Simulating Reliability of IoT Networks with *RelIoT*

Kazim Ergun[1], Xiaofan Yu[1], Nitish Nagesh[1], Ludmila Cherkasova[2], Pietro Mercati[3], Raid Ayoub[3], Tajana Rosing[1]
[1]*University of California San Diego* [2]*Arm Research* [3]*Intel Corporation*

***Abstract*—The Internet of Things (IoT) networks are expected to operate reliably for many years while meeting the needs of a growing range of applications. However, a dependable operation may not always be maintained due to the reliability degradation of IoT devices. From low-power sensors to multi-core platforms, IoT devices age and degrade, leading to failures that necessitate maintenance. Reliability-aware design and management were shown to delay failures and improve the lifetime of individual devices. Even though the IoT networks can also radically benefit from this, the unavailability of network simulators that provide reliability modeling makes it impossible to assess reliability-aware strategies. To bridge this gap and enable reliability analysis at an early design phase, we introduce an integrated reliability framework called *RelIoT* for IoT networks, implemented in the ns-3 simulator. The framework also includes modeling of power, performance, and temperature, which are required to model reliability. We validate the simulations done using our framework and demonstrate that *RelIoT* accurately captures the power, temperature, and reliability dynamics of real networked IoT devices.**

I. Introduction

The emerging paradigm of the Internet of Things (IoT) connects billions of heterogeneous devices ranging from low-power sensors with limited computational capabilities to multi-core platforms on the high-end. The rapid development of IoT has lead to a tremendous increase in the number of devices connected to the Internet. By 2025, the IoT is expected to connect 41 billion devices [1].

The inherent large-scale and heterogeneity of the IoT brings a maintainability challenge with it. IoT devices, as any electronic or mechanical system, are prone to failures due to the aging and degradation of electronic circuits, batteries, and other components. Eventually, these devices require maintenance for the repair or replacement of defective parts. According to Cisco, for 100K devices that operate IoT smart homes, around $6.7M/year will be spent for administration and technical diagnosis related to system failures, comprising between 30% to 70% of total costs [2]. Despite being a major concern, there is a lack of thorough studies on the *reliability* of IoT networks in the context of device aging and degradation.

The reliability of IoT networks can be improved with proper system design and reliability management strategies. Prior work has studied the management of reliability degradation on processor-based systems, showing notable gains in the lifetime of individual devices [3]–[5]. To extend this work and explore reliability management strategies on a network-level, there is a need for a convenient tool. Simulators are widely used tools in research and industry to evaluate and validate networks, but currently it is not possible to analyze the reliability of IoT networks with any of the available network simulators. The most popular ones, e.g., ns-3 [6], OMNeT++ [7], and OPNET [8], are designed only for analyzing communication performance (throughput, delay, utilization, etc.) under different protocols.

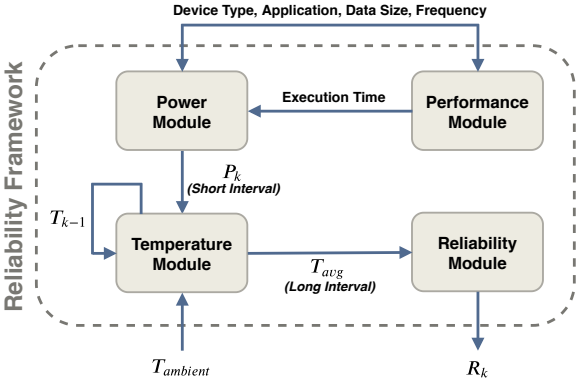

Fig. 1: Reliability framework structure

To enable reliability evaluation and analysis in IoT networks, we present a simulation framework, called *RelIoT*, implemented as a set of modules for the ns-3 [6] network simulator. The proposed framework, besides *reliability* module, incorporates three other interrelated modules for *power, performance*, and *temperature*. The use of other modules are required to be able to evaluate reliability. Moreover, with this design choice, *RelIoT* allows users to explore trade-offs between power, performance, and reliability of network devices. The following are the two key aspects of the design and implementation of *RelIoT*:

- **Scalability:** *RelIoT* is implemented in ns-3 [6], which is a simulator with low computational overhead and low memory demands. Up to one billion nodes can be simulated with ns-3 [9], [10]. *RelIoT* incurs only a marginal performance overhead on the default ns-3, making it scalable for simulating large networks.
- **Flexibility:** The framework is extensible and enables the integration of new models as well as the configuration of existing models. Also, due to the modular design of *RelIoT*, users can conduct simulations with various combinations of power, performance, temperature and reliability models.

To the best of our knowledge, *RelIoT* [1] is the first reliability analysis framework for heterogeneous IoT networks, taking thermal characteristics as well as power and performance into account. In this paper, we discuss the design of the framework and and describe the provided models in detail. We validate our framework against a real experiment setup, showing *RelIoT* estimates power, performance and temperature with errors of less than 3.8%, 4.5% and $\pm 1.5°C$ respectively. We validate reliability models against the results from existing literature.

[1]The code is available at https://github.com/UCSD-SEELab/RelIoT.

II. RelIoT Framework: Design and Implementation

In this section, we describe the overall structure of *RelIoT*, the functionality of the proposed modules and interfaces, and present underlying models in detail.

A. Overview of the Proposed Framework

To allow reliability simulation in ns-3, *RelIoT* integrates the following modules as shown in Fig. 1.

- *Performance Module.* IoT devices can run some applications to process the sensed or collected data before sending it to a central entity. The performance module provides performance predictions (e.g., execution time) for these applications.

- *Power Module.* Estimates power consumption of the device for various applications with configurable power models.

- *Temperature Module.* Predicts the temperature of a device based on its power consumption and ambient temperature.

- *Reliability Module.* Evaluates the device reliability using the existing thermal-based degradation models [4], [5], [11].

The modules operate on two different time scales. Performance and power values are updated every *Short Interval*, which is on the order of milliseconds. The reliability value is updated every *Long Interval*, on the order of days, because reliability estimation is computationally expensive. Therefore, the reliability module uses the averaged temperature over each *Long Interval*. The underlying mechanisms of each module is explained in more detail in the remainder of the paper.

B. Power Module

The power module encompasses power models and various functions. The main task of the power module is to update power values at the predefined period *Short Interval*. If an application is designated to run on a device, the power module sets the state of that device to *Busy* and calculates the power value according to the selected power model. If the device is not running any applications, the state is set to *Idle*.

In this work, we focus on the power modeling of the CPU of an IoT device. There are numerous power modeling techniques from cycle-accurate to functional-level at different levels of abstraction. Low-level models usually have high computational complexities because they use a fine-grain representation of the CPU. Since many nodes should be simulated concurrently in network simulators, estimation with low-level models is very time consuming, which is highly undesirable. We offer two CPU power models in our framework, having low model complexity while still providing good estimation accuracy.

Frequency & Utilization-based Power Model. We use linear models to estimate the CPU power consumption based on CPU frequency and utilization. Similar models were leveraged previously by many works to a great extent because they provide sufficient accuracy using easily accessible metrics. For example, in [12], the authors characterized the CPU power of a smartphone as a linear combination of frequency and utilization with an average error less than 2.5%. The CPU power consumption P_{CPU} can be expressed as:

$$P_{CPU}(t) = a \cdot f(t) + b \cdot u(t) + c \tag{1}$$

where $f(t)$ and $u(t)$ are CPU frequency and utilization at time t respectively. The coefficients a, b, c are learned through linear regression based on datasets collected on real devices. The frequency and utilization traces may not be always available to the users in practice. In that case a functional-level model is convenient, presented next.

Application-based Power Model. The CPU power consumption of IoT devices varies depending on the data processing application it is running. If there are only high-level functional properties available, an application-based power model is useful. In our framework, we adopt the modeling methodology proposed in [13], where the authors characterize and verify power models of running machine learning (ML) algorithms on edge devices (i.e. Raspberry Pi) and servers. The observation is that different ML algorithms show different power trends (i.e. linear, exponential etc.) with increasing input data size. Besides, the amount and the characteristics of power consumption resulted by running the same applications alter for different devices.

We leverage the same methodology to build models for Raspberry Pi's, servers, and for microcontrollers such as Arduinos. Taking the size of processed data by applications as the model input, we train, test and cross-validate four regression models (linear, polynomial, log and exponential regression), and select the one that best predicts power consumption. Our framework delivers the 22 ML algorithms modeled in [13], and a CPU power model for Multilayer Perceptron (MLP) based on the number of MAC (multiply-accumulate) operations. The same modeling approach can further be applied to other neural network architectures such as Convolutional Neural Networks (CNNs). To improve the extensibility of the simulator for custom applications, we have included functionality for users to add new models to the power module. Parameters of the power models are configurable through external interfaces.

C. Performance Module

IoT systems need to satisfy various performance requirements to provide acceptable Quality of Service (QoS) to users. The performance module is designed to evaluate and monitor the performance of applications and hence the overall network performance. Different metrics can be used to quantify performance, e.g., throughput, response time, etc. The performance metric is application-specific; for example, delay and throughput are critical in multimedia streaming applications whereas information accuracy is the main criterion for performance in some ML applications.

Execution Time Model. We use the input data size of the application or the number of MAC operations it needs to perform to estimate the application execution time. To build the model, we measure the execution times of various applications on a target device, then fit regression models to the collected data. Certain performance metrics can be calculated using the execution time value. For example, if t_{exec} is the execution time of an application, then its throughput can be obtained as D/t_{exec} where D is the input data size.

D. Temperature Module

The temperature module estimates device temperature (based on device power consumption and ambient temperature) and calculates average temperature over a *Long Interval*. For thermal modeling, we adopt a strategy that can be used for any IoT device. To have an acceptable level of complexity

in our simulator, we work on high-level information gathered from the coarse-grained thermal sensors of the device's key heat sources. Such information is available in most of the devices today like smartphones and single-board computers (e.g., Raspberry Pi).

Let the number of the heat sources be n and let $T_k \in \mathbb{R}^n$ represent the vector of temperatures observed by thermal sensors and $P_k \in \mathbb{R}^n$ be the power consumed by the heat sources at time instant k. Each heat source is assumed to have one thermal sensor measuring its temperature. Then, temperature T_{k+1} at time instant $k+1$ can be predicted given the current temperature T_k and power P_k at time k. The discrete-time state-space model of the device's thermal behavior is expressed in Equation (2) [14].

$$T_{k+1} = A \cdot T_k + B \cdot P_k + C \cdot T_k^{env} \tag{2}$$

where $A, B \in \mathbb{R}^{n \times n}$ are defined as the state and the input matrices. T_k^{env} is the ambient temperature and C is a vector of coefficients which weighs the impact of ambient temperature on each heat source's internal temperature. We use system identification methods to derive the model from measured power and temperature traces. A, B and C parameters are different for each class of devices, so we offer multiple device thermal models and made the parameters configurable through the temperature module API. The order of the model is equal to the number of the heat sources n. In our initial work, we have $n = 1$, where the only source is CPU. However, the extension to multiple sources is straightforward in our framework. For example, if a power model for GPU is provided, then power consumption values from both CPU and GPU can be used to predict temperature.

The temperature module updates the states in Equation (2) at a time resolution of *Short Interval*, the same time granularity as power estimation updates. On the other hand, average temperature $\bar{T}$ is calculated for every *Long Interval* denoted *LI*. $\bar{T}$ is the exponential moving average of past temperature values in the interval k to $k + LI$.

$$\bar{T}_{k+1} = \alpha \cdot T_k - (1 - \alpha) \cdot \bar{T}_{LI} \tag{3}$$

where α is a weighing coefficient that is configured depending on the length of interval LI.

E. Reliability Module

Reliability is defined as the probability of not having failures before a given time t. The effects of different failure mechanisms should be combined to obtain the overall reliability of a processor. We use the sum-of-failure-rates model in our reliability module, which states that the processor is a series failure system; the first instance of a failure due to any mechanism causes the entire processor to fail [11]. The reliability module computes the single device reliability as a product of the reliabilities due to different failure mechanisms. Some examples for these mechanisms are: Time Dependent Dielectric Breakdown (TDDB), Negative Bias Temperature Instability (NBTI), Hot Carrier Injection (HCI), Electromigration (EM) and Thermal Cycling (TC). In this work, we implement the TDDB model, but the framework allows the integration of other models.

Reliability degradation develops by consistent stress over long time intervals rather than instantaneous stress. Compared to power and temperature, reliability is a more slowly changing variable. Since reliability models are usually highly compute-intensive, we calculate reliability sparsely in our framework. The reliability module does estimation every *Long Interval* (on the order of hours or days), using temperatures averaged over the interval. Due to the aforementioned properties, the models do not lose significant accuracy but the simulations speed up.

Time Dependent Dielectric Breakdown (TDDB) Reliability Model. Due to gate oxide degradation in transistors, which is a non-reversible mechanism with a cumulatively increasing impact, there is a risk of breakdown that may shorten the device's lifetime. The reliability of a single transistor subject to oxide degradation can be expressed as [15]:

$$R_i(t) = e^{-a(\frac{t}{\gamma})^{x_i \beta}} \tag{4}$$

where t is the time-to-breakdown, x_i is the oxide thickness, a is the device area normalized with respect to the minimum area, and γ and β are respectively the scale parameter and shape parameter. The scale parameter γ represents the characteristic life, which is the time where 63.2% of devices fail, and it depends on voltage and temperature. The shape parameter β, instead, is a function of the critical defect density, which in turn depends on oxide thickness, temperature and applied voltage. $R(t)$ indicates the probability of the system not failing before time t. It is a monotonically decreasing function with values in the range of [0, 1].

The reliability of the entire chip R_C can then be expressed as the product of single transistor reliabilities:

$$R_C(t) = \prod_{i=1}^{m} R_i(t) = e^{\sum_{i=1}^{m} -a_i(\frac{t}{\gamma_i})^{\beta_i x_i}} \tag{5}$$

m is the number of transistors on the entire chip. It can be on the order of millions, which possesses a large complexity on the computation of Equation (5). However, this complexity can be reduced by assuming the same scale and shape parameters over the chip [5] for the reason that different regions of the chip have similar temperatures.

The R_C expression in Equation (5) is only representative of static systems because it assumes a constant temperature applied from time $t = 0$. To capture the dynamics of reliability under varying temperature, we discretize the time and calculate reliability at each time step as shown in Equation (6). The temperature is assumed to be constant between time steps.

$$R_k = R_{k-1} - \Big(R_C(t_{k-1}, T_{k-1,k}) - R_C(t_k, T_{k-1,k}) \Big) \tag{6}$$

In Equation (6), k indicates the k^{th} time instant and $T_{k-1,k}$ is the temperature experienced by the chip between the time instants $k-1$ and k. We set this interval between adjacent time steps as the *Long Interval* and let $T_{k-1,k}$ be equal to the average temperature $\bar{T}_{LI}$ of the corresponding *LI*.

The reliability module can work with any failure mechanism or combination of multiple mechanisms as long as the mechanism can be described by a function $R_C(t)$, as in Equation (5). For example, the module can be extended to include NBTI and HCI if we describe the reliability functions associated with these mechanisms, respectively R_{NBTI}

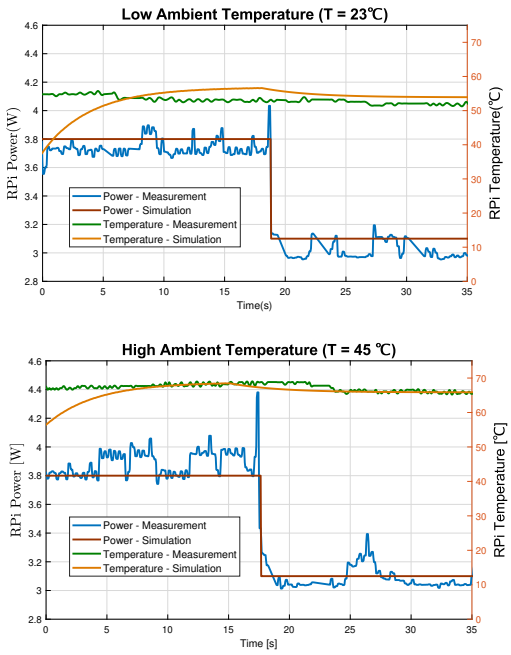

Fig. 2: Power and temperature traces for RPi3

and R_{HCI}. Then, by the sum-of-failure-rates approach, the reliability module calculates the total system reliability as the product of the functions associated with the single mechanisms as $R_C(t) = R_{TDDB}(t) \cdot R_{NBTI}(t) \cdot R_{HCI}(t)$. Equation (6) would not need any modifications since it is general and does not depend on a specific $R_C(t)$.

III. Experiments and Results

In this section, we present validation results on a three-node network topology, comparing power, performance and temperature measurements in experiments with simulated traces.

A. Validation and Evaluation

We use a simple three-node network to verify the functionality of the simulator modules to validate the device models. The network consists of an ESP8266 WiFi microchip with microcontroller, a Raspberry Pi 3 (RPi3), and a PC. The devices communicate with MQTT protocol over WiFi (IEEE 802.11b). The ESP8266 samples data as a sensor node, runs a data preprocessing application, then sends the preprocessed data to the RPi3. Data is further processed by an application on the RPi3, or the computation is offloaded to the PC. This type of computation offloading is common in IoT edge devices and is representative of their usual operation [16]. If the application is chosen to be offloaded, then the RPi3 is only responsible of relaying incoming data to the PC. In our three-node experiments, we collected: (i) RPi3 power consumption, (ii) RPi3 CPU temperature, (iii) ESP8266 power consumption, (iv) ESP8266 CPU temperature, and (v) ambient temperature measurements synchronously.

Two example traces from measurements and simulations are presented in Fig. 2 under two different ambient temperatures. We here show a temporal view of the simulator output, in a dynamic case where the simulated device has a varying workload. In the experiment, the RPi3 runs a data processing application with incoming data input from ESP8266 for the first 15-20 seconds. After that, the application is offloaded to the PC and the RPi3 only relays data while its CPU is $Idle$. As shown in Fig. 2, the simulator output follows the real power and temperature traces with a mean error of 3.42% and 6.19% in low ambient temperature, and with a mean error of 2.69% and 3.97% in high ambient temperature. There is a discrepancy between real and simulated temperatures at the beginning of each plot because the initial condition set for the temperature is low ($25°C$) and it reaches the steady state value after some time. Overall, we estimate the execution time and energy consumption of the RPi3 for 23 different ML applications with average errors of 3.8% and 4.5%, respectively. For the CPU temperature, the state-space model predictions stays within $\pm 1.5°C$ of measurements at steady state, for all applications.

IV. Conclusion

In this work, we introduced a novel framework for reliability analysis of IoT networks implemented in the ns-3 network simulator. *RelIoT* can assist researchers and engineers in exploring trade-offs between power, performance and reliability of devices in IoT networks. We are working on leveraging our framework to do design space exploration (DSE) in IoT networks. We can simulate, explore, and evaluate the optimality of different network configurations for different objectives such as reliability, performace, and energy efficiency. We believe that *RelIoT* will help researchers to assess the reliability degradation problems in large-scale networks.

V. Acknowledgements

This work was partially supported by SRC task #2805.001, NSF grants #1911095, #1826967, #1730158 and #1527034, and by KACST.

References

[1] International Data Corporation, "The Growth in Connected IoT Devices," https://www.idc.com/getdoc.jsp?containerId=prUS45213219, 2019, [Online].
[2] Cisco, ""The Hidden Costs of Delivering IIoT," https://www.cisco.com/c/dam/m/en_ca/never-better/manufacture/pdfs/hidden-costs-of-delivering-iiot-services-white-paper.pdf, 2016, [Online].
[3] P. Mercati, F. Paterna, A. Bartolini, L. Benini, and T. S. Rosing, "Warm: Workload-Aware Reliability Management in Linux/Android," *IEEE TCAD*, vol. 36, 2016.
[4] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Reliability Modeling and Management in Dynamic Microprocessor-Based Systems," in *DAC*, 2006.
[5] C. Zhuo, D. Sylvester, and D. Blaauw, "Process Variation and Temperature-Aware Reliability Management," in *DATE*, 2010.
[6] "The ns-3 Network Simulator," https://www.nsnam.org/, [Online].
[7] "OMNeT++ Discrete Event Simulator," https://omnetpp.org/, [Online].
[8] X. Chang, "Network Simulations with OPNET," in *WSC'99. 1999 Winter Simulation Conference Proceedings.'Simulation-A Bridge to the Future'*, vol. 1, 1999.
[9] S. Nikolaev, E. Banks, P. D. Barnes, D. R. Jefferson, and S. Smith, "Pushing the envelope in distributed ns-3 simulations: One billion nodes," in *Proceedings of the 2015 Workshop on Ns-3*, ser. WNS3 '15, 2015.
[10] S. Nikolaev, P. D. Barnes, J. M. Brase, T. W. Canales, D. R. Jefferson, S. Smith, R. A. Soltz, and P. J. Scheibel, "Performance of distributed ns-3 network simulator," in *SIMUtools*, 2013.
[11] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The Case for Lifetime Reliability-Aware Microprocessors," in *Computer Architecture News*, vol. 32, 2004.
[12] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones," in *Proc. of the 8th IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2010.
[13] W. Cui, Y. Kim, and T. S. Rosing, "Cross-Platform Machine Learning Characterization for Task Allocation in IoT Ecosystems," in *CCWC*, 2017.
[14] F. Beneventi, A. Bartolini, A. Tilli, and L. Benini, "An Effective Gray-Box Identification Procedure for Multicore Thermal Modeling," *IEEE Transactions on Computers*, vol. 63, no. 5, 2012.
[15] J. H. Stathis, "Physical and Predictive Models of Ultrathin Oxide Reliability in CMOS Devices and Circuits," *IEEE Transactions on Device and Materials Reliability*, vol. 1, no. 1, 2001.
[16] F. Samie, V. Tsoutsouras, D. Masouros, L. Bauer, D. Soudris, and J. Henkel, "Fast operation mode selection for highly efficient iot edge devices," *IEEE TCAD*, 2019.