared to protocols mentioned above. RT-Link [33] and ROCS [13] both explore to synchronize clocks to a certain type of radios. Specific hardware are required in those proposals to generate periodical signals. The majority of those solutions utilize radios for the clock calibration at the expense of higher power consumption. Although clock calibration can be performed periodically, the energy consumption for each calibration could not be reduced and the accumulated power draining for time synchronization is still high. [14] designs passive radio receivers for synchronizing nodes to radio stations. In [15], authors introduce to synchronize clocks to Wi-Fi beacons without the extra hardware support. Such a global reference (i.e., Wi-Fi beacons), however, may not be stable enough due to channel contention and collisions. In [12], the authors propose to use power lines for clock calibration. However, due to dramatically decay, the periodical signals can only be detected in the vicinity of power lines. Extra hardware is required as well in [12]. On the contrary, FLIGHT is not constrained by those limitations and FLIGHT can precisely calibrate clocks in an energy-efficient manner without any specific hardware. In [34], the compact fluorescent light bulbs have been used for the proximity sensing. The technique in [34] utilizes the special property of the bulbs when they are illuminated, while we focus on the special property of the fluorescent light itself.

## 8. CONCLUSION

In this paper, we develop a new clock synchronization approach called FLIGHT, which leverages the fact that the light intensity from fluorescent lamps varies with a stable period. FLIGHT does not require any extra hardware and radio operations. By sampling light sensor or camera, FLIGHT can intelligently extract periods for the clock calibration and retain a common notion of time in the system. We give in-depth analysis on various practical issues to ensure the calibration accuracy. We implement FLIGHT in TelosB motes and conduct experiments using a 12-node test-bed under

both static and mobile settings. During one week measurements, Over one-week measurement results show that FLIGHT can achieve a tightly synchronized time with low energy consumption.

## Acknowledgement

## 9. REFERENCES

[1] J. Qiu, D. Chu, X. Meng, and T. Moscibroda, "On the feasibility of real-time phone-to-phone 3d localization," in *Proc. of ACM Sensys*, 2010.

[2] T. Gu, Z. Wu, X. Tao, H. Pung, and J. Lu, "epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition," in *Proc. of IEEE PerCom*, 2009.

[3] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, and J. Stankovic, "An advanced wireless sensor network for health monitoring," in *Proc. of D2H2*, 2006.

[4] A. Natarajan, B. de Silva, K. Yap, and M. Motani, "Link layer behavior of body area networks at 2.4 ghz," in *Proc. of ACM Mobicom*, 2009.

[5] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proc. of ACM/IEEE IPSN*, 2005.

[6] Z. Zhong, P. Chen, and T. He, "On-demand time synchronization with predictable accuracy," in *Proc. of IEEE Infocom*, 2011.

[7] "Product List of Seiko Instruments Inc., 2009," http://speed.sii.co.jp/pub/compo/quartz/productListEN.jsp.

[8] J. Elson, L. Girod, and D. Estrin, "Fine grained network time synchronization using reference broadcasts," in *Proc. of ACM OSDI*, 2002.

[9] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing sync protocol for sensor networks," in *Proc. of ACM Sensys*, 2003.

[10] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. of ACM Sensys*, 2004.

[11] C. Lenzen, P. Sommer, and R. Wattenhofer, "Optimal clock synchronization in networks," in *Proc. of ACM Sensys*, 2009.

[12] A. Rowe, V. Gupta, and R. Rajkumar, "Low-power clock synchronization using electromagnetic energy radiating from ac power lines," in *Proc. of ACM Sensys*, 2009.

[13] L. Li, G. Xing, L. Sun, W. Huangfu, R. Zhou, and H. Zhu, "Exploiting fm radio data system for adaptive clock calibration in sensor networks," in *Proc. of ACM Mobisys*, 2011.

[14] Y. Chen, Q. Wang, M. Chang, and A. Terzis, "Ultra-low power time synchronization using passive radio receivers," in *Proc. of ACM/IEEE IPSN*, 2011.

[15] T. Hao, R. Zhou, G. Xing, and M. Mutka, "Wizsync: Exploiting wi-fi infrastructure for clock synchronization in wireless sensor networks," in *Proc. of IEEE RTSS*, 2011.

[16] D. Allan and H. Machlan, "Time transfer using nearly simultaneous reception times of a common transmission," *26th Annual Symposium on Frequency Control*, 1972.

[17] M. Li, D. Ganesan, and P. Shenoy, "Presto: feedback-driven data management in sensor networks," *IEEE/ACM Transactions on Networking (TON)*, 2009.

[18] Y. Wu and X. Wang, "Achieving full view coverage with randomly-deployed heterogeneous camera sensors," in *Proc. of IEEE ICDCS*, 2012.

[19] X. Chang, R. Tan, G. Xing, Z. Yuan, C. Lu, Y. Chen, and Y. Yang, "Sensor placement algorithms for fusion-based surveillance networks," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, no. 99, pp. 1–1, 2011.

[20] S. Zhao, L. Fu, X. Wang, and Q. Zhang, "Fundamental relationship between node density and delay in wireless ad hoc networks with unreliable links," in *Proc. of ACM Mobicom*, 2011.

[21] A. Kashyap, S. Ganguly, and S. Das, "A measurement-based approach to modeling link capacity in 802.11-based wireless networks," in *Proc. of ACM Mobicom*, 2007.

[22] S. Ganeriwal, I. Tsigkogiannis, H. Shim, V. Tsiatsis, M. Srivastava, and D. Ganesan, "Estimating clock uncertainty for efficient duty-cycling in sensor networks," *IEEE/ACM Transactions on Networking (TON)*, 2009.

[23] B. Hamilton, X. Ma, Q. Zhao, and J. Xu, "Aces: adaptive clock estimation and synchronization using kalman filtering," in *Proc. of ACM Mobicom*, 2008.

[24] J. Koo, R. Panta, S. Bagchi, and L. Montestruque, "A tale of two synchronizing clocks," in *Proc. of ACM Sensys*, 2009.

[25] T. Schmid, P. Dutta, and M. Srivastava, "High resolution, low power time synchronization an oxymoron no more," in *Proc. of ACM/IEEE IPSN*, 2010.

[26] Z. Zhang, X. Zhou, W. Zhang, Y. Zhang, G. Wang, B. Zhao, and H. Zheng, "I am the antenna: accurate outdoor ap location using smartphones," in *Proc. of ACM Mobicom*, 2011.

[27] A. Bhartia, Y. Chen, S. Rallapalli, and L. Qiu, "Harnessing frequency diversity in wi-fi networks," in *Proc. of ACM Mobicom*, 2011.

[28] S. Rallapalli, L. Qiu, Y. Zhang, and Y. Chen, "Exploiting temporal stability and low-rank structure for localization in mobile networks," in *Proc. of ACM Mobicom*, 2010.

[29] J. R. Barry, *Digital Communications*, 2003.

[30] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proc. of ACM/IEEE IPSN*, 2011.

[31] T. Schmid, Z. Charbiwala, Z. Anagnostopoulou, M. Srivastava, and P. Dutta, "A case against routing-integrated time synchronization," in *Proc. of ACM Sensys*, 2010.

[32] Z. Yang, L. Cai, Y. Liu, and J. Pan, "Environment-aware clock skew estimation and synchronization for wireless sensor networks," in *Proc. of IEEE Infocom*, 2012.

[33] A. Rowe, R. Mangharam, and R. Rajkumar, "Rt-link: A time-synchronized link protocol for energy-constrained multi-hop wireless networks," in *Proc. of SECON*, 2006.

[34] S. Gupta, K. Chen, M. Reynolds, and S. Patel, "Lightwave: using compact fluorescent lights as sensors," in *Proc. of ACM UbiComp*, 2011.