Powering Indoor Sensing with Airflows: A Trinity of Energy Harvesting, Synchronous Duty-Cycling, and Sensing*

Tianyu Xiang, Zicheng Chi, Feng Li, Jun Luo
School of Computer Engineering
Nanyang Technological University
{txiang001, zcchi, fli3, junluo}@ntu.edu.sg

Lihua Tang, Liya Zhao, Yaowen Yang
School of Civil and Environmental Engineering
Nanyang Technological University
{tanglh, lyzhao1, yyw}@ntu.edu.sg

ABSTRACT

Whereas a lot of efforts have been put on energy conservation in *wireless sensor networks*, the limited lifetime of these systems still hampers their practical deployments. This situation is further exacerbated indoors, as conventional energy harvesting (e.g., solar) ceases to work. To enable long-lived indoor sensing, we report in this paper a self-sustaining sensing system that draws energy from indoor environments, adapts its duty-cycle to the harvested energy, and pays back the environment by enhancing the awareness of the indoor microclimate through an "energy-free" sensing.

First of all, given the pervasive operation of *heating, ventilation and air conditioning* (HVAC) systems indoors, our system harvests energy from airflow introduced by the HVAC systems to power each sensor node. Secondly, as the harvested power is tiny (only of hundreds of μW), an extremely low but synchronous duty-cycle has to be applied whereas the system gets no energy surplus to support existing synchronization schemes. So we design two complementary synchronization schemes that cost virtually no energy. Finally, we exploit the feature of our harvester to sense the airflow speed (which can be used to infer the indoor microclimate) in an energy-free manner. To our knowledge, this is the first indoor wireless sensing system that encapsulates energy harvesting, network operating, and sensing all together.

Categories and Subject Descriptors

C.2.4 [**Computer Communication Networks**]: Distributed Systems]

General Terms

Design, Algorithms, Performance, Experiments

Keywords

Indoor energy harvesting, duty-cycle, synchronization

*This work was supported in part by AcRF Tier 2 Grant ARC15/11.

SenSys'13, November 11–15, 2013, Rome, Italy.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) have been involved in many environmental monitoring applications [18], including in particular indoor environmental monitoring [20, 35]. For such applications, one of the most challenging problems is the conflict between limited lifetime of the WSNs and their purpose for a long-term monitoring [18]. Although harvesting energy from surrounding environment offers a potential solution [26, 38], conventional power sources (e.g., solar) are often unavailable in an indoor environment.

Being able to harvest energy from vibrations, piezoelectric materials have long been claimed as suitable for indoor energy harvesting [11, 36]. In fact, dedicated sensor nodes designed to accept piezoelectricity power can be found in the market [3]. However, the following questions have never been fully answered: i) what kind of vibrations can be harvested indoors to produce sufficient energy? ii) how much power can be produced to support indoor sensing? iii) how to build a power management module for commonly used sensor nodes? and finally, iv) how should a WSN operate under such a harvester? These are indeed questions we intend to tackle in our paper.

One typical indoor application of WSNs is *microclimate control* [32, 33]. As *heating, ventilation and air conditioning* (HVAC) systems are extensively installed indoors and cost a huge amount of energy, indoor microclimate control aims to adapt the output of HVAC systems to population density, which may save energy on one hand while improving the comfort level of occupants on the other hand. To this end, a sensing system is required to "sit" beside the outlets and measure the speed of airflow, whose output is in turn fed to the control system for precise actuation. Obviously, the sensor nodes of this system (close to the outlets) are often far from power grid, and it is not affordable to frequently maintain them (e.g., changing batteries) given the large number of outlets. In fact, such a sensing system (if ready) can serve other indoor applications, such as *illumination control* or *air quality monitoring*. Therefore, we dedicate a prototype of energy harvesting WSN in this paper to such applications.

In this paper, we present Trinity as a self-sustaining sensing system. Trinity harvests energy from the airflow produced by HVAC outlets to power the sensor nodes, and it adapts nodes' duty-cycle to the harvested energy. Moreover, we take advantage of the physical properties of our harvester to i) synchronize sensor nodes such that a sender does not miss its receiver under very low duty-cycle, and ii) create an "energy-free sensor" for detecting the speed of airflow. Note that, because the energy harvested from HVAC system is

tiny (only of hundreds of μW), our WSN has to operate under an extremely low and synchronous duty-cycle. Moreover, the off-the-shelf airflow sensors consume higher energy than what our harvester can supply, so an energy-free sensing is necessary for Trinity to work. In summary, we make the following main contributions:

- We design a special type of energy harvester for drawing energy from indoor airflows with a speed of 2 to 6m/s. Our harvester makes use of a bimorph (cantilever with two layers of piezoelectric materials) to convert airflow induced vibration into electric power.
- We produce a general-purpose power management module to accept piezoelectricity from the harvester. It then powers a sensor node while charging the surplus energy (if any) into two thin-film batteries.
- We present two complementary synchronization strategies with very low energy consumptions. The first one calibrates the native clock of a sensor node using the periodic output of our harvester when there is no data traffic, and the other one relies on the data traffic to perform constant synchronization among nodes.
- We innovate in proposing an energy-free sensor for detecting the airflow speed, using the physical features of our energy harvester.
- We build Trinity as a prototype of a self-sustaining indoor airflow sensing system. We evaluate its performance by using our research center as a testing site.

In the following, we first briefly introduce the targeted application/problem as well as the Trinity system in Section 2. Then we present the three key modules of Trinity, namely energy harvesting, synchronous duty-cycling, and power-free sensing, in Section 3, 4, and 5 respectively. We report our field tests in Section 6, and we survey literature while discussing related issues in Section 7 before finally concluding our paper in Section 8.

2. PROBLEM AND SYSTEM

We first briefly explain the problem context, then we give a general overview of our Trinity system, as well as the rationales behind its design.

2.1 Indoor Microclimate Control and Sensing

Existing HVAC systems in large buildings are mostly controlled in a centralized manner through an *air handling unit* (AHU). Although tuning the output of certain outlets is possible, it is often not preferred as it normally requires mechanics to physically approach those outlets. Also, as tuning HVAC systems in such a way is not real-time, the outcome may lag far behind the need. For example, when a big but short-term gathering takes place in a certain section of a building, air conditioning may need to run more intensively to cool down the surrounding area. Whereas an adjustment at the AHU causes excessive power consumption for the whole building, tuning the outlets close to the concerning area could be too tardy to be useful for serving the gathering. Consequently, there is an increasing desire on *indoor microclimate control* in recently years, i.e., controlling the individual outlets of a HVAC system such that they adapt to the changes of occupancy in real-time.

A key component of indoor microclimate control for precise actuation, airflow speed sensing at individual HVAC outlets delivers an essential feedback to the control unit(s), along with other components such as temperature sensing and occupancy detection [10]. Figure 1 illustrates an indoor microclimate control system with its airflow speed sensing module highlighted. Apparently, a WSN could be a perfect candidate for this sensing task, but this task is special in that it has to be performed right at the outlets, which makes it very challenging to tackle the power supply issue. As most of the HVAC outlets are not close to the power grid, wiring the sensor nodes may cause a lot of troubles to, for example, indoor wire planning (which is what we try to avoid by wireless sensing), and is hence not a preferred choice. Moveover, a normal battery-powered WSN is not competent either, as changing batteries from time to time for a large building with tens of thousands HVAC outlets is definitely not feasible. Therefore, one would need a self-powered WSN that performs sensing and networking operations (for gathering sensed data) in an autonomous manner. Although BACnet [12] based systems might be used for the same sensing purpose, the penetration of such systems cannot be guaranteed even in developed countries such as Singapore. Moreover, a power-free WSN can complement BACnet based systems to further suppress the energy consumption of sensing.

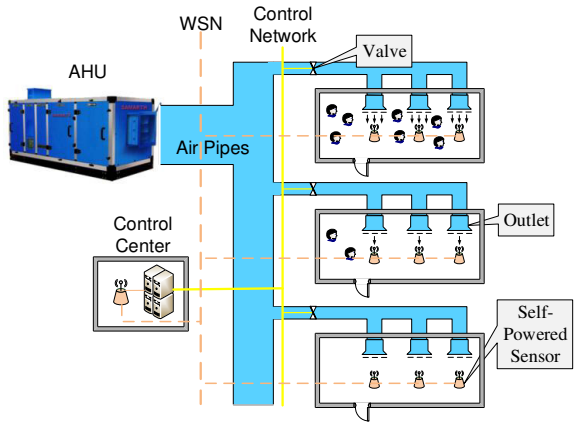


Figure 1: An indoor microclimate control system.

To tackle the aforementioned issue, we aim to develop a prototype of a *self-sustaining sensing system for airflow monitoring*. In particular, we revive the long-envisioned idea of piezoelectric energy harvesting and make a concrete case of applying it to power *indoor* WSNs. In addition, although the sensing module is designed for airflow monitoring, the energy harvesting and networking modules may serve other indoor sensing requirements (e.g., for temperature, light, and/or air quality). Finally, we intend to answer the four questions raised in Section 1 based on our field tests; these answers can yield intriguing insights and hence provide guidance for full-fledged developments and deployments of self-powered indoor sensing systems.

2.2 Trinity Sensing System Overview

We sketch the schematic of Trinity in Figure 2. The harvester is a frame with piezoelectric sheets (bimorph and unimorph) attached in the middle. We use a MicaZ Mote run-

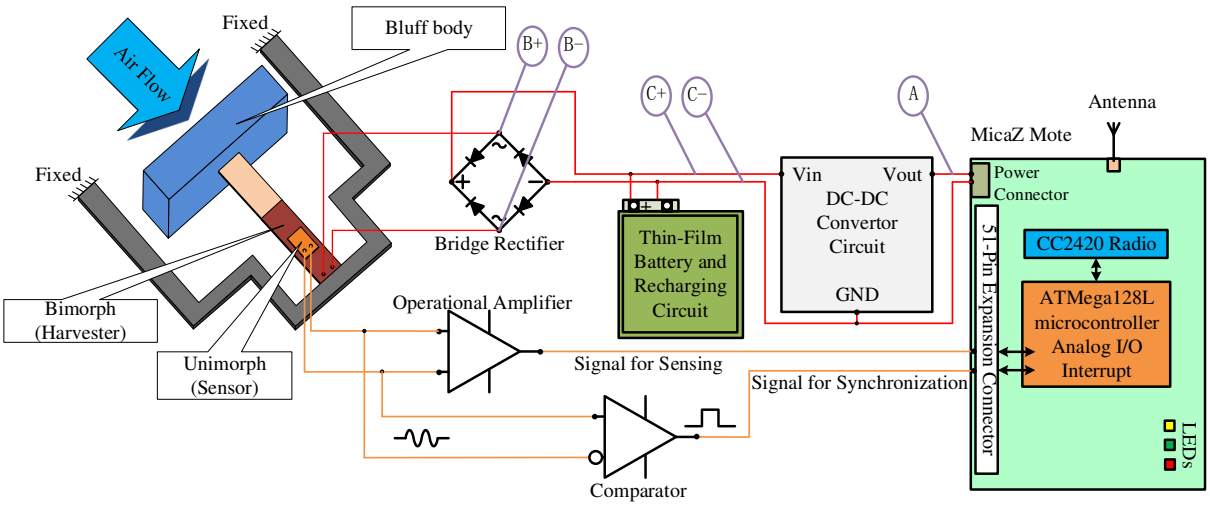
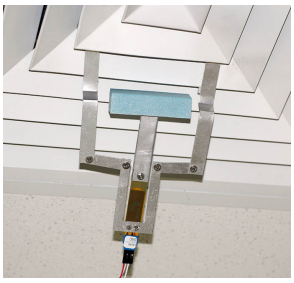


Figure 2: Trinity Indoor Sensing System.

ning TinyOS 2.1 as the sensor node platform, but *Trinity can be readily adapted to all other commonly used nodes*. Three "paths" exist between the harvester and the node. The upper one is for power supply; it goes through our power management module. The middle one samples the voltage and thus serves as our sensor. The lower one transforms the voltage signal into square waves for synchronization purpose.

2.2.1 Energy Harvesting and Power Management

As our sensor nodes are deployed right beside individual outlets, drawing power from the airflow issued by the outlets comes in handy. We choose piezoelectricity for energy harvesting because it is shown to be more efficient than other schemes (e.g., a micro turbine) given the low airflow speed [36]. In Figure 3, we show our harvester prototype



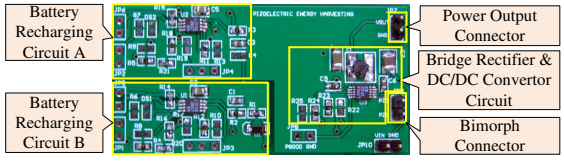
(a) The harvester

(b) A bimorph

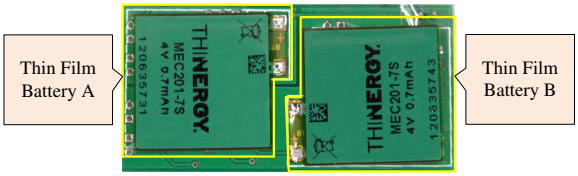
Figure 3: Energy harvester and its key component.

along with its key component, a bimorph. The harvester consists of a fixed frame that can be stuck to an HVAC outlet and a bimorph (as shown in Figure 2) with bluff body attached to better absorb the power carried by the airflow.

The power generated by the bimorph cannot be directly used by a sensor node due to the extremely high internal resistance (in the scale of mega-ohm, or MΩ) of the bimorph. In addition, we would like to have the operations of a sensor node (power consumer) as independent of the harvesting procedure (power supplier) as possible. Generally speaking, the power produced by a harvester (hundreds of micro-Watts, or μWs) is far lower than the power consumption of a sensor node in its active mode (tens of milli-Watts, or mWs). However, a naive operation mode that a bunch of packets can be sent or received only upon sufficient energy (hence voltage) is accumulated appears to be highly undesirable. Therefore, while low duty-cycle is necessary to keep a node "alive" under the tiny power supply, the actual percentage of the duty-cycle should be constrained only by the average power output from the harvester, instead of by the time period to accumulate a sufficiently high voltage. The power management module that we have produced is shown in Figure 4. It has several integrated circuits on the



(a) Front side



(b) Back side

Figure 4: Power management module.

front side to regulate the power (both voltage and current), while having two thin film rechargeable batteries attached on the back side: they serve as energy buffers to decouple the power supplier from the power consumer. Details on the energy harvesting and power management module will be elaborated in Section 3.

2.2.2 Synchronous Duty-Cycling

Given the low power output of our harvester, our WSN must perform in a very low duty-cycle manner. Further-

more, the duty-cycling of the sensor nodes needs to be synchronized so that the receivers wake up in time to receive data packets sent by the respective senders, as otherwise asynchronous duty-cycling requires extra overhead to notify either receivers or senders and it may wake up nodes not involved in the transmissions [14, 21, 30]. Unfortunately, synchronization needs to be repeated in a regular basis in order to compensate for the clock drifts of individual sensor nodes. Existing synchronization protocols (e.g., FTSP [27]) rely on periodically flooding, resulting in an unaffordable load on our system. In Figure 5, we illustrate the relative drift between two MicaZ Motes, assuming each node has a sleeping-active period of 30s including a 45ms active slot. It shows that, without a regular (re)synchronization, two initially synchronized nodes may only communicate with each other for about 50 minutes (each period takes 30s).

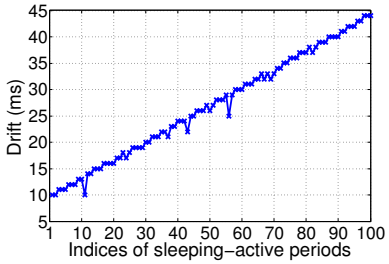


Figure 5: Relative clock drift between two nodes.

Our Trinity system first employs a low-power self-calibration strategy. Based on the physical property of our harvester that its vibration frequency depends only on its structure and material, the output of the harvester can be used as an external oscillator to calibrate the native clock of the connected sensor node. In order to avoid the interference from the network operations, we use an additional small piece of unimorph (as shown in Figure 2) for the purpose of synchronization. Although we borrow this idea from the FM synchronization reported in [24], our strategy consumes far less energy as the external oscillation comes directly from the harvester and hence needs no additional receiver or signal processing. Nevertheless, this self-calibration process still incurs an mW scale energy consumption, so we want to avoid invoking it as much as possible. Consequently, we also propose a complementary strategy for synchronization. This approach piggybacks control information with data traffic: for each transmission link, the sender synchronizes its sleeping time with that of the receiver based on the information piggyback with the acknowledgements sent by the receiver. According to our field tests, our per-link synchronization is sufficient in most cases to maintain an adequate level of successful transmissions. We will provide more details on these two strategies in Section 4.

2.2.3 *Airflow Speed Sensing*

Being an important aspect for indoor microclimate monitoring, airflow sensing can be preformed by many off-the-shelf products. However, almost all the sensor products have a high demand on power. For example, Figure 6 shows a typical airflow sensor. It has a current consumption of 15mA and requires an input voltage of 10.8 to 26.4V, which entails about 400mW power consumption. Such a huge power consumption is beyond the capacity of our energy harvester.

To this end, Trinity takes a novel approach towards sensing the airflow speed. As it can be shown (in Section 3.1) that the peak output voltage of our harvester is a function of the airflow speed, we may simply sample the voltage of the harvester to infer the airflow speed. The sampling procedure involves an amplifier circuit (shown in Figure 2) and the ADC module of MicaZ Mote's MCU; the resulting total power consumption is below 500μW. Similar to our clock calibration strategy, we again take the output from the unimorph to avoid interference from the power generation. We shall provide more details in Section 5.

Figure 6: MEMS airflow sensor D6F-10A produced by OMRON Electronic Components LLC.

3. HARVESTING AND MANAGING TINY ENERGY

In this section, we first discuss the physical principles of our energy harvester, then we explain the design of the power management module.

3.1 Harvesting Energy from Airflow

The basic idea behind our energy harvesting scheme is the *piezoelectric effect*. More specifically, certain materials produce electric charges (or *piezoelectricity*) on their surfaces as a consequence of applying mechanical stress. In our case, the mechanical stress is expected to be applied by the airflows. Therefore, our design, as shown by Figure 2 and 3(a), is meant to attach a bimorph (containing piezoelectric materials) to a frame fixed on an HVAC outlet such that the bimorph (a type of cantilever with a tip mass, the bluff body) can exhibit forced oscillation under the blow of airflow. As a result, the oscillation exerts mechanical stress on the bimorph that in turn accumulates electric charges.

Based on aforementioned principles, we may establish the following coupled lumped parameter model [36] to emulate the electromechanical coupling behavior of our harvester. The model is characterized by two governing equations:

$$M_{\mathit{eff}}\ddot{w} + C\dot{w} + Kw + \Theta V = F \quad (1)$$

$$\frac{V}{R_L} + C_p\dot{V} - \Theta\dot{w} = 0 \quad (2)$$

where w is the tip displacement in the direction normal to the airflow, C and K are the damping coefficient and stiffness of the harvester, respectively, V is the generated voltage, R_L is the applied load resistance, and C_p is the total capacitance of the piezoelectric sheets. Other parameters need a bit more elaborations. $M_{\mathit{eff}} = \frac{33}{140}M_b + M_{blu}$ is the effective mass, where M_b is the distributed mass of the cantilever, and M_{blu} is the mass of the bluff body. $\Theta = \sqrt{(\omega_{noc}^2 - \omega_{nsc}^2)M_{\mathit{eff}}C_p}$ is the electromechanical coupling term, where ω_{noc} and ω_{nsc} denote the open circuit and short circuit *natural* frequencies of the harvester, respectively. F represents the aerodynamic force acting on the tip body in the direction normal to the airflow, it can be computed as

$$F = \frac{1}{2}\rho_a h L_{blu} U^2 C_F, \quad (3)$$

where ρ_a is the air density, $h \cdot L_{blu}$ denotes the windward area of the bluff body, U is the airflow speed, and C_F =

$\sum_i A_r \alpha^r$ $(r = 1, 2, 3, ...)$ is a function of the angle of attack α, and can be determined through experiments (e.g., [13]). Here A_r denotes an empirical coefficients for the polynomial fitting. The attack α is defined by $\alpha = \frac{\dot{w}}{U} + w_a$ where w_a denotes the rotation angle of the cantilever at the free end.

The mechanical part of the equation system (1–2) describes an aeroelastic phenomenon termed *across-wind galloping* [16], whose frequency is determined by the natural frequency of the harvester. Moreover, as the system is coupled with the piezoelectric effect, there exists an *implicit function relation* between the airflow speed U and the voltage V. Finally, the vibrations incurred by galloping are converted into an alternating output voltage with a fixed frequency. We shall use these features later for our synchronization and sensing modules (see Section 4.2 and 5), but we focus only on power generation for now.

According to our estimates and measurements, different bimorphs may produce a peak-peak open-circuit voltage ranging from 20V to 50V and a short-circuit current around 100μA. This suggests a large internal resistance R_I in the scale of MΩ. Given a certain load resistance R_L, the power output is $P = \frac{V^2}{R_L}$, but this V is lower than the open-circuit voltage and is affected by the ratio between R_L and R_I. In Table 1, the parameters of two harvesters (among several others that we have tried) are shown; the main difference comes from the distinct bimorphs used. Apparently, to efficiently utilize the harvested power, a power management circuit should be added between the harvester and sensor node in order to match R_L to R_I, so that sufficient power can be generated regardless of the node's working modes.

Table 1: Two Different Harvester Designs

Parameters	Harv. 1	Harv. 2
Open circuit frequency ω_{noc} (Hz)	13.23	10.99
Short circuit frequency ω_{nsc} (Hz)	13.14	10.87
Capacitance C_p (nF)	52	18
Bimorph Type [1]	V21BL	V22BL
Mass of bluff body M_{blu} (kg)	0.0017	0.0017
Windward area of bluff body(cm^2)	20	20
Airflow speed (m/s)	5.5	5.5
Equiv R_L^a for active mode (kΩ)	1	1
Power P^a under R_L^a (μW)	4.51	2.11
Equiv R_L^s for sleeping mode (kΩ)	425	425
Power P^s under R_L^s (μW)	622	258
Power P^* under optimal load (μW)	667	268

3.2 Regulating and Buffering Energy

In general, a power management module that draws power from our harvester and supplies power to a MicaZ Mote has to achieve the following three objectives:

1. Regulate the alternating voltage (also its amplitude) to suit the 3.3V direct voltage input of a MicaZ Mote.
2. Deliver an equivalent load resistance R_L that matches the internal resistance R_I of the harvester, largely independent of the working modes of a sensor node.
3. Buffer generated power to decouple the power generation from power consumption.

[1]The specification for the two types of bimorphs can be found in `http://www.mide.com/pdfs/Volture_Datasheet_001.pdf`

Our power management module (shown in Figure 2 and 4) consists of three components: a low-loss full-wave bridge rectifier, a DC/DC converter (with a high efficiency of 85%) and a battery recharging system that can work under a low operating current. The first two components are built in chip LTC3588-1 [5]: it rectifies the alternating power generated by the bimorph into a DC power by a bridge rectifier. The output of the bridge is divided into two parts. One is converted to the working voltage of a MicaZ Mote for direct power supply. The other one is employed to recharge two batteries via a battery recharging circuit. The battery recharging/enegy storage sub-system is made by two LTC4071 chips [6] and two thin film batteries (THINERGY MEC201 from Infinite Power Solutions [9]).

The charging circuit LTC4071 has a low operating current of 550nA. It also has a disconnect function to protect the rechargeable batteries from a deep discharge and potentially irreparable damage. The thin film battery MEC201 has several desirable features, including 0.17mm thickness, 490mg weight, ultralow self-discharge rate, and a long cycle life. The battery recharging system not only stores the residual energy when the sensor node is in low-power sleep mode, but also plays a key role of power backup in case that the harvester temporarily stops working. According to our field test, even if the harvester is detached, a node may keep working for up to 20 hours with the supply delivered by MEC201.

4. NETWORK OPERATIONS UNDER EXTREMELY LOW POWER SUPPLIES

According to Table 1, the power generated by our harvester is several hundreds of μWs, but a MicaZ Mote consumes several tens of mWs in various working modes (see Table 2). Due to this deficit in power supply, a Trinity WSN has to operate under a very low duty-cycle. Fortunately, as we shall show in our field tests, such a low duty-cycle does not compromise the functionality of Trinity as an indoor sensing system. In this section, we first introduce the duty-cycling configuration for Trinity in Section 4.1, and then we propose two complementary strategies in Section 4.2 and Section 4.3, respectively, that provide a crucial synchronization service for Trinity to properly operate a WSN.

4.1 Duty-Cycling Configuration

A Trinity sensor node works in a periodic on-off manner and, as shown in Figure 7, it has a sleeping-active period T_{dc} consisting of a sleep duration T_{sleep} and an active duration T_{active}. In addition, the sensor node usually needs to spend T_{wake} (3 to 4ms) to recover from sleeping mode to active mode. Obviously, $\mathcal{R}_{dc} = \frac{T_{active}}{T_{dc}}$ is the duty-cycle.

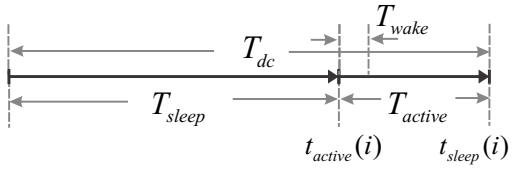


Figure 7: The sleeping-active period of a sensor node.

The large deficit in energy supply requires Trinity to not only employ an extremely low duty-cycle but also reduce the

power consumption during T_{sleep} as far as possible. According to Table 2,[2] the most energy-consuming components of

Table 2: Current draws under different modes.

Sleeping mode	20∼60μA
Active mode (radio off but MCU on)	8mA
Active mode (receiving)	22.4mA
Active mode (transmitting, 0dBm)	23.4mA
Active mode (transmitting, -10dBm)	21.8mA
Active mode (transmitting, -25dBm)	21.4mA

our MicaZ Motes are the radio module (CC2420 [2]) and MCU (ATmega128L [1]). While the radio and MCU together may draw up to 23.4mA, the MCU alone already needs 8mA. Therefore, only shutting down the radio during T_{sleep} is not enough, we have to shutdown everything except the oscillator, which leads to the sleeping mode shown in Table 2. The actual power consumption of a node under the sleeping mode varies from 66 to 198μW (given the 3.3V working voltage of a node); this allows for several hundreds of μW power surplus to be charged into the battery. Driven by these measurements, we can figure out an upper bound on the duty-cycle $\mathcal{R}_{dc}$. Let I_h, I_{active}, I_{sleep}, and I_c denote the output current of the harvester, the current draws within T_{active} and T_{sleep}, and the current consumed by the power management module (approximate 8 μA), respectively, and $\mathcal{R}_{dc}$ should satisfy

$$\frac{I_{active}V_{node}}{V_{harv}\delta}\cdot\mathcal{R}_{dc}+\frac{I_{sleep}V_{node}}{V_{harv}\delta}\cdot(1-\mathcal{R}_{dc})+I_c\leq I_h, \quad (4)$$

where V_{harv} and V_{node} are the the output voltage of the harvester and the working voltage of a node, respectively, and $\delta = 0.85$ is the efficiency of the DC-DC convertor. Any $\mathcal{R}_{dc}$ satisfying this constraint guarantees that the excessive energy cost during T_{active} can be compensated during T_{sleep}.

Detailed calculation of $\mathcal{R}_{dc}$ will be postponed to Section 6.2, we hereby briefly introduce our strategy of adaptively adjusting $\mathcal{R}_{dc}$ to satisfy (4). We fix the active duration T_{active} and empirically specify the shortest sleep-active period T_{dc} according to $\mathcal{R}_{dc}$ measured under the maximum airflow speed offered by an HVAC setting. Upon waking up, a sensor node first reads its voltage to check if it is maintained to the same level as before. If not, it doubles T_{dc} until a sustainable $\mathcal{R}_{dc}$ is reached.

Given an mA scale I_{active} and a μA scale I_h, it is obvious that the duty-cycle has to be below 1%. Now how a Trinity WSN operates under such a low duty-cycle is a key question we need to address. Obviously, asynchronous low power MACs such as [14, 21, 30] would not work (which will also be demonstrated later in Section 6.3.2), because the *effective* $\mathcal{R}_{dc}$ cannot be made very low given that a sender has to either send a long preamble or wait for a receiver to wake up. Consequently, a Trinity WSN has to operate in a synchronous manner, such that a pair of sender and receiver wake up at roughly (with accuracy down to millisecond scale) the same time. This cannot be done without a constant synchronization service running at the background, as two native clocks of an arbitrary pair of nodes may always exhibit relative drift [24] (also see Figure 5). Nevertheless, the synchronization service has to be almost energy-free, since Trinity does not have much energy budget for it. Therefore, we hereby propose two complementary synchronization strategies in the following.

[2]The data reported in Table 2 are obtained by our own experiments, which may differ significantly from the data sheet of CC2420 [2].

4.2 Self-Calibration with Harvester Vibration

One important property of the harvester vibration (briefly discussed in Section 3.1) is that, its frequency (thus that of the output voltage) is consistent with the natural frequency of the harvester, which relies only on the structure and material of the harvester [31]. For our Trinity system, each node is connected to a harvester whose natural frequency is accurately measured. As the structure of a harvester (thus its natural frequency) is robust to environment changes, its output voltage can serve as an external reference oscillator (with a known frequency) to calibrate the native clocks of the connected node in order to eliminate its clock drift. As explained in Section 2.2.2 (also shown in Figure 2), we attach an additional unimorph to the bimorph, so that they share the same frequency but come with independent voltage outputs. This is meant to avoid the interference from the load (the power draw from the node). In the following, we first explain the rationale of our self-calibration strategy, before diving into the technical details.

4.2.1 Principle of Self-Calibration

Our strategy is based on a clock calibration method, whose goal is to estimate the drift rate of a certain clock with respect to a presumably more accurate reference clock. Being aware of the instant clock drift rate, a node in a WSN would easily maintain a native clock that is synchronized with the reference clock. This idea was recently proposed for sensor node clock calibration using FM radio signal [24]. However, while the method proposed in [24] requires an independent sensing and signal processing module (which entails a power consumption comparable to that of the data transmission), our method directly gets the reference clock signal from the harvester to trigger interrupts of a node's MCU, resulting in a much lower power consumption.

Given a certain time period $\mathcal{T}$, we measure it using the reference clock by counting the number of periods of reference clock and dividing it by the known frequency f_{ref}. We denote this measurement by $\mathcal{T}_{ref}$. During the same period, a node is also able to measure it using its native clock (similarly by counting the number of ticks and dividing it by the nominal frequency f_{node} of the internal oscillator), which is denoted by $\mathcal{T}_{node}$. Comparing the two measurements on the same time period $\mathcal{T}$, we can compute the clock drift ξ [3] of the node's native clock with respect to the reference clock:

$$\vartheta(\mathcal{T}_{ref})+\mathcal{T}_{ref}=\mathcal{T}_{node}(1+\xi) \quad (5)$$

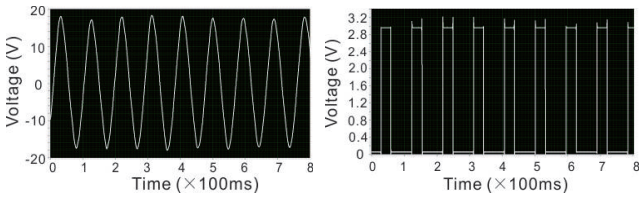
where $\vartheta(\mathcal{T}_{ref})=\mathcal{T}_{ref}-\mathcal{T}$ is a zero-mean Gaussian variable representing the error of the reference clock when measuring $\mathcal{T}$. As the clock drift rate may vary due to the environmental factors, e.g., battery level, temperature, and so on, the calibration procedure may need to be performed periodically, such that the estimation of the clock drift rate can be updated in response to the drift jitter. Therefore, we take the linear regression approach used in [24]: each node periodically measures both $\mathcal{T}_{ref}$ and $\mathcal{T}_{node}$, and maintains a table of k tuples sampled in the last k periods $\mathfrak{T}=$

[3]If the node's native clock runs slower than the reference clock, $\xi > 0$; otherwise, $-1 < \xi < 0$.

$\{\langle \mathcal{T}^k_{ref}, \mathcal{T}^k_{node}\rangle, \langle \mathcal{T}^{k-1}_{ref}, \mathcal{T}^{k-1}_{node}\rangle, \cdots\}$. Then a linear regression is applied to fit these points and hence to estimate ξ (as $\vartheta(\mathcal{T}_{ref})$ is zero-mean, the intercept of the line should goes to zero with a large k). In the following, we will discuss how to apply this principle to the context of Trinity.

4.2.2 Trinity Self-Calibration

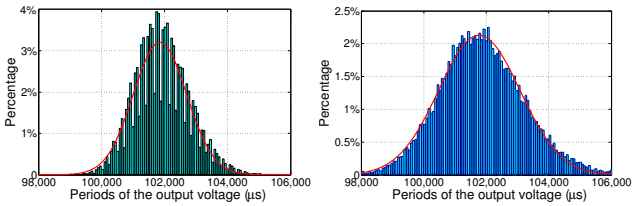
The output voltage of the unimorph is a sine wave, whose actual measurements are shown in Figure 8(a). In order to capture the zero-passing points that can be used as the ticks for the reference clock, we employ a comparator (see Figure 2) to convert the sine wave to a square wave shown in Figure 8(b). Using every two rising edges to trigger an interrupt in a node's MCU, we get a reference clock for self-calibration, whose frequency is indeed the natural frequency of the harvester (measured when it is manufactured). As the power consumption of the comparator is 1.3mW and that of MCU is negligible,[4] our self-calibration is extremely energy efficient. To show the reference clock is sufficiently stable to



(a) Sine wave (b) Square wave

Figure 8: The signals of the reference clock.

measure the drift rate, we first use an accurate clock to measure the interrupt period in Figure 9(a); the Gaussian fitting gives a mean of 101800μs.Then we use the interrupt period to calibrate the native clock of a node in Figure 9(b), resulting a mean of 101900μs. Obviously, the 100μs difference results in the drift of this native clock. As both histograms can be perfectly fitted by a Gaussian distribution, the noise around the mean will be eliminated by the linear regression.



(a) Accurate clock (b) Native clock

Figure 9: Frequency stability of the reference clock under an accurate clock (a) and the native clock of a node (b).

Our self-calibration algorithm is summarized in **Algorithm 1**. The basic idea is to perform the calibration during T_{active} in a periodic manner. As the drift is minor within each duty-cycle period, so the period of calibration can be longer than that of duty-cycling. A slight difficulty here is that, as the MCU and the comparator are both shut down during T_{sleep}, the clock cycle of the reference clock is not counted between two calibrations. Therefore, the measurement of the reference clock at the j-th calibrations, $\mathcal{T}^j_{ref}$ cannot be directly measured. Within the active duration for the j-th calibration, the node simply waits for the next interrupt (driven by the rising edges shown in Figure 8(b)). When an interrupt is fired, the native clock time $\mathcal{T}^j_{node}$ is recorded (line 2). As no interrupt has been counted since the last calibration, we have to figure out the index of this interrupt in order to compute $\mathcal{T}^j_{ref}$. Using $\mathcal{T}^j_{node}$, f_{ref} and the last estimated drift rate ξ^{j-1}, this index can be computed by rounding $(1+\xi^{j-1})\mathcal{T}^j_{node} \cdot f_{node}$ to the nearest integer, which in turn suggests $\mathcal{T}^j_{ref}$ (line 3). The remaining part simply maintains the table $\mathfrak{T}$ and performs a linear regression to estimate the drift rate ξ^j. The correctness of this computation relies on an assumption that the drift jitter of the native clock of the node can only cause a misalignment less than half of the reference clock period, which can be easily guaranteed by choosing a proper calibration period. As our reference clock has a frequency of around 10Hz, so given a drift jitter of 1ms/minute, this condition can still be guaranteed even if the calibration period is about an hour.

Algorithm 1: Self-Calibration (the j-th period)

Input: The reference clock frequency f_{ref}; the estimated drift rate of the last calibration ξ^{j-1}
Output: The estimation for the clock drift rate ξ^j

1 **upon** Interrupt
2 Record $\mathcal{T}^j_{node}$
3 $\mathcal{T}^j_{ref} \leftarrow \lfloor (1+\xi^{j-1})\mathcal{T}^j_{node} \cdot f_{node} + 0.5 \rfloor \cdot f^{-1}_{ref}$
4 **insert** $\langle \mathcal{T}^j_{ref}, \mathcal{T}^j_{node}\rangle$ into $\mathfrak{T}$
5 **if** $j > k$ **then delete** $\langle \mathcal{T}^{j-k}_{ref}, \mathcal{T}^{j-k}_{node}\rangle$ from $\mathfrak{T}$
6 Fit a line $\left(1+\xi^j\right)x + b$ to the points in $\mathfrak{T}$ using linear regression and **return** ξ^j

Assuming all nodes are initially synchronized, the calibrations at individual nodes allow every node to slightly adjust its T_{sleep} and T_{active}, such that the duty-cycling remains synchronized. This synchronization does not have to be perfect, a several-millisecond misalignment between a pair of sender and receiver (as far as it is stable) is tolerable. We shall discuss in the following a complementary strategy that can be used to perform the initial synchronization and also to further reduce the power consumption of synchronization.

4.3 Per-Link Synchronization through ACKs

The self-calibration presented in Section 4.2 only compensates the clock drift for individual sensor nodes without actually synchronizing them. Moreover, its mW scale power consumption may still be a burden for Trinity. So we hereby propose a per-link synchronization strategy. It virtually consumes no power as the information is piggyback with ACKs, and it complements the self-calibration, allowing nodes to be synchronized with each other. We first introduce the basic protocol, then we discuss its limitations and extensions.

4.3.1 The Synchronization Protocol

The objective of our per-link synchronization is to let the sender and receiver wake up and go dormant at the same time. In practice, we actually require the receiver to wake

[4] As the calibration only takes place during T_{active}, the MCU is considered to have a "flat rate" current draw of 8mA.

up slightly before the sender in order to accommodate certain system errors. In the following, we use lower-case t to represent a point in time and upper-case T for a time period. Also let t (or T) denote the wall-clock time, $\hat{t}$ (or $\hat{T}$) denote the reading from the native clock of a node, and a superscript s (resp. r) denote the sender (resp. the receiver).

Given $t^s_{sleep}(i)$ and $t^r_{sleep}(i)$ as the times for the sender and receiver to go dormant in the i-th sleeping-active period, the times for them to wake up in the $i+1$-th period are

$$t^s_{active}(i+1) = t^s_{sleep}(i) + \hat{T}^s_{sleep}(\xi^s + 1) \quad (6)$$

$$t^r_{active}(i+1) = t^r_{sleep}(i) + \hat{T}^r_{sleep}(\xi^r + 1) \quad (7)$$

where ξ^s and ξ^r are the clock drift rates of the sender and receiver, respectively. Assuming $t^s_{sleep}(i) = t^r_{sleep}(i)$ and the drift rates are known, we can (re)set the duty-cycling parameters (i.e., $\hat{T}^s_{sleep}$ and $\hat{T}^r_{sleep}$) for the sender and receiver to guarantee that $t^r_{active}(i+1) < t^s_{active}(i+1)$. As the drift rates are already estimated by our self-calibration strategy, the per-link synchronization problem is now reduced to synchronizing t^s_{sleep} and t^r_{sleep} for each sleeping-active period.

For the actual protocol, the time can only be measured by the native clock for a node. Let us consider one specific active period. The sender records a timestamp $\hat{t}^s_{tx}$ when sending a data packet. When the receiver gets the incoming packet, it also records a timestamp $\hat{t}^r_{rx}$, and it computes its time-to-dormant as $\Lambda = \hat{t}^r_{sleep} - \hat{t}^r_{rx}$. The receiver then piggybacks Λ and ξ^r with the ACK. Upon receiving this feedback, the sender sets its sleep time locally as follows

$$\hat{t}^s_{sleep} = \hat{t}^s_{tx} + \left[\Delta T + \Lambda(1+\xi^r)^{-1}\right](1+\xi^s) \quad (8)$$

where ΔT is the time cost in packet delivery (including MCU interrupt handling, decoding/encoding and propagation delay) and can be estimated offline. In practice, ΔT is only hundreds of microseconds, so omitting ΔT does not cause any problem in our synchronization protocol. As the clock drift during Λ (caused by ξ^s and ξ^r) is also negligible, we may simplify (8) to

$$\hat{t}^s_{sleep} = \hat{t}^s_{tx} + \Lambda \quad (9)$$

We give two examples in Figure 10 to visualize our per-link synchronization strategy. Basically, it shows that, regardless of the drifting direction of the sender with respect to the receiver, the two nodes can be re-synchronized as far as the two-way communications can be preserved.

While combining (6), (7) and (8) allows for an accurate per-link synchronization by making use of the outcome of our self-calibration, the per-link synchronization can also be a stand-alone protocol if we apply (9) and the following:

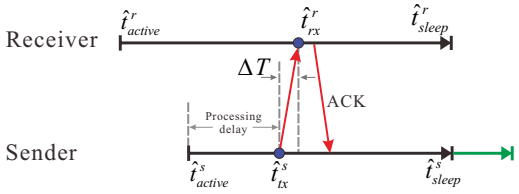
$$t^s_{active}(i+1) = t^s_{sleep}(i) + \hat{T}^s_{sleep} \quad (10)$$

$$t^r_{active}(i+1) = t^r_{sleep}(i) + \hat{T}^r_{sleep} \quad (11)$$

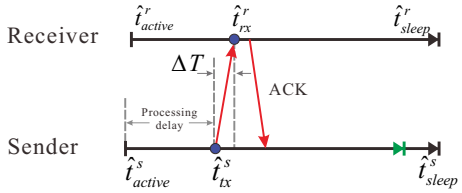
Under this simplified protocol, special care has to be taken to guarantee the receiver receives the data packet sent by the sender in the $(i+1)$-th duty-cycle period. Typically, we would require

$$0 < t^s_{active}(i+1) - t^r_{active}(i+1) < T^r_{active}$$

According to our experience, setting $T^r_{sleep} - T^s_{sleep} = 5$ms is sufficient to guarantee the difference in active time is larger than 0, and the clock drift during the sleep period cannot result in a gap larger than T^r_{active}.



(a) Sender drifts forward with respect to receiver



(b) Sender drifts backward with respect to receiver

Figure 10: Per-link synchronization between two nodes. The green arrows denote the original dormant times for the sender before synchronizing with the receiver.

4.3.2 *Limitations and Extensions*

In order for the per-link synchronization to be stable in a WSN, the network topology has to satisfy certain conditions. Instead of speculating on what topologies could accommodate this synchronization protocol, we directly organize our Trinity WSN into a tree with its root at the sink. This makes sense as the data collection functionality of a WSN is often performed on a tree topology. Within the tree, non-root nodes report data to their respective parent nodes, hence get synchronized with these parent nodes. Obviously, as our per-link synchronization forces a sender to keep up with its receiver, running this protocol within a tree network simply lets every node to stay roughly synchronized with the sink. The apparent drawback of sticking to a tree topology rather than allowing for flexible route selections based on link quality (e.g., [22]) is a potential reduction in packet delivery ratio. Fortunately, as a Trinity WSN is often deployed on the ceiling, the link quality is rather stable due to the absence of common disturbance such as human movements. Our field tests reported in Section 6.4 show that our Trinity WSN achieves an adequate packet delivery performance.

Packet or ACK loss may reduce the opportunity for nodes to get synchronized. However, our Trinity WSN has a rather high packet delivery ratio (more than 90% per link), so losing less than 10% of the synchronization opportunities should not cause much trouble. Moreover, with the drift rate estimated by the self-calibration, nodes may stay synchronized for quite a long time even without data traffic.

In fact, a straightforward extension of the per-link synchronization protocol can be used to achieve the initial synchronization. With the help of the energy pre-stored in the rechargeable batteries, the sensor nodes remain active (i.e., no duty-cycling) long enough before being synchronized. We let the sink push a SYNC message downstream along the tree. The protocol remains the same except that this SYNC replaces the ACKs: the parent nodes notify their respective child nodes of their time-to-dormant Λ, and the synchronized dormant time t_{sleep} sets a common entry point to

duty-cycling. Such a procedure could be completed in tens of seconds for a WSN with several tens of nodes. This initialization synchronization can definitely be supported by Trinity's rechargeable batteries.

5. ENERGY-FREE SENSING

As mentioned in Section 2.1, airflow speed sensing is a crucial part of an indoor microclimate control system. However, the off-the-shelf airflow sensors often consume more power than a sensor node. Obviously, Trinity has to look for an alternative solution. According to Section 3.1, there exits an implicit function relation between airflow speed U and the harvester output voltage V. Therefore, if we could experimentally obtain this function, our harvester may also serve as an airflow sensor. In fact, we do not take the voltage from the bimorph, as the voltage there may not be stable given the varying load (due to node duty-cycling, also see Section 6.2). Instead, our sensing module shares the unimorph with the self-calibration module (see Section 4.2.2): while the former samples the magnitude of the voltage, the latter only uses the zero-passing points to trigger interrupts.

According to our experience, the peak output voltage of the unimorph is up to 20V, whereas the MCU of our MicaZ Mote only supports a voltage input of 3.3V. Therefore, we employ an amplifier circuit to proportionally reduce the output voltage. The reduced voltage is introduced into the ADC module of MicaZ Mote's MCU via the 51-pin expansion connector, as shown in Figure 2. We perform several experiments in a wind tunnel (where wind speed can be controlled) with our three different types of unimorphs attached to our harvester. The resulting function relations between wind speeds and the peak voltage values read by our MicaZ Mote are shown in Figure 11. We show both mean values and standard deviations in the figure: the voltage value under a given wind speed has a variation only up to $\pm$0.06V, which suggests that using voltage to infer wind speed has an adequate accuracy with an error of $\pm$0.2m/s. We choose the third unimorph for Trinity, as it has the widest dynamic range (in terms of measuring wind speed).

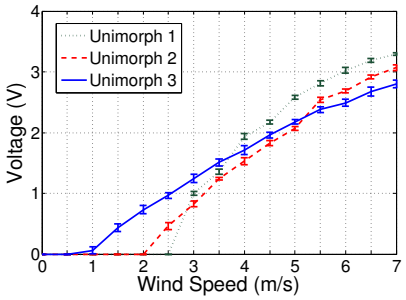


Figure 11: Peak voltage under different wind speeds.

Based on our measurement, the power consumption of the amplifier circuit is no more than 500μW. As the vibration of our harvester results in an alternative voltage whose frequency is about 10Hz (see the measurements in Section 4.2.2), the node only needs to sample the voltage for at most 100ms in order to catch a peak voltage, entailing an extremely low energy cost. Given the far higher sample rate of the ADC module (up to 76.9kHz), the error in capturing the voltage peak due to discrete sampling is negligible. Consequently, Trinity now has an almost energy-free airflow speed sensor with a satisfactory accuracy.

6. SYSTEM EVALUATIONS

We have built a prototype of Trinity and deployed a small scale WSN in our research center for more than one month. In the following, we report various evaluations that we have performed on this prototype. We start with a brief description of the deployment and evaluation settings, then we present the results concerning individual components, namely energy harvesting, networking, and sensing.

6.1 System Settings

Our Trinity prototype is built using the schematic shown in Figure 2. Our hardware mainly contains i) an energy harvester with a bimorph as the power generator (the second type shown in Table 1) and a unimorph (the third type shown in Figure 11) to produce voltage signal for both self-calibration and airflow speed sensing, ii) a power management module (see Figure 4) along with a few circuits for preprocessing the voltage signal from the unimorph, and iii) a MicaZ Mote to serve as the processing and networking unit. We implement the software part (including the two synchronization strategies and the airflow speed sensing) using NesC under TinyOS-2.1. To accurately measure the power supplying process under different networking scenarios, we make use of an NI9229 (a 4-Channel, 24-Bit analog input module) [8] and NI LabVIEW [4] to monitor the current and voltage in Trinity. Finally, we use a hand-held anemometer to verify the airflow sensing data collected by our Trinity nodes. All these equipments are shown in Figure 12, including two sets of Trinity and the monitoring equipments.

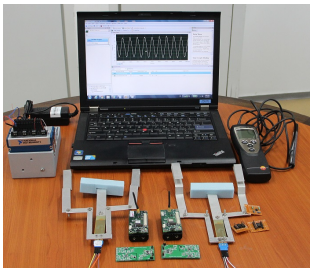


Figure 12: Lab setting for measurements.

We deploy a Trinity WSN consisting of 17 nodes on the ceiling of our research center; each node is fixed on an HVAC outlet. The WSN constantly monitors the airflow speeds of individual outlets, and reports them to the sink (node n_0), possibly through multi-hop transmissions. Given the need for a synchronous duty-cycling, we deliberately make the network topology a tree rooted at the sink (see explanations in Section 4.3.2), and the WSN uses this topology to perform data collection.

6.2 Energy Harvesting and Power Supplying

To observe the interactions between the energy harvester and the energy consumer (power management module + sensor node), we use NI9229 and LabVIEW to monitor the current and voltage at the three points marked in Figure 2. We first use Figure 14 to illustrate the voltage changing at point C.[5] The node has a 0.26% duty-cycle (T_{sleep} = 15s and T_{active} = 40ms), this satisfies the constraint (4) given $I_h = 70\mu$A for the harvester output. Therefore, the voltage should exhibit a stable and periodic pattern, which is appar-

[5]We are actually concerned with variations of the input voltage of a node, but the DC-DC converter conceals them by stabilizing its output voltage. Therefore, the testing point has to be set before the DC-DC converter.

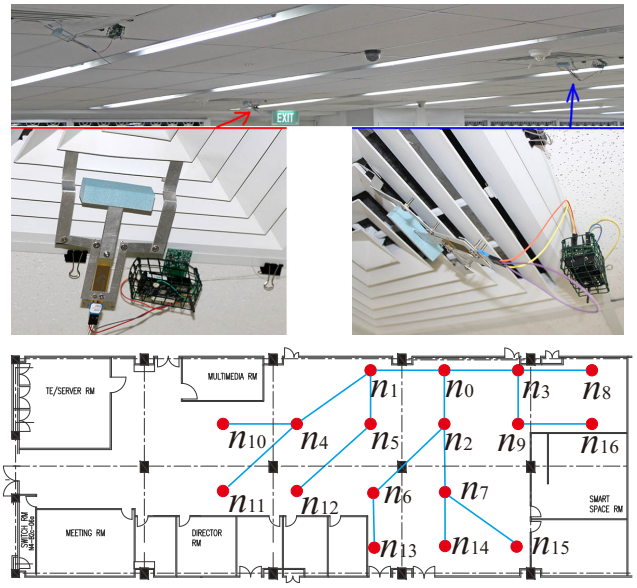


Figure 13: The Trinity WSN deployed in our 800m^2 research center. Node n_0 serves as the sink, while other nodes sense the airflow speeds of individual outlets to which they are attached and report the data to the sink.

ently the case in Figure 14. When the node becomes active (with both radio and MCU enabled), its current draw is increased to about 20mA. Due to a large internal resistance of the power supplier, the voltage suffers a drastic drop, but the power supplier (assisted by the rechargeable batteries) can still sustain the node. After the node goes dormant, the current draw is reduced to μA level, which allows the voltage to gradually recover and the batteries are hence recharged. The critical things are that the voltage should be able recover to the same level and that the voltage drop should not reach the threshold triggering the protecting circuit; these can be guaranteed if the constraint (4) is satisfied.

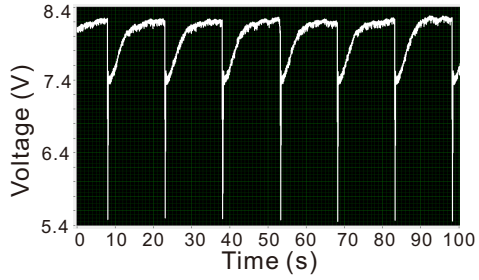


Figure 14: Voltage variations at testing point C during several sleeping-active periods.

In Figure 15, we set four different duty-cycles to verify their effects on the power supplier. Keeping $T_{active} = 40$ms and varying T_{sleep} to be 15s, 10s, 5s and 1s, we end up with four duty-cycles at 0.26%, 0.4%, 0.79%, and 3.85%. While each row is dedicated to a certain duty-cycle, columns show the measurements at different testing points: the first column is the node's current draw sampled at point A, while the other two are output voltage of the harvester and that of the batteries sampled at points B and C, respectively. Note that the first two columns have a sample period of only a minute (in order to avoid blending signals together), whereas the last column has a sample period of one hour. For each figure, we attach a small zoomed figure to show the detail of each period.

The first column shows that there is always a current spike whenever the node wakes up; this obviously corresponds to the drastic voltage drop in the last column. The actual power consumption during T_{active} is shown by the "plateau" following the spike, which can only be observed in the zoomed figure. The second column is the output voltage of the harvester, which is supposed to be close to a sine wave under open circuit (see Figure 8(a)). However, it is "deformed" (in both shape and magnitude) by the varying load. As a result, we cannot use it for self-calibration and sensing purpose. The last column is meant to demonstrate the effect of over-spending power by increasing duty-cycle. As the voltage is taken at the output of the batteries, its reduction signals the depletion of the batteries. With 0.2% duty-cycle, the voltage remains the same after one hour. Doubling the duty-cycle leads to a very small but barely discernable decrease in voltage. Further doubling the duty-cycle results in a discernable reduction, and the zoomed figure shows that the voltage can barely be recovered at the end of each period. The last one, 3.8% duty-cycle, is apparently too large for Trinity to handle: the continuous decreasing of voltage triggers the protecting circuit of the battery charging system (briefly mentioned in Section 3.2) after about 37 minutes running. Once this protecting circuit is triggered, the output of the DC-DC convert will never rise beyond 3.3V, forcing the node to shut down.

In fact, given different airflow speeds, we can estimate the equivalent output currents of the harvester. Then using the constraint (4), we may figure out the upper bounds on the corresponding sustainable duty-cycles. Table 3 shows such a correspondence within the range of airflow speeds that are meaningful for indoor environments (our outlets offer airflow speeds varying from 3 to 6m/s), assuming $V_{harv} = 9.2$V, $V_{node} = 3.3$V, $I_{active} = 25$mA, $I_{sleep} = 60\mu$A, and $I_c = 8$mA.

Table 3: DC bounds for different airflow speeds.

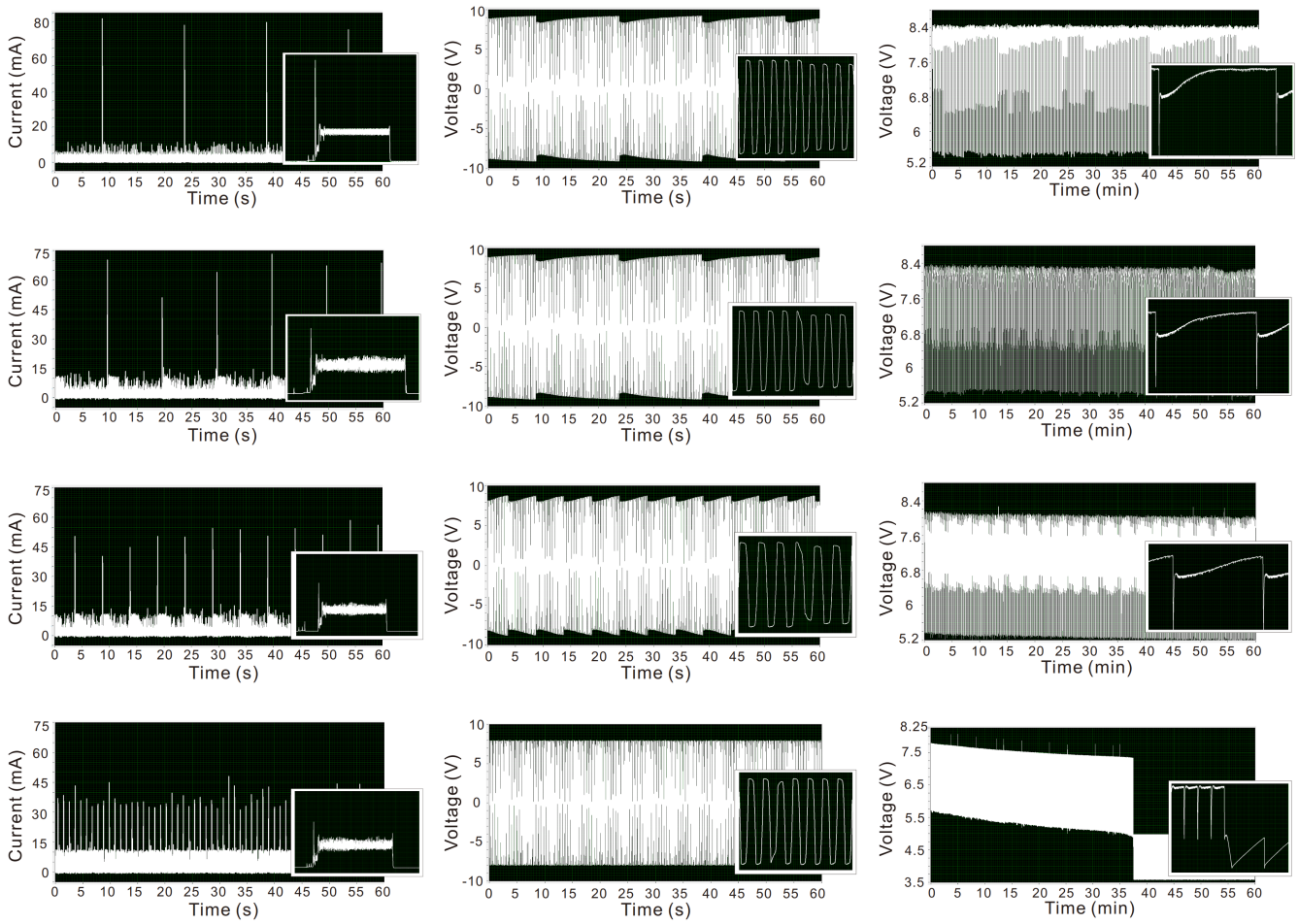
Airflow speed (m/s)	I_h (mA)	Duty-cycle
3	37.1	0.04%
3.5	44.5	0.10%
4	50.9	0.17%
4.5	58.3	0.24%
5	62.6	0.28%
5.5	68.9	0.34%
6	72.1	0.37%
6.5	75.3	0.40%
7	77.4	0.42%

6.3 Synchronous Duty-Cycling

We evaluate the effectiveness of our synchronization strategies by checking an arbitrary link within our Trinity testbed. To demonstrate the necessity of a synchronous duty-cycling, we employ A-MAC [21] on that link and compare it with our duty-cycling in terms of sustainability.

6.3.1 *The Effectiveness of Trinity Synchronization*

We apply our two synchronization strategies separately on an arbitrarily link in our Trinity WSN, and we run the



(a) The current at testing point A (b) The voltage at testing point B (c) The voltage at testing point C

Figure 15: Monitoring energy harvesting and power supplying at different testing points marked in Figure 2. Rows correspond to different duty-cycles: 0.26%, 0.4%, 0.79%, and 3.85%.

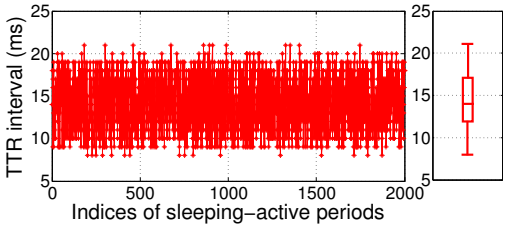
tests for more than 8 hours with CSMA enabled. As the objective is to avoid the receiver missing the packet from the sender rather than perfectly synchronize the sensor nodes in μs scale, we report the results as, instead of synchronization accuracy, the *time-to-receive* (TTR) interval between the active time and the receiving time at the side of receiver, i.e., $\hat{t}^r_{rx} - \hat{t}^r_{active}$ in Figure 16. Besides the original data shown in the left side, we also plot the interquartile range and median value to facilitate understanding the experiment results. According to our measurements, the average time in sending one packet is about 10ms for MicaZ Motes. Considering the receiver wakes up 5ms earlier than the transmitter, the receiver is supposed to get the data packet in 15ms after waking up, which accords with the median values shown in the box plots. The fluctuation in TTR intervals stems from the variations in channel accessing and sending. In summary, both our synchronization strategies could effectively correct the clock drift between a pair of sender and receiver.

6.3.2 Synchronous vs. Asynchronous

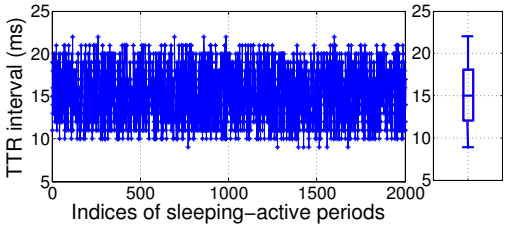
In order to show the necessity of using synchronous duty-cycling for Trinity, we apply A-MAC [21] to one of the links in our Trinity WSN using the same duty-cycle (0.26% with $T_{active} = 40$ms and $T_{sleep} = 15$s) as Trinity. [6] In the network, every node generates a packet during T_{active}. The basic idea of A-MAC, a receiver initiated link layer protocol, is that a sender waits for (radio on) the receiver to wake up in order to send a packet, so the effective duty-cycle could be larger due to the waiting time. From our knowledge, many factors may result in a large effective duty-cycle for A-MAC, including the discrepancy of wake-up times and also the loss of the notification from the receiver upon waking up. In fact, our comparison favors A-MAC: as the whole WSN is initially synchronized, the sender does not need to wait long at the beginning, but the effective duty-cycle will get larger due to the relative clock drift between the sender and receiver.

Figure 17(a) shows that the sender of the link running our synchronous duty-cycling still works after a continuous operation of 2 hours (in fact, the link has been working for more than a month since our tests). However, running A-

[6]The sender-initiated low-power listening (LPL) [30] does not support very low duty-cycle due to the limit on the preamble length.

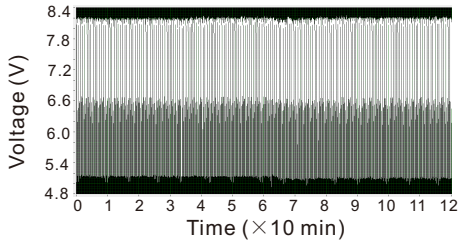


(a) Self-calibration only

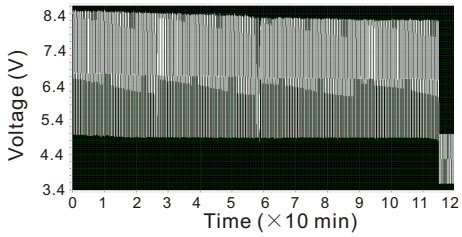


(b) Per-link synchronization only

Figure 16: Performance of Trinity synchronization.



(a) Trinity synchronous duty-cycle



(b) A-MAC asynchronous duty-cycle

Figure 17: Comparing our synchronous duty-cycling with A-MAC.

MAC "kills" that node in less than 2 hours. According to the clock drift rate between the two nodes obtained experimentally, the link can remain "alive" for at most 9 hours due to the increasing effective duty-cycle caused by the clock drift. However, if a notification from the receiver gets lost, the sender is forced to wait for several seconds until the next notification arrives. This is shown by the "glitches" in Figure 17(b), and as such a situation drastically increases the effective duty-cycle, it kills a node sooner than expected.

6.4 Network Performance

Our Trinity WSN shown in Figure 13 has been running for months, hence demonstrating the validity of its self-powered design. However, as nodes in the WSN are all running under very low duty-cycles (determined by the airflow speeds of the individual outlets to which they are attached), there might be a suspicion on the *packet delivery ratio* (PDR) for individual nodes. Therefore, we compare our Trinity WSN with the same network running CTP [22] but powered by alkaline batteries. Note that the routing topology of CTP is not fixed, but rather determined by the link qualities. For fairness purpose, the sending rates of individual nodes are the same for both scenarios (i.e., one packet per 15.04 seconds), but the CTP WSN does not involve duty-cycling (to serve as a convincing reference). As a result, whereas we collect data through the whole lifetime (months for now) of the Trinity WSN, the data from CTP is obtained for a couple of days since batteries run out.

Due to the low packet generation rate, CTP can achieve an almost 100% PDR for every node: a PDR of 96.8% can be reached even in the worst case (e.g., for node 16). This makes CTP a good reference for us to evaluate the performance of Trinity. We calculate the ratio of PDR (relative PDR) between our Trinity WSN and the CTP WSN for every node. As shown by Figure 18, Trinity works sufficiently well in the sense that most of the relative PDRs are beyond 80% and 8 out of 16 are over 90%. Moreover, the relative PDR is lower for a node further (in hop) from the sink. This is expectable as Trinity uses a fixed tree topology for data collection, where CTP adapts its routing paths to the link qualities. However, this is a (small) price Trinity has to pay for being able to operate under a very low duty-cycle.

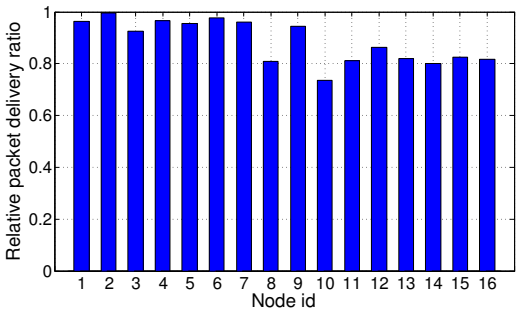


Figure 18: Relative PDRs between Trinity and CTP.

6.5 Airflow Monitoring in Action

According to Section 5, our Trinity WSN can sense the airflow speeds of the outlets that they are attached to, with an almost zero energy consumption. Therefore, we collect the readings from our airflow sensors through the WSN during the February 2013. At the same time, we also let someone use the hand-held anemometer to check the same spots intermittently. We first plot Figure 19 to show the peak voltage outputs of five different sensors. Although the voltage values keep varying, the variances are negligible and the mean values can definitely be used to derive the airflow speeds (by consulting a table that is used to produce Figure 11). Since our research institute does not allow the HVAC system to be freely tuned, we simply observe a constant airflow speed at a given spot. In Figure 20, we show the monthly averages of the airflow speeds measured at the 16 outlets by both Trinity and the hand-held anemometer. The differences between the two sets of measurements are minor, and they can result from the errors of both approaches. Most importantly, we are now convinced that we can totally rely on Trinity to re-

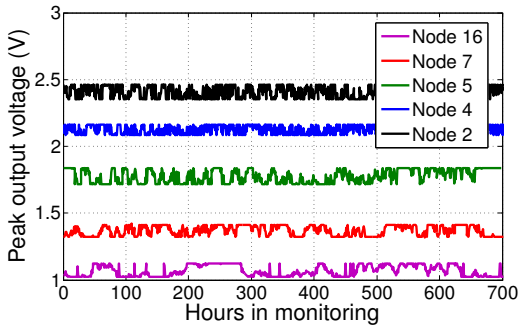


Figure 19: Five nodes' voltage readings in 30 days.

port the measurements, without bothering to have someone climbing up to the ceiling several times a day.

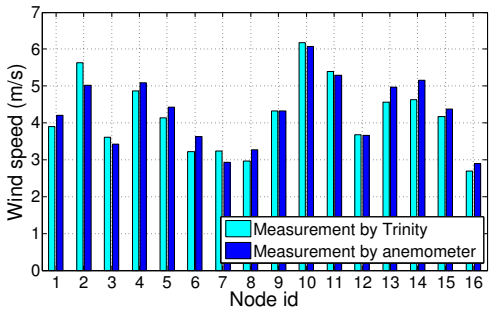


Figure 20: Average airflow speeds measured at 16 outlets by Trinity and a hand-held anemometer.

7. RELATED WORK AND DISCUSSIONS

The recent decade has witnessed an increasing interest in harvesting energy for sustainable WSNs [15]. As the WSNs are usually applied to monitor surrounding environment, a natural choice is to extract ambient energy, e.g., solar [17, 26, 38], vibration [34], heat [28, 37], and radio [29].

Solar power is popularly explored by many proposals. As the first solar-based system integrating energy harvesting and power management, Heliomote [26] equips Mica2 Motes with solar panels and uses an adaptive schedule mechanism to bridge the applications at upper layer to the power management at lower layer. Corke *et al.* [17] studies the effects of combining a DC-DC converter with NiMH batteries to build a solar-drive wireless sensor networks. Zhu *et al.* [38] build their solar-based platform using ultra-capacitors instead of rechargeable batteries. Keeping an eye on the leakage of the capacitor, the network behaviors can be adaptively adjusted thereby prolonging the lifetime of the network.

Thermal energy harvesting is a recently explored field. Temperature difference of the steam pipeline is used to produce power up to 0.8W [37], and the thermoelectric harvester is again used in [28] to harness the small temperature difference in water. Due to the heavily limited energy supply, the latter system adopts an extremely low power wake up strategy: a node wakes up only upon a water flow event that supplies it with sufficient energy. Our work goes close to [28, 37] in that we all deliver an integrated sensing system that draws energy from the sensing objects, but Trinity differs from these existing proposals by positioning itself as an indoor sensing system, and it indeed faces great challenges very different from those in [28, 37]. While the energy that can be harvested by Trinity is far lower than that from steam pipelines [37], Trinity has to constantly monitor the environment instead of being triggered only by events [28].

Trinity has been tested only in Singapore (a tropical country) where the HVAC systems are always on. The airflow speeds may be decreased for energy-saving purpose, but a minumum air flow speed required by our harvesters can be guaranteed. For more general cases where some outlets even the whole HVAC system may be turned off, a re-synchronization is required and it may leverage the initialization operation presented in Section 4.3.2. The system building cost, which is about 80USD except the radio board, may be another concern. The current cost is reasonable for a prototype, but we are working towards a better system integration that may further lower the cost.

8. CONCLUSIONS

We have presented Trinity as a self-sustaining and integrated indoor sensing system. Trinity encapsulates three main components: energy harvesting, synchronous duty-cycling, and sensing, and it fully sustains itself by harvesting the energy from the airflow issued by the HVAC outlets. Being the very first one of its kind (to the best of our knowledge), Trinity endows us with the privilege to answer several questions that were never fully addressed. First of all, we are now convinced that the vibration caused by indoor airflow is a promising resource for piezoelectric energy harvesting. Secondly, we have obtained the first-hand experiment results on the amount of power that can be generated (under various conditions) by this type of piezoelectric energy harvesting (which is far lower than what can be obtained outdoors). Thirdly, we have gained sufficient experience on designing a proper power management module to marry a piezoelectric energy harvester and a commonly used sensor node. Finally, although indoor energy harvesters can only generate limited energy, Trinity serves as a perfect demonstration that a meaningful indoor sensing system can sustain itself by relying on these energy harvesters, with a carefully designed network operation (in particular duty-cycling) mode.

At the meantime, we are testing Trinity in a data center, where a sensing system monitoring rack temperatures is crucial [25] but the HVAC system works in a different manner. We are also integrating Trinity with a novel multi-channel MAC [23] to improve its reliability. Moreover, a new integrated design that packs everything into one circuit board is being studied. Finally, we intend to add more nodes into our current deployment and open it for serving public testing. Different from existing testbeds (e.g., [7, 19]), our Trinity-based testbed will be dedicated to testing indoor applications that accommodate a low duty-cycle.

9. REFERENCES

[1] 8-bit Atmel Microcontroller with 128KBytes In-System Programmable Flash. `www.atmel.com/Images/doc2467.pdf`.

[2] Chipcon's CC2420 2.4G IEEE 802.15.4/ZigBee-ready RF Transceiver. `www.ti.com/lit/ds/symlink/cc2420.pdf`.

[3] EH-Link?Energy Harvesting Wireless Node. `www.microstrain.com/energy-harvesting/eh-link`.

[4] LabVIEW System Design Software. `www.ni.com/labview/`.

[5] LTC3588-1 Piezoelectric Energy Harvesting Power Supply. `cds.linear.com/docs/en/datasheet/35881fa.pdf`.

[6] LTC4071 Li-Ion/Polymer Shunt Battery Charge System with Low Battery Disconnect. `cds.linear.com/docs/en/datasheet/4071fc.pdf`.

[7] MoteLab: Harvard Sensor Network Testbed. `motelab.eecs.harvard.edu/`.

[8] NI 9229 4-Channel, 24-Bit Analog Input Module. `sine.ni.com/nips/cds/view/p/lang/en/nid/208796`.

[9] THINERGY MEC201 Solid-State Rechargable Micro-Energy Cell. `www.infinitepowersolutions.com/images/stories/downloads/controlled_documents/DS1012.pdf`.

[10] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng. Occupancy-driven Energy Management for Smart Building Automation. In *Proc. of the 2nd ACM BuildSys*, pages 1–6, 2010.

[11] S. Anton and H. Sodano. A Review of Power Harvesting Using Piezoelectric Materials (2003-2006). *Smart Materials and Structures*, 16(3):R1–R21, 2007.

[12] ASHRAE. ANSI/ASHRAE Standard 135-1995: BACnet, 1995.

[13] A. Barrero-Gil, A. Sanz-Andrés, and M. Roura. Transverse Galloping at Low Reynolds Numbers. *J. Fluids Struc.*, 25(7):1236–1242, 2009.

[14] M. Buettner, G. Yee, E. Anderson, and R. Han. X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks. In *Proc. of the 4th ACM SenSys*, pages 307–320, 2006.

[15] A. Chandrakasan, R. Amirtharajah, S. Cho, J. Goodman, G. Konduri, J. Kulik, W. Rabiner, and A. Wang. Design Considerations for Distributed Microsensor Systems. In *Proc. of IEEE CICC*, pages 279–286, 1999.

[16] R. Clark, D. Cox, H. Curtiss, J. Edwards, K. Hall, D. Peters, R. Scanlan, E. Simiu, F. Sisto, T. Strganac, and E. Dowell. *A Modern Course in Aeroelasticity.* Springer, 1995.

[17] P. Corke, P. Valencia, P. Sikka, T. Wark, and L. Overs. Long-duration Solar-powered Wireless Sensor Networks. In *Proc. of the 4th ACM EmNets*, pages 33–37, 2007.

[18] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore. Environmental Wireless Sensor Networks. *PIEEE*, 98(11):1903–1917, 2010.

[19] M. Doddavenkatappa, M. Chan, and A. Ananda. Indriya: A Low-cost, 3D Wireless Sensor Network Testbed. *Testbeds and Research Infrastructure. Development of Networks and Communities*, 90:302–316, 2012.

[20] Q. Dong, L. Yu, Z. Hong, and Y. Chen. Design of Building Monitoring Systems Based on Wireless Sensor Networks. *Wireless Sensor Netowrks*, 2(9):703–709, 2010.

[21] P. Dutta, S. Dawson-Haggerty, Y. Chen, C. Liang, and A. Terzis. Design and Evaluation of a Versatile and Efficient Receiver-initiated Link Layer for Low-power Wireless. In *Proc. of the 8th ACM SenSys*, pages 1–14, 2010.

[22] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proc. of the 7th ACM SenSys*, pages 1–14, 2009.

[23] F. Li, J. Luo, G. Shi, and Y. He. FAVOR: Frequency Allocation for Versatile Occupancy of spectRum in Wireless Sensor Networks. In *Proc. of the 14th ACM MobiHoc*, pages 39–48, 2013.

[24] L. Li, G. Xing, L. Sun, H. Wei, R. Zhou, and H. Zhu. Exploiting FM Radio Data System for Adaptive Clock Calibration in Sensor Networks. In *Proc. of the 9th ACM MobiSys*, pages 169–182, 2011.

[25] C. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao. RACNet: A High-Fidelity Data Center Sensing Network. In *Proc. of the 7th ACM SenSys*, pages 15–28, 2009.

[26] K. Lin, J. Yu, J. Hsu, S. Zahedi, D. Lee, J. Friedman, A. Kansal, V. Raghunathan, and M. Srivastava. Heliomote: Enabling Long-lived Sensor Networks Through Solar Energy Harvesting. In *Proc. of the 3rd ACM SenSys*, pages 309–309, 2005.

[27] M. Maroti, B. Kusy, G. Simon, and A.Ledeczi. The Flooding Time Synchronization Protocol. In *Proc. of the 3th ACM/IEEE IPSN*, pages 39–49, 2004.

[28] P. Martin, Z. Charbiwala, and M. Srivastava. DoubleDip: Leveraging Thermoelectric Harvesting for Low Power Monitoring of Sporadic Water . In *Proc. of the 10th ACM SenSys*, pages 225–238, 2012.

[29] H. Nishimoto, Y. Kawahara, and T. Asami. Prototype Implementation of Ambient RF Energy Harvesting Wireless Sensor Networks. In *Proc. of IEEE Sensors*, pages 1282–1287, 2010.

[30] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of the 2nd ACM SenSys*, pages 95–107, 2004.

[31] J. Sirohi and R. Mahadik. Harvesting Wind Energy Using a Galloping Piezoelectric Beam. *J. Vib. Acoust.*, 134(1):011009.1–011009.6, 2012.

[32] J. Taneja, A. Krioukov, S. Dawson-Haggerty, and D. Culler. Enabling Advanced Environmental Conditioning with a Building Application Stack. In *Proc. of the 4th IGCC*, 2013.

[33] N. Watthanawisuth, A. Tuantranont, and T. Kerdcharoen. Microclimate Real-time Monitoring based on ZigBee Sensor Network. In *Proc. of IEEE Sensors*, pages 1814–1818, 2009.

[34] C. Williams and R. Yates. Analysis Of a Micro-electric Generator For Microsystems. In *Proc. of the 8th Int'l Conf. on Solid-State Sensor and Actuators and Eurosensors*, pages 369–372, 1995.

[35] Z. Wu, Z. Liu, X. Huang, and J. Liu. Real-time Indoor Monitoring System Based on Wireless Sensor Networks. In *Proc. of SPIE*, pages 22–25, 2009.

[36] Y. Yang, L. Zhao, and L. Tang. Comparative Study of Tip Cross-Sections for Efficient Galloping Energy Harvesting. *Appl. Phys. Lett.*, 102(6):064105:1 – 064105:4, 2013.

[37] C. Zhang, A. Syed, Y. Cho, and J. Heidemann. Steam-powered Sensing. In *Proc. of the 9th ACM SenSys*, pages 204–217, 2011.

[38] T. Zhu, Z. Zhong, G. Yu, T. He, and Z. Zhang. Leakage-aware Energy Synchronization for Wireless Sensor Networks. In *Proc. of the 7th MobiSys*, pages 319–332, 2009.