Real-time Distributed In-Situ Benchmarking of Energy Harvesting IoT Devices

Ashok Samraj Thangarajan
imec-DistriNet, KU Leuven
ashoksamraj.thangarajan@cs.kuleuven.be

Fan Yang
imec-DistriNet, KU Leuven
fan.yang@cs.kuleuven.be

Wouter Joosen
imec-DistriNet, KU Leuven
wouter.joosen@cs.kuleuven.be

Danny Hughes
imec-DistriNet, KU Leuven
danny.hughes@cs.kuleuven.be

ABSTRACT

The deployment of Internet of Things (IoT) devices is accelerating across a wide range of applications. The majority of today's IoT devices are powered by batteries that can operate for a maximum of a few years, after which they need to be replaced. This introduces two problems. First, the effort that is required to manually replace batteries cannot economically scale to support the next billion IoT devices. Secondly, treating billions of toxic batteries as disposable is not environmentally friendly. Together, these problems form a critical road-block in deploying IoT solutions. The biggest problem facing the designers of IoT applications is ensuring that their application software is energy efficient enough to operate within the strict power envelope that is provided by batteries or energy harvesting hardware. In this paper, we tackle this problem through the introduction of a distributed benchmarking middleware that rapidly and accurately quantifies the power consumption of different software configurations. Critically, our middleware operates in real-time across a distributed network of devices, allowing developers to experiment with code changes at runtime. This makes it significantly easier for developers to write applications that operate within the power constraints of batteries or energy harvesting systems. We evaluate our approach on a real world energy harvesting testbed and demonstrate that benchmarking results are accurate, with limited overhead for developers.

CCS CONCEPTS

• **Networks** → **Network measurement**; *Middle boxes / network appliances*; *Network performance analysis*; Network manageability;

KEYWORDS

Internet of Things (IoT), Energy Harvesting, Benchmarking, Self Adaptive Networks, Industrial IoT Networks

M4IoT'18, December 10–11, 2018, Rennes, France

ACM ISBN 978-1-4503-6118-7/18/12…$15.00
https://doi.org/10.1145/3286719.3286724

ACM Reference Format:
Ashok Samraj Thangarajan, Fan Yang, Wouter Joosen, and Danny Hughes. 2018. Real-time Distributed In-Situ Benchmarking of Energy Harvesting IoT Devices. In *5th Workshop on Middleware and Applications for the Internet of Things (M4IoT'18), December 10–11, 2018, Rennes, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3286719.3286724

1 INTRODUCTION

The adoption of IoT across domains such as smart cities, e-health and manufacturing is occurring at an estimated rate of 127 devices every second [20]. Analysts claim that in the year 2020, the number of deployed IoT devices will exceed 20 billion [1]. The rapid penetration of the IoT has increased comfort with smart devices, enhanced safety in medical and automotive domains, and improved efficiency in industries with predictive and condition based maintenance, thus delivering great economic benefits [18].

The majority of IoT devices are estimated to last for between a few months and a year [1]. Energy harvesting IoT devices offer great potential in this space. It is therefore critical that IoT application developers are supported in measuring energy efficiency. On the platform side, S. Sudevalayam et al. [28] categorized the energy harvesting platforms into two major types namely, *(i) Harvest-Use* and *(ii) Harvest-Store-Use.* Although some IoT devices work based on a single hop star routing approach, many IoT deployments (e.g.industrial IoT solutions) require the end nodes to be part of a network which performs muti-hop routing. Such kind of networks, cannot rely on *Harvest-Use* architectures, because of the uncontrollable and unpredictable nature of most energy harvesting sources. They instead use a storage medium like battery or a supercapacitor [22, 26, 27, 29] and sometimes even a multi-tier energy storage architecture [12, 19]. It is proven in this research landscape, that the best way to get tens of years of deploy-and-forget IoT networks, in an environmentally friendly way [31] is by using supercapacitors and a multi-tier energy storage architecture to ensure reliable energy supply to the IoT devices. This introduces high dynamism and many more parameters that effect the system. Given the architectural complexity of multi-tier storage systems, the benchmarking scheme should be able to handle this complexity.

Most of the IoT systems are distributed in nature, and their network and device specific parameters are distributed across the entire network. All of these parameters effect the way in which the networks perform. Moreover, reproducing the uncontrollable environment dynamism in a lab environment is very difficult. To benchmark such a system, all of the parameters must be made

available to the user and the benchmarking should be able to deliver real-time results from field deployments.

It is in fact common that after IoT networks are deployed, their field applications are expanded through over the air (OTA) updates. Although developers test the applications during development, it is difficult to say that the performance of the application will be the same on a field-deployment. There are problems similar to those explained in *Tragedy of the Coulombs* [7], were a specific component like reading a sensor or a specific behaviour in the software, which when triggered by field specific events, turns out to be power hungry exhausting all the available energy. This makes it essential to benchmark the software after deployment, in the field and on a real-time basis, for application specific parameters.

In this paper, we treat all three of the problems discussed above by introducing a middleware that offers the: (i) ability to handle complex energy harvesting architectures, (ii) benchmarking for distributed parameters across the network in real-time, (iii) benchmarking during development and after OTA updates in field deployments. The framework acquires network parameters that are distributed across the end-points and delivers it for analysis, in tandem with the network information available in the gateways. The output of this acquisition is a set of modifiable network parameters that can be assigned to the end-points without affecting the network stability. The framework then gathers statistical data on the energy profile of the devices against various network parameters.

The remainder of the paper is organized as follows, in the next Section 2 we discuss related work and highlight the gap that this framework fulfills. Section 3 provides a high level architecture of the framework. Design and implementation of the framework is discussed in Section 4. Section 5 presents the evaluation platform and a use case study from a real-world deployment. In Section 6, we discuss two areas of research that we intent to focus on following this work. Finally, Section 7 presents concluding remarks.

2 RELATED WORK

Kruger et al. [13] present a micro and macro benchmarking scheme in their paper, however this was done to evaluate the suitability of commercial components such as Raspberry Pi and BeagleBone as IoT gateways. A benchmarking of IoT middlewares were done by Cardoso et al. [4] which facilates selection of appropriate IoT middleware that is suitable for a particular deployment. In contrast, we focus on enabling development and network maintenance team with better benchmarking tools. Hardware-level power analysis [15], focusses on components of CPU's and not on the network as a whole. Simulators for TinyOS called TOSSIM [17] and its extension powerTOSSIM [25] were presented, together they can estimate energy consumption using an abstract model of a Mica2 sensor node running on x86. Titzer et al. [30] presented Avrora, a simulator that can simulate sensor network code at machine level like ATEMU [21] and at the same time scalable like TOSSIM. AEON [16] builds upon Avrora and delivers energy consumption of the node and the network based on real development code. Gaglione et al. [6] demonstrated developing a battery-free kinetic energy harvester, based on vibration data collected from a bridge and simulating and benchmarking in a laboratory environment using the collected data. Ritter et al. [23] use supercapacitors and software instrumentation to estimate lifetime of end-nodes in a laboratory set up and compare it against previously developed mathematical models.

Some of the solutions discussed above delivers very fine grained results, but most of these are simulations. Few others, were done on a laboratory set up, or with previously collected model data of the energy harvesting environment. We believe that, after the initial deployments a component level energy profiling on individual nodes is unnecessary. What we consider as a necessity for developers and network maintenance teams is a tool for evaluating their changes in-situ, and deliver real-time results from continuous testing with minimal effort and time. Additionally, by enabling real-time evaluation on field deployments, the framework facilitates Agile [2] like development methodology, with continuous delivery of application software on end-nodes with field test results, after every newly deployed firmware.

3 FRAMEWORK ARCHITECTURE

Architectures that facilitate a generic approach to energy measurement on IoT networks that support energy harvesting end-nodes must meet three guidelines: (i) Able to measure energy consumption against local and global network parameters, end-node parameters and application specific parameters (ii) Able to benchmark end-nodes on live deployments remotely without degrading performance and (iii) Able to handle complexities that arise from multi-tier energy harvesting architectures. We have ensured that the guidelines are followed by the architecture of the framework and the delivered result covers all the parameters that impacts energy consumption, some of which shall be discussed in the evaluation of the framework.

3.1 Sub-Component Architecture

Our design, targets a traditional three layer IoT platform, that includes end-nodes for sensing and actuation, gateways to channel the messages from end-nodes to the back-end, and a back-end platform for data storage and analysis. The end-nodes and gateways are linked by some form of wireless technology, which may range from LPWAN links like LoRa and SigFox to short range wireless links like Bluetooth, ZigBee, Smart Mesh IP. In such a three layered IoT network, we assume that there are a wide range of network parameters, distributed across the end-nodes and the gateway. It is a co-operation of all the parameters that maintains these networks optimally. In the case of energy harvesting, it is essential that all of these parameters are better understood, and are made available at a central point on this network for monitoring and good control. This framework delivers upon all of these requirements with a multi-tier architecture as seen in Figure 1.

3.1.1 ***Tier-I: End Nodes.*** The first tier represents the end-nodes, that holds local network parameters, along with some device specific parameters that can be tuned to deliver better energy effeciency. In this tier, we have designed three components on top of the network stack and the device configurations using the SDK.
(a) Network Control Application (NCA)
(b) Sensor Data Processor (SDP)
(c) Control Interface (CI)
The *Network Control Application* establishes the initial connection and is responsible for maintaining the connection. The *Sensor Data*

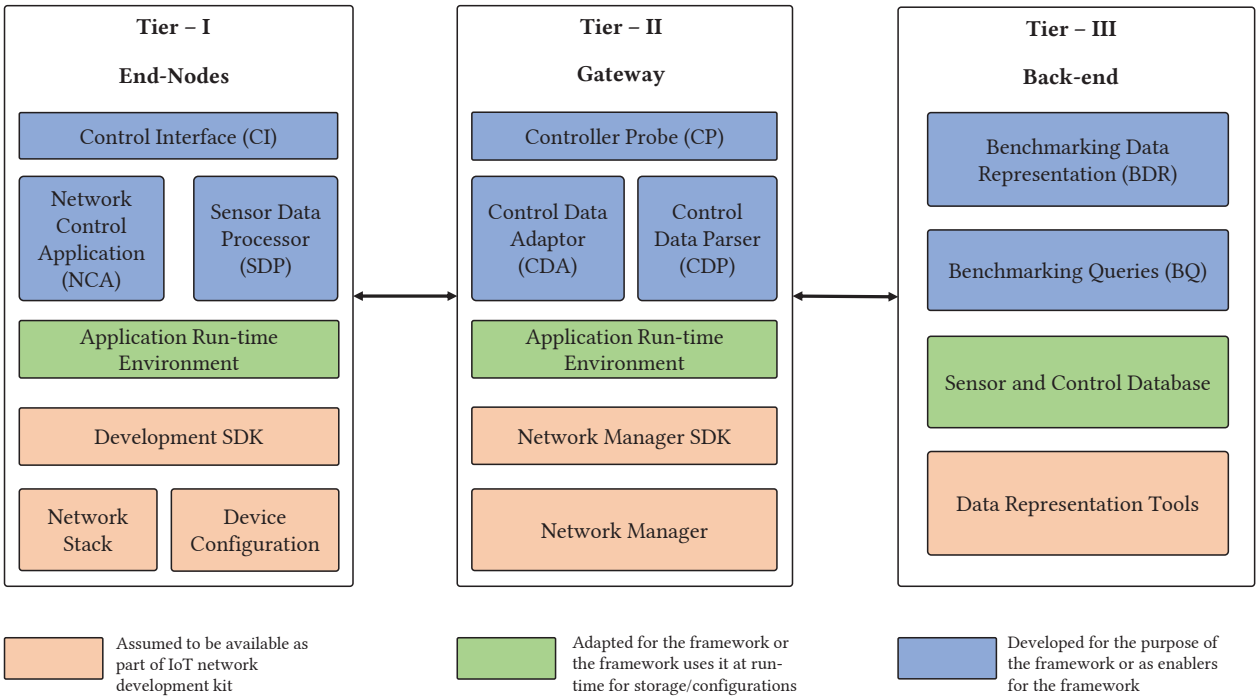


Figure 1: Architecture of the in-situ energy benchmarking framework.

Processor is an enabler in the framework to deliver energy related sensor information (battery usage, current battery level, temperature etc.), when it receives commands from the *Control Interface*. The *Control Interface* is the manager of the framework from end-nodes side, that provides a set of services to other external interfaces. External interface in the context of *Control Interface* are other sub-modules deployed within the end-node and devices accessing its services through the network (e.g. gateway). This provides flexibility in the framework to either control the benchmarking from within the end-nodes or from the gateway. However, running the complete benchmarking on the end-nodes incurs energy cost, that has to be kept low and as a known bounded value. Considering this aspect, the three tier solution is established, as a result of which the energy cost of running the benchmarking itself is kept negligible, and the cost incurred in transmitting benchmarking data is used for calculating the energy profile.

3.1.2 ***Tier-II: Gateway.*** The second tier consists of
(a) Controller Probe (CP)
(b) Control Data Parser (CDP)
(c) Control Data Adaptor (CDA)
The following was taken in to consideration when designing the *Controller Probe* (i) Has the ability to exploit global parameters of the network (ii) Has the capability to query and manipulate local network and devices parameters of the end-node (iii) Run the control logic to generate the benchmarking parameters in tandem with backend. The *Control Data Parser* as the name suggests, basically is a parser that receives the data from the network and converts it to a format understandable by the backend. The *Control Data Adaptor* bundles data and uploads it to the database.

3.1.3 ***Tier-III: Back-end.*** The third tier has two submodules, which are:
(a) Benchmarking Queries (BQ)
(b) Benchmarking Data Representation (BDR)
Benchmarking Queries exposes a set of API's, using which a set of queries are run over the data from Sensor and Control Database. The *Benchmarking Data Representation* is essentially a data representation module that shows the transformed data, that are queried using the *Benchmarking Queries*, in the form of charts and graphs that enables better data comparison methods. It is important to note that the *Benchmarking Queries* and *Benchmarking Data Representation* works together to present the data in an user-understandable format. Hence, these two modules can be designed or selected in such a way that a single module transforms and presents the data. This layer is also designed in such a way that it features a plugin-like capability. To enable this, the data is posted using Message Queuing Telemetry Transport (MQTT), streamed via webserver and REST API's are exposed for the plugin's to query for information. If the user wants to do some form of transformation of the data (e.g. forecasting) in real-time or at a later stage, this can be done by writing their own transformation functions over data received from one of the methods given above. This feature is exploited in Section 5 for an usecase study.

4 DESIGN OF THE FRAMEWORK

4.1 Benchmarking Application Software

For the purpose of this design and implementation, let's assume EN_i is the end-node instance and GW_j is the gateway instance, that

runs their respective portions of the benchmarking software. The description below follows the modules depicted in Figure 1.

4.1.1 ***Tier-I.*** Consider EN_i is powered up for the first time, it starts searching for the network advertisements beacons from the GW. Once a network is identified, the NCA triggers a join request that results in the network stack, negotiating an optimal connection and placement of the EN_i in the network. This network is continually optimized by the GW during its lifetime, to achieve best in class performance. The NCA is also responsible for re-negotiating the connection, in case of a connection drop.

The CI was implemented as a request-response based message exchange entity, which delivers a set of services that can be used by sub-modules within the EN_i or by other network entities like the GW. The CI delivers good control on individual parameters and do not directly work on completing a request. That responsibility lies with either the sub-modules or the GW (as in our case) whoever is weilding control of EN_i through the CI.

The SDP is exercised by the CI to sample sensor data, at the rate in which CI needs, packages data into a format requested by CI and pushes it into the network or a specific target set by the CI.

4.1.2 ***Tier-II.*** The CP in GW_j interfaces with the CI of the EN_i for control and data to manage the benchmarking framework. CP configures the framework according to user requirements and commands the CI to deliver relevant data to the framework.

When EN_i receives command from CP to transmit data, it asynchronously transmits the data, which is received by the CDP and parsed in such a way that the data can be loaded on to a database. Based on whether the data is to be used for benchmarking or for decision making in the benchmarking process, the data is either pushed to a database or passed to CP. The CP receives data and adjusts the benchmarking parameters to closely match the user requirements.

The parsed data from the CDP is passed to the CDA. The task of the CDA is to package this data and pass it to the backend in an interface that is required by the server.

4.1.3 ***Tier-III.*** The CDA pushes data from EN_i to the database. Specific queries are written to pull relevant measurement data from the database to be represented graphically. These queries are represented by the BQ and graphical representation is delivered by the BDR. Additionally, as discussed in the Section 3.1.3, this layer features a plugin architecture. To exploit this, we intent to force the system to work only on supercapacitor, and write a plugin to forecast the data for the rest of operating range of the hardware, with a minimum training data set obtained from the voltage drop of capacitor. However, users of the framework, can either use the queries within the BQ, or can develop their own plugin, or they can use both for analysis.

The raw value from the capacitor drop over time will have lot of residuals. So an initial smoothing of the value has to be done before the data is used by the plugin. To keep this initial smoothing simple, we did a moving average over a specific window that we calculated, the proof of which, is as follows. Given a sequence:

$$\{X_i\}_{i=1}^{N}, \qquad \text{N = number of data points}$$

Moving average of the sequence defined with a window n is

$$\{S_i\}_{i=1}^{(N-n)+1}$$

$$\{S_i\} = \frac{1}{n}\sum_{j=1}^{i+n-1} X_j$$

So the moving average sequence is represented as:

$$S_n = \{\frac{1}{n}\sum_{j=i}^{i+n-1} X_j\}_{i=1}^{(N-n)+1}$$

The window n of simple moving average is obtained from the equation

$$n = \{max(X_{i+1}, X_i)\}_{i=1}^{N}$$

4.2 Achieving the Design Objectives

In the architecture section, we discussed three guidelines behind which this framework is designed. Here we will take these guidelines one-by-one and analyze how these objectives are achieved by the framework.

The CI and CP discussed in Sections 4.1.1 and 4.1.2 respectively, is able to query and modify device specific, local and global network parameters on behalf of sub-modules within the IoT device or from other devices that are in the same network. The CP and CI does this in such a way, so as to not threaten the stability of the network, thus delivering an in-situ benchmarking capability with minimal support required from the user of the benchmarking framework. Together with this and an algorithm to sequencially modify these parameters and generate energy profile of the devices, we achieve the two of the three design objectives. To handle complexities arising from multi-tier energy harvesting architectures, the middleware parameterizes all the power states, the operating mode, conditions and capacity of both the supercapacitor and the battery. By doing this all of the parameters that affects the network even in a complex architecture, can be benchmarked and monitored remotely.

5 EVALUATION PLATFORM AND USE CASE STUDY

5.1 Evaluation platform

5.1.1 ***Hardware.*** We selected the representative LTC5800 from ADI [10] as our IoT end-node. The LTC5800 is an IETF Class-1 device [3] which is implemented as a System-on-Chip (SoC) that integrates a low power 2.4 GHz IEEE 802.15.4e radio with a 20MHz ARM Cortex-M3 32-bit microprocessor, with 12KB RAM and 32KB flash memory available for applications. The LTC5800-IPM consumes 800 nA in deep-sleep mode, 1.3 mA in active mode, 4.5 mA when receiving a packet and 9.7 mA when transmitting a packet. For the energy harvesting, a two staged storage approach was followed with a super-capacitor as primary supply and a battery for back-up. The battery output is regulated through a coulomb counter [9], which is used to measure the charge consumed, when the microcontroller is powered by battery. An analog multiplexer [8], that is controlled by a sub-module of the software framework, provides the following capabilities: (i) Switch on/off charging via solar panel (ii) Switch from battery to supercapacitor and vice-versa (iii) Measure the supercapacitor charge using the ADC when its not powering the board. For the gateway, we used the same LTC5800 in network manager mode, connected it via UART to an Intel Edison platform [5].

5.1.2 ***Basic Software.*** The LTC5800 EN is a 6LoWPAN device, which supports IPv6 and communicates with the gateway using the

Constrained Application Protocol (CoAP) over User Datagram Protocol (UDP). The end-node runs SmartMesh-IP stack over Micrium's μCOS-II real-time operating system on the ARM Cortex-M3 core. The application on the end-node side, that realises the framework runs on this chip. On the GW side, the Intel Edison platform runs embedded Linux, over which the control algorithms are developed. For the backend, we configured a server with a time series database influxDB [11] to store data and Grafana [14], an open source time series analytics platform as the graphical representation tool.

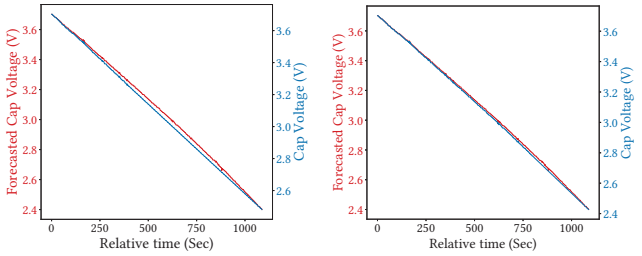
5.2 Case Study

For the evaluation, we built a network that has 35 end-nodes connected to a gateway. Two of the nodes were energy harvesting nodes, that we ran our experiments on, and the remaining nodes provide scale. The end-nodes were deployed across three floors of a five floor building. We evaluated our benchmarking framework based on its ability to quantify changes to the following parameters:
(a) With default parameters
(b) With routing off (default on)
(c) With join duty cycle as 255 (default 64)
(d) With antenna gain 8 (default 2)

For the purpose of evaluation, packets that send the voltage information to the gateway were used to simulate sensor data sampling rate. For all the experiments the sampling rate was set to 2 seconds. Every experiment was run from a voltage drop in capacitance from 3.7V to 2.4V. Below 2.4V there were inconsistencies in the way different hardwares went to poweroff due to low voltage. Hence a lower bound of 2.4V was selected to ensure consistency across all the measurements.

For the case study, we developed a plugin for the framework, that provides a way to forecast the data with a minimal data set. To achieve this, first we forced the system to operate on the supercapacitor, then we monitored the voltage drop due to the change in parameters mentioned above. We then took a minimal data set and forecasted the rest of the data. By doing this, we can reduce the time and hence drastically reduce the duration over which we run benchmarking. Selection of the forecasting algorithm is more of an empirical method, the explanation is as follows. It is important to note that, using the framework's plugin architecture, the user can select any data transformation method to work out their needs. We tried forecasting using linear regression, gaussian process regression, however the below approach seems to best suit our data set.

To determine the forecasting algorithm, we ran experiments with a sample application that sends sensor data every 10 seconds. Along with the sensor data, we packed the capacitor voltage, this ensures there is not much overhead for sending energy data, other than the additional five bytes of voltage and an one byte identifier. We used the BQ and BDR to plot the capacitor voltage drop over time. The capacitor voltage exhibited a downward trend and an additive seasonality, that were reminiscent of Holt-Winters (HW) exponential smoothing [24]. For the α, β and γ values of the HW forecasting, we allowed the algorithm to select the values, to yield best results.



(a) Training Data 200 **(b) Training Data 400**

Figure 2: Forecasts with Two Different Training Data Ranges.

End-Node	Modified Parameter	MAPE
EN1	Default Parameters	0.49576
	Routing OFF	1.03798
	Join Duty Cycle = 255	0.53191
	Antenna Gain = 8	1.01371
EN2	Default Parameters	0.77386
	Routing OFF	0.60651
	Join Duty Cycle = 255	1.47794
	Antenna Gain = 8	0.76909

Table 1: MAPE on Forecasts for all Parameter Changes with O_{dp} = 150

To verify the correctness of the forecasting, from one complete set of experimental data, we extracted few data points and forecasted the rest, and plotted it against the actual data to get a good comparison. In Figure 2, we have shown two images with a total data point of 545 and training data of 200 and 400 data points, and compared it with the actual data. The red line shows the forecasted data and the blue line shows the actual data. It is seen that as the amount of forecasting data grows, the red line, more closely follows the blue line. Note that in the figure, for simplicity, voltage is plotted against relative time, with time at start of experiment as 0 and voltage is collected every 2 seconds.

By experimentation, we selected an optimal number of training data points (O_{dp}) required for forecasting. Then we used the obtained O_{dp} to forecast for the rest of the experimental data. With calculated Mean Absolute Percentage Error (MAPE) values for all the forecasted data, we derived the accuracy of forecasting as shown in Table 1. The possibilities with this framework are many, and this case study shows one possible method to use the framework and its plugin architecture.

6 FUTURE WORK

Our future work will focus on exploring areas of continuous self-benchmarking, that assists decision making for self-adaptive algorithms, for delivery of optimal energy performance of IoT networks. Gathering data to a central location incurs energy cost, which increases as the amount of data increases. So we will explore methods

for optimizing data handling such as data compression schemes or posting data using, unused fields of mandatory network messages.

7 CONCLUSION

As we move towards energy harvesting IoT devices, the need to benchmark and test the devices on live networks increase, due to the dynamism of the sources from where energy is harvested. We have presented a novel approach for real-time, in-situ benchmarking of energy harvesting devices. The framework discussed in the paper delivers on a method to benchmark live networks, benchmark against all the available local and global parameters of the network and device specific parameters that can be remotely controlled. A case study of the plugin architecture was presented with Holt-Winters forecasting algorithm. The framework can be used to benchmark the end-nodes, against a wide range of network and IoT device specific parameters, with little to no dependency on the energy harvesting architectures and can gracefully handle complex energy harvesting architectures. The framework essentially enables developers to continuously test their deliveries and updates, and network maintenance teams to fine tune the network with real-time, in-situ benchmarking results.

REFERENCES

[1] 7, February 2017. 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. *Gartner Press Relese* (7, February 2017). https://www.gartner.com/newsroom/id/3598917

[2] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. 2001. Manifesto for agile software development. (2001).

[3] C. Bormann, M. Ersue, and A. Keranen. 2014. *Terminology for Constrained-Node Networks.* RFC 7228. IETF. http://www.rfc-editor.org/rfc/rfc7228.txt

[4] João Cardoso, Carlos Pereira, Ana Aguiar, and Ricardo Morla. 2017. Benchmarking IoT middleware platforms. In *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2017 IEEE 18th International Symposium on.* IEEE, 1–7.

[5] Intel Corporation. 2016. Intel Edison Compute Module Datasheet. https://www.intel.com/content/dam/support/us/en/documents/edison/sb/edison-module_HG_331189.pdf

[6] Andrea Gaglione, David Rodenas-Herraiz, Yu Jia, Sarfraz Nawaz, Emmanuelle Arroyo, Cecilia Mascolo, Kenichi Soga, and Ashwin Sheshia. 2018. Energy neutral operation of vibration energy-harvesting sensor networks for bridge applications. In *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks (EWSN '18).*

[7] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. 2015. Tragedy of the Coulombs: Federating Energy Storage for Tiny, Intermittently-Powered Sensors. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15).* ACM, New York, NY, USA, 5–16. https://doi.org/10.1145/2809695.2809707

[8] Analog Devices Inc. 2018. ADG811 Datasheet. http://www.analog.com/media/en/technical-documentation/data-sheets/ADG811_812.pdf

[9] Analog Devices Inc. 2018. LTC3335 Datasheet. http://www.analog.com/media/en/technical-documentation/data-sheets/3335f.pdf

[10] Analog Devices Inc. 2018. LTC5800 Datasheet. http://www.analog.com/media/en/technical-documentation/data-sheets/5800ipmfa.pdf

[11] InfluxData. 2018. InfluxDB - Getting Started. https://docs.influxdata.com/influxdb/v1.6/introduction/getting-started/

[12] X. Jiang, J. Polastre, and D. Culler. 2005. Perpetual environmentally powered sensor networks. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.* 463–468. https://doi.org/10.1109/IPSN.2005.1440974

[13] Carel P Kruger and Gerhard P Hancke. 2014. Benchmarking Internet of things devices. In *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on.* IEEE, 611–616.

[14] Grafana Labs. 2018. Grafana - Getting Started. http://docs.grafana.org/guides/getting_started/

[15] Paul E Landman and Jan M Rabaey. 1996. Activity-sensitive architectural power analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15, 6 (1996), 571–587.

[16] O. Landsiedel, K. Wehrle, and S. Gotz. 2005. Accurate prediction of power consumption in sensor networks. In *The Second IEEE Workshop on Embedded Networked Sensors, 2005. EmNetS-II.* 37–44. https://doi.org/10.1109/EMNETS.2005.1469097

[17] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. 2003. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03).* ACM, New York, NY, USA, 126–137. https://doi.org/10.1145/958491.958506

[18] James Manyika, Michael Chui, Peter Bisson, Jonathan Woetzel, Richard Dobbs, Jacques Bughin, and Don Aharon. June 2015. The Internet of Things: Mapping the Value Beyond the Hype. *McKinsey Global Institute, McKinsey & Company* (June 2015).

[19] C. Park and P. H. Chou. 2006. AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes. In *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, Vol. 1. 168–177. https://doi.org/10.1109/SAHCN.2006.288421

[20] Mark Patel, Jason Shangkuan, and Christopher Thomas. May 2017. What's new with the Internet of Things? *McKinsey & Company Article* (May 2017). https://www.mckinsey.com/industries/semiconductors/our-insights/whats-new-with-the-internet-of-things

[21] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras. 2004. ATEMU: a fine-grained sensor network simulator. In *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.* 145–152. https://doi.org/10.1109/SAHCN.2004.1381912

[22] Vijay Raghunathan, Aman Kansal, Jason Hsu, Jonathan Friedman, and Mani Srivastava. 2005. Design Considerations for Solar Energy Harvesting Wireless Embedded Systems. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05).* IEEE Press, Piscataway, NJ, USA, Article 64. http://dl.acm.org/citation.cfm?id=1147685.1147764

[23] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso. 2005. Experimental evaluation of lifetime bounds for wireless sensor networks. In *Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005.* 25–32. https://doi.org/10.1109/EWSN.2005.1461996

[24] Prajakta S Kalekar. 2004. Time series Forecasting using Holt-Winters Exponential Smoothing. (01 2004).

[25] Victor Shnayder, Mark Hempstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. 2004. Simulating the Power Consumption of Large-scale Sensor Network Applications. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems (SenSys '04).* ACM, New York, NY, USA, 188–200. https://doi.org/10.1145/1031495.1031518

[26] Farhan Simjee and Pai H. Chou. 2006. Everlast: Long-life, Supercapacitor-operated Wireless Sensor Node. In *Proceedings of the 2006 International Symposium on Low Power Electronics and Design (ISLPED '06).* ACM, New York, NY, USA, 197–202. https://doi.org/10.1145/1165573.1165619

[27] Phillip Stanley-Marbell and Diana Marculescu. 2007. An 0.9 &Times; 1.2", Low Power, Energy-harvesting System with Custom Multi-channel Communication Interface. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '07).* EDA Consortium, San Jose, CA, USA, 15–20. http://dl.acm.org/citation.cfm?id=1266366.1266372

[28] S. Sudevalayam and P. Kulkarni. 2011. Energy Harvesting Sensor Nodes: Survey and Implications. *IEEE Communications Surveys Tutorials* 13, 3 (Third 2011), 443–461. https://doi.org/10.1109/SURV.2011.060710.00094

[29] Jay Taneja, Jaein Jeong, and David Culler. 2008. Design, Modeling, and Capacity Planning for Micro-solar Power Sensor Networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN '08).* IEEE Computer Society, Washington, DC, USA, 407–418. https://doi.org/10.1109/IPSN.2008.67

[30] B. L. Titzer, D. K. Lee, and J. Palsberg. 2005. Avrora: scalable sensor network simulation with precise timing. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.* 477–482. https://doi.org/10.1109/IPSN.2005.1440978

[31] A. S. Weddell, N. R. Harris, and N. M. White. 2008. Alternative Energy Sources for Sensor Nodes: Rationalized Design for Long-Term Deployment. In *2008 IEEE Instrumentation and Measurement Technology Conference.* 1370–1375. https://doi.org/10.1109/IMTC.2008.4547256