Convex Optimization and Model Identification for Reliable and Energy Efficient QoS-Aware IoT Systems

Pietro Mercati and Tajana Simunic Rosing
UCSD, CSE Department - San Diego CA, USA. Email: pimercat@ucsd.eng.edu, tajana@ucsd.edu

***Abstract*—In 2020 there will be more than 50 billions of interconnected devices composing the Internet of Things (IoT) to enable fine-grained sensing and actuation in a variety of fields, from smart environments to healthcare. A large part of IoT devices are powered by batteries that need to be changed or recharged frequently. Also, IoT networks require replacement of defective parts for meeting Quality of Service (QoS) targets, with high associated costs. Maintenance and energy efficiency represent important limiting factors to such growth. In this paper we formulate a comprehensive problem for reducing IoT energy consumption subject to QoS and reliability constraints. We then propose a framework to dynamically manage reliability and energy consumption using convex optimization. The proposed framework leverages model identification techniques for efficiently identifying reference values at runtime.**

I. Introduction

The Internet of Things (IoT) indicates the seamless interconnection of an extremely large number (billions) of devices to enable pervasive sensing, communication, computation and actuation in the physical world [11]. A distinctive aspect of IoT is its rapid development. In the recent years, smart devices outnumbered human beings and it is expected to reach 50 billions of interconnected devices in 2020 [18], with an average of 6.5 devices per person. Recent studies say that the IoT market is expected to grow from $157.05 billion in 2016 to $662 billion in 2021, at a Compound Annual Growth Rate (CAGR) of nearly 33 percent during that time interval [3]. With such numbers involved, it is no surprise that IoT has been recognized as a fundamental component of the fourth industrial revolution [20].

The IoT spans across applications in every field of human life: industry, environment, agriculture, smart cities, smart homes, healthcare and more [4]. In every application, the goal of IoT is to deliver Quality of Service (QoS) to the user, which can be defined in terms of accuracy, availability, stability and user satisfaction.

The industry is shaping around this new paradigm. Every major tech company today, such as Samsung, Intel, Microsoft and Apple, have an IoT department, and IoT-related startups are flourishing [8], [1].

Like any revolution, the IoT faces dramatic challenges that put limitations to its growth. The IoT is composed by different devices coming from different vendors, so standardization is required for seamless integration. The large amount of data flowing also poses problems of interoperability, data fusion, noisy data. The communication of sensitive data creates problems related to security and privacy. A large part of IoT devices is powered by batteries that need to be frequently changed or recharged, making energy efficiency a primary requirement. Moreover, the large networks of devices composing the IoT require continuous maintenance for the replacement of defective parts, with high associated costs [5].

To better motivate this work, we consider the SmartSantander project [19]. The Santander facility in the SmartSantander project employs around 2000 devices installed in the city center for environmental monitoring. Nodes in the network can cease to operate for either battery discharging or for hardware failures. If we assume that a network like the one described in [19] is composed by low cost devices powered with coin cell batteries that are $1 each, then replacing all the batteries in the network would be a cost of $2000. Also, the price of a single IoT device may range from $20 up to $200. Replacing all the devices in the Santander facility could range from $40000 to $400000. Such estimates do not account for the cost of continuous assistance, human labor and lost productivity, which can be an important portion of the total cost. These costs will scale at least linearly with the size of the IoT network, which will grow consistently in the next years. For example, Cisco is currently tracking 28 millions devices on its IoT network, and adding a million devices a month [2].

Dynamic management of systems has been used extensively for energy, thermal and reliability management in traditional devices such as personal computers, mobiles and data centers [6], [14]. However, a flexible framework for IoT energy and reliability management is still missing.

In this work we address these two key problems in IoT: energy efficiency and reliability, intended as integrity and functioning of IoT devices. We formulate the problem of reducing energy consumption subject to QoS and reliability constraints. We then propose a framework solving this problem to dynamically manage the IoT networks. Such framework relies on model identification techniques to extract reference values and employs convex optimization to adjust control decisions.

II. Problem Formulation

We consider the structure of IoT illustrated in Figure 1. We assume that the IoT devices are organized in **networks** communicating with the **Cloud** through a number of intermediate stages, also called the IoT **fog** (for example, gateways and routers).

The IoT devices have limited computation and storage capabilities, while the Cloud has virtually unlimited storage and

can run computations quickly and efficiently. We assume that a network implementing a certain IoT application is composed by N devices. We also assume that each device $i = 1, ..., N$ is characterized by M utilization metrics, contained in a $M-$dimensional vector u_i.

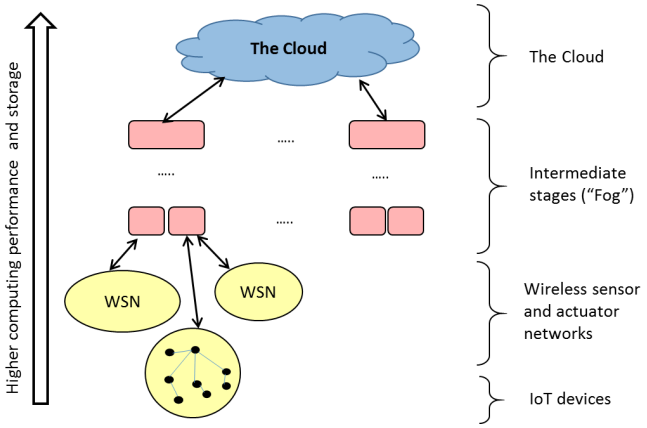

Fig. 1: Internet of Things structure

The problem of reducing energy consumption subject to QoS and reliability constraints is formulated for each network of sensors as follows.

$$\min_{\mathbf{u}} \quad \|P(\mathbf{u}, t)\|^2 = \|\sum_i P_i(u_i, t)\|^2 \tag{1}$$

$$\text{s.t.} \quad QoS(\mathbf{u}) \geq QoS_{target} \tag{2}$$

$$R_{predicted}(\mathbf{u}_{predicted}, t_{life}) \geq R_{target} \tag{3}$$

In this formulation, each constraint should be maintained for each for $k = k_0, ..., k_0 + L$, within a predefined horizon L. The term $\mathbf{u}$ is a $M \times N$ vector, containing the utilization metrics of each sensor. The total power consumption of the network P is a function of $\mathbf{u}$ and of time t, and can be expressed as the sum of power consumptions P_i for each sensor i. The quality of service QoS is a function of $\mathbf{u}$ and it is required to be greater or equal than the target value QoS_{target}. The predicted reliability $R_{predicted}$ at the target lifetime t_{life} depends on the predicted utilization metrics $\mathbf{u}_{predicted}$, and must be greater or equal than the target value R_{target}.

At each activation interval k, the problem is solved and the solution is a vector $\mathbf{u}$, determining the desired utilization metrics for the next interval.

As illustrated in [7], an optimization problem belongs to the class of convex problems if the objective function and the constraint functions are convex. The objective function is clearly convex, as it is expressed as a squared norm of a vector. As for the two constraints, we require that they are identified at runtime as convex functions.

If the problem is convex, it is possible to use an embedded convex solver. In this work, we use the CVX package. CVX is a package to specify and solve convex programs [10], [9]. Furthermore, problems that are correctly formulated for the CVX package can be translated into C++ code using the CVXGEN tool [13]. Therefore, an embedded convex solver can be cross-compiled to run on a target IoT device. In the following, we explain the nature and the role of the problem parameters more into details.

The utilization metrics in $\mathbf{u}$ should be identified as parameters that strongly influence the power consumption of the system. These are, for example, the activity ratio of the device, the residency of processor power states, operating frequency, transmitting and receiving rates. A fundamental requirement is that they can be monitored at runtime. We assume that the utilization metrics can be influenced by a set of control knobs $\mathbf{C}$, such as duty cycling, power gating of hardware components, dynamic frequency scaling (if supported) and manipulation of transmitting and receiving rates.

Thanks to this, it is possible to build a model for power consumption of the IoT devices based on the utilization metrics. To make this possible, we require at least one device for each class of device to be equipped with a monitor to measure the real power consumption, so that model identification techniques can be used to extract the function $P(u)$.

Maintaining a target Quality of Service (QoS) is the goal of IoT applications. Since the IoT is a user-centric paradigm, the QoS_{target} is defined as the minimum QoS that meets user expectation and provides quality experience. For each application and for each network, the QoS should be modeled and expressed as a function $QoS(\mathbf{u})$ of the utilization metrics, which takes values in the range $[0, 1]$. To build such a function, it is required to collect feedback from users at runtime. This can be obtained directly from users, with online evaluations on personal mobile devices. This is possible since many IoT applications have their own mobile app. Otherwise it can be obtained indirectly, for example by monitoring facial expressions of the user through the built-in cameras of their mobile devices [21].

Reliability $R(t)$ is defined as the probability that a system does not fail before time t. It is a monotonic (decreasing) function of time and assumes values in the range $[0, 1]$. In this work, we assume that each IoT device has its own reliability function $R_i(u_i, t)$, that depends on the utilization metrics and that can be computed online. Assuming a certain predicted utilization $u_{predicted_i}$ for each device, from the current time instant until t_{life}, the same model can be used to estimate the predicted reliability at the target lifetime $R_{predicted_i}(u_{predicted_i}, t_{life})$. Then, the predicted reliabilities of the individual devices are combined to express the predicted reliability of the IoT network $R_{predicted}$. We assume the case of a general series-parallel combination model, where the reliability of a series is $R_{series} = R_1 \times R_2 \times ...$ and the reliability of a parallel is $R_{parallel} = 1 - (1 - R_1) \times (1 - R_2) \times ....$ The formulated problem assumes that the IoT network target lifetime t_{life} target reliability R_{target} are predefined and provided as input parameters.

III. Framework Architecture

Figure 2 shows the components of the proposed framework. This is composed by models for reliability (**R**), power (**P**), quality of service (**QoS**) and control decisions (**C**), and by a

Control Engine. This is subdivided in an **Upper Layer** and a **Lower layer**.

All the computations involved in the control decisions and the model updates take place in the Cloud. The IoT device is only responsible for updating its own *utilization log* and communicate it periodically to the Control Engine. The utilization log is a vector of numbers where the first element is a time stamp and the remaining elements are the components of u_i, where each entry is value accumulated from the last sampling interval.

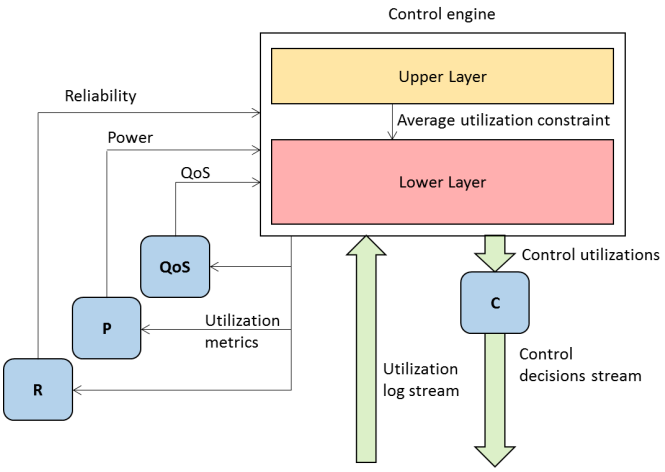

Fig. 2: Control framework architecture

The reliability model **R** is derived from accelerated testing on devices and the impact of physical degradation phenomena. For example, degradation of integrated circuits is described in terms of models for Negative Bias Temperature Instability (NBTI), Time Dependent Dielectric Breakdown (TDDB) and other degradation mechanisms [14]. We assume each IoT device to have its own reliability fun1ction, expressed as a function of the utilization metrics.

The power model **P** is identified online. Assuming a network of IoT devices of the same kind, at least one of them should be equipped with a power monitor. Such nodes are called *power spies*. By sampling power and utilization metrics at a constant rate, data can be fed to a model identification algorithm. As an example we consider the Recursive Least Square (RLS) algorithm to derive a linear model [12]. The main advantage if RLS is its very fast convergence. The drawback is high computational complexity, which can be resolved with low overhead by the Cloud. The power spy transmit utilization metrics log together with a power log to the control engine, which process the data and updates the parameters in the stored power model. No computation is performed by the power spy itself. In the end, the control engine maintain and updates a model for each type of devices.

The idea of RLS is to derive a linear model of the kind $y = ax$ given noisy measured data $y^* = ax + \epsilon$, where y is the system output (for example power), x is the input (in this case, the utilization metrics), a is a vector of coefficients and ϵ is noise. RSL will implement recursive formulas to update the coefficients $\hat{a}$ at runtime to minimize the following quantity over the previous k sampling intervals.

$$\min_{\hat{\mathbf{a}}} \quad \sum_{i=0}^{k-1} (\hat{a}x_i - y_i)^2 \tag{4}$$

The quality of service model **QoS** is also identified online. The control engine receives periodical feedback from users to determine the current status of quality of a specific IoT application, and correlates it with the utilization metrics of the devices participating in such application. Then, similarly as for power, it employs model identification such as RLS to derive a model for QoS.

Finally, our framework requires a model **C** to convert the solution of the optimization problem, expressed in terms of desired utilization metrics, into actual control decisions. This requires characterizing how each control knob impacts on the utilization metrics. For example, for obtaining an activity ratio (utilization metric) of 50%, the control decision can issue a duty cycling of 50% using power gating (control decision).

In the control engine, for computation efficiency the problem is subdivided into a upper layer and a lower layer. The motivation is that reliability changes takes place on a higher time scale with respect to that of interest for power management decisions. The upper layer activates at a slow rate, while the lower layer activates at a faster rate. Given this, the lower layer solves the same problem of Equation 1, where the constraint in Equation 7 is replaced by the following.

$$\mathbf{u}_{average} \leq \mathbf{u}_{ref_{UL}} \tag{5}$$

Where $\mathbf{u}_{ref_{UL}}$ is the average utilization constraint computed by the upper layer by solving the following optimization.

$$\min_{\mathbf{u}} \quad \|R_{predicted}(\mathbf{u}_{predicted}, t_{life})\| \tag{6}$$

$$\text{s.t.} \quad R_{predicted}(\mathbf{u}_{predicted}, t_{life}) \geq R_{target} \tag{7}$$

Where $\mathbf{u}_{predicted}$ indicates the average of utilization metrics predicted from the current time interval until t_{life}. Such prediction can be computed with an exponential moving average [14] where the new sample is the most recent $\mathbf{u}$. If $R_{predicted}$ is a convex function of $\mathbf{u}$, then this problem can be solved with an online convex solver such as CVX.

IV. Results of Previous Research

Our previous research in power and reliability management for mobile devices demonstrates the importance of having a QoS-aware and user-centric management. Also, we highlight the significance of designing a framework with a low overhead implementation and that is easy to port and modify.

In our previous work in [16] we proposed *application-dependent power states*: a simple online model for mobile power consumption. Based on that we formulate a management problem to improve performance under battery lifetime constraints, and implement the controller on a real Android device. The lightweight implementation can be installed on any Android-based device. In our experimental evaluation, we

verify that the proposed strategy can meet user experience requirements with a battery lifetime extension of at least 25%.

In our previous work in [17] we proposed a framework for user experience-aware power management of mobile devices. The proposed technique modulates the operating conditions based on application-specific user preferences to meet quality of experience requirements. The implementation is lightweight and does not require restructuring the operating system. We implemented the technique on a Google Nexus 5 and we demonstrated that it can achieve up to 46% of application-specific savings on power consumption and up to 35% savings at the device level. More detailed results of the proposed QoS-aware manager are reported in Figure 3, compared to a QoS-agnostic manager, for a range of popular mobile applications.

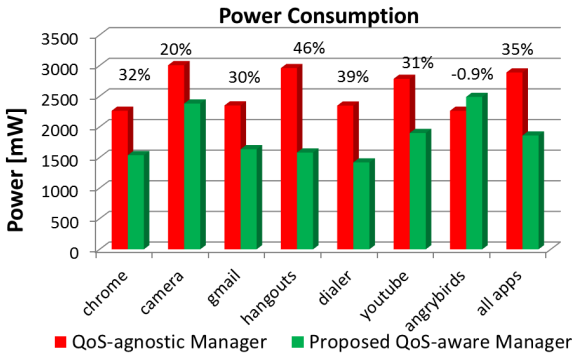

Fig. 3: Power savings [17]

In our previous work in [14] we proposed a hierarchical controller for multicore processor dynamic reliability management, exploiting the major gap between the time scales of workload variations and reliability loss. We improved user experience by locally relaxing reliability-induced operating point constraints, while meeting them over the large time windows relevant for reliability. With respect to the state-of-the-art, this solution guarantees timely execution of 100% of QoS-critical tasks. Work in [15] then presented the first complete software implementation on a real device of a low-overhead, Android-compatible workload-aware dynamic reliability manager for mobile multiprocessors. We discussed the design challenges and the runtime overhead involved. We also show the effectiveness of our governor in guaranteeing the predefined target lifetime and show that it achieves up to 100% of lifetime improvement with respect to traditional governors, while providing comparable performance for critical applications.

REFERENCES

[1] https://angel.co/internet-of-things.

[2] http://www.recode.net/2016/6/1/11835156/cisco-iot-28-million-connected-devices-cars.

[3] http://www.marketsandmarkets.com/market-reports/internet-of-things-market-573.html. 2016.

[4] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Comput. Netw.*, 54(15):2787–2805, Oct. 2010.

[5] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal. Sentinel: Occupancy based hvac actuation using existing wifi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 17:1–17:14, New York, NY, USA, 2013. ACM.

[6] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini. Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller. *IEEE Trans. Parallel Distrib. Syst.*, 24(1):170–183, Jan. 2013.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[8] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller. A survey of commercial frameworks for the internet of things. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–8, Sept 2015.

[9] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[10] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1, Mar. 2014.

[11] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.*, 29(7):1645–1660, Sept. 2013.

[12] M. H. Hayes. *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1996.

[13] J. Mattingley and S. Boyd. CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, Mar. 2012.

[14] P. Mercati, A. Bartolini, F. Paterna, T. S. Rosing, and L. Benini. Workload and user experience-aware dynamic reliability management in multicore processors. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 2:1–2:6, New York, NY, USA, 2013. ACM.

[15] P. Mercati, A. Bartolini, F. Paterna, T. S. Rosing, and L. Benini. A linux-governor based dynamic reliability manager for android mobile devices. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–4, March 2014.

[16] P. Mercati, V. Hanumaiah, J. Kulkarni, S. Bloch, and T. Rosing. Blast: Battery lifetime-constrained adaptation with selected target in mobile devices. In *Proceedings of the 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services on 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MOBIQUITOUS'15, pages 80–89, ICST, Brussels, Belgium, Belgium, 2015. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[17] P. Mercati, T. S. Rosing, V. Hanumaiah, J. Kulkarni, and S. Bloch. User-centric joint power and thermal management for smartphones. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, pages 98–105, Nov 2014.

[18] C. Perera, C. H. Liu, S. Jayawardena, and M. Chen. A survey on internet of things from industrial market perspective. *IEEE Access*, 2:1660–1679, 2014.

[19] L. Sanchez, L. Munoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, and D. Pfisterer. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61:217 – 238, 2014.

[20] K. Schwab and W. E. Forum. *The Fourth Industrial Revolution*. World Economic Forum, 2016.

[21] A. Shye, Y. Pan, B. Scholbrock, J. S. Miller, G. Memik, P. A. Dinda, and R. P. Dick. Power to the people: Leveraging human physiological traits to control microprocessor frequency. In *2008 41st IEEE/ACM International Symposium on Microarchitecture*, pages 188–199, Nov 2008.