Adaptive Sensing Using Internet-of-Things with Constrained Communications

Mahmudur Rahman
Bangladesh University of Engineering and Technology, Bangladesh

Amatur Rahman
Bangladesh University of Engineering and Technology, Bangladesh

Afia Afrin
Bangladesh University of Engineering and Technology, Bangladesh

Hua-Jun Hong
National Tsing Hua University, Taiwan

Pei-Hsuan Tsai
National Tsing Hua University, Taiwan

Md Yusuf Sarwar Uddin
University of California, Irvine, USA

Nalini Venkatasubramanian
University of California, Irvine, USA

Cheng-Hsin Hsu
National Tsing Hua University, Taiwan

ABSTRACT

In this paper, we design and implement an Internet-of-Things (IoT) based platform for developing cities using environmental sensing as driving application with a set of air quality sensors that periodically upload sensor data to the cloud. Ubiquitous and free WiFi access is unavailable in most developing cities; IoT deployments must leverage 3G cellular connections that are expensive and metered. In order to best utilize the limited 3G data plan, we envision two adaptation strategies to drive sensing and sensemaking. The first technique is an *infrastructure-level* adaptation approach where we adjust sensing intervals of periodic sensors so that the data volume remains bounded within the plan. The second approach is at the *information-level* where application-specific analytics are deployed on board devices (or the edge) through container technologies (Docker and Kubernetes); the use case focuses on multimedia sensors that process captured raw information to lower volume semantic data that is communicated. This approach is implemented through the EnviroSCALE (Environmental Sensing and Community Alert Network) platform, an inexpensive Raspberry Pi based environmental sensing system that periodically publishes sensor data over a 3G connection with a limited data plan. We outline our deployment experience of EnviroSCALE in Dhaka city, the capital of Bangladesh. For information-level adaptation, we enhanced EnviroSCALE with Docker containers with rich media analytics, along Kubernetes for provisioning IoT devices and deploying the Docker images. To limit data communication overhead, the Docker images are preloaded in the board but a small footprint of analytic code is transferred whenever required. Our experiment results demonstrate the practicality of adaptive sensing and triggering rich sensing analytics via user-specified criteria, even over constrained data connections.

ARM'17, December 11–15, 2017, Las Vegas, NV, USA

ACM ISBN 978-1-4503-5168-3/17/12…$15.00
https://doi.org/10.1145/3152881.3152887

ACM Reference Format:
Mahmudur Rahman, Amatur Rahman, Afia Afrin, Hua-Jun Hong, Pei-Hsuan Tsai, Md Yusuf Sarwar Uddin, Nalini Venkatasubramanian, and Cheng-Hsin Hsu. 2017. Adaptive Sensing Using Internet-of-Things with Constrained Communications. In *ARM'17: ARM'17: Adaptive and Reflective Middleware, December 11–15, 2017, Las Vegas, NV, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3152881.3152887

1 MOTIVATION

The emerging trend to incorporate Internet of Things (IoT) based smart devices and sensors to instrument communities and cities around the world has enhanced environmental awareness in urban regions. New smart city applications aim to monitor environmental conditions to ensure health and safety of the public at large. For example, effectively monitoring and managing air quality is an age-old problem that is becoming more important with increasing urbanization. In one study, WHO reported that global urban air pollution is increased by 8% during the period 2008-2013. Rise in pollution levels has catastrophic consequences to the environment - awareness of air quality can help individuals take measures to improve health awareness.

In practice, however, the ability to gather environmental information in cities and communities at finer levels of temporal and spatial granularity is limited because of high deployment costs and maintenance overheads. The issue is further aggravated in developing nations such as Bangladesh, where it is not always possible to deploy permanent sensors due to lack of reliable, cheap and ubiquitous WiFi connectivity nationwide. The presence of low-cost sensing (insitu and mobile) with improved connectivity options are required to make such services accessible to a larger community who can subscribe to receive the associated information. While some countries in South Asia (e.g., Bangladesh) are ahead in terms of 3G cellular provisioning [10], exploiting 3G connectivity for continuous and autonomous real time sensing with IoT platforms can be prohibitively expensive.

In this paper, we propose to develop inexpensive air quality monitoring solutions to accurately gather and communicate information to cloud platforms where the information can be further analyzed and acted upon. Due to the dynamic nature of the environment and phenomena being monitored (e.g., toxic plumes in a fire), the proposed schemes must be inherently adaptive. We have

developed a combined hardware/software platform that includes off-the-shelf sensors and compute platforms (Raspberry Pi), along with a modular software stack to collect and upload data to the cloud. In particular, our work is an extension of an IoT based sensing platform SCALE (currently used for public safety applications), developed at UC Irvine [4]. We construct an air quality measuring box, EnviroSCALE, with a suitable set of sensors for collecting environmental sensing data and uploading them over a cellular connection. As articulated earlier, continuous operation of the EnviroSCALE platform is expensive due to the limited data plan. One of the core challenges in EnviroSCALE is to support meaningful sampling of sensor data for the application at hand while ensuring that the data budget of 3G cellular connections are not exceeded. This will require adaptation of how sensing, collection, and communication is performed (*infrastructure-level adaptation*). We explore strategies to downsample (reduce sampling rate) adaptively based on observed sensing patterns so as to reduce communication load. We carefully formulate, solve, and evaluate this adaptation problem to reduce the data volume over the 3G cellular connections.

Another adaptation strategy is to enable simple IoT analytics on board and send lower-volume processed information over the limited communication networks (*information-level adaptation*). For this, we employ container-based middleware to support a more flexible deployment of the EnviroSCALE IoT platform, where analytics can be deployed in a plug-and-play manner based on the type of information processing needed.

With the ability to process sensor data on-board, we can now deploy richer sensing modalities such as video and audio sensing. Our work attempts to add intelligence in the IoT platform to automatically activate a range of multimedia sensors, such as microphones and cameras, for a more comprehensive analysis. We note that these multimedia sensors generate a large volume of data that can overwhelm the constrained data connections. Our strategy is to adaptively deploy media analytics on the devices colocated or close to the media sensors. Containers, fortunately, allow us to package the rich sensing analytics, along with any associated software dependencies, for easy deployment. The container based approach enables multiple usage scenarios to be configured and reconfigured; we list some possibilities below:

- Air quality sensors may report high PM 2.5 pollutants, and activate cameras and rich sensing analytics to identify the pollution sources (such as garbage, construction sites, and others).
- Microphone arrays may report gunshots, and activate cameras for collecting evidence and locate the suspects.
- Gas sensors may detect the existence of toxic or flammable gas, and activate cameras to identify and alert the persons in the affected zone.

In this paper, we build a prototype system to explore the need for adaptation and demonstrate the practicality of our container-based approach for IoT deployments.

2 RELATED WORK

The Safe Community Awareness and Alerting Network (SCALE) [4] platform is an IoT system created in the context of the *SmartAmerica Challenge* effort to aid public and personal safety using inexpensive off-the-shelf IoT platforms and devices. SCALE supported data exchange between IoT content producers and consumers using a Pub/Sub architecture through the use of protocols such as MQTT. SCALE2 enhanced the basic SCALE platform with resilience capabilities including multiple network [14] capabilities. Similar smart community efforts around the world have gained popularity in recent years. For example, the Padova Smart City [16] implements techniques to collect temperature, humidity and light sensitivity data and store them in an HTTP-CoAP proxy server. AQBox [7], an earlier version of EnviroSCALE, illustrates the promise of off-the-shelf sensors for air quality monitoring. U-Sense [5] is a low-cost sensing system using sensors and wireless access point to upload data. Our earlier work [8, 13] advocates container based smart city platforms, that leverage centralized controllers for resource allocation. In the current work, we demonstrate the utility of automated rich sensing and associated analytics triggered by sensor readings.

Several recent efforts focus on the use of IoT platforms for air quality sensing and monitoring. A basic architecture and process for distributed air quality monitoring is discussed in [15]. Air Quality Egg [6] is a community driven sensing network hosted on Xively IoT platform that is similar to our hosting process. Among the commercially available air monitoring devices, AirBeam [1] is a wearable mapping, graphing, and crowdsourcing platform, and Awair [2] focuses on indoor air quality measurement. The key innovation of our work lies in designing a methodology that system that considers data plan constraints in the communication channels, focusing specifically on 3G cellular networks.

The other novel aspect of our work lies in the ability to integrate modular methods for dynamic adaptation - this is essential to support resource provisioning for community IoT systems, where there are frequent changes in the sensing and communication contexts. Dynamic application deployment on IoT devices utilizing lightweight virtualization methods are gaining popularity [3, 9, 11, 12]. Techniques we will explore to enhance QoS and performance in such settings include support for layered virtualization images for reduced network overhead and event-driven application deployment.

3 ENVIROSCALE: A FLEXIBLE ARCHITECTURE FOR AIR QUALITY AWARENESS

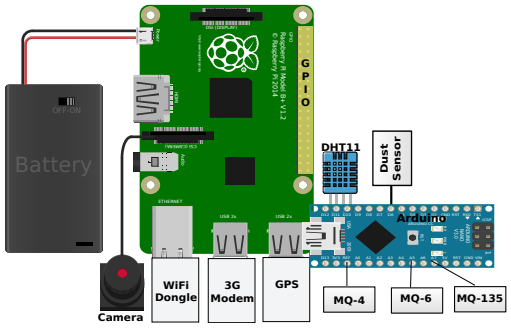
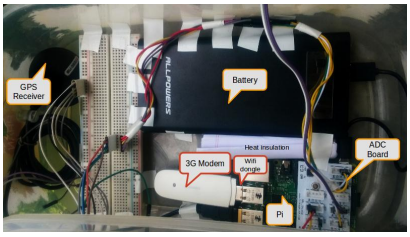


Figure 1: Hardware schematic of EnviroSCALE.

(a) Outside



(b) Inside

Figure 2: Photos of our EnviroSCALE box.

In this section, we describe the design and implementation of the hardware and software components of the EnviroSCALE prototype.

3.1 Hardware Setup

We give the schematic of our prototype system in Fig. 1. The basic platform is designed using a Raspberry Pi for control and sensing tasks and an Arduino Nano for additional sensing tasks. The MQ series gas sensors are connected to the analog input pins of Arduino Nano. The platform uses three gas sensors: MQ-4 for detecting combustible gases, MQ-6 for butane and LPG, and MQ-135 for measuring air quality (sensitive to benzene, alcohol, smoke, ammonia, sulfide, etc.). Two digital sensors: DHT11 temperature-humidity sensor and the Sharp GP2Y1010AU0F dust sensor are interfaced through the digital input pins of Arduino Nano. The platform also has a GPS receiver (GlobalSat BU-353S4), an HSPA + 3G cellular model (Huawei E303) and a WiFi dongle (D-Link), all connected via the USB port. The RPi runs on an 8GB microSD card loaded with the standard Linux images and the SCALE software stack. The whole setup, when deployed outdoors, is powered by an external battery of 19200 mAh capacity. Fig. 2 is an image of our the initial EnviroSCALE prototype.

3.2 Software Stack

The key components of the EnviroSCALE software stack include a sensing and computation module, an upload module and a data publishing module (MQTT broker). A key adaptation feature in the prototype software stack is the adjustment of data capture parameters. During operation, the sensing and computation module adjusts the sampling intervals of sensors to ensure that the platform operates under the constrained data connection without loss of significant sensor information. The module reads the connected sensors, converts the sensor data into a valid representation, and inserts them into an in-memory queue. An upload routine is invoked periodically to upload data from the queue (using the MQTT protocol in our initial usecase). Note that prior to uploading, relevant data entries are tied together into a bundle and the MQTT protocol is used for the purpose of uploading. The data bundle is encoded into a binary payload instead of using verbatim XML or JSON format, which are arguably expensive in terms of byte usage over a constrained 3G data plan. A central server is maintained that collects these encoded payloads, decodes them and republishes them in a self-descriptive JSON format under a suitable MQTT topic name so that other users can consume the published sensor data. Any device that can be connected to the MQTT broker can subscribe to the topic to consume the sensor data.

In the following two sections, we discuss two focused forms of adaptation that were performed using the EnviroSCALE setting. The first use case studies how sampling rates for various sensors can be adapted based on 3G dataplan budgets that constrained communication capabilities. The second use case enhanced the basic EnviroSCALE hardware/software platform with additional audiovisual sensors and container-based analytics to adaptively trigger and conduct appropriate on-board analytics prior to communication.

4 BUDGETED OPERATION OF ENVIROSCALE IN 3G CELLULAR NETWORKS

The basic EnviroSCALE prototype performs the task of monitoring air quality through sensors. Each sensor performs a sensing task, which constitutes reading from the sensor periodically at a certain interval. The readings, in most cases, are real-valued numbers obtained from the sensor at a particular time and location. Hence, all sensors readings, called *samples*, are timestamped and geotagged with latitude and longitude values obtained from the GPS.

As the platform deploys multiple sensors and each sensor may have a different degree of temporal sensitivity to their readings, they all may not be sampled at the same periodic interval, rather at different intervals. For example, LPG sensors can measure the level of LPG in air in every 10 seconds (because it may change suddenly) whereas the dust level can be observed in every minute (as it may remain fairly constant in a certain area). These intervals are called the *sampling intervals* of the sensors. Since every data sending operation over 3G has an overhead, it would not be wise to communicate every reading as and when they are obtained. Instead, we accumulate readings for a certain duration and upload them as a bundle. The length of this interval, which we refer to as *upload interval*, should be small enough to ensure that we do not lose the timeliness of the sensed data and large enough to amortize the cost of additional transmission overhead while uploading this information using 3G networks.

It can be argued that the overall volume of data that can be sent over a 3G connection is limited and is usually dictated by a *data plan*. A data plan is a contract between the 3G operator and a user restricting how much data the user can send over an extended period of time. For example, a 10MB per week data plan limits sending (and receiving) a total of 10MB over a week period. Typically, it is up to the user to decide how to use this allocated amount; she may prefer to use up the entire 10MB in first hour or first day or save progressively to use it throughout the whole week. Although a data plan is usually accounted for both uploading

and downloading bytes, in our case we assume the data plan is for upload bytes as the upload traffic dominates in our platform.

4.1 Adaptive Sampling for Budgeted Operation

Given a data plan and an upload interval, the question becomes one of how to choose sampling intervals for sensors so that the overall data generated during the data plan duration remains bounded with the data plan budget. Let T_i be the sampling interval of sensor i and S_i be the size of each reading. Let tuple (M, D) denote a data plan that enables uploading M bytes of data for a duration of D seconds and T be the upload interval. We split the entire data budget, D, *equally* across all the upload events. That means, the platform can upload at most $\frac{M}{D}T$ bytes at every upload interval. At every upload event, sensor readings are accumulated since the last upload event and the accumulated bytes are uploaded with an additional overhead of α bytes. Therefore, the following constraint should hold for a platform with k sensors:

$$\alpha + \sum_{i=1}^{k} \frac{T}{T_i} S_i \leq \frac{M}{D} T \tag{1}$$

which can be reduced to:

$$\sum_{i=1}^{k} \frac{S_i}{T_i} \leq \frac{M}{D} - \frac{\alpha}{T} \tag{2}$$

Therefore, the budgeted operation over 3G requires us to determine the values of T_i so that the above constraint holds. Since we should best utilize the data budget, we can hold the constraint with equality. Let $\beta = \frac{M}{D} - \frac{\alpha}{T}$, so we have:

$$\sum_{i} \frac{S_i}{T_i} = \beta \tag{3}$$

A generalized solution to (3) can be obtained as: $\frac{S_i}{T_i} = \omega_i \beta$, for some suitably defined *weight*, ω_i, for each sensor where $\sum_i \omega_i = 1$. This gives the upload intervals, T_i, as follows:

$$T_i = \frac{1}{\beta} \frac{S_i}{\omega_i} \tag{4}$$

We apply two simple approaches to determine weights to adapt data collection based on plan budgets:

- *Equal weights*: In this, we assume all ω_i's are equal. Hence, $\omega_i = \frac{1}{k}$ and $T_i = \frac{k}{\beta} S_i$.
- *Proportional weights*: Each sensor can be assigned an application specific "value" based on the importance of the sampling interval for a particular sensor: higher value means higher sampling frequency, that is, lower sampling interval. This value can be derived from the samples themselves as well. For example, samples showing lower degree of variance can be sampled at a lower frequency (i.e., at a higher time interval) and high variation in samples results in shorter interval. To this end, we use the co-efficient of variation (cv), $cv = \frac{stdev}{mean}$, of samples as a way of measuring ω_i. Hence, ω_i's are given by:

$$\omega_i = \frac{\sigma_i}{\mu_i} / \sum_i \frac{\sigma_i}{\mu_i} \tag{5}$$

where μ_i and σ_i are the mean and the standard deviation of sensor readings as observed before the current upload event. Putting ω_i in Eq (4) gives sampling intervals T_i.

4.2 Experimental Results

We conducted two types of experiments with EnviroSCALE. The first set of experiments utilized the real hardware setup deployed in diverse settings in a real city - Dhaka, Bangladesh; air quality data was collected under different scenarios and conditions. The second set of experimental results are derived from a simulation-based study where we analyze our proposed adaptive sampling techniques under various parameters.

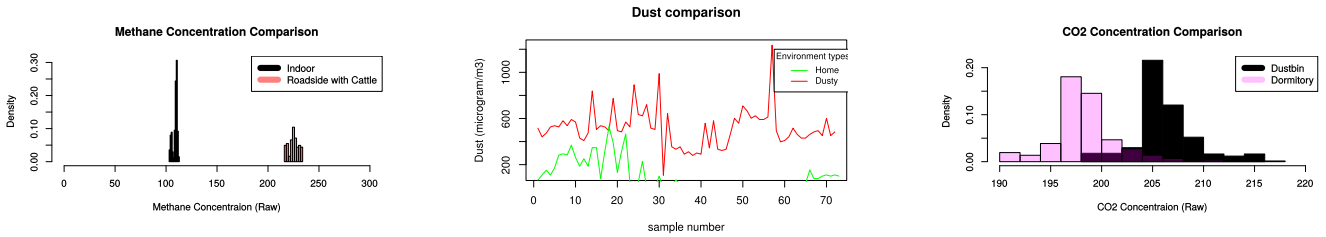
4.2.1 Data Collection from EnviroSCALE Deployment. We deployed EnviroSCALE at different places of Dhaka and collected air quality data. We tried to validate our data collected from different contexts. For instances, we compared two datasets of methane concentration readings, one collected in indoor setup in a regular day and the other one was collected from a roadside with considerable amount of cow excretion, a large source of methane. Figure 3a shows the histograms of these two, and it shows that in the later one, methane concentration shows a higher value, which should actually be the case. Figure 3b and 3c show comparisons between CO_2 and dust sensor readings in different scenarios.

4.2.2 Analysis of Different Adaptation Techniques. Now we present the comparison of the two approaches of adaptive sampling discussed earlier. We wrote a custom trace-driven and message-level simulator to model the scenario we worked in, namely generating samples from sensors (actually replay of earlier recorded data), enforcing a certain data plan and actuating network up/down time. Our performance metrics are the "overall error" in uploaded samples and the utilization of the data plan. The error measures how much the time series graph constructed from down-sampled readings deviates from the original data trace. Utilization estimates what fraction of data plan is used by the schemes.

The results depicted in Figure 4a show that the "equal weights" approach exhibits very small error compared to the "proportional weights" scheme. This might be to due to the fact that the observed sensor readings (which was collected for a short duration, roughly an hour) did not have much variation in them to have their co-efficient of variation varied across the sensors. In that, assigning equal weights across sensors works fairly well. We are currently looking forward to a longer duration operation. We also observe a decline in error as the data budget increases. Fig. 4c shows the CDF of error values to highlight the detail of the reported errors. Fig. 4b shows the utilization of data plan.

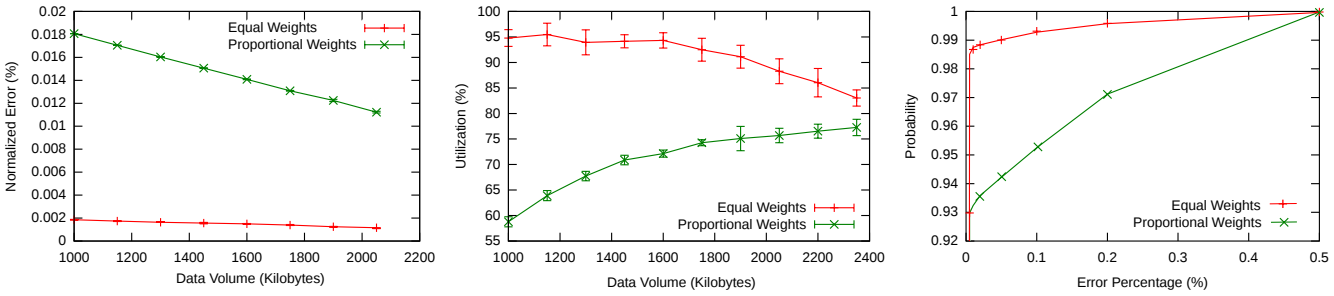
5 ENHANCING ENVIROSCALE: CONTAINER-BASED RICH SENSING ANALYTIC

We describe a container-based IoT platform in this section. This is followed by a sample usage scenario: once a gas sensor detects poisonous gas, it triggers cameras to search for the presence of people. The video (rich sensing) analytic alerts the people or notifies the authority.



(a) Methane (b) Dust (c) CO_2

Figure 3: Different sensor readings from the EnviroSCALE setup.



(a) Normalized error for different data volume. (b) Utilization for different data volume. (c) CDF of errors for a fixed data volume.

Figure 4: Different characteristics of adaptive sampling method.

5.1 System Overview

So far, we focus on data from air quality sensors, which are relatively concise compared to multimedia sensors, such as microphones and cameras. Because air quality sensors produce a few scale values, adaptively down-sampling such sensor data to meet the data plan constraints is feasible as demonstrated above. However, some comprehensive event detection may require data from multimedia sensors, which generate a larger amounts of data, where simply down-sampling sensor data may *not* work. Therefore, we have to dynamically deploy rich sensing analytic close to the multimedia sensors, and only send the processed results over the constrained data connections. In this way, we may more aggressively reduce the sensor data amount while supporting comprehensive event detection. Doing so, however, is no easy task, because the IoT platform is

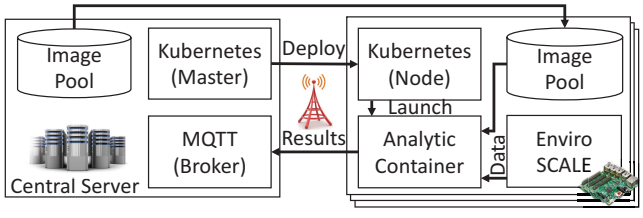


Figure 5: Container-based IoT platform for rich analytic.

resource constrained, and the resource consumption must be carefully monitored, managed, and planned. To this end, we propose a centralized management platform for rich sensing analytic, as illustrated in Fig. 5, which contains a server on the left and multiple IoT devices on the right. In addition to the EnviroSCALE, we also adopt Docker containers for dynamic deployments of rich sensing analytic and Kubernetes for management of multiple IoT devices that are heterogeneous in capabilities and locations. We assume near-by IoT devices communicate via short-range wireless networks. Next, we briefly introduce Docker containers and Kubernetes.

Docker containers enable us to package rich sensing analytic along with its dependencies, such as run-time libraries and configurations into Docker images. Docker provides tools to simplify the process of creating, deploying, and launching these Docker images on heterogeneous IoT devices. Docker containers provide some protections via isolation similar to virtual machines, but docker containers do not contain complete operating systems, and thus incur less overhead. Moreover, multiple docker containers may share the same underlying Linux kernel for faster response time.

Kubernetes provides tools to deploy, manage, and migrate rich sensing analytic as container images. It adopts client-server model, where multiple IoT devices are connected to a server and the clients periodically report the resource levels of the IoT devices to the server. Then, the Kubernetes server instructs the clients to launch appropriate Docker images on the IoT devices at the right locations. The problem of disseminating the Docker images over constrained data connections can be partially solved by the image pools, which are caches of Docker images. That is, a Docker image is sent to an IoT device only once, even if it is launched multiple times. Moreover,

Docker images have layered structures, and thus several images may share the same lower layers.

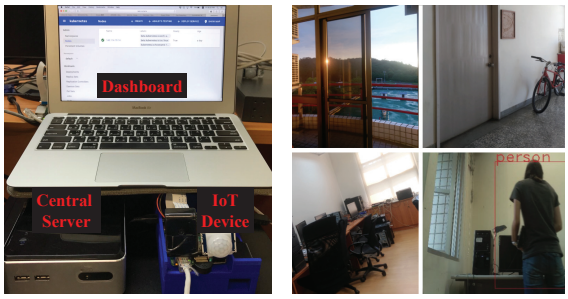


Figure 6: Our testbed (left); a detected person (right).

5.2 Case Study: On-Demand Person Detection

As a proof of concept of *on-demand* container-based IoT analytics, we implement a person detector on our platform, as illustrated in Fig. 6. We use an Intel i5 Ubuntu box installed with Kubernetes as our Central Server. The IoT devices are Raspberry Pi 3s installed with Kubernetes, Docker, and EnviroSCALE. EnviroSCALE monitors air for toxic or flammable gases and once detected, it notifies the Central Server to deploy an analytic Docker image that would activate a camera, capture photos and detect from the captured images if there is any human present in the scene (who may be exposed to the gas and can be alerted right away). Fig. 6 shows a sample detected person. The person detector adopts OpenCV for processing images, uses TensorFlow for detecting the persons and is packaged into a layered Docker container image, where the base-image layer includes a Debian operating system and required libraries and the analytic layer includes the analytic algorithm.

We conduct experiments to measure the performance of our platform. First, we measure the size of the layered-Docker images, which are 227 MB for the base layer and $<$ 1 KB (precisely, 877 bytes) for the analytic layer. We note that the base layer is cached on the IoT devices. Hence, when we want to dynamically change the analytic algorithm, we only need to send the analytic layer to the IoT device. Second, we measure the running time (Table 1) of the detector algorithm, which includes four steps: (i) EnviroSCALE triggers the deployment of the person detector, (ii) the platform transfers the analytic layer and combines it with the base layer, (iii) RPi's execute the person detection, and (iv) the person detector send the results back to the Central Server. We observe that updating the analytic algorithm (step 2) needs around five seconds, which makes the dynamic deployment feasible. One limitation of our platform is that it is difficult to run complex analytic algorithm (step 3) on a single IoT device in real-time. In future, we would look into distributing different steps on different IoT devices.

6 CONCLUSION

We study the problem of adaptive sensing with IoT devices over constrained data connections. We start with a basic adaptation approach, where air quality sensor readings are down-sampled based on the data budget. We further consider a more general container based rich sensing analytic platform, in which containers with specific sensing analytics are automatically activated once being triggered by some sensor readings. We demonstrate the feasibility of the usage scenarios on a platform built upon Docker and Kubernetes. We show that our approaches reduce the data volumes befitting to limited data plans over 3G and are applicable, in general, to IoT platforms with constrained data connections.

Table 1: Running time (in seconds) of each step

	Step 1	Step 2	Step 3	Step 4	Total
Mean	0.0010	4.9754	27.9286	0.0080	32.9228
Std	0.0008	0.4900	1.8370	0.0007	2.1192

ACKNOWLEDGEMENT

The research is funded by National Science Foundation under awards No. CNS-0958520, CNS-1450768, and CNS-1528995.

REFERENCES

[1] AirBeam. 2017. AirBeam. (2017). http://www.takingspace.org/aircasting/airbeam
[2] Awair. 2017. Know what's in the air you breathe. (2017). https://www.getawair.com/
[3] Paolo Bellavista and Alessandro Zanni. 2017. Feasibility of fog computing deployment based on docker containerization over raspberrypi. In *Proceedings of the 18th International Conference on Distributed Computing and Networking (ICDCN)*. ACM, Hyderabad, India, 16.
[4] Kyle Benson et al. 2015. SCALE: Safe Community Awareness and Alerting Leveraging the Internet of Things. *IEEE Communications Magazine* 53, 12 (2015), 27–34.
[5] Simone Brienza, Andrea Galli, Giuseppe Anastasi, and Paolo Bruschi. 2015. A low-cost sensing system for cooperative air quality monitoring in urban areas. *Sensors* 15, 6 (2015), 12242–12259.
[6] Air Quality Egg. 2017. Air Quality Egg. (2017). https://airqualityegg.wickeddevice.com/
[7] Mahmudur Rahman Hera, Amatur Rahman, Afia Afrin, Md Yusuf Sarwar Uddin, and Nalini Venkatasubramanian. 2017. AQBox: An air quality measuring box from COTS gas sensors. In *Networking, Systems and Security (NSysS), 2017 International Conference on*. IEEE, Dhaka, Bangladesh, 191–194.
[8] Hua-Jun Hong, Pei-Hsuan Tsai, and Cheng-Hsin Hsu. 2016. Dynamic module deployment in a fog computing platform. In *Network Operations and Management Symposium (APNOMS), 2016 18th Asia-Pacific*. IEEE, Kanazawa, Japan, 1–6.
[9] Roberto Morabito and Nicklas Beijar. 2016. Enabling data processing at the network edge through lightweight virtualization technologies. In *Sensing, Communication and Networking (SECON Workshops), 2016 IEEE International Conference on*. IEEE, London, Uk, 1–6.
[10] OpenSignal. 2016. Global state of mobile networks. (2016). https://opensignal.com/reports/2016/08/global-state-of-the-mobile-network/
[11] Claus Pahl, Sven Helmer, Lorenzo Miori, Julian Sanin, and Brian Lee. 2016. A container-based edge cloud PaaS architecture based on Raspberry Pi clusters. In *Future Internet of Things and Cloud Workshops (FiCloudW), IEEE International Conference on*. IEEE, Vienna, Austria, 117–124.
[12] Claus Pahl and Brian Lee. 2015. Containers and clusters for edge cloud architectures–a technology review. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*. IEEE, Rome, Italy, 379–386.
[13] Pei-Hsuan Tsai et al. 2017. Distributed Analytics in Fog Computing Platforms Using TensorFlow and Kubernetes. In *Asia-Pacific Network Operations and Management Symposium (APNOMS)*. Springer, Seoul, Korea, 1–6.
[14] Md Yusuf Sarwar Uddin, Alexander Nelson, Kyle Benson, Guoxi Wang, Qiuxi Zhu, Qing Han, Nailah Alhassoun, Prakash Chakravarthi, Julien Stamatakis, Daniel Hoffman, et al. 2016. The Scale2 Multi-Network Architecture for IoT-Based Resilient Communities. In *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE, St. Louis, Missouri, 1–8.
[15] Alejandro Velasco, Renato Ferrero, Filippo Gandino, Bartolomeo Montrucchio, Maurizio Rebaudengo, Theodore E Simos, Zacharoula Kalogiratou, and Theodore Monovasilis. 2015. On the design of distributed air quality monitoring systems. *AIP Conference Proceedings* 1702, 1 (2015), 180014.
[16] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. 2014. Internet of things for smart cities. *IEEE Internet of Things journal* 1, 1 (2014), 22–32.