EleTrack: Ultra-Low-Power Retrofitted Monitoring for Elevators

Mobashir Mohammad
Ackcio Pte. Ltd.
mobashir@ackcio.com

Raj Joshi
National University of Singapore
rajjoshi@comp.nus.edu.sg

Mun Choon Chan
National University of Singapore
chanmc@comp.nus.edu.sg

Abstract

As elevators are essential in high rise buildings, their safety and reliability are naturally a major concern. Elevators are mechanical devices that require regular maintenance and monitoring. However, retrofitting elevators with a remote monitoring solution is quite challenging due to lack of available power source and the difficulty in accessing the data remotely.

In this paper, we present EleTrack, an ultra-low-power solution to retrofit existing elevators for remote monitoring. EleTrack, a *plug-n-play* solution, requires minimal human intervention, uses a novel sensor-assisted duty-cycling of radio for energy savings and provides round-the-clock collection of elevator-related sensor data for predictive maintenance. Our evaluation shows that EleTrack is up to 3.2 times more power-efficient than state-of-the-art duty-cycling mechanisms and can last for close to three years on a coin cell battery.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

monitoring, elevator maintenance

Keywords

elevator, monitoring, sensor-assisted duty-cycling

1 Introduction

Elevators are essential in high rise buildings. The number of elevators installed in the three most populous countries (China, India, and the USA) exceeds 4 million[1]. Globally, elevators make 18 billion passenger trips each day [1]. According to the U.S. Bureau of Labor Statistics and the Consumer Product Safety Commission, injuries arising from the use of elevators affect about 10,000 people each year[2]. Elevator safety and reliability is, therefore, a major concern with an estimated elevator modernization market of $13.92B[3] targeting many old elevators globally.

Elevators are mechanical devices with an expected lifespan of up to 40 years[4]. Thus, they require regular maintenance and monitoring. Monitoring is required not only for safety reasons but also for optimizing the elevator travel algorithm from time to time to improve the efficiency of elevator operations. Our vision is to provide a low-cost, plug-n-play retrofit to old elevators which lack built-in remote monitoring system. However, retrofitting is challenging due to the following reasons. First, a power source might not be readily available for the new monitoring system. Second, there is no easy way to connect the installed monitoring system to a remote gateway that collects and processes the information. While wireless communication is an option, the elevator's movement and the obstacles created by the building infrastructure pose a significant challenge for data exchange between the sensors on the elevator and the gateway.

Given these challenges, a retrofitted monitoring system for elevator needs to be: (i) low-power so that it can operate for a long duration without a walled power supply, (ii) able to reliably communicate the sensor data so as to identify usage patterns and abnormalities and (iii) easily deployable without requiring any user input or calibration.

In this paper, we present EleTrack, a monitoring system designed to retrofit existing (legacy) elevators. EleTrack provides a small but important set of elevator information (current level, moving/stationary, door open/closed, etc.) using two sensors (barometer and magnetometer), which could be easily extended further. The key novelty of EleTrack's ultra-low-power design is that it uses low-power sensors to track the elevator state and actuate higher power sensors and radio only when required. In other words, it performs *sensor-assisted aggressive duty-cycling*.

EleTrack has two components, a small, battery-powered wireless monitoring device that is placed *inside* the elevator cabin for easy deployment and a gateway device with Internet access that is placed on one of the levels close to the elevator door. EleTrack's design can be summarized as following:

[1] www.astarlifts.com

[2] consumerwatch.com/workplace-public-safety/elevators

[3] http://www.businesswire.com

[4] https://www.whatech.com

- A barometer sensor is used to track elevator movement and the level reached and a magnetometer sensor is used to track door opening/closing events.
- A level tracking algorithm uses barometer inputs to first *learn* the building characteristics (ceiling height, level count, etc.) and then starts tracking the detailed elevator usage without requiring any user input or calibration.
- A sensor-assisted duty cycling algorithm that uses a set of sensors to identify when communication with the gateway should be performed (when the door is open and the elevator is at a level where the gateway device is installed).
- A network-coding-based transmission scheme with block acknowledgment (B-ACK) to improve transmission throughput and reliability.

We have fully implemented the monitoring device on the TI SensorTag platform running the Contiki OS. The gateway is implemented using a RaspberryPi attached to another SensorTag. Our evaluation shows that:

- The barometer-based level tracking algorithm is able to *learn* the building characteristics in less than 10 journeys with real-world elevator usage patterns. Further, it is able to correctly track the levels of a 5-floor and a 7-floor buildings with 100% accuracy (evaluated over 13 hrs combined) and that of a 25-floor building with 97.8% accuracy (evaluated over an hour).
- With our sensor-assisted duty-cycling algorithm, EleTrack draws only 40.76 μA of current leading to a standby time of 2.8 years on 1 Ah coin cell battery. This is a 3.19-fold improvement over the default ContikiMAC duty-cycling for the purpose of detecting transmission opportunity.
- EleTrack's network coded B-ACK transmission scheme offers up to 21.6 times higher throughput than the standard CSMA/CA based per-packet acknowledgment scheme. This directly translates to significant reduction in energy consumption due to much lower radio-on time.

The rest of the paper is organized as follows. §2 provides more details on the challenges involved and based on that we present the design of EleTrack and its components in §3. §4 describes the implementation details while §5 presents the evaluation results. §6 puts light on the real-world elevator abnormalities found by EleTrack so far while §7 summarizes the related work. We discuss the limitations and potential improvements for EleTrack in §8 and finally conclude in §9.

2 Motivation

The elevator is a complex mechanical system that requires continuous monitoring to provide useful insights into its regular operation and also to provide the operators with sensor data for predictive maintenance and diagnosis. Many sensors can be installed on elevators to measure parameters such as position, speed, acceleration, temperature, load, etc. However, depending on the design of the elevator, many of these sensors may not be pre-installed, and if installed, the sensor data may not be easily available for remote data collection and analysis. Therefore, it is desirable to design a monitoring system that can easily retrofit any given elevator. For the ease of deployment, the solution should involve close to zero effort in terms of parameter inputs, calibration or arranging for a persistent power supply. To avoid the need of a persistent power source, the solution should also be energy efficient to last for years, running on small batteries. Finally, the system should be small and unobtrusive to easily blend in any elevator cabin. In this section, we highlight the key challenges in realizing such a monitoring system.

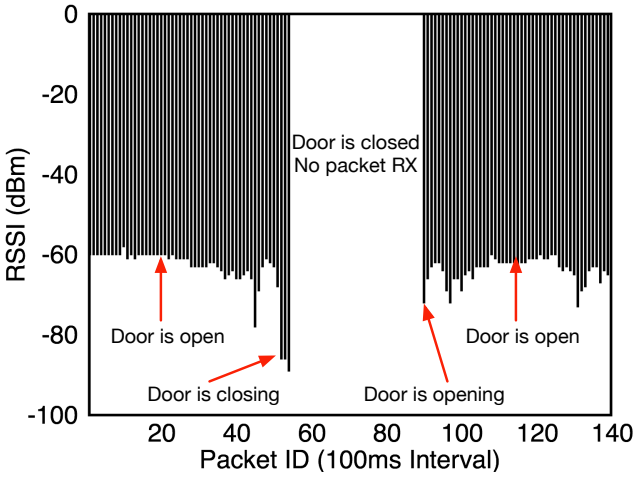


Figure 1. Impact of elevator doors on packet reception.

Limited Communication Opportunity. The monitoring device needs to be placed *inside* the elevator cabin since it is the elevator cabin's parameters that are to be monitored. As elevator cabins are often built using multiple layers of steel and installed inside reinforced concrete channels in a building, the metallic frame and the walls simulate a **Faraday Cage Effect**. This phenomenon coupled with absorption, reflection, and scrambling of radio waves by the elevator enclosure leads to data loss and the creation of *dead zones* for any form of radio communication that requires external electromagnetic signals.

To understand the impact of the elevator structure on radio communications, we placed a 802.15.4 transmitter inside the elevator cabin which continuously (every 100 ms) transmitted packets to a gateway placed about 3 m outside the elevator cabin. Figure 1 shows the signal strength of the packets received by the gateway when the door of the elevator cabin is in different states. We see that when the door is open, all the packets are received successfully with RSSI around -60 dBm. When the door starts closing (around packet id 50), the RSSI starts to drop until no packets are received when the door is fully closed. Communication is restored (around packet id 90) when the door starts to open again.

Limited Network Coverage. In the light of the aforementioned Faraday Cage effect, communication opportunities would increase if multiple gateways are installed on different floors to provide improved coverage. However, this would add unnecessary deployment costs to the system, especially for high-rise buildings. A more pragmatic solution would be to install a gateway at only one level. Elevator sensor data can be buffered and then transferred to the gateway

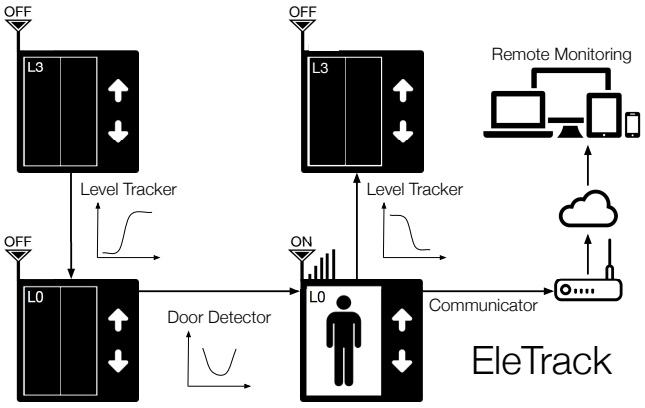


Figure 2. EleTrack System Design Overview.

when the elevator stops at the gateway level. This significantly reduces the deployment cost and effort. However, it increases the system complexity since the sensors now have to track which level the elevator is at and communicate only when the gateway level has arrived and when the door has opened.

Unreliable Communication. In the wake of small communication opportunity and limited network coverage, the buffered sensor data can be transferred to the gateway only during a short period (when the door is open at the gateway level). With cross-technology interference (in particular if 2.4 GHz band is used) transferring the buffered sensor data in a small time window demands a reliable and high throughput transmission scheme. Typical low-power wireless communication requires ACKs for every packet to ensure reliability. This to and fro packet exchanges significantly slow down the data transfer, especially when the links are unreliable.

Design for Ultra-Low-Power. Elevators usually have a direct power supply available for lighting, ventilation, etc. However, an energy efficient and easily deployable solution that does not require any power supply tampering or maintenance downtime of a running mechanical system is still desirable. For energy efficiency, default low-power sensor network protocols involve duty-cycling of the radio to conserve energy. These protocols either involve periodic clear channel assessment (CCA) [7] or periodic low-power probing (LPP) [22]. For an elevator, as seen above, most of the time, the device inside the elevator cabin is unable to communicate with the outside world. Periodic radio-on for CCA or LPP makes sense for typical low-power sensor network applications. However, when it comes to the target scenario, such protocols are highly inefficient and would waste a lot of energy.

3 System Design

In this section, we present *EleTrack*, an ultra-low-power retrofit for **ele**vators to **track** and remotely monitor various elevator parameters. It is designed for easy deployment in a *plug-n-play* fashion and is capable of auto-calibration with no user inputs.

3.1 EleTrack in a Nutshell

EleTrack consists of two physical components:

1. A gateway device that is fixed outside the elevator door at one of the levels.
2. A monitoring device that is fixed inside the elevator cabin, near the door.[5]

The gateway device is connected to the Internet and has a direct power supply. The monitoring device is battery powered to allow quick *plug-n-play* deployment without requiring to tamper with the elevator's electrical system. The overall functionality requires the monitoring device to collect the sensor data and wirelessly communicate it to the gateway device. The gateway device, in turn, forwards it to a cloud-hosted data store for real-time monitoring and analytics for predictive maintenance. For its basic operations, EleTrack uses only two sensors: barometer and magnetometer. Additional sensors, such as an accelerometer, temperature sensor, etc. can be added as required by the monitoring application.

3.1.1 Deployment and Operations

System deployment involves a user installing the gateway device outside the elevator at any arbitrary floor. He then calls the elevator to the same[6] floor and when the elevator arrives, he simply *sticks* the monitoring device inside the elevator cabin (near the door) and the system starts. Note that the system has no information about the total number of floors in the building, the building's ceiling height or the starting floor where the gateway is installed. The monitoring device has a functional component called the **level-tracker** (§3.2) which after a few initial elevator "journeys" automatically learns the building's ceiling height (in terms of pressure difference) and starts tracking the current level of the elevator accurately. When the level-tracker finds that the elevator has reached the floor at which the gateway is installed, another component called the **door-detector** (§3.3) activates. The door-detector identifies the door open and door close events, which further activates the **data communicator** (§3.4) functional component whenever the door is open. The data communicator uses a reliable high-throughput data transmission scheme to send out a large amount of information to the gateway device within the short time when the door is open. Figure 2 shows the three functional components that constitute EleTrack. The following subsections discuss each of the functional components in detail.

3.2 Level Tracker

The level tracker uses barometer signal to track the current level of the elevator. In this section, we first introduce the notion of maintaining a cumulative sum of pressure differences (tracking delta) which forms the key idea behind our level tracking algorithm. We then provide an overview of the level tracking algorithm followed by the details of individual sub-components.

3.2.1 Notion of Tracking Delta

For tracking the elevator levels, there are two reasons why absolute air pressure measured by the barometer cannot be used. First, the semidiurnal atmospheric pressure tide [18]

[5]The sensing device is placed inside the elevator is to help applications to capture vibrations, accelerations, occupancy, humidity, temperature, etc.

[6]in principle, the monitoring device could be installed when the lift is at a different floor. For the sake of simplicity of the discussion, we assume the two floors to be the same.

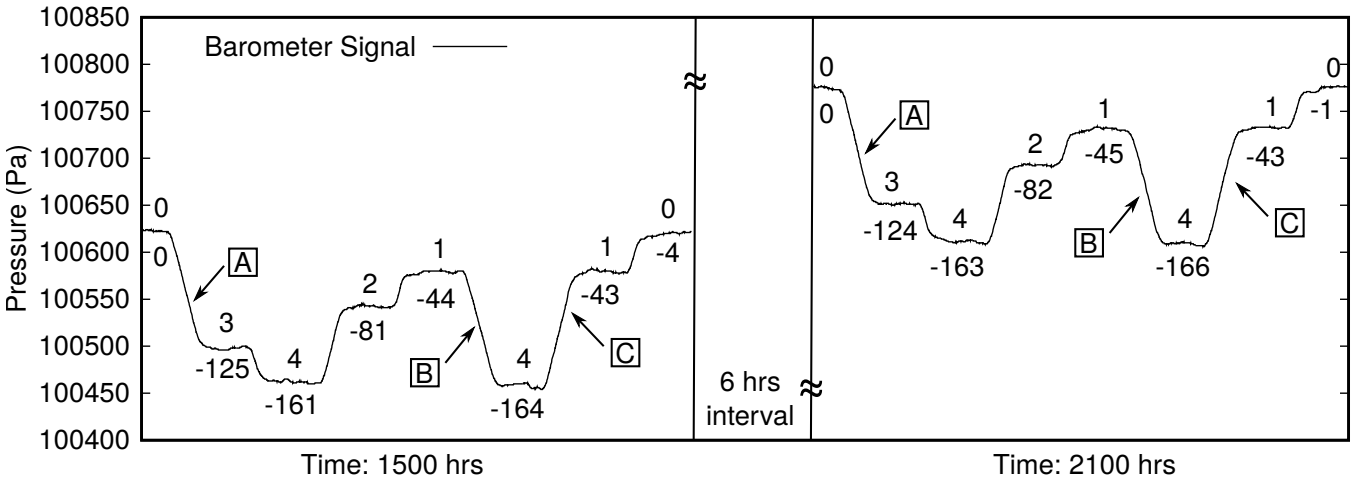


Figure 3. Atmospheric pressure for a sequence of elevator journeys at two different times of a day. Numbers above the line are the ground truth levels of the elevator while those below are the 'tracking delta' when the elevator is at that level.

causes the absolute air pressure to *drift* during the course of a day. Such a drift in absolute pressure values can be seen in Figure 3 which shows the barometer signal of an elevator when it makes the same sequence of journeys at two different times of a day. Because these pressure tide drifts affect the absolute air pressure, the barometer readings of two elevator journeys with the same start and end level can be very different. Second, the barometer sensor's absolute accuracy is limited. Commodity barometer sensors such as the Bosch BMP280 available on the iPhone6 and TI SensorTag provides *absolute* accuracy of about 100 Pa which translates to about 10 m in height. This error is too large for level detection even if there is no atmospheric pressure drift.

Our approach is to consider only the *difference* in pressure between the start level and the end level of an elevator journey. For example, consider the journeys A, B, and C in Figure 3 taken at 3 pm and 9 pm on the same day. Even though the absolute readings are very different, the pressure difference is (almost) the same for these journeys. Measurement studies [23] have shown this property to hold in general. Further, the Bosch BMP280 sensor specifies the *relative* accuracy to be ± 12 Pa, which translates to about $\pm$ 1m in height and is acceptable for our application.

Based on this observation, our insight is that if we maintain a cumulative sum of pressure differences (call it the ***tracking delta***) of all the journeys so far, **the cumulative sum would be the same each time the elevator returns to the same level**. The error depends on the relative accuracy which is much better than the absolute accuracy [3]. In Figure 3, the numbers below the line denote the tracking deltas corresponding to each level.

How stable is the tracking delta? One question with using tracking delta is whether the relative pressure change remains stable over time. We conducted an 8-hour long experiment on an elevator that traveled between 5 building levels and recorded the tracking deltas at each level. Figure 4 shows how the tracking delta for a given level varies about

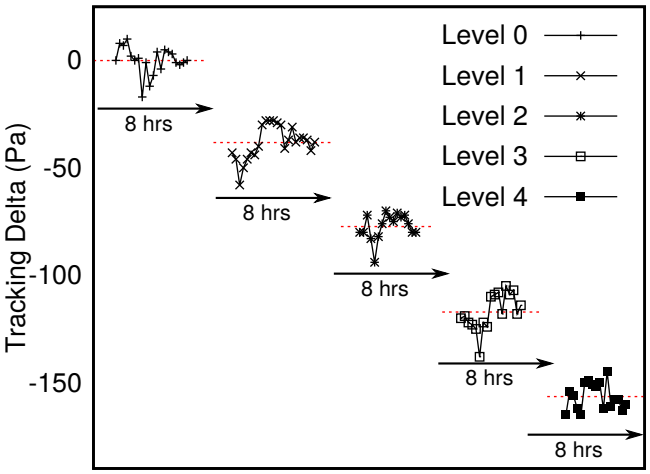


Figure 4. Tracking deltas at each level of a 5-level elevator over a period of 8 hours.

the expected value (dotted line). We make two observations. First, while there are variations in the pressure values measured when the elevator returns to the same level, these values form distinct clusters that are well separated. Second, the values do not trend always upward or always downward. While this measurement does not illustrate that the tracked level would always be correct, it shows that the system could track the correct level with high probability.

If the tracking delta is stable, then in principle, when sufficient data has been collected from all the levels, the data from each level will naturally cluster together and can be identified. The clusters would be well separated as long as the floor level separation is sufficiently higher than the relative accuracy of the barometer. The International Building Code [12] dictates the minimum floor level separation (ceiling height) to be 2.16 m ($\sim$ 26 Pa or 7 ft). Since the relative accuracy of today's commodity sensors is about $\pm$1 m, the

clustering of tracking deltas should work in principle as the clusters would be separated by at least 26 Pa. Such a clustering of tracking deltas could, however, take a significant amount of time if there are many levels in the building and if some levels are rarely visited. Next, we present how this process can be escalated under specific conditions.

3.2.2 Fast Level Tracking Algorithm

A faster level tracking algorithm can be designed based on the assumption that all floors have the same height. With this assumption, once we know the floor level separation in terms of pressure difference (the ***minimum delta***, which should be equal or larger than 26 Pa), we can determine what level the elevator is at by simply dividing the *tracking delta* at that level by the *minimum delta* of the building and rounding off to an integer:

$$curr\_level = roundINT\left(\frac{curr\_tracking\_delta}{minimum\_delta}\right) \quad (1)$$

Algorithm 1. Fast Level Tracking Algorithm

```
sampleWindow ← HeadDropFifoQueue(LEN)
curr_tracking_delta ← 0
curr_level ← 0
state ← NOT_MOVING
MOTION_THRESH ← 2
foreach sample ∈ BarometerSignal do
    sampleWindow.add(sample);
    mean_bmp, std = GetMeanStd(sampleWindow);
    if state == NOT_MOVING then
        curr_mean_bmp = mean_bmp;
        if std ≥ MOTION_THRESH then
            state = MOVING;
    else if state == MOVING then
        if std < MOTION_THRESH then
            state = NOT_MOVING;
            prev_mean_bmp = curr_mean_bmp;
            curr_mean_bmp = mean_bmp;
            journey_delta = curr_mean_bmp -
              prev_mean_bmp;
            min_delta = GetMinDelta(journey_delta);
            curr_tracking_delta += journey_delta;
            curr_level =
              round(curr_tracking_delta/min_delta);
end
```

For the elevator considered in Figure 3, the *minimum delta* is 38.2 Pa [7](explicitly measured). Thus, when the tracking delta is -81 Pa, the current level is -81 / 38 ≈ -2 i.e. two levels above[8] the reference level (level 0).

Algorithm 1, our fast level tracking algorithm, is based on the notion of *tracking delta* (§3.2.1). This algorithm is able to quickly estimate the floor levels reached including new unseen levels once the minimum tracking delta is found. The algorithm first uses a motion detection technique (details in §3.2.3) to detect that a journey has occurred. At the end of a journey, the pressure difference between the start and the end level is calculated and added to the tracking delta. The level tracking algorithm uses the method GetMinDelta to get the (best estimate of) minimum delta of the building (details in §3.2.4). Dividing the current tracking delta by the minimum delta and rounding off to an integer yields the current level relative to the reference level.

In the next two subsections, we describe the motion detection technique and the method to determine the minimum delta.

[7]Equivalent to a height change of roughly 3.18 m

[8]Negative values indicate an increase in level as pressure decreases with elevation.

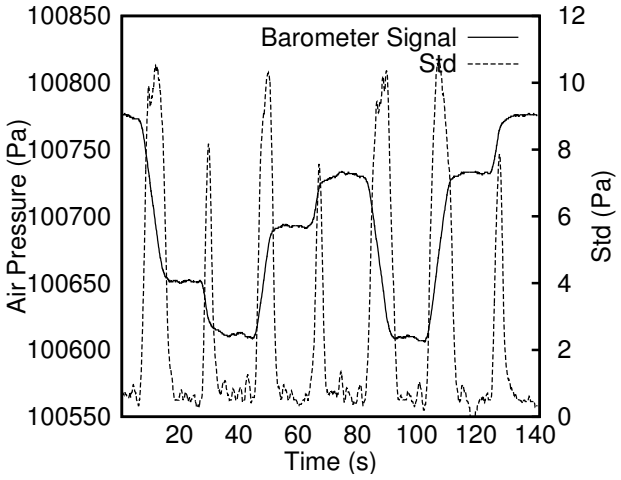


Figure 5. Motion Detection using standard deviation.

3.2.3 Motion Detection

From Figure 3, it is clear that to detect the start and end of the elevator's motion, we need to detect when the signal changes from being flat to being on a gradient and vice-versa. To detect these changes, we employ a simple standard deviation based edge detection algorithm. Figure 5, shows a barometer signal for a sequence of 7 journeys and the corresponding standard deviation calculated over a moving window of samples. It is clear that when the standard deviation increases above a threshold, the motion has started. When it falls below the same threshold, the motion has stopped. In principle, we expect the standard deviation to remain close to zero when the elevator is stationary. However, due to sensor noise, it occasionally reaches 1. Therefore, in practice, we found that a threshold of 2 was sufficient to avoid false positives.

3.2.4 Determining the minimum delta

Recall that the minimum delta is the ceiling height of the building in terms of the pressure difference. It is thus the pressure difference for a journey of length 1 level. A straight forward way of knowing the minimum delta would be to have a calibration phase where the elevator takes a 1 level journey and the system records the pressure difference as the minimum delta. However, we show that such a calibration can be avoided and the system can *learn* the minimum delta by itself. Procedure GetMinDelta summarizes our method to *learn* the minimum delta and is explained as follows.

Procedure GetMinDelta(journey_delta)

```
   Input: journey_delta
   Output: min_centroid
   Data: Cluster set: C
1  MIN_CLUSTER_DST ← 26;
2  jd_list = [abs(journey_delta - c.centroid) for c in C];
3  jd_list.add(journey_delta);
4  foreach jd in jd_list do
5      if len(C) == 0 then
6          C.add(new Cluster([jd]));
7      else
8          closest_cluster = getClosestCluster(jd);
9          min_distance = abs(closest_cluster.centroid -
             jd);
10         if min_distance < MIN_CLUSTER_DST then
11             closest_cluster.add([jd]);
12         else
13             C.add(new Cluster([jd]));
   end
14 min_centroid = min([c.centroid for c in C])
```

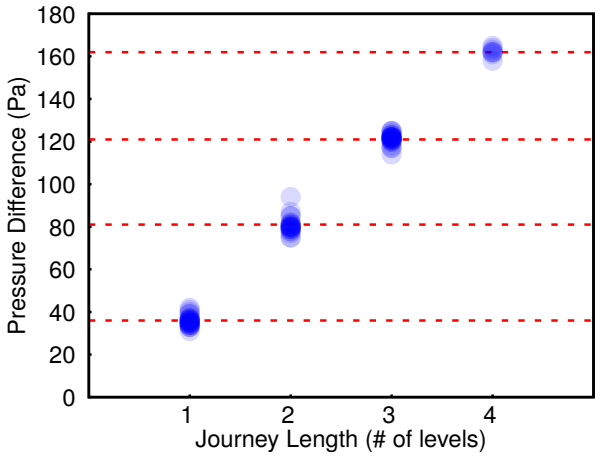


Figure 6. Journey pressure differences for different journey lengths form well-separated clusters. The dotted lines indicate the centroids.

Learning* the minimum delta.** In Figure 6, we plot the journey pressure differences (deltas) corresponding to 92 different journeys of varying lengths that an elevator took over a period of 8 hours. We can see that the journey pressure differences for different journey lengths form well-separated clusters. Based on this observation, our ***key idea is that if we keep clustering the journey pressure differences as journeys take place, then at any given moment, the minimum centroid among all the clusters is our best guess of the minimum delta. Therefore, the moment the elevator takes a journey of length 1 level, the minimum centroid among the clusters would be the required minimum delta.

Speeding up the learning. To speed up the learning process, for every new journey pressure delta to be clustered, we generate *additional* pressure deltas and cluster them. The additional pressure deltas are generated by taking the difference between the new to-be-clustered pressure delta and the existing cluster centroids. Line 2 in procedure GetMinDelta represents this step. The intuition behind this is that all the pressure deltas are integer multiples of the minimum delta with some noise. Therefore the differences between the pressure deltas are also valid pressure deltas for clustering. Thus, when the elevator takes a new journey whose length differs by 1 level from any of the prior journeys, then generating additional pressure deltas would yield the pressure delta of 1 level which is the required minimum delta.

3.2.5 Back correcting initial journeys

To learn an accurate minimum delta (§3.2.4) it takes at least a few journeys. During this time, the current level which is given by equation 1 would be incorrect as the minimum delta has not been correctly determined. However, when the elevator reaches the reference level (level 0), the current tracking delta would be close to zero, and thus the current level would be correctly identified as 0 (due to integer rounding), even if the minimum delta is incorrect. This is indeed not a bad thing. Being at level 0, the monitoring device can communicate the journey data collected so far to the gateway device even if the journey data has an incorrect minimum delta. Since the minimum delta is eventually learned, the gateway device can go back and correct the level information for the initial journeys[9].

In summary, EleTrack's level tracker does not waste ***any*** journey information and accurately tracks the levels, including for the journeys in the initial part of the learning/self-calibration process.

3.3 Door Detector

While the level tracker helps detect the reference level at which the gateway is installed, this information is not sufficient to identify the communication opportunities. Due to the Faraday Cage Effect, the monitoring device can communicate reliably only when the door is *open* at the gateway level. The radio should be turned on only during such communication opportunities to conserve energy.

To identify the communication opportunity, simple heuristics based on elevator motion and time may not work for the following reasons. First, the door may not open every time the elevator stops. For instance, elevators often move to a default level, come to a stop but do not open the door. In such cases, the radio may turn on even when there is no communication opportunity. Also, when the elevator door opens, the duration before it closes back is not deterministic. It depends both on the specific elevator as well as the movement of people in and out of the elevator cabin. Overall, estimating the communication opportunity based on elevator motion and hard-coded door timings can be unreliable.

In EleTrack, we make novel use of the magnetometer to develop a more generic door open/close detector. The idea of using magnetometer is based on the fact that when the

[9]The journey data consists of tracking delta values for the *from* and the *to* levels and the minimum delta. This allows EleTrack to use the correct minimum delta at a later point of time to compute correct levels using equation 1.

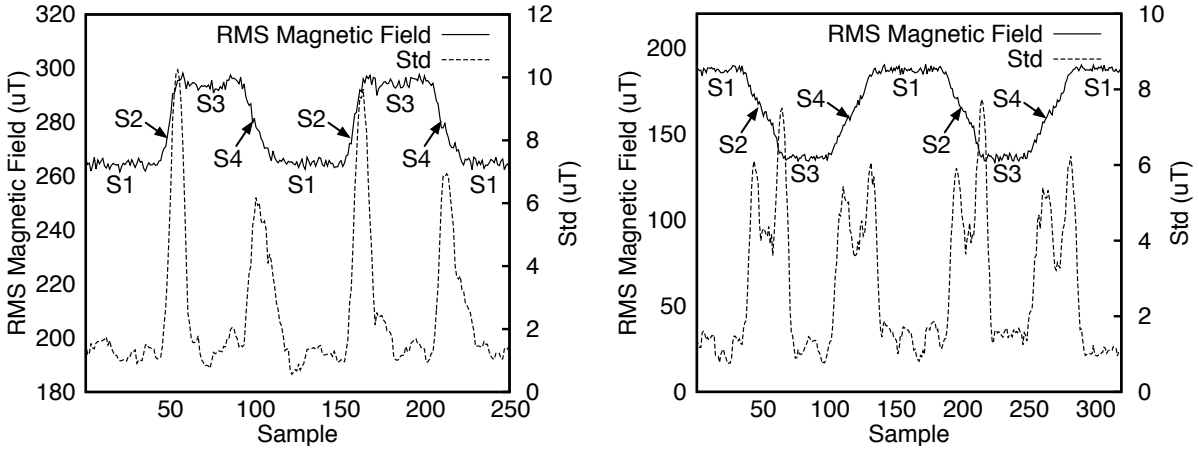


Figure 7. Root mean square of magnetic field and the corresponding standard deviations shown along with door states.

elevator door opens or closes, the strength of the magnetic field near the door changes (due to metal movement). As a result, door events (movements) generate distinct patterns in magnetic field strength across the different axes of a magnetometer. To make the detector orientation independent for calibration-free deployment, we consider the root mean square (RMS) values across all the three magnetometer axes. Figure 7 shows the changes in the RMS magnetic field when there are door open/close events for two different elevators. It also shows the corresponding standard deviation of the RMS signal. The different door states S1, S2, S3, and S4 are door closed, door opening, door open, and door closing respectively. What is interesting to note is that even though the magnetic field strength can either rise (as shown in Figure 7a) or drop (as shown in Figure 7b) due to the door opening, there is always a ***change*** in the magnetic field strength whenever there is door movement. A simple standard deviation based edge-detection algorithm is thus able to identify if the door's current state (open or close) is being changed to the other.

3.4 Data Communicator

Traditional per-packet acknowledgment-based schemes involve a lot of overhead especially when the link is interfered. There is a likelihood of not just data packets, but also acknowledgments getting lost. This leads to MAC layer retransmissions even though the actual data might have been delivered. Block acknowledgments or B-ACKs were introduced in IEEE 802.11e to improve the MAC layer efficiency. Instead of every packet being acknowledged, an acknowledgment is sent for every block of packets. The B-ACK notifies the transmitter about all the packets that couldn't go through which are subsequently retransmitted. The process repeats until all the packets are delivered. While the use of B-ACKs reduces the ACK transmissions in a non-interfered setting, the number of B-ACKs can still be significant if the channel is severely interfered [22]. Both the above schemes involving per-packet ACKs or B-ACKs significantly reduce the achievable throughput and become unsuitable when the elevator sensor data needs to be transmitted to the gateway in a very small amount of time.

In this section, we propose a transmission scheme that uses streaming network coding with B-ACKs for 802.15.4. The design choice to use 802.15.4 for communication was to achieve a sweet spot between throughput and power consumption (LoRaWan provides extremely low throughput while 802.11 is power hungry.). Our scheme involves two components:

1. **Sender**: Runs on the monitoring device. Performs network coding on blocks of data and transmits a stream of encoded codewords until a B-ACK is received.
2. **Receiver**: Runs on the gateway device. Accumulates just enough codewords from the incoming stream to decode the transmitted block of data and responds with a B-ACK.

3.4.1 Sender

Fountain codes have been used for high throughput low-power wireless communication [4, 16]. The sender implements Luby Transform codes [20] (a variant of Fountain codes [21]) as it has relatively simple encoding and decoding algorithms that need to run on the resource constraint monitoring device.

Encoding. Assume that the monitoring device has buffered data for N journeys j_1,...,j_N before it detects a communication opportunity. The device then starts encoding packets in blocks of K journeys and generates an infinite stream of codewords x_1,x_2,... which are then transmitted. The encoding operation involves standard XOR operation of d journey's data where d is drawn randomly from a degree distribution. The transmission of codewords for the current block of journeys stops on reception of a B-ACK from the gateway. This triggers transmission of codewords encoded from the next block of K journeys until no more journeys are left.

Degree. Selecting a proper degree distribution is crucial for fast decoding of the journeys. Various distributions like the standard Soliton or the improved Robust Soliton have

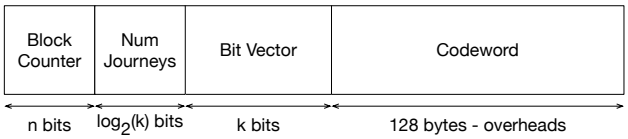


Figure 8. EleTrack packet structure.

been proposed in the past. However, EleTrack uses *Growth Codes* [15] as it enables faster decoding with a small number of codewords. By using *Growth Codes*, the device transmits codewords of increasing degree instead of choosing the degree randomly from a distribution. As the gateway decodes more data of the current block, higher degree codewords are transmitted to speed up the decoding process.

3.4.2 Receiver

The receiver executes an online *On-the-fly Gaussian Elimination* [2] decoding algorithm. The decoding happens per block of journeys and a B-ACK is transmitted after success to notify the sender to shift its window to the next block. To facilitate the decoding of individual journey's data, each packet includes (1) the window/block counter, (2) a bit vector to identify which journeys of the block form the codeword and (3) the actual codeword. Since the last window may contain journeys less than K, each transmitted packet also contains the maximum number of the journeys in the current block as shown in Figure 8. This allows the gateway to acknowledge quickly if fewer journeys are expected in the last block.

3.5 Sensor Assisted Duty Cycling

With all the components (§3.2, §3.3, §3.4) put together, EleTrack knows if the elevator is moving or stationary, which floor it is at and whether the door is open or close. This enables EleTrack to achieve sensor-assisted aggressive duty cycling by using low-power components to actuate higher power components only when required. The low-power barometer triggers the moderate-power magnetometer for door detection only when the elevator has stopped at the gateway level. The moderate-power magnetometer, in turn, triggers the high-power radio only when the door opens. Further, the high throughput reliable transmission scheme makes data transmission fast with reduced radio-on time. Such a prudent use of sensors and radio is what makes EleTrack ultra-low-power. Moreover, the level tracking and the door detection being zero-input and zero-calibration algorithms, it makes EleTrack readily deployable.

4 System Implementation

We have implemented the monitoring device using Texas Instrument's SensorTag[10]. The CC2650STK version of SensorTag is battery powered and provides a host of sensors including barometer and magnetometer along with the radio, all in an extremely small form factor as shown in Figure 9. The gateway is composed of another CC2650STK connected serially to a RaspberryPi[11] that logs all the data. This can in turn be connected to the Internet to push data to the cloud for real-time remote monitoring.

[10] http://www.ti.com/tool/cc2650stk

[11] https://www.raspberrypi.org/

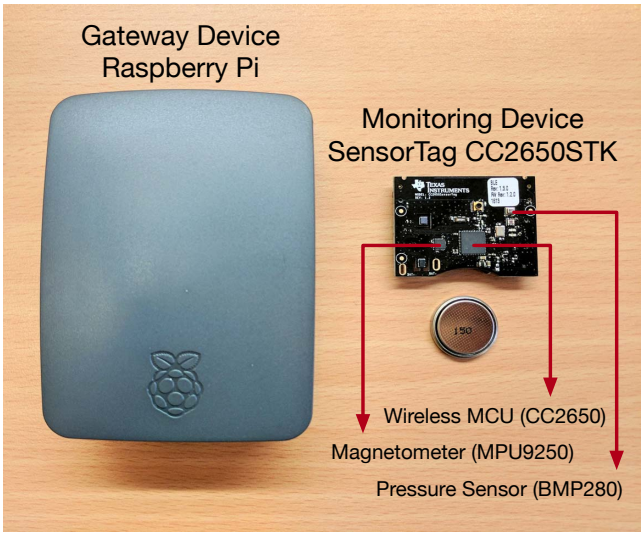


Figure 9. Texas Instrument's SensorTag and Raspberry Pi used to implement the monitoring device and the gateway.

Table 1. Level Tracking Accuracy.

	Ele A	Ele B	Ele B*	Ele C	Ele D
Max Elevator Floors	3	5	5	7	25
Total Journeys	24	25	92	300	45
Accuracy after learning minΔ	100%	100%	100%	100%	97.8%
Accuracy of back correction	100%	100%	100%	100%	100%

* trace-driven

EleTrack uses 5 Hz sampling frequency for both the barometer and the magnetometer. CC2650STK has been programmed using Contiki [6] and without the default ContikiMAC [5] radio duty-cycling layer. The duty-cycling of the radio is governed by the magnetometer which in turn is duty-cycled by the barometer.

5 Evaluation

In this section, we evaluate the different components of EleTrack.

5.1 Level Tracking

When it comes to the maintenance of an elevator, tracking the usage is important. The number of journeys that the elevator has covered at any point in time, what are the most common journeys, which building levels are most frequently visited, etc. are some information that can reveal how intensively an elevator has been used over the past. This section evaluates the accuracy of EleTrack to track the journeys.

5.1.1 Accuracy

We evaluated EleTrack's level tracking component by deploying it on four elevators in buildings having different levels. Table 1 summarizes the results. After learning the minimum delta, for elevators A, B, and C, EleTrack was able to correctly track all the levels while for elevator D it was incorrect for only one stop (level) out of the total 46 stops. On investigating, we found that this was because the sensor's

(a) Sliding Telescopic Door **(b) Center Opening Door**

Figure 10. Two popular elevator doors found in buildings.

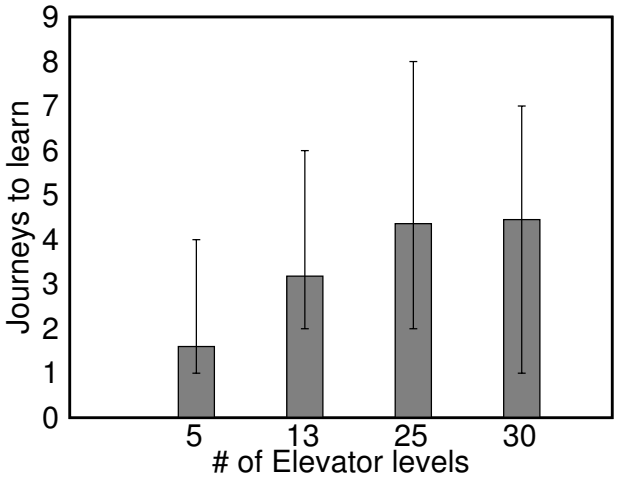


Figure 11. Number of journeys required to learn the Δ_{min}.

noise had momentarily added up in the tracking delta too far in one direction. But eventually, it did revert to the mean. In §8, we discuss how such an occasional noise build-up can be mitigated.

5.1.2 Learning the minimum delta

Recall that our clustering-based method would *learn* the minimum delta (§3.2.4) once the sequence of journeys taken by the elevator contains two journeys with lengths differing by 1. In practice, the total number of journeys it would take to get two such journeys would depend on the elevator usage pattern. Figure 11 shows the minimum, average and maximum number of journeys it took for four different elevators to learn the minimum delta. The values are computed over a set 10 arbitrary journey *sequences* that the elevator users undertook on a typical day. We see that in general, it takes less than 10 journeys to learn the minimum delta with real-world elevator usage patterns.

5.2 Door Detection

This section evaluates the accuracy of using a magnetometer for detecting door status change on elevators for different door types. We placed EleTrack's monitoring device inside different elevators having two types of doors commonly found in modern elevators: center opening (Figure 10b) and sliding telescopic (Figure 10a). Figure 12 plots the RMS magnetic field observed for two elevator doors of each type. We observe a clear pattern of change in magnetic field resulting in ***accurate*** door event detection. We evaluated on 4 more different elevator doors and the door detection accuracy was 100%.

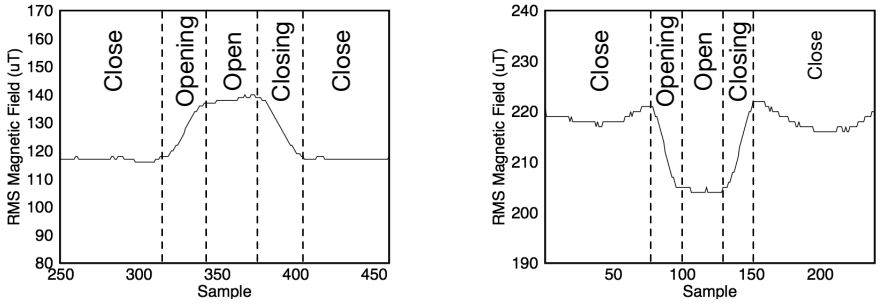
5.3 Energy Consumption

In this section, we evaluate EleTrack's standby and transmission energy consumption. We use Monsoon power meter [14] to measure the current drawn by the monitoring device.

5.3.1 Standby Energy Consumption

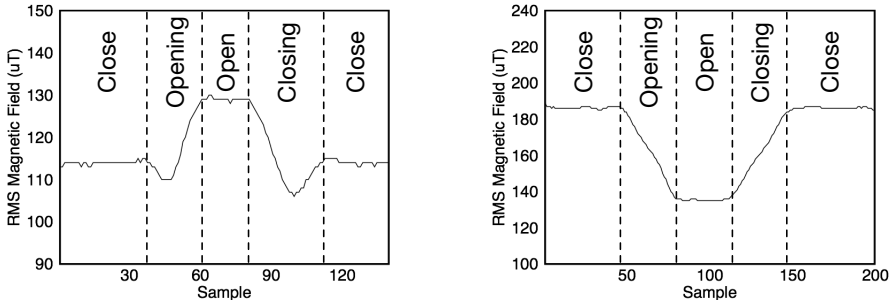
We evaluated EleTrack's standby energy savings achieved through sensor-assisted duty cycling in a real world scenario and compare it to the state-of-the-art ContikiMAC duty-cycling. In ContikiMAC, without information from the sensors, the radio wakes up every 125ms (8 Hz) to sample the channel to detect from transmission from the gateway. In the comparison, we consider only the energy needed to trigger data transmission to the gateway, without data transmission. Data transmission energy is considered in the next section (Section 5.3.2).

Note that EleTrack's standby energy consumption is not deterministic and depends on the elevator's usage pattern. Therefore, our evaluation presented in Table 2 is based on the energy consumption recorded during a an 8-hour long operation of a 5-level elevator on a typical weekday. During these 8 hours, the elevator stopped at the gateway level 20 times and the magnetometer sampling (@ 5 Hz) was turned on (using the scheme described in §3.3) for a total of about 80 seconds (0.3% of the total time). Other than the magnetometer, EleTrack also had the barometer sampling (@ 5 Hz) all the time. For the above experiment, EleTrack drew an average current of 40.76 μA leading to a stand-by time of 2.8 years on a 1 Ah coin cell battery. This is a 3.19 times improvement over the traditional duty cycling approach.



(a1) Elevator 1 (a2) Elevator 2

(a) Sliding Telescopic Door



(b1) Elevator 3 (b2) Elevator 4

(b) Center Opening Door

Figure 12. Magnetometer signatures of door opening and closing events for different elevator door types.

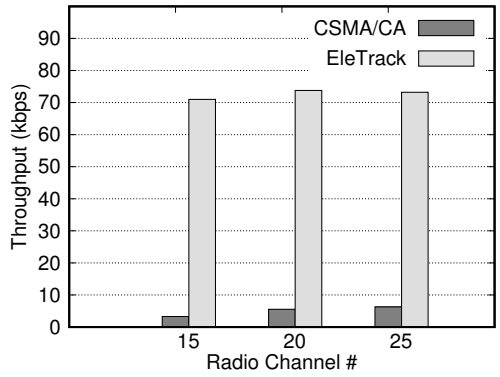


Figure 13. Throughput comparison for EleTrack and ContikiMAC during transmission mode.

Note that an elevator monitoring application would typically keep track of the level. Hence, the level tracking using barometer is a part of the application itself. In such a case, by excluding the energy consumed by the barometer sensor, EleTrack consumes only 35.76 μA on the average. This gives up to 3.64 times improvement compared to traditional duty cycling approaches.

5.3.2 Transmission energy consumption

In addition to the sensor-assisted duty-cycling, EleTrack's reliable high throughput protocol also helps reduce the "radio on" time required for transmitting a unit of data. We selected three 802.15.4 channels to transmit information from the monitoring device to the gateway using EleTrack and compared the throughput with typical CSMA/CA-based scheme. Figure 13 illustrates the results from a real-world experiment where we transmit 32 KB of data for each setting. As seen, EleTrack provides up to 22 times improvement in achievable throughput in comparison to typical CSMA/CA MAC. This directly translates into energy savings due to low radio-on time required for transmitting the same amount of data.

6 Sample Monitoring Applications

In this section, we share some of the abnormalities that we found while experimenting with elevators retrofitted with EleTrack.

6.1 Rough Journeys

Elevator rides are supposed to be quiet and smooth. However, this may not always be the case, especially for old elevators. It is thus crucial to monitor the smoothness of the journeys in order to schedule maintenance. We performed an 8 hour experiment to identify the smoothness of journeys on an old elevator. Figure 14 illustrates the z-axis of the accelerometer on CC2650STK deployed inside the elevator. Initially, the elevator rides were smooth with well defined accelerations. However, 4 hours into the operation, the acceleration readings recorded significant signal spikes indicating rough rides.

Table 2. Power consumption comparison based on 8 hrs of real world usage on a working day.

		ContikiMAC		EleTrack		EleTrack (free barometer)	
	Avg Current (uA)	Active Time Fraction	Actual Current (uA)	Active Time Fraction	Actual Current (uA)	Active Time Fraction	Actual Current (uA)
MCU	35	100%	35	100%	35	100%	35
Barometer @ 5 Hz	5	0%	0	100%	5	0%	0
Magnetometer @ 5 Hz	255	0%	0	0.3%	0.76	0.3%	0.76
Radio @ 8 Hz	95	100%	95	0%	0	0%	0
Consumption			130		40.76 (3.19 times lower)		35.76 (3.64 times lower)

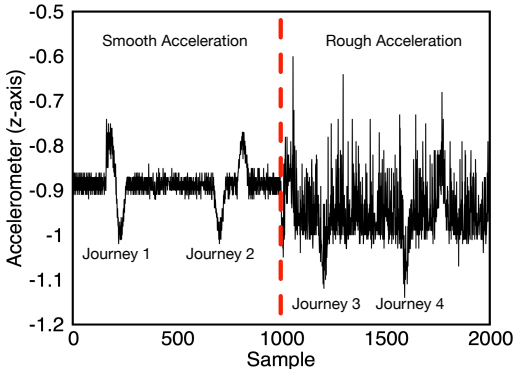


Figure 14. Accelerometer readings for the same elevator for 2 smooth and 2 rough journeys 4 hours apart.

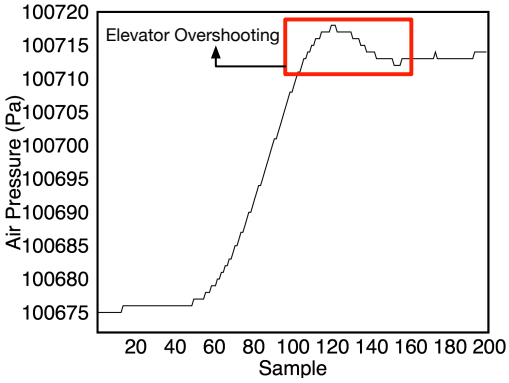


Figure 15. Elevator overshoots the destination level due to imprecise acceleration.

6.2 Overshooting Rides

Improper accelerations not only cause rough rides but can also lead to elevators not being able to stop correctly at the destination level. For instance, as shown in Figure 15 one of the elevators in our sample set always overshot the destination level while coming down. After overshooting, it took some time to come back and open the door thus causing additional delays.

7 Related Work

In this section, we cover some of the commercial elevator monitoring solutions and few interesting works related to the three EleTrack functional components.

7.1 Elevator monitoring

We have seen some dramatic shift in the elevator industry with "smart" features built into the new elevators. MelEye™ from Mitsubishi[12], E-Link™ from Kone[13], REM®5.0 from Otis[14] are some of the remote monitoring solutions from popular elevator manufacturers. They offer a suite of sensors for round the clock operational status and fault monitoring of elevators through a web dashboard. Unfortunately, a significant proportion of existing elevators are old and do not have such sophisticated monitoring features. EleTrack is a *plug-n-play* solution that retrofits such old elevators with ultra-low-power wireless monitoring solution.

[12]http://www.mitsubishielectric.com/elevator/
[13]http://www.kone.com
[14]http://www.otisworldwide.com

7.2 Level tracking

Building level tracking has been explored in the context of indoor localization. There have been accelerometer-based approaches [27] which are known to accumulate errors and degrade over time [19]. Moreover, they consume much more power compared to using barometer [23]. Among the barometer-based solutions, Muralidharan et al. [23] use training data to build a predictor for the number of floors changed, given a journey's pressure difference. Ichikari et al. [13] use inputs from other localization infrastructure to extract the altitude-dependent component from a pressure reading. B-Loc [26] uses a barometer fingerprint map that is built using crowdsourcing. These barometer-based methods involve calibration, and also need computational resources that are not available on ultra-low-power devices such as the SensorTag. In contrast, our relatively simple level tracking algorithm requires zero calibration, works in an online fashion and is computationally lightweight.

7.3 Door detection

Today's commercial monitoring solutions come with a cabled network that is laid during the installation of elevators. Since wireless solutions work only when the door is open, door detection is critical. Wu et al. [25] use pressure differences to detect door open/close events in insulated buildings for intrusion detection. Since elevators are not perfectly insulated, no noticable pressure differences are observed when door opens or closes. EleTrack, on the other hand, uses magnetometer to exploit the changes in the magnetic field in-

duced by moving metal.

7.4 Energy-efficient Communication

Significant effort has been put in improving low-power sensor network communication. There are energy-efficient radio duty-cycling MAC protocols like A-MAC [9], RI-MAC [24], ContikiMAC [5], etc. For reliability, numerous approaches have been exploited to achieve optimal routing [10, 8, 17] and to mitigate the impact of cross-technology interference [11, 22]. In our setting, standard duty-cycling approaches waste a significant amount of energy by periodically turning on the radio to check for communication opportunities. EleTrack on the other hand performs ultra-low-power sensor-assisted duty-cycling to conserve energy.

8 Discussion

Even though the relative accuracy of the barometer is fairly good, error may still add up in one direction beyond the tolerable threshold like what happened with elevator C in §5.1.1. One approach to resolve this problem would be to avoid directly adding a journey's pressure difference to the tracking delta. Instead, the journey's pressure difference could be clustered first (which is already being done for learning the minimum delta) and the centroid of that cluster be added to the tracking delta. The rationale behind this is that if errors of the pressure difference measurements are zero mean, their clustered centroid is likely to have the least error.

Nevertheless, error in level detection can still occur. Such errors can be detected by observing the signal strengths (RSSI) of communication between the monitoring device and the gateway. For instance, if the elevator stops at one level above or below the gateway level and tries to initiate communication and if it happens to hear the gateway response, the RSSI would be very weak. This indirectly tells the system that it has lost track of the levels. For realignment, the device needs to wakeup more often (say one level below and above the *perceived* gateway level) to find the correct level before reverting back to the sensor-assisted duty cycling.

9 Conclusion

We have presented EleTrack, a *plug-n-play* retrofit for remote monitoring of elevators that can operate for over a year on a coin cell battery. The system has been fully implemented using a TI SensorTag as the monitoring device and a RaspberryPi as the gateway. Through real-world experimentation on different types of elevators, we have demonstrated EleTrack's ability to track the elevator's state with close to 100% accuracy. Moreover, EleTrack's novel sensor-assisted duty-cycling provides an order of magnitude lower power consumption than the traditional duty-cycling techniques. Since EleTrack needs no calibration, it is a practical and promising approach to retrofit existing elevators and is currently being deployed commercially. In future, we envision such sensor-assisted duty-cycling approach to be exploited in other applications where possible.

10 Acknowledgements

This work is partially supported by the Singapore Ministry of Education Academic Research Fund Tier 1 (R-252-000-621-114).

11 References

[1] K. Al-Kodmany. Tall Buildings and Elevators: A Review of Recent Technological Advances. *Buildings*, 2015.

[2] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno. On the fly Gaussian elimination for LT codes. *IEEE Communications Letters*, 2009.

[3] Bosch. BMP280 datasheet, 2017. `https://www.bosch-sensortec.com/bst/products/all_products/bmp280`.

[4] W. Du, J. C. Liando, H. Zhang, and M. Li. When pipelines meet fountain: Fast data dissemination in wireless sensor networks. In *Proceedings of SenSys*, 2015.

[5] A. Dunkels. The ContikiMAC radio duty cycling protocol. 2011.

[6] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of LCN*, 2004.

[7] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. Orchestra: Robust mesh networks through autonomously scheduled TSCH. In *Proceedings of SenSys*, 2015.

[8] S. Duquennoy, O. Landsiedel, and T. Voigt. Let the tree bloom: Scalable opportunistic routing with ORPL. In *Proceedings of SenSys*, 2013.

[9] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *Proceedings of SenSys*, 2010.

[10] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of SenSys*, 2009.

[11] A. Hithnawi, H. Shafagh, and S. Duquennoy. TIIM: technology-independent interference mitigation for low-power wireless networks. In *Proceedings of IPSN*, 2015.

[12] I. IBC. International building code. *International Code Council, Inc.*, 2006.

[13] R. Ichikari, L. C. M. Ruiz, M. Kourogi, T. Kurata, T. Kitagawa, and S. Yoshii. Indoor floor-level detection by collectively decomposing factors of atmospheric pressure. In *Proceedings of IPIN*, 2015.

[14] M. S. Inc. FTA22J Power Meter, 2017. `http://msoon.github.io/powermonitor/`.

[15] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: Maximizing sensor network data persistence. In *Proceedings of SIGCOMM*, 2006.

[16] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in the air: Practical wireless network coding. In *Proceedings of SIGCOMM*, 2006.

[17] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low power, low delay: opportunistic routing meets duty cycling. In *Proceedings of IPSN*, 2012.

[18] F. Le Blancq. Diurnal pressure variation: the atmospheric tide. *Weather*, 2011.

[19] X. Lin, X.-W. Chang, and X. Liu. LocMe: Human locomotion and map exploitation based indoor localization.

[20] M. Luby. LT codes. In *Proceedings of FOCS*, 2002.

[21] D. J. MacKay. Fountain codes. *IET Proceedings-Communications*, 2005.

[22] M. Mohammad, X. Guo, and M. C. Chan. Oppcast: Exploiting spatial and channel diversity for robust data collection in urban environments. In *Proceedings of IPSN*, 2016.

[23] K. Muralidharan, A. J. Khan, A. Misra, R. K. Balan, and S. Agarwal. Barometric phone sensors: More hype than hope! In *Proceedings of HotMobile*, 2014.

[24] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of SenSys*, 2008.

[25] M. Wu, P. H. Pathak, and P. Mohapatra. Monitoring building door events using barometer sensor in smartphones. In *Proceedings of UbiComp*, 2015.

[26] H. Ye, T. Gu, X. Tao, and J. Lu. B-loc: Scalable floor localization using barometer on smartphone. In *Proceedings of MASS*, 2014.

[27] H. Ye, T. Gu, X. Zhu, J. Xu, X. Tao, J. Lu, and N. Jin. FTrack: Infrastructure-free floor localization via mobile phone sensing. In *Proceedings of PerCom*, 2012.