Contents lists available at ScienceDirect
Journal of Systems Architecture
journal homepage: www.elsevier.com/locate/sysarc

An adaptive IoT platform on budgeted 3G data plans

Mahmudur Rahman[a,*], Amatur Rahman[a], Hua-Jun Hong[b], Li-Wen Pan[b], Md Yusuf Sarwar Uddin[c], Nalini Venkatasubramanian[c], Cheng-Hsin Hsu[b]

[a] Bangladesh University of Engineering and Technology, Bangladesh
[b] National Tsing Hua University, Taiwan
[c] University of California, Irvine, USA

ARTICLE INFO

Keywords:
Internet-of-Things (IoT)
Adaptive middleware
Container-based analytics
Air quality monitoring

ABSTRACT

In this paper, we design and implement an Internet-of-Things (IoT) based platform for developing cities using environmental sensing as a driving application. Since ubiquitous and free WiFi access is not available in most developing cities, IoT deployments must leverage 3G cellular connections that are metered and expensive. In order to best utilize the limited 3G data plan, we propose two adaptation strategies to drive sensing and data collection. 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 allocated data budget. The second approach is at the *information-level* where application-specific analytics are deployed on-board devices. This use case focuses on multimedia sensors that process captured raw media data to lower volume semantic data that is communicated. We implement the two adaptation strategies on the EnviroSCALE (Environmental Sensing and Community Alert Network) platform, which is an inexpensive Raspberry Pi based environmental sensing system that employs air quality sensors and 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, particularly in the direction of infrastructure-level adaptation. For information-level adaptation, our testbed experiment results demonstrate the practicality of adaptive sensing and triggering rich sensing analytics via user-specified criteria over budgeted 3G connections.

1. Introduction

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, World Health Organization (WHO) reported that global urban air pollution was increased by 8% during the period 2008–2013 [75]. 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 (in-situ 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 [52], 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 platform (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 [6]. SCALE was a low cost IoT platform without any data budget consideration, whereas we de-

* Corresponding author.
E-mail addresses: mahmudhera93@gmail.com (M. Rahman), amatur003@gmail.com (A. Rahman), hua.j.hong@gmail.com (H.-J. Hong), helen36974@gmail.com (L.-W. Pan), ysarwar@gmail.com (M.Y. Sarwar Uddin), nalini@ics.uci.edu (N. Venkatasubramanian), chsu@cs.nthu.edu.tw (C.-H. Hsu).

https://doi.org/10.1016/j.sysarc.2018.11.002
Received 10 March 2018; Received in revised form 18 October 2018; Accepted 8 November 2018
Available online xxx
1383-7621/© 2018 Published by Elsevier B.V.

sign our software to work with both cellular and WiFi connection with a data budget constraint. 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.

Another adaptation strategy is to enable simple IoT analytics on board and send lower-volume processed information over the limited communication networks. We refer to this as *information-level adaptation* in this work. In particular, 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 sample scenarios 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).
- Gas sensors may detect the existence of toxic or flammable gas, and activate cameras to identify and alert the persons in the affected zone.
- Microphone arrays may report gunshots, and activate cameras for collecting evidence and locate the suspects.

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.

The rest of this paper is organized as follows. Section 2 surveys the literature. The EnviroSCALE prototype is detailed in Section 3. Section 4 formulates and solves the adaptation problems to meet the data plan budget. This is followed by the evaluations reported in Section 5. Section 6 concludes the article.

2. Related work

The uprising trend in Internet of Things has inspired researchers around the world to devise various solutions using IoT. Various application scenarios, architectural elements and future directions in IoT are presented in [19]. The application scenarios are extent from home to community and national level.

IoT based smart home automation. Home automation solutions are available in literature [33,55,58], which include Raspberry Pi (RPi) based monitoring and access control system for smart home [13,56]. Vujović and Maksimović [74] identified RPi as a potential sensor web node for home automation. In [32], an IoT based smart parking system for smart community was implemented. Environmental condition monitoring possibilities were addressed in the contributions of [31]. A dedicated IoT-based family robot based was implemented in [38]. Raspberry Pi for surveillance purpose is used in the works of [68].

Smart agriculture. IoT solutions are not limited to smart homes. Possibilities of smart agriculture based on IoT and cloud computing was studied in [18,65,80]. In [4], authors discussed development of smart devices to solve security and monitoring problems for agriculture. An autonomous percipient irrigation system was developed in [27] using Raspberry Pi. Another Raspberry Pi based solution for Green IoT Agriculture and healthcare was proposed in [50]. An intelligent agriculture management information system based on IoT was designed in [76].

Healthcare. Healthcare is another area which experienced IoT development. In [28], authors made a comprehensive study on IoT for healthcare. Being easy to interface with health monitoring devices, Raspberry Pi is popular for healthcare based IoT solutions. IoT based patient monitoring systems were fashioned in [21,35,36]. IoT based biometrics have also gained popularity in recent times [69]. The works of [16] uses KVM virtualization in industrial applications.

Environment monitoring. Several recent efforts focus on the use of IoT solutions for environment monitoring. Among these, there are solutions to monitor water quality in real time [73], smart environment monitoring devices [26], IoT based urban climate monitoring systems [70]. IoT enabled air quality monitoring system was developed in [3]. All these four solutions [3,26,70,73] were based on Raspberry Pi. A basic architecture and process for distributed air quality monitoring is discussed in [72]. Air Quality Egg [14] 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 use of Raspberry Pi as the core module in IoT system has its own benefits, as seen in and highlighted in [40]. It compares Raspberry Pi based device with other devices, and points out its flexibility in use and its ability to house a large number of external peripherals. These attributes make Raspberry Pi particularly suitable for use in air quality monitoring application. Another air quality monitoring IoT system that uses MQTT to measure and report humidity, carbon dioxide, dust, and air pressure is discussed in [37]. This system is installed in India and uses IoT IBM BlueMix, which is a PaaS (Platform-as-a-Service) provided mainly to enterprises. Another air quality monitoring IoT system that uses MQTT to measure and report humidity, carbon dioxide, dust, and air pressure is discussed in [37]. These IoT based solutions are plain implementations and do not consider any strategy to narrow the data volume or employ any virtualization technique.

Safe and smart community. Besides other possible IoT based solutions in various scenarios, a keen and promising area to explore is community wellness. The Safe Community Awareness and Alerting Network (SCALE) [6] 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 devices. SCALE supported data exchange between IoT content producers and consumers using a Pub/Sub architecture through the use of lightweight M2M (machine to machine) protocols such as MQTT. SCALE2 enhanced the basic SCALE platform with resilience capabilities including multiple network [61] capabilities. Similar smart community efforts around the world have gained popularity in recent years. For example, the Padova Smart City [78] implements techniques to collect temperature, humidity and light sensitivity data and store them in an HTTP-CoAP proxy server. AQBox [22], an earlier version of EnviroSCALE, illustrates the promise of off-the-shelf sensors for air quality monitoring. U-Sense [8] is a low-cost sensing system using sensors and wireless access point to upload data.

Raspberry Pi as the core module in IoT system. The use of Raspberry Pi as the core module in IoT system has its own benefits, as seen highlighted in [40]. It compares Raspberry Pi based device with other devices, and points out its use in flexibility and its ability to house a large number of external peripherals make it suitable for use in air quality monitoring application. Among the previously mentioned solutions, most of them were implemented using Raspberry Pi [3,13,19–21,26,27,32,33,35,36,38,50,55,56,58,62,69,70,73,74]. In [79], authors explored IoT applications using Raspberry Pi thoroughly. The works of [20] explained with examples how to implement IoT devices using RPi. Choice of RPi is also popular for building middleware [7,49]. The necessity of a IoT middleware platform for managing distributed source of streams is recognised and discussed in breadth in [66]. The framework they propose to bridge the gap in the lack of availability of IoT middleware uses TensorFlow as computing module and Raspberry Pi to run the smart home applications to validate their architecture. A scalable and low cost MQTT broker and implemented using RPi was proposed in [30]. In [39], authors propose an architecture for IoT as a service and implemented so using RPi. Being so popular among IoT researchers and easy to use, EnviroSCALE was also built using RPi.

3G as IoT gateway. Having RPi as the base hardware, the choice for IoT gateway can both be WiFi and 2G/3G networks. 3G network has been identified as one of the promising IoT gateways in literature [12,19,82]. The works of [25] proposed an IoT framework that makes use of 3G networks for connectivity. Implementation of IoT solutions in various areas using 3G is also popular [15,17,29,41,57]. Although these use 3G for connecting to the Internet, they do no consider constraints due to the expensive 3G data.

Container based virtualizations. With rise in IoT devices and envisioning that everything will eventually be connected to the Internet, container based virtualizations have been gaining its popularity for providing lightweight services. In [45], the authors compare virtual machines with lightweight virtualization technologies, and present three usage scenarios in smart cities. The authors of [11] explore container-based virtualization on smart objects for IoT Cloud scenarios. In [48], the authors develop a container-based platform that deploys several smart-car applications on Raspberry Pis. The deployment on in-vehicle networks improves the efficiency of the resources allocation. Morabito et al. [46] adopt two different lightweight virtualization technologies on a real IoT testbed to quantify the performance with and without the virtualization overhead. In [67], authors propose a container-based resource allocation scheme. In [53], a container-based PaaS architecture for RPi clusters was proposed. The works of [10] presented a three-layer container-based architecture. A new concept of Software Defined IoT Units was introduced [51] to provision container-based IoT cloud systems. Morabito et al. [47] implement a lightweight gateway using Raspberry Pi. The gateway contains two modules: northbound and southbound, for higher flexibility. The northbound module communicates over the Internet and the southbound module communicates with the sensors. Yin et al. [77] employ container based virtualization with task-scheduling algorithm and reallocation mechanism to mitigate the problems caused by devices failure and task delays. Brost et al. [9] leverage the isolated execution supports of containers to realize a secure architecture. Dynamic application deployment on IoT devices utilizing lightweight virtualization methods are gaining popularity [5,44,53,54,64]. Various container technologies are also evaluated in literature [42,43]. With container-based virtualization, we could build process fast and easily to manage resources and diverse services on heterogeneous device. Most important, the overheads such as CPU utilization, power consumption, and the latency significantly decrease. Our earlier work [24,60] 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.

Adaptive operations in IoT. Several IoT systems focus on adapting operation based on 'power' very similar to the adaptivity of budget we are employing in this paper. In [81], a low power wide area network is implemented. AdaM is an adaptive monitoring framework for IoT devices [59]. In [34], the authors devised an adaptive rule based IoT engine for health data acquisition. An adaptive lossless entropy compressor was proposed in [71]. None of these techniques use virtualization.

The key innovation of our work is to design a system that considers data plan constraints, focusing specifically on 3G cellular networks. The idea is to employ dynamic adaptation to collect samples from the sensors, without losing information content. This adaptation is essential to support resource provisioning for community IoT systems, where there are frequent changes in the sensing and communication contexts. The various adaptation 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: IoT platform for environmental monitoring

We first describe the two adaptation strategies. We then describe in detail the physical sensors based EnviroSCALE box and then the container-based analytic modules and their deployment and management.

3.1. Overview of the EnviroSCALE prototype

EnviroSCALE is a multi-sensor IoT platform that is designed for measuring environmental pollution through a series of gas, dust, air quality sensors. The platform is constructed to be deployed outdoor where WiFi connections may not be available, hence it uses 3G cellular connections to upload sensor data to the cloud. Since 3G connections are metered and have usually fixed data budget (dictated by the associated data plans), the platform needs to operate in a budget-aware fashion and needs to shape its data traffic matching the data budget by adjusting sampling frequencies of on-board sensors or data pruning (if required).

Our first version of EnviroSCALE, namely EnviroSCALE *box*, contained only a set of *physical* sensors that mostly generated time-series scalar data. EnviroSCALE *box* only had infrastructure-level adaptations. Later on, we augmented the platform so that it can make rich sense-making that can involve multi-media sensors, such as camera and microphones. The augmented version supports both infrastructure- and information-level adaptations.

For example, whenever a gas sensor detects a poisonous gas exceeding a certain level, it can trigger a camera to capture images and detect if there is any human in the area so that she can be alerted. This kind of *event* detection needs data from multimedia sensors which are much larger than other sensors, say gas sensors. To accommodate sensors with large values, we instrumented EnviroSCALE with the capability of deploying rich sensing analytic close to the multimedia sensors. This enables the platform to send processed results instead of raw multimedia data over the constrained data connections. Since these analytics are application specific and may need to perform rich but arbitrary computation on sensor data, they are hosted through containers. These container-backed analytic modules effectively function as *virtual* sensors in the platform and generate event data, which are then uploaded over the constrained 3G connections. In this way, we do reduce the sensor data amount more aggressively while supporting comprehensive event detection.

For the point of deploying the containers for these analytics, There are multiple ways to so. Mature or stable analytics modules can be shipped with the SD cards for Raspberry Pis. For experimental or new analytics modules, administrators may go to the proximity of Raspberry PIs and push the containers over short-range wireless networks, such as WiFi Direct. In the case of urgent, such as security, updates, the containers can be downloaded through 3G downlinks during the off-peak time. In the rest of the article, we assume that the containers are preloaded on Raspberry PIs before being launched.

3.2. EnviroSCALE box

We give the schematic of the box 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 8 GB microSD card loaded with a standard Linux image and the SCALE software stack. The whole

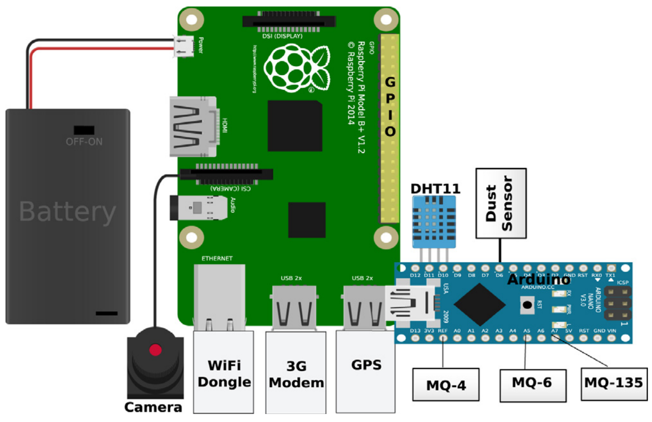


Fig. 1. Hardware schematic of EnviroSCALE box.

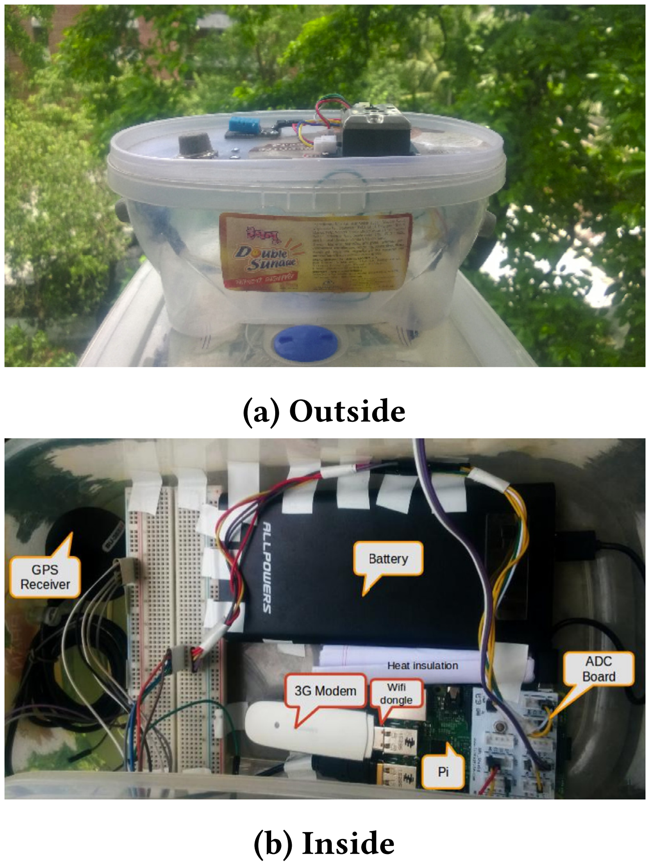


Fig. 2. Photos of our EnviroSCALE box.

setup, when deployed outdoors, is powered by an external battery of 19,200 mAh capacity. Fig. 2 is an image of our the initial EnviroSCALE prototype.

The software stack of the EnviroSCALE box includes 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. This adjustment is done dynamically to adaptively change the sampling rate based on the constrained data connection. 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 to an MQTT sever). The upload routine also handles data coming from the virtual sensors (i.e., analytic modules) and shape the traffic to fit in the data budget (which we describe in the next section).

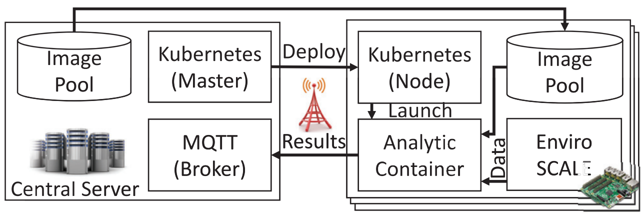


Fig. 3. Container-based IoT platform for rich analytic.

As an effort to reduce data volume even more, the upload routine ties together data entries into a bundle and encodes them 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 publishes them back 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.

3.3. *Container-based analytics for rich sensemaking*

As we argued, EnviroSCALE hosts analytic modules running on containers to make rich sensemaking, such as event detection. Doing so, however, is no easy task, because the IoT platform is 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. 3, which contains a server on the left and multiple IoT devices on the right. In addition to the EnviroSCALE box, 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.

4. Adaptive sensemaking under budgeted data plans

In this section, we first give the architecture of EnviroSCALE. We then present the two adaptation strategies.

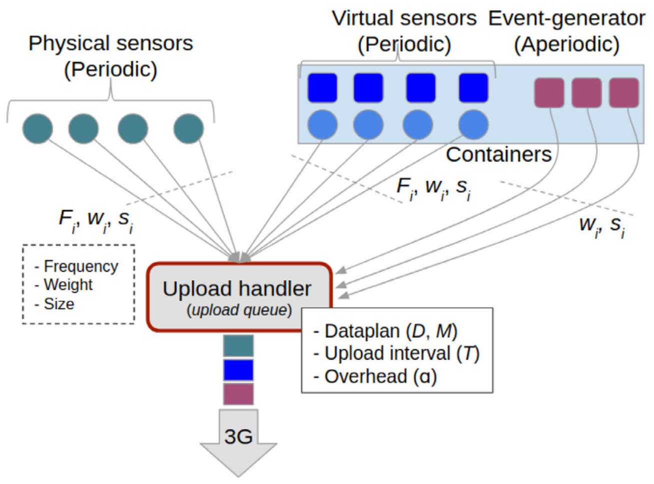


Fig. 4. Architecture of EnviroSCALE.

4.1. *Architecture of EnviroSCALE*

Fig. 4 present the architecture of the EnviroSCALE prototype, which consists of physical sensors, virtual sensors, and event-generator. The physical and virtual sensors generate periodic sensor readings, while the event-generators produce event data aperiodically, more precisely whenever an event is detected. 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. Compared to physical sensors, the virtual sensors and event generators are realized as analytic logics packaged in containers, so as to extract the more condense information out of the physical sensor readings. These containers are preloaded on Raspberry Pi, and then dynamically launched and teared down following the sensing demands and available resources. The core of EnviroSCALE is an *Uploader Handler* that determines: (i) the upload frequency of individual sensors and (ii) whether to launch or tear down containers (of virtual sensors and event-generators) under the constraint of 3G wireless bandwidth. The later decisions on which containers to launch or tear down are associated with some overhead and thus are longer-term decisions; while the former decisions distribute the data budget in real time, in order to provide as frequent sensor readings as possible. Notice that event-generators produce aperiodic sensor readings, which are transmitted once in a while and thus have the highest priority. The residue data budget (with size of aperiodic sensor readings deducted) is allocated among the physical and virtual sensors.

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 (or sampling *frequency* is expressed in Hertz, the reciprocal of the intervals). Sensor readings also have a certain size per sample (in bytes). The sampling frequencies together with the sample sizes determine the overall data volume of operation.

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 10 MB 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 10 MB 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.

Given a certain data plan, it can happen that the total volume of data generated by all sensors when configured to run at their specified frequencies may exceed the data budget. In that, the frequencies of different sensors need to be adjusted: more precisely frequencies are adjusted to lower values. The adjusted frequencies are referred to as *operating frequency* per sensor. Given a data plan, the central question of the upload handler is as how to choose sampling frequencies for sensors so that the overall data generated during the data plan duration remains bounded with the data plan budget. We describe this next.

4.2. *Handling data uploads under constrained data plans*

The data characteristics of each sensor i, (both physical and virtual) are described by three parameters as shown in Fig. 4: (F_i, s_i, w_i), where F_i is the sampling frequency (and $T_i = \frac{1}{F_i}$ is the sampling interval), s_i be the size of each reading (we assume all readings from a given sensors are of equal sizes) and w_i is a *weight* denoting the degree of importance of its samples, which can be specified by the application or can be derived from the sensor readings themselves. The weights also indicate the degree of sensitivity to the adjustment to their requested frequencies (if ever needs to be done by the upload handler). The aperiodic event-generators, however, do not have periods (hence sampling frequencies), instead they send data as events get detected.

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.

Given a data plan and an upload interval, the question becomes one of how to choose sampling frequency or intervals for sensors so that the overall data generated during the data plan duration remains bounded with the data plan budget. 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 $\rho \times T$ bytes at every upload interval, where $\rho = \frac{M}{D}$ is the *upload rate* (measured in bytes/sec). This definition of ρ, however, gives only the initial value of ρ as it changes throughout the operation depending on the total remaining budget. 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. This overhead is fixed for a data packet sent via network channel, in other words, it is the per packet protocol overhead. Therefore, the following constraint should hold for a platform with k sensors:

$$\alpha + \sum_{i=1}^{k} \frac{T}{T_i} s_i \le \rho \times T \tag{1}$$

which can be reduced to:

$$\sum_{i=1}^{k} \frac{s_i}{T_i} \le \rho - \frac{\alpha}{T} \tag{2}$$

Let $A = \rho - \frac{\alpha}{T}$, so we have:

$$\sum_{i} \frac{s_i}{T_i} \le A \tag{3}$$

Writing in terms of sampling frequency, it becomes:

$$\sum_{i} s_i F_i \le A \tag{4}$$

Therefore, the budgeted operation over 3G requires us to adjust the values of F_i so that the above constraint holds. If the constraint holds, that means the sensors can operate in their specified frequency and no adjustment to their frequencies are required. Otherwise, an adjustment is made. Let f_i be the operating frequency of sensor i (instead of its specified F_i) so that total data volume remains bounded. Obviously, f_i is strictly less than F_i for at least one sensor if not all. Since we intended to best utilize the data budget, we may want the constraint 4 to hold with equality, which gives:

$$\sum_i s_i F_i = A \tag{5}$$

As the sampling frequencies are adjusted, we define a loss function, $L_i(f_i)$, which gives the perceived *loss* in sensor readings whenever the sensor's frequency is reduce from the base frequency F_i. Technically, this loss function is a convex function over the difference between f_i and F_i (the value of loss increases as the difference grows). The function can even be different across different sensors. We, nonetheless, assume a generic function: $L_i(f_i) = (F_i - f_i)^2$. Our goal is to choose f_i so that the overall weighted loss over all sensors is minimized. We have:

$$\min \sum_i w_i L_i(f_i) \text{ s.t. } \sum_i s_i f_i = A \tag{6}$$

The strategy we adopt to find the local minima is based on Lagrange multiplier. Introducing Lagrange multiplier, we have

$$\mathcal{L}(\lambda, \bar{f}_i) = \sum w_i L_i(f_i) - \lambda(A - \sum s_i f_i) \tag{7}$$

We are to find f_i's and λ so as to minimize $\mathcal{L}(\lambda, \bar{f}_i)$. Differentiating with respect to f_i and λ gives:

$$\frac{\partial \mathcal{L}}{\partial f_i} = w_i \mathcal{L}'(f_i) + \lambda s_i = 0 \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = -A + \textstyle\sum_i s_i f_i = 0 \tag{9}$$

Plugging in the defined loss function and its derivative, we get the following solution for f_i's:

$$f_i = F_i - \frac{\lambda s_i}{2w_i} \tag{10}$$

where

$$\lambda = \frac{2}{\sum_i \frac{s_i}{w_i}} \left(\sum_i s_i F_i - A \right)$$

Given (F_i, w_i, s_i) triples per sensor, the upload handler computes the operating frequencies, f_i, using Eq. (10). If we denote $\Delta = \sum_i s_i F_i - A$ as the amount by which the data volume exceeds budget (A) when the sensors operate in their base frequencies, we have:

$$f_i = F_i - \frac{s_i / w_i}{\sum_i s_i / w_i} \Delta \tag{11}$$

As we see from the above expression, frequency adjustment is done only when Δ is +ve, that is, data budget is exceeded by the total volume of data. If Δ is zero or negative, no adjustment is required and the sensors operate in their base frequency (F_i).

Once the operating frequencies for both physical and virtual sensors are determined, now the question is how the upload handler realizes these frequencies in action. For the physical sensors, the platform would ultimately sample the sensors at those frequencies, that is, it would take periodic sensor readings at an interval of $\frac{1}{f_i}$. For virtual sensors, however, this cannot be made because they operate inside a container and their operating frequencies can be hardly changed. Instead, the upload handler would take all readings at an interval $\frac{1}{F_i}$ but prune some data points (so as to match data volume with that of f_i). Two options can be tried in this regard:

- Each data reading reported from the virtual sensor may contain a measure of importance with it depending on the application specific value assigned to the reading. One way to measure this importance is to calculate how different a reading is from its previous measurements. When the system runs under $f_i < F_i$, it keeps only readings with higher importance level and drops the lower valued ones.
- When the system cannot find the less important data to prune or there is no important level assigned to the readings, the handler *randomly* drops incoming data items with probability $1 - \frac{f_i}{F_i}$, which shapes the frequency from F_i to f_i, which in turn would meet the budget.

While the upload handler shapes incoming data received from both the physical and virtual sensors by adjusting their operating frequencies, what remains is to describe how it handles data coming from the aperiodic event-generators. Event-generators produce data occasionally and the generated data is immediately pushed to the upload queue (being ready to be uploaded in the next upload event). Once uploaded, the handler deducts the total volume of data uploaded from the remaining data plan budget and updates data *upload rate*, ρ, for the next upload event as follows (if budget and duration still remain):

$$\rho = \frac{\text{remaining data budget from the data plan}}{\text{duration left until data plan expires}}$$

Alternative solutions for f_i There can be occasions when the base frequencies of sensors, F_i, (as well as weights, w_i) may be unknown, especially for the physical sensors. In that case, we cannot solve Eq. (6) to find f_i. Instead, we are simply left with the constraint: $\sum_i s_i f_i = A$, which can have a generalized solution of the form: $s_i f_i = \omega_i A$, for suitably define weights where $\sum_i \omega_i = 1$. In that, operating frequencies are given by:

$$f_i = \frac{\omega_i A}{s_i} \tag{12}$$

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

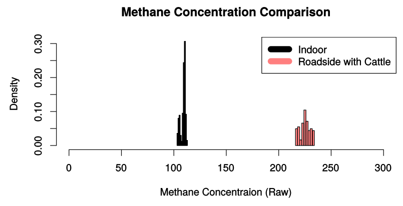
- *Equal weights*: In this, we assume all ω_i's are equal. Hence, $\omega_i = \frac{1}{k}$ and $f_i = \frac{A}{s_i k}$.
- *Proportional weights*: Each sensor can be assigned an application specific "value" based on the importance of the sampling interval for a particular sensor: higher value means higher 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), $cv = \frac{\text{stdev}}{\text{mean}}$, of samples as a way of measuring ω_i. Hence, ω_i's are given by:

$$\omega_i = \frac{\sigma_i}{\mu_i} / \sum_i \frac{\sigma_i}{\mu_i} \tag{13}$$

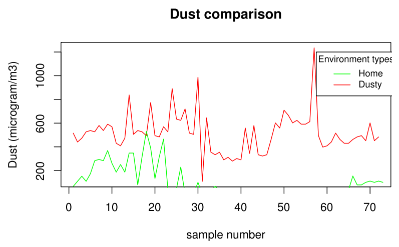
where μ_i and σ_i are the mean and the standard deviation of sensor readings as observed before the current upload event. Putting ω_i in Eq. (12) gives operating frequency f_i.

4.3. Deployment and activation of container-driven analytics

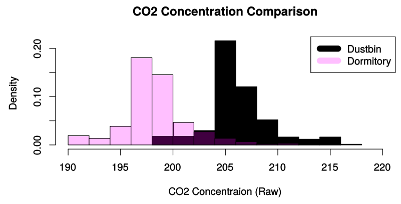
To dig useful information from multimedia data, we leverage the container technology running with diverse analytics to realize it. The analytics are launched in periodical or aperiodical ways. E.g., air pollution analytics may update pollution heatmaps every 10 s. In contrast to air pollution, some emergent events, such as fire fighting event are launched in aperiodical way. There are two possible ways to launch the analytics: (i) launching the analytics on a cloud server, which receives sensing data (image) from data originates (Raspberry Pis) for further analyzing or (ii) launching the analytics on Raspberry Pis and send analyzed results to the cloud server. Sending images to the cloud server



(a) Comparison of methane sensor readings (raw ADC output) in collected from indoor and outdoor environments



(b) Comparison of dust particle amount (raw ADC reading) of clean home air and dusty outdoor air



(c) Comparison of CO_2 concentrations between clean dormitory air and air beside a dustbin

Fig. 5. Different sensor readings from the EnviroSCALE setup.

consumes large amount of network resources, while launching all the analytics on Raspberry Pi suffers from limited resource budgets, such as RAM. Hence, we formulate our information-level adaptation problem to minimize the overall network consumption with RAM and network budgets.

We formulate the problem as a ILP problem, as illustrated in formulation (14). This problem will be solved in every T 'tes. The decision variable is a_i, which is a boolean value to decide whether we launch analytics i. Every analytic has its desired frequency F_i. Moreover, $Z(i, a_i)$ gives consumed network traffic of launching i or sending data to the cloud without launching i, where $i \in \mathbf{J}$ and $\mathbf{J}$ is a set of analytics. The objective function Eq. (14a) minimizes overall network traffic. Eq. (14b) ensures that the total consumed bandwidth is smaller than the network budget ($rate*T$) of our 3G cellular networks. Eq. (14c) ensures that the consumed RAM resources of launched analytics are smaller than the total remaining RAM budgets (S) of the devices connected through the same 3G networks.

$$\min \sum_{i\in\mathbf{J}} Z(i, a_i) * F_i * T/w_i \tag{14a}$$

$$st: \sum_{i\in\mathbf{J}} Z(i, a_i) \le rate * T; \tag{14b}$$

$$\sum_{i\in\mathbf{J}} a_i r_i \le S; \tag{14c}$$

$$a_i \in \{0, 1\}\ \forall i \in \mathbf{J}. \tag{14d}$$

5. Experimental results

We conducted three sets 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. The third set of results correspond to deployment of container-based analytics and an application usecase of on-demand activation of analytics.

5.1. 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 instance, 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. Fig. 5a 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. Fig. 5b and 5 c show comparisons between CO_2 and dust sensor readings in different scenarios.

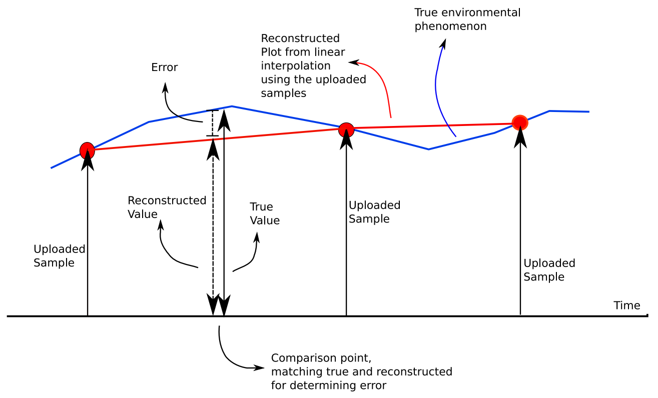

True environmental phenomenon
Reconstructed Plot from linear interpolation using the uploaded samples
Error
Reconstructed Value
True Value
Uploaded Sample
Uploaded Sample
Uploaded Sample
Uploaded Sample
Time
Comparison point, matching true and reconstructed for determining error

Fig. 6. Errors are measured from the trace and the reconstructed data points.

5.2. Analysis of different adaptation techniques

Now we present the comparison of the three adaptive sampling approaches. 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.

The simulator mimics EnviroSCALE by taking the data plan (*D, M*) as its configuration. It can be configured to imitate variable number of periodic sensors (both physical and virtual), and aperiodic event-generators. In the experiments, both physical and virtual sensors use data traces from previous deployments for generating sensor readings. Aperiodic events happen as a Poison process with a certain rate where each event has a certain size. Table 2 shows a set of important simulation settings. Each simulation runs for 500 s with an upload interval of 50 s. The detailed simulation scenario along with the simulator code and data trace is publicly available here [63].

Due to the adjustment of sampling frequencies, the sensor data streams that are eventually uploaded differ from the original trace. In fact, the uploaded data stream may contain less number of data points from the original data stream . We are interested to see to what extent these two streams vary. Fig. 6 shows how the two streams are constructed and compared. When a given sensor reading is not uploaded—which we referred to as being missed in the reconstructed stream—the missing value is estimated by linear interpolation from the two nearby uploaded readings as shown in Fig. 6. The difference between these two values is an *error*. In particular, we compute the relative errors across all missing values and take its average. We refer to this quantity as Mean Absolute Percentage Error (MAPE), which is defined as:

$$MAPE = \frac{100}{N} \sum_{i=1}^{N} \frac{|\mathrm{Tr}_i - \mathrm{Rc}_i|}{\mathrm{Tr}_i} \tag{15}$$

Table 1
List of sensors tested.

Sensors	Measurement
MQ−4	Combustible gases and high sensitivity for natural gases
MQ−6	Butane and LPG
MQ−135	Air quality sensor (sensitive for Benzene, Alcohol, smoke, Ammonia, Sulfide etc.)
DHT11	Temperature and humidity of surrounding environment
GP2Y1010AU0F	Optical dust sensor to measure dust particles
GlobalSat BU−353−S4	GPS co-ordinates

Table 2
Simulation setting: some important parameters.

ID	Sensor Name	Desired Frequency (Hz)	Size of a Reading (bytes)	Type
0	CO2	5	16	Periodic Physical
1	Methane	5	16	
2	Dust	5	16	
3	VS1	1	16	Periodic Virtual
4	VS2	1	16	
5	ET1	aperiodic	2048	Aperiodic Event-Triggered

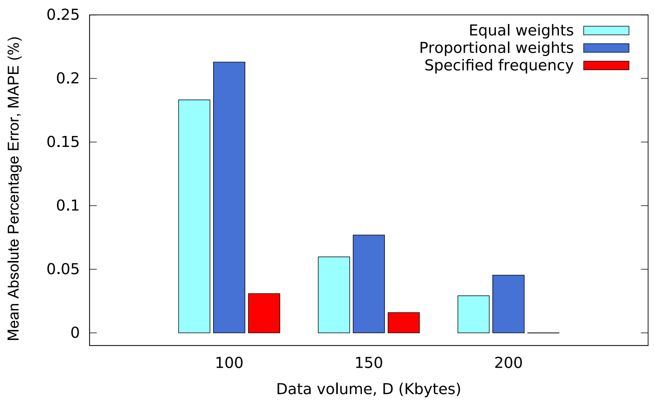


Fig. 7. Effect of D on MAPE calculated from Dust.

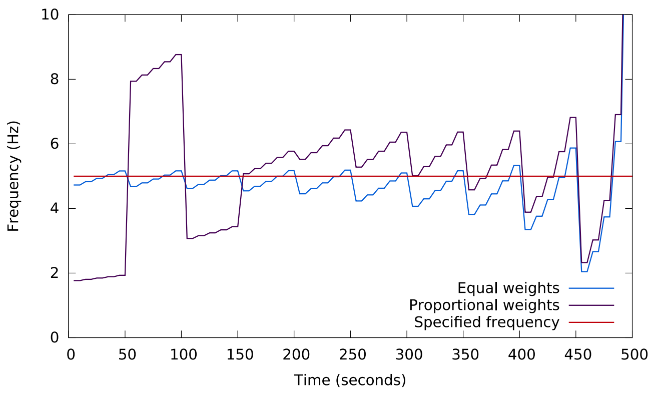


Fig. 8. Change of the sampling period over running time for the three approaches (D = 200 Kbytes).

where Tr_i and Rc_i denote the true and interpolated value respectively and N is the total number of missing points. Obviously, if all data points are uploaded, MAPE becomes zero, otherwise it takes a positive value: higher value indicating higher degree of error.

We compute and compare results for three frequency adaption techniques. One technique is the one that adjusts the specified (i.e., base) frequencies of periodic sensors using Eq. (11). The other two techniques are due to Eq. (12) with "equal" and "proportional" weights.

The results for comparing the trace of Dust with the collected samples are presented in Fig. 7. The reason for choosing Dust sensor is simply because data collected from this show the highest variation; and therefore this sensor is particularly more important than others. It is clear that "Specified frequency" method gives the least error. Among the other two, "Equal weights" is the better one; as we had already investigated in our earlier works.

The reason behind "Specified frequency" approach performing better than the other two is investigated in Fig. 8. This illustrates the frequencies that were employed to sample the Dust sensor over the runtime. The frequency chosen by "Specified frequency" approach is static, and does not change. This is because the base frequencies of the sensors given in the simulation setting do not overflow the data budget, and therefore the stable frequency of 5 Hz is used throughout the entire time. On the other hand, the other two approaches do not care for base frequency, rather try to maximize bytes used (of course, by keeping within the data plan). Therefore, the frequencies given by these are not so stable over time. Although the other two are often able to operate with higher frequencies (compared to "Specified frequency"), the moments when the frequency is lower contributes to the comparatively larger error.

It is important to note that these frequencies that were configured into the simulator were calculated by analyzing the data trace. We collected samples from trace using various sampling frequencies, and found that 5 Hz is convincingly fast enough for resulting in very low error. Thus, "Specified frequency" approach, with this little analysis, brings about a priori knowledge of the previously collected data into the whole adaptive environment, and performs better.

We also outlined similar plots for this approach, in this case, forcing it to adjust its frequencies by allowing it a lighter data plan. Due to the light data budget, the initial frequencies are lower, but quickly rises close to the base frequency. For the heavier data plan, on the other case, the base frequencies do not overflow the budget, and thus the base frequencies are used through and through. Of course, this means sampling at a speed slower than we are capable of, thus resulting in lower utilization (Fig. 10). Then again, operating in base frequencies is exactly the goal of this adaptive sampling scheme after all.

Finally, we present *Utilization*, defined as ratio of bytes actually used, to bytes installed in the data plan (*D*), in Fig. 10. The purpose of this study is to assess if our methods are in fact capable of operating within the data plan, or they exceed it. Utilization for "Equal weights" and "Proportional weights" both exhibit an utilization close to 100, whereas "Specified frequency" shows a lower value for two cases. For the lower utilizations, the frequencies are not challenged by the data plan, and

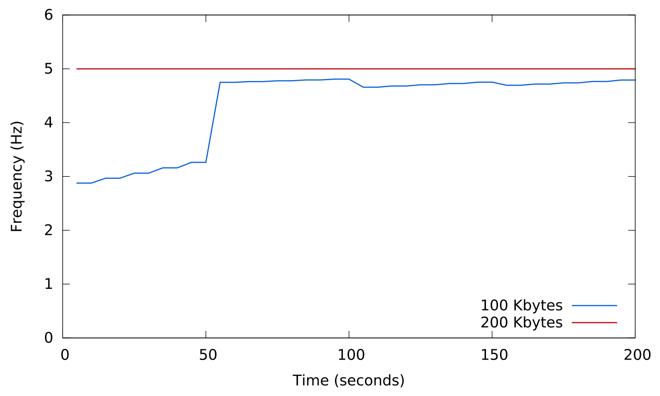


Fig. 9. Change of the sampling period over running time for "Specified frequency" approach.

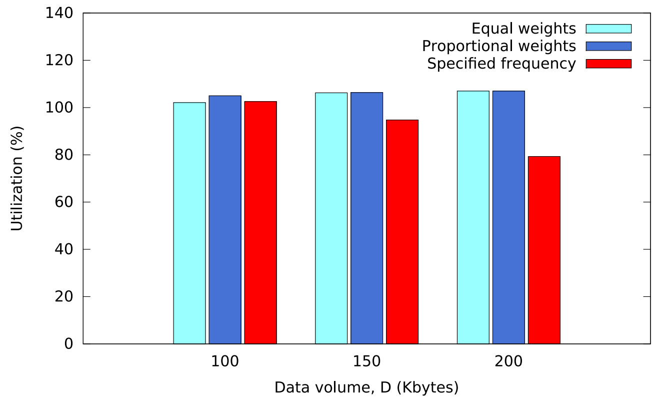


Fig. 10. Utilization from various runs.

therefore the base frequencies are maintained altogether, resulting in lower utilization.

We conclude this section by pointing out that for some cases, the utilization is more than 100%, thus exceeds the data plan. However, the extra usage is at most 5% and therefore, considered tolerable.

5.3. Optimal information-level adaptation

We optimally solve our information-level adaptation problem in Eq. (14) using CPLEX, which is referred to OPT. Moreover, we implement two baseline algorithms, called ALL and NONE for comparisons. The ALL algorithm launches all the on-board analytics, which minimizes the network consumption without considering the RAM budget. The NONE algorithm analyzes everything in the cloud, which consumes the network data plan budget the most. In our experiments, we vary the number of analytics $|\mathbf{J}|$={5, 10, 15, **20**, 25}, where 20 is the default setting. In each run, the analytics are randomly selected from two sample analytic applications implemented by us: *sound recognizer* and *object detector*. The sound recognizer is a periodic analytics application and the object detector is an aperiodic analytics application. We set the 3G bandwidth to 2 Mbps and T to 10 min, so that the network resource budget is rate $\times$ T=2 Mbps $\times$ 10 Min. The frequency of periodic analytic (sound recognizer) is $F_i = 2/T$, while the event generation frequency of the aperiodic analytic (object detection) is randomly generated with a mean of $1/T$. We conduct a measurement study to measure the required network resources of the analytics for every analyzing task with/without launching the containers on the Raspberry Pi. The measurement results are reported in Table 3. We use the system model derived by Hong et al. [23] to model the RAM resource consumptions of our analytics. The RAM budget S={1,2,4,**8**} GB, where 8 is the default setting. We vary the values of D and S and conduct our experiments 5 times. We report the average results with 95% confidence intervals whenever applicable.

Table 3
The network resource requirements of the analytics with/without launching the analytics on Raspberry Pis.

	With	Without
Sound Recognizer	708 B	61 KB
Object Detector	709 B	194 KB

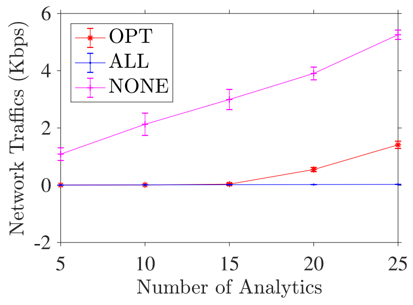


Fig. 11. Network traffic consumption of the three algorithms.

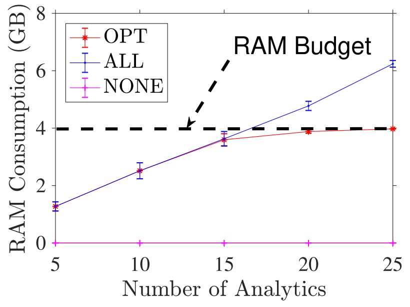


Fig. 12. RAM resource consumption of the three algorithms.

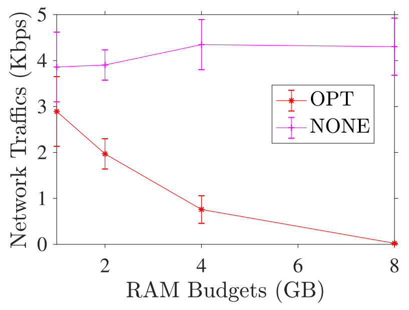


Fig. 13. Network traffic consumption under different RAM budgets.

Fig. 11 shows that compared to the NONE algorithm, our OPT algorithm saves significant amount of network resources. Moreover, when the number of analytics is smaller than 15, our OPT algorithm achieves the same performance as the ALL algorithm. This however does not mean that the ALL algorithm outperforms our OPT algorithm, as explained by Fig. 12. This figure shows that the ALL algorithm does not carefully consider the RAM budget. When the number of analytic is greater than 15, the ALL algorithm overloads the Raspberry Pi, rendering the solution impractical. Fig. 13 reports the network resource consumptions under different RAM budgets. It shows that while the RAM budget is increased, we can save more and more network resources. We note that we do not include the ALL algorithm in Fig. 13 because it does not comply with the RAM budgets.

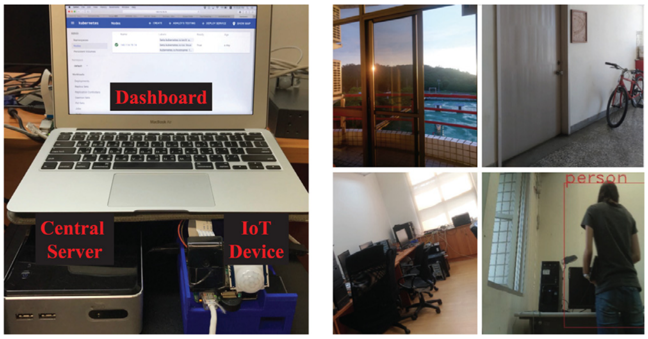


Fig. 14. Our testbed (left), and a detected person (right).

5.4. Case study: on-demand person detection

As a proof of concept, we implement a person detector on our platform for real experiments, as illustrated in Fig. 14. We use an Intel i5 Ubuntu box installed with Kubernetes as our Central Server. The IoT end devices have Raspberry Pi 3 as the base hardware, 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 analyze the captured images to see if there is any human present in the scene (who may be exposed to the gas and should be warned right away). Fig. 14 shows a sample detected person. The person detector adopts OpenCV library for processing images, uses TensorFlow framework for detecting the person and is packaged into a layered Docker container image. The base-image layer includes system software and libraries; and the analytic layer includes the analytic application. Through this case study, we demonstrate the practicality of our multi-sensor IoT platform: instead of sending an H.264 video sequence to the cloud for analysis, our platform only sends the image with the detected person. Throughout our experiments, an average of ~ 150 times of data amount reduction is observed.

6. Conclusion

We study the problem of adaptive sensemaking with IoT devices over constrained data connections, especially under limited 3G data plans. We start with an adaptation approach where 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 data volumes befitting to limited data plans over 3G and are applicable, in general, to IoT platforms with constrained data connections.

Acknowledgment

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

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.sysarc.2018.11.002.

References

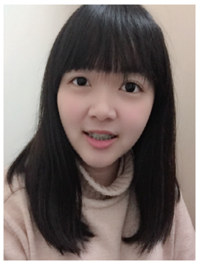
[1] AirBeam, 2017, AirBeam 2017.http://www.takingspace.org/aircasting/airbeam.
[2] Awair, Know What's in the Air you Breathe 2017, (2017), https://www.getawair.com/.
[3] C. Balasubramaniyan, D. Manivannan, Iot enabled air quality monitoring system (AQMS) using raspberry Pi, Indian J. Sci. Technol. 9 (39) (2016).
[4] T. Baranwal, P.K. Pateriya, et al., Development of Iot based smart security and monitoring devices for agriculture, in: Proceedings of the IEEE 6th International Conference Cloud System and Big Data Engineering (Confluence), 2016, pp. 597–602.
[5] P. Bellavista, A. Zanni, 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, 2017, p. 16.
[6] K. Benson, E.A. Kyle Benson, et al., SCALE: Safe community awareness and alerting leveraging the internet of things, IEEE Commun. Mag. 53 (12) (2015) 27–34.
[7] S. Bhave, M. Tolentino, H. Zhu, J. Sheng, Embedded middleware for distributed raspberry pi device to enable big data applications, in: Proceedings of the IEEE International Conference on Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), volume 2, 2017, pp. 103–108.
[8] S. Brienza, A. Galli, G. Anastasi, P. Bruschi, A low-cost sensing system for cooperative air quality monitoring in urban areas, Sensors 15 (6) (2015) 12242–12259.
[9] G.S. Brost, M. Huber, M. Weib, M. Protsenko, J. Schütte, S. Wessel, An ecosystem and Iot device architecture for building trust in the industrial data space, in: Proceedings of the 4th ACM Workshop on Cyber-Physical System Security (CPSS'18). ACM, 2018, pp. 39–50.
[10] A. Celesti, M. Fazio, M. Giacobbe, A. Puliafito, M. Villari, Characterizing cloud federation in Iot, in: Proceedings of the IEEE 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2016, pp. 93–98.
[11] A. Celesti, D. Mulfari, M. Fazio, M. Villari, A. Puliafito, Exploring container Virtualization in Iot clouds, in: Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP), 2016, pp. 1–6.
[12] H. Chen, X. Jia, H. Li, A brief introduction to Iot gateway, in: Proceedings of the IET International Conference on IET Communication Technology and Application (ICCTA), 2011, pp. 610–613.
[13] M.N. Chowdhury, M.S. Nooman, S. Sarker, Access control of door and home security by raspberry pi through internet, Int. J. Sci. Eng. Res 4 (2013) 550–558.
[14] Air Quality Egg 2017 Air Quality Egg, 2017, https://airqualityegg.wickeddevice.com/.
[15] I. Ganchev, J. Zhanlin, M. O'Droma, A Generic IoT Architecture for Smart Cities, in: Proceedings of the IET Irish Signals & Systems Conference (ISSC) and China-Ireland International Conference on Information and Communications Technologies (CIICT), 2014, pp. 196–199.
[16] M. Garcí-Valls, J. Ampuero-Calleja, L.L. Ferreira, Integration of data distribution service and raspberry Pi, in: Proceedings of the International Conference on Green, Pervasive, and Cloud Computing. Springer, 2017, pp. 490–504.
[17] A. Gaur, B. Scotney, G. Parr, S. McClean, Smart city architecture and its applications based on iot, Proc. Comput. Sci. 52 (2015) 1089–1094.
[18] N. Gondchawar, R.S. Kawitkar, Iot based smart agriculture, Int. J. Adv. Res. Comput. Commun. Eng. (IJARCCE) 5 (6) (2016) 177–181.
[19] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): a vision, architectural elements, and future directions, Future Generat. Comput. Syst. 29 (7) (2013) 1645–1660.
[20] D. Guinard, V. Trifa, Building the Web of Things: With Examples in Node. js and Raspberry Pi, Manning Publications Co., 2016.
[21] M.S.D. Gupta, V. Patchava, V. Menezes, Healthcare based on IoT using raspberry Pi, in: Proceedings of the IEEE International Conference on Green Computing and Internet of Things (ICGCIoT), 2015, pp. 796–799.
[22] M.R. Hera, A. Rahman, A. Afrin, M.Y.S. Uddin, N. Venkatasubramanian, AQBox: an air quality measuring box from COTS gas sensors, in: Proceedings of the International Conference on Networking, Systems and Security (NSysS), IEEE, Dhaka, Bangladesh, 2017, pp. 191–194.
[23] H. Hong, Y. Uddin, P. Tsai, A. Cheng, N. Venkatasubramanian, C. Hsu, Supporting internet-of-things analytics in a fog computing platform, in: Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong-Kong, 2017.
[24] H.-J. Hong, P.-H. Tsai, C.-H. Hsu, Dynamic module deployment in a fog computing platform, in: Proceedings of the 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), IEEE, Kanazawa, Japan, 2016, pp. 1–6.
[25] H.-C. Hsieh, C.-H. Lai, Internet of things architecture based on integrated Plc and 3G communication networks, in: Proceedings of the IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS), 2011, pp. 853–856.
[26] M. Ibrahim, A. Elgamri, S. Babiker, A. Mohamed, Internet of things based smart environmental monitoring using the raspberry-Pi computer, in: Proceedings of the IEEE Fifth International Conference on Digital Information Processing and Communications (ICDIPC), 2015, pp. 159–164.
[27] A. Imteaj, T. Rahman, M.K. Hossain, S. Zaman, IoT based autonomous percipient irrigation system using raspberry Pi, in: Proceedings of the 19th International Conference on IEEE Computer and Information Technology (ICCIT), 2016, pp. 563–568.
[28] S.M.R. Islam, D. Kwak, M.D.H. Kabir, M. Hossain, K.-S. Kwak, The internet of things for health care: a comprehensive survey, IEEE Access 3 (2015) 678–708.
[29] K.-J. Jeong, W.-J. Kim, The implementation of smart raising environment management system based on sensor network and 3G telecommunication, J. Korea Inst. Electron. Commun. Sci. 6 (4) (2011) 595–601.
[30] P. Jutadhamakorn, T. Pillavas, V. Visoottiviseth, R. Takano, J. Haga, D. Kobayashi, A scalable and low-cost MQTT broker clustering system, in: Proceedings of the IEEE 2nd International Conference on Information Technology (INCIT), 2017, pp. 1–5.
[31] S.D.T. Kelly, N.K. Suryadevara, S.C. Mukhopadhyay, Towards the implementation of IoT for environmental condition monitoring in homes, IEEE Sens. J. 13 (10) (2013) 3846–3853.

[32] A. Khanna, R. Anand, IoT based smart parking system, in: Proceedings of the IEEE International Conference on Internet of Things and Applications (IOTA), 2016, pp. 266–270.
[33] J.-W. Kim, A smart home prototype implementation using raspberry Pi, J. Korea Inst. Electron. Commun. Sci. 10 (10) (2015) 1139–1144.
[34] M.S. Kiran, P. Rajalakshmi, K. Bharadwaj, A. Acharyya, Adaptive rule engine based IoT enabled remote health care data acquisition and smart transmission system, in: Proceedings of the IEEE World Forum on Internet of Things (WF-IoT), 2014, pp. 253–258.
[35] M. Koshti, S. Ganorkar, L. Chiari, Iot based health monitoring system by using raspberry pi and ECG signaly, Int. J. Innovat. Res. Sci. Eng. Technol. 5 5 (2016).
[36] R. Kumar, M.P. Rajasekaran., An Iot Based Patient Monitoring System Using Raspberry Pi, in: Proceedings of the IEEE International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE), 2016, pp. 1–4.
[37] S. Kumar, A. Jasuja, Air Quality Monitoring System Based on Iot Using Raspberry Pi, in: Proceedings of the International Conference on Computing, Communication and Automation (ICCCA), 2017, pp. 1341–1346. https://doi.org/10.1109/CCAA.2017.8230005.
[38] X.-Q. Li, X. Ding, Y. Zhang, Z.-P. Sun, H.-W. Zhao, Iot Family Robot Based on Raspberry Pi, in: Proceedings of the IEEE International Conference on Information System and Artificial Intelligence (ISAI), 2016, pp. 622–625.
[39] P. Maiti, B. Sahoo, A.K. Turuk, S. Satpathy, Sensors data collection architecture in the internet of mobile things as a service (ioMTaas) platform, in: Proceedings of the IEEE International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2017, pp. 578–582.
[40] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, . Branko Perišić, Raspberry Pi as internet of things hardware: performances and constraints, Des. Issues 3 (2014) 8.
[41] K. Mandula, R. Parupalli, C.H.A.S. Murty, E. Magesh, R. Lunagariya, Mobile based home automation using internet of things (IoT), in: Proceedings of the IEEE International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2015, pp. 340–343.
[42] M. Roberto, A Performance Evaluation of Container Technologies on Internet of Things Devices, in: Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2016, pp. 999–1000.
[43] M. Roberto, Virtualization on internet of things edge devices with container technologies: a performance evaluation, IEEE Access 5 (2017) 8835–8850.
[44] R. Morabito, N. Beijar, Enabling data processing at the network edge through lightweight virtualization technologies, in: Proceedings of the IEEE International Conference on Sensing, Communication and Networking (SECON Workshops), IEEE, London, Uk, 2016, pp. 1–6.
[45] R. Morabito, V. Cozzolino, A.Y. Ding, N. Beijar, J. Ott, Consolidate IoT edge computing with lightweight virtualization, IEEE Netw. 32 (2018) 102–111.
[46] R. Morabito, I. Farris, A. Iera, T. Taleb, Evaluating performance of containerized IoT services for clustered devices at the network edge, IEEE Internet Things J. 4 (2017) 1019–1030.
[47] R. Morabito, R. Petrolo, V. Loscri, N. Mitton, LEGIOt: a lightweight edge gateway for the internet of things, Future Generat. Comput. Syst. 81 (2018) 1–15.
[48] R. Morabito, R. Petrolo, V. Loscri, N. Mitton, G. Ruggeri, A. Molinaro, Lightweight virtualization as enabling technology for future smart cars, in: Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 1238–1245.
[49] Y. Nakamura, H. Suwa, Y. Arakawa, H. Yamaguchi, K. Yasumoto, Design and implementation of middleware for IoT devices toward real-time flow processing, in: Proceedings of the IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2016, pp. 162–167.
[50] C.S. Nandyala, H.-K. Kim, Green iot agriculture and healthcare application (GAHA) 2016, Int. J. Smart Home 10 (4) (2016) 289–300.
[51] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, S. Dustdar, Provisioning Software-defined Iot Cloud Systems, in: Proceedings of the IEEE 2nd International Conference on Future Internet of Things and Cloud (FiCloud), 2014, pp. 288–295.
[52] OpenSignal, Global State of Mobile Networks, 2016,. https://opensignal.com/reports/2016/08/global-state-of-the-mobile-network/.
[53] C. Pahl, S. Helmer, L. Miori, J. Sanin, B. Lee, A Container-based edge cloud paas architecture based on raspberry Pi clusters, in: Proceedings of the IEEE International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), IEEE, Vienna, Austria, 2016, pp. 117–124.
[54] C. Pahl, B. Lee, Containers and clusters for edge cloud architectures–a technology review, in: Proceedings of the 3rd International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, Rome, Italy, 2015, pp. 379–386.
[55] V. Patchava, H.B. Kandala, P.R. Babu, A smart home automation technique with raspberry Pi using IoT, in: Proceedings of the IEEE International Conference on Smart Sensors and Systems (IC-SSS), 2015, pp. 1–4.
[56] D. Pavithra, R. Balakrishnan, IoT based monitoring and control system for home automation, in: Proceedings of the IEEE Global Conference on Communication Technologies (GCCT), 2015, pp. 169–173.
[57] P. Pyykönen, J. Laitinen, J. Viitanen, P. Eloranta, T. Korhonen, IoT for intelligent traffic system, in: Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2013, pp. 175–179.
[58] P.B. Rao, S.K. Uma, Raspberry pi home automation with wireless sensors using smart phone, Int. J. Comput. Sci. Mob. Comput. 4 (5) (2015) 797–803.
[59] D. Trihinas, G. Pallis, M.D. Dikaiakos, Adam: An Adaptive Monitoring Framework for Sampling and Filtering on IoT Devices, in: Proceedings of the IEEE International Conference on Big Data (Big Data), 2015, pp. 717–726.
[60] P.-H. Tsai, Distributed analytics in fog computing platforms using Tensorflow and Kubernetes, in: Proceedings of the Asia-Pacific Network Operations and Management Symposium (APNOMS), Springer, Seoul, Korea, 2017, pp. 1–6.
[61] M.Y.S. Uddin, A. Nelson, K. Benson, G. Wang, Q. Zhu, Q. Han, N. Alhassoun, P. Chakravarthi, J. Stamatakis, D. Hoffman, The Scale2 Multi-network architecture for IoT-based resilient communities, in: Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP), IEEE, St. Louis, Missouri, 2016, pp. 1–8.
[62] P.A. Shinde, Y.B. Mane, Advanced vehicle monitoring and tracking system based on raspberry Pi, in: Proceedings of the IEEE 9th International Conference on Intelligent Systems and Control (ISCO), 2015, pp. 1–6.
[63] S. EnviroSCALE Trace, 2018, https://github.com/EnviroSCALE/infrastructure-level-adaptation.
[64] S. Souranil, B. Aruna, A highly resilient and scalable broker architecture for IoT applications, in: Proceedings of the 10th International Conference on Communication Systems and Networks (COMSNETS). IEEE, 2018, pp. 3–7.
[65] F. TongKe, 2013. Smart agriculture based on cloud computing and IOT, J. Converg. Inf. Technol. 8 (2) (2013).
[66] P. Ravindra, A. Khochare, S.P. Reddy, S. Sharma, P. Varshney, Y. Simmhan, ECHO: an adaptive orchestration platform for hybrid dataflows across cloud and edge, in: Proceedings of the International Conference on Service-Oriented Computing, Springer, 2017, pp. 395–410.
[67] T. Renner, M. Meldau, A. Kliem, Towards Container-based Resource Management for the Internet of Things, in: Proceedings of the IEEE International Conference on Software Networking (ICSN), 2016, pp. 1–5.
[68] M. Sajjad, M. Nasir, K. Muhammad, S. Khan, Z. Jan, A.K. Sangaiah, M. Elhoseny, S.W. Baik, Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities, Future Generat. Comput. Syst. (2017).
[69] D. Shah, IoT based biometrics implementation on raspberry Pi, Proc. Comput. Sci. 79 (2016) 328–336.
[70] R. Shete, S. Agrawal, IoT based urban climate monitoring using raspberry Pi, in: Proceedings of the Communication and Signal Processing (ICCSP), 2016 International Conference on. IEEE, 2016, pp. 2008–2012.
[71] M. Vecchio, R. Giaffreda, F. Marcelloni, Adaptive lossless entropy compressors for tiny Iot devices, IEEE Trans. Wireless Commun. 13 (2) (2014) 1088–1100.
[72] A. Velasco, R. Ferrero, F. Gandino, B. Montrucchio, M. Rebaudengo, T.E. Simos, Z. Kalogiratou, T. Monovasilis, On the design of distributed air quality monitoring systems, Proceedings of the AIP Conference 1702(1) (2015) 180014.
[73] N. Vijayakumar, R. Ramya, The real time monitoring of water quality in IoT environment, in: Proceedings of the International Conference on IEEE Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015, pp. 1–5.
[74] V. Vujović, M. Maksimović, Raspberry pi as a sensor web node for home automation, Comput. Electr. Eng. 44 (2015) 153–171.
[75] WHO., Ambient and Household Air Pollution and Health, 2016,. http://www.who.int/phe/health_topics/outdoorair/databases/en/.
[76] Y.-e. Duan, Design of intelligent agriculture management information system based on IoT, in: Proceedings of the International Conference on IEEE Intelligent Computation Technology and Automation (ICICTA), vol. 1, 2011, pp. 1045–1049.
[77] L. Yin, J. Luo, H. Luo, Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing, IEEE Trans. Ind. Inf. 14 (2018) 4712–4721.
[78] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, 2014, Internet of things for smart cities, IEEE Internet Things J. 1 (1) (2014) 22–32.
[79] C.W. Zhao, J. Jegatheesan, S.C. Loon, 2015, Exploring iot application using raspberry pi, International Journal of Computer Networks and Applications 2 (1) (2015) 27–34.
[80] J.-c. Zhao, J.-f. Zhang, Y. Feng, J.-x. Guo, The Study and Application of the IOT Technology in Agriculture, in: Proceedings of the Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on. IEEE, volume 2, 2010, pp. 462–465.
[81] K. Zheng, S. Zhao, Z. Yang, X. Xiong, W. Xiang, 2016, Design and implementation of LPWA-based air quality monitoring system, IEEE Access 4 (2016) 3238–3245.
[82] Q. Zhu, R. Wang, Q. Chen, Y. Liu, W. Qin, Iot Gateway: Bridgingwireless Sensor Networks into Internet of Things, in: Proceedings of the Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on. Ieee, 2010, pp. 347–352.

Mahmudur Rahman Hera obtained his B.Sc. degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Mahmudur is now serving as a lecturer in the same university. Topping his class throughout all the eight terms of B.Sc. studies, Mahmudur has received Dean's List Scholarship and University Merit List Scholarship. For his excellent academic records, he is to receive Department Gold Medal and Faculty Gold Medal in the next convocation. He has also received Prime Minister Gold Medal Award 2017, a prestigious recognition offered by the Prime Minister of Bangladesh.

Amatur Rahman was born in Dhaka, Bangladesh. She obtained her Bachelor degree on Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET) in the year 2017. She had been awarded with Board Scholarship, Dean's List Award and University Merit Scholarship. She is now serving as a lecturer at the Department of Computer Science and Engineering at United International University in Bangladesh.

Hua-Jun Hong is a PhD student at the National Tsing Hua University, Taiwan. He received his B.S. and M.S. in Computer Science from National Tsing Hua University. His research interests are in fog computing, cloud gaming, and multimedia networking.



Li-Wen Pan is a senior undergraduate student at National Tsing Hua University. Her research interests are in Internet-of-Things, embedded systems, and software-defined networks.

Md Yusuf Sarwar Uddin is an Associate Professor at Computer Science and Engineering of Bangladesh University of Engineering and Technology (BUET). He is a postdoctoral fellow at University of California, Irvine. He completed his PhD from Department of Computer Science at the University of Illinois at Urbana-Champaign, USA. His research interest includes distributed computing systems, mobile computing, wireless and sensor networks and distributed cyber-physical systems.

Nalini Venkatasubramanian is a Professor of Computer Science in the Donald Bren School of Information and Computer Sciences at the University of California, Irvine. Born and raised in Bangalore, she received her Ph.D. in Computer Science from the University of Illinois, Urbana-Champaign in 1998. From 1991 to 1998, she was a member of technical staff and software designer engineer for Hewlett-Packard. In 1998, she joined UC Irvine as an Assistant Professor of Computer Science. Her research interests are Multimedia Computing, Networked and Distributed Systems, Internet technologies and Applications, Ubiquitous Computing and Urban Crisis Responses. Dr. Venkatasubramanian's research focuses on enabling effective management and utilization of resources in the evolving global information infrastructure. She also addresses the problem of composing resource management services in distributed systems.

Cheng-Hsin Hsu (S'09–M'10–SM'16) received the B.Sc. degree in Mathematics and M.Sc. degree in Computer Science and Information Engineering from National Chung-Cheng University, Taiwan. He received the M.Eng. degree in Electrical and Computer Engineering from the University of Maryland, College Park and the Ph.D. degree in Computing Science from Simon Fraser University, Burnaby, BC, Canada. He is an Associate Professor at National Tsing Hua University at Hsin Chu, Taiwan. Before that, he was with Deutsche Telekom, Motorola Inc., and Lucent Technologies. His research interests include multimedia networking, mobile computing, and computer networks.