Automated Construction of Fast and Accurate System-Level Models For Wireless Sensor Networks

Lan S. Bai† Robert P. Dick† Pai H. Chou‡ Peter A. Dinda*
{lanbai, dickrp}@umich.edu phchou@uci.edu pdinda@northwestern.edu
†University of Michigan ‡University of California, Irvine *Northwestern University

***Abstract*—Rapidly and accurately estimating the impact of design decisions on performance metrics is critical to both the manual and automated design of wireless sensor networks. Estimating system-level performance metrics such as lifetime, data loss rate, and network connectivity is particularly challenging because they depend on many factors, including network design and structure, hardware characteristics, communication protocols, and node reliability. This paper describes a new method for automatically building efficient and accurate predictive models for a wide range of system-level performance metrics. These models can be used to eliminate or reduce the need for simulation during design space exploration. We evaluate our method by building a model for the lifetime of networks containing up to 120 nodes, considering both fault processes and battery energy depletion. With our adaptive sampling technique, only 0.27% of the potential solutions are evaluated via simulation. Notably, one such automatically produced model outperforms the most advanced manually designed analytical model, reducing error by 13% while maintaining very low model evaluation overhead. We also propose a new, more general definition of system lifetime that accurately captures application requirements and decouples the specification of requirements from implementation decisions.**

I. Introduction

Any sensor network design process, whether manual or automated, requires that the designer or synthesis toolchain estimate the quality of prospective designs. Many performance metrics exist, and the relevant quality metric is often application-dependent. The faster the metric can be estimated for a prospective design, the better, as this permits more of the solution space to be evaluated in the same amount of time. However, the estimate must also have sufficient accuracy and fidelity to support appropriate design decisions.

The modeling work in this paper is part of a project on automated synthesis of sensor networks driven by very high-level specifications written by application domain experts. The goal of the synthesis process is to produce a sensor network implementation that meets the specifications and optimizes or bounds system-level performance metrics such as lifetime, price, and sampling resolution. Our work and related automated synthesis research [1], [2] share the need to rapidly and accurately estimate such metrics for prospective designs in the "inner loop" of the synthesis process. Accurate system-level performance models can be used to rapidly evaluate a multi-objective optimization function and find Pareto-optimal designs.

There are currently three approaches to estimating system-level performance metrics; each has a different tradeoff between efficiency and accuracy. *Measurement-based approaches* are based on data from real wireless sensor network deployments. They are accurate, but also the most costly in terms of hardware and human effort, and are particularly challenging to use for metrics relevant to long-term behavior. *Simulation-based approaches* are based on simulation of the prospective design. Detailed network simulation can handle numerous performance metrics but is so slow that relying solely on simulation for design space exploration is impractical. *Analytical approaches* are based on manually constructed models that quickly compute specific performance metrics for a prospective design. However, such models are less accurate than measurement or simulation because simplifying assumptions must be made in their construction, particularly in regards to network and environment behavior.

We have developed a technique for the automated construction of fast and accurate models for estimating system-level sensor network performance metrics. Our technique combines the accuracy of simulation-based approaches with the rapid evaluation time of analytical approaches. The key idea is to automatically derive a model for a system-level performance metric from measured component behavior and detailed simulation results. Model construction is done offline and may be time-consuming without cause for concern, as it needs not be repeated during the design or synthesis process. Once the model is constructed, it can be rapidly and repeatedly evaluated.

Automated Model Construction: Our technique is based on fitting a statistical model to the multidimensional simulated quality metric data that characterize a design space. The black-box technique we propose can be readily automated and permits rapid evaluation of the resulting models. Numerous stochastic processes influence metrics such as system lifetime. Models constructed with the proposed process support prediction of the values of deterministic variables, and the distributions of stochastic variables. This allows a variety of metrics to be computed. In our system lifetime example, metrics such as mean time to system failure or time to n-probability of system failure can also be readily computed. As more simulation data are included, the model improves at the cost of increased model construction time. Our iterative sampling technique allows desired model accuracy to be achieved with few simulation runs. We have considered a range of alternative modeling techniques, and have found that Kriging (an interpolation method) is most appropriate [3].

Our technique also incorporates known component time-dependent characteristics into the models it builds for system-level metrics. This makes it possible to capture long-term behavior that might not be observed in measurement or simulation spanning short time intervals. One important behavior is component failure. Node failures are common in deployed wireless sensor networks because sensor nodes are generally constructed using inexpensive components and often operate in harsh environments. However, node fault processes are often ignored when considering system-level metrics, such as lifetime. Most previous work equates node lifetime and battery lifetime. In our system lifetime example, our model considers both node-level fault processes and battery depletion. We conducted experiments in which device faults were measured for a specific sensor network platform. The node temporal fault distribution we use is consistent with our 21 months of measurement data.

Definition of system lifetime: We evaluate our model construction technique using the system lifetime performance metric. System lifetime has generally been defined as the duration from the start of operation until the sensor network ceases to meet its operating requirements, but most existing work uses a limited definition of "operating requirements" to simplify the system lifetime estimation problem. Past work has defined network failure as (1)

This work was supported in part by the National Science Foundation under awards CNS-0721978, CNS-0910816, and CNS-0347941.

978-3-9810801-7-9/DATE11©2011 EDAA

first node failure [4], (2) first link disconnection, (3) failure of a specific number or percentage of nodes [5], and (4) disconnection of a specific number or percentage of nodes. These definitions have unfortunate implications for system design because they are often poorly related to specific application requirements. More importantly, *lifetime metrics based on such criteria conflate specification and implementation decisions*. Consider an application in which one must sample temperature with a spatial resolution of one sample per square meter. The common metrics would not appropriately capture the lifetimes of implementations that use redundant nodes for fault tolerance because the failure of a number or percentage of nodes differs from the inability to gather data at the required spatial resolution. Coupling specification and implementation is especially troublesome if the application domain expert, e.g., a geologist or biologist, is not an expert in embedded system design. Reasoning about the relationship between network-level and application-level behaviors requires understanding the low-level system components and how they interact with each other. Domain experts rarely have the time or inclination to develop this understanding.

We believe that the definition of system lifetime should capture the requirements of application domain experts while limiting ties to implementation decisions. The definition should also be flexible enough to support a class of applications instead of a specific application. Section V-B presents and provides support for such a definition of sensor network lifetime, which can be summarized as follows: *system lifetime is the duration from the start of operation until the sensor network ceases to meet the specified application-dependent but implementation-independent data gathering requirements*. More generally, our automated construction process makes it possible to generate a model based on the application domain expert's preferred system lifetime metric.

Using our proposed definition of system lifetime, we applied our automatic model construction technique to modeling system lifetime for data gathering applications. Our iterative sampling technique supports construction of a predictive model with 3.6% error relative to exhaustive simulation based on simulation of only 0.27% of the design space.

Contributions: Our work makes the following contributions.

1. We are the first to propose an automatic method to construct fast and accurate models of multiple system-level metrics in wireless sensor networks.

2. We evaluate our framework by using it to build a model of system lifetime, and comparing this model with the most advanced analytical model in the literature, which it surpasses in accuracy. The resulting model itself is therefore a contribution.

3. We propose a new definition for system lifetime that better represents application requirements than current definitions and allows sensor network specification be decoupled from implementation.

4. We present a measurement-based model for node-level fault processes, and use it for system-level reliability modeling.

Those interested in using the techniques described in this paper can find more information at the associated project website [6].

II. Related Work

Model construction from simulation or measurements with statistical methods or machine learning techniques has been used to model processor design spaces [7], [8], [9]. Previous work has demonstrated that accurate predictive models can be built by sampling a small percentage of points in the design space. We are the first to apply simulation-based model generation methods to system-level sensor network performance metrics. We focus on defining appropriate system-level performance metrics and developing a framework to automatically construct models to estimate them.

Researchers have previously proposed definitions and models for system lifetime [4], [5], [10]. Generally, node-level fault processes have been ignored. However, a lifetime model that considers only battery lifetime is insufficient, because node-level faults can occur before battery depletion and they also influence system performance [11], [12]. Our problem is formulated using a system lifetime definition that, as we will later argue, is more general and better suited for use by application designers. Lee et al. constructed analytical models for sensor network aging analysis using a network connectivity metric [13]. They consider node fault processes in addition to battery depletion. In contrast, we use a definition of system lifetime that decouples specification from implementation and describe a regression technique to automatically construct system-level lifetime models based on node-level characteristics. We also provide evidence that our automatically derived model is more accurate than their manually constructed analytical model when evaluated using their system lifetime definition.

Node-level lifetime models can be used as a foundation for estimating system-level lifetime. Most work assumes that node lifetime equals battery lifetime, which is estimated by computing time spent in each power state [14]. A few researchers directly measured device fault processes. The developers of the ZN1 sensor node module [15] accelerated aging by inducing rapid thermal cycling in order to estimate node lifetime. Our work considers both factors, battery depletion and device faults.

III. Node-Level Modeling

This section describes methods of building models for device fault processes and battery energy depletion. They are two key factors that determine the lifetimes of individual wireless sensor network nodes.

III.A. Fault Modeling

Node-level fault models relate functionality to time, node characteristics, and node operating modes; they may be used as building blocks to estimate system-level lifetime. Models for node-level fault processes can be obtained in three ways. (1) The node manufacturer may evaluate the reliability of sensor node modules via direct testing and provide a fault model to users [15]. Models obtained in this way, however, may not characterize the in-field behavior if the deployment environment differs from the expected operating environment. (2) Node-level lifetime models may be derived from reports on prior deployments of the nodes under consideration. (3) Finally, it is possible for application developers to experimentally characterize the sensor nodes being considered. This approach allows a controlled testing environment and workload.

We conducted experiments to model the lifetime fault distribution of ultra-compact Eco wireless sensor node [16]. The nodes were used for various wearable applications including infant monitoring, gesture-based input devices, and water pipe monitoring. We wrote programs to test the ADC, radio, and EEPROM node components in the field, and tracked the status of 250 Eco nodes manufactured during June 2007 for 21 months.

Figure 1 shows the accumulated failure rate. Seven global node status evaluations were conducted during this study. Almost half of the nodes failed after 20 months. The Weibull extreme value distribution is widely used in reliability models, and is the appropriate distribution for modeling the first component fault in a node composed of many components with arbitrary temporal fault distributions [17]. We tentatively fit a Weibull distribution to the measured data. Figure 2 shows the log plot of time and $1/R(t)$. $R(t)$ is the reliability function. The Weibull distribution implies a linear relationship between $\ln(t)$ and $\ln(\ln(1/R(t)))$. The resulting Weibull distribution has shape parameter 0.33 and scale parameter 0.02. Its

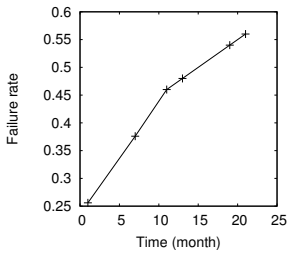

Fig. 1. Device failure of Eco nodes.

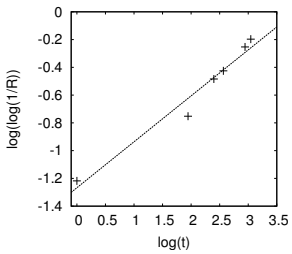

Fig. 2. Fit failure data to Weibull distribution.

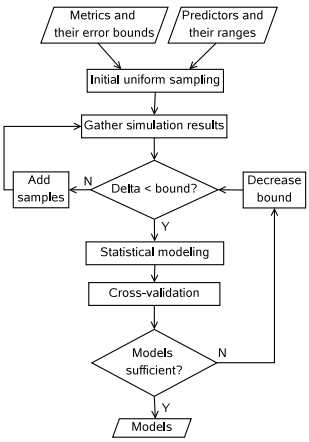

Fig. 3. Overview of the model construction technique.

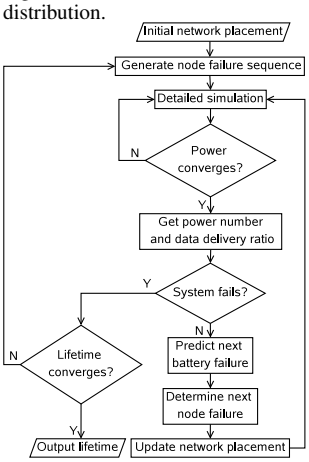

Fig. 4. Monte Carlo simulation for system lifetime distribution computation.

standard residual error is 0.08 and its R^2 is 0.96. The statistical significance test shows that the result is significant (p-value 0.04%). These results indicate that the measured data are consistent with those that would be produced by a fault process with a Weibull distribution. We use the resulting model in our characterization of system-level lifetime.

III.B. Battery Energy Dissipation Modeling

Battery models are used to predict the remaining energy of a battery and node failure time due to battery depletion. We adopt a simple battery model that assumes a constant deliverable energy capacity that is independent of variation in discharge rate. A battery is depleted when the total consumed energy equals the rated battery capacity. This model is accurate when the battery's internal resistance and the device current are low [18]. Most sensor nodes meet these conditions. The proposed model generation technique could easily be used with more complex battery models.

IV. Automatic Model Construction

This section describes our framework to automatically generate models for system-level sensor network performance metrics.

IV.A. Overview

Figure 3 gives an overview of the automatic model construction process, which takes four types of inputs: performance metrics to be modeled (response variables), constraints on prediction error associated with the performance metrics, design parameters (predictor variables), and their associated ranges. It outputs a model for each performance metric. Our model construction technique starts with a sparse and uniformly distributed sample set. It then incrementally adds more samples in rough regions (regions where the magnitude of cost differences for adjacent points are large) according to prior simulation results. The process is iterative and contains two loops. As shown in Figure 3, the first loop iteratively augments the sample set until differences in response variables of already sampled points that are close in the design space are below a threshold. The second loop adjusts the bound parameter if currently derived models do not meet accuracy requirements. Each sample represents a possible value assignment to design parameters. The values of performance metrics for each design are determined with Monte Carlo trials based on detailed sensor network simulations. Statistical modeling is used to fit the simulation results for the sampled points. Cross-validation is used to estimate the prediction errors for the derived models. The procedure terminates when the estimated prediction errors meet the specified requirements. The steps in this procedure will be explained later in this section.

Our framework models multiple performance metrics simultaneously in order to reduce total simulation time. Response surfaces for different metrics may have different shapes. As a consequence, the minimum sample set required to model different metrics may differ. The model may be used by designers with different multiobjective cost functions, making it necessary to consider the surface roughness associated with each metric. However, all the metrics are modeled with the same set of samples. We choose this option for two reasons. (1) The total number of simulation runs depends on the metric that requires the largest number of samples. This technique better utilizes the available simulation results and can therefore generate more accurate models than an alternative technique using subsets of available samples to model different metrics. (2) It reduces implementation complexity. The only disadvantage is that model construction time for some metrics may be longer than necessary. However, since modeling is done offline, this is acceptable.

A wireless sensor network design can be evaluated with various performance metrics. We are interested in developing design tools that are accessible to domain experts who are generally not embedded system experts. To this end, we focus on system-level performance metrics that directly reflect application requirements from a domain expert's perspective. For example, domain experts may have specific requirements for end-to-end data delivery latency, but are rarely interested in node-to-node data transmission latency. System-level performance metrics such as data delivery rate, event miss rate, query response time, and unattended lifetime are affected by numerous factors. Some are specified by domain experts to characterize functionality, requirements, and the operating environment. They are fixed for the application and cannot be adjusted by design tools. Examples are size of deployment field and required sensor readings. Other factors, defined as design parameters (e.g., communication protocols, network size, and node positions) are implementation options that can be determined either manually by the designer or automatically by a design tool. The interdependencies among these factors and their complex impact on system-level performance metrics make deriving accurate closed-form analytical models for them a challenging or intractable problem.

Our technique has the following beneficial features.

1. Using a detailed sensor network simulator allows the use of realistic simulation models, e.g., wireless communication models that consider signal attenuation, interference, and contention.

2. Adaptive sampling and statistical modeling allows production of models that have accuracies comparable to exhaustive on-line simulation. However, only a small part of the design space must be simulated.

3. Our technique can be used to model any system-level performance metric. Our examples consider system lifetime and data latency.

4. The constructed models can be reused by multiple application developers and synthesis tools. The pool of models can be expanded to support new hardware platforms or deployment environments.

IV.B. Sampling Technique

The sampling procedure determines which design points to simulate. Using fine-grained sampling results in a long simulation time, while coarse-grained sampling results in inaccurate models. Adaptively increasing the number of samples can reduce simulation time without sacrificing model accuracy. A straightforward approach is to increase the uniform sampling resolution until accuracy requirements are met. However, this approach has significant drawbacks. Increasing the resolution for any parameter requires either invalidating all prior samples due to the new inter-sample spacing, or requires the resolution for the parameter to double. If uniform sampling is used, doubling the resolution of any parameter is very costly; even adding a single new parameter value requires m new samples, where m is the product of value counts for all other parameters. Finally, uniform sampling may introduce new samples in smooth regions of the parameter space, which will have little impact on accuracy.

We propose an algorithm that starts with sparse uniform sampling and incrementally adding samples to the rough regions. The iteration terminates when the difference in each response variable between adjacent samples is smaller than a threshold. Each iteration of the algorithm does the following. (1) For each sample point, the differences (delta) of output values between its K nearest neighbors and itself are computed. K is an empirically determined variable. (2) If the difference in output value between the sample point and any of its neighbors is larger than the given bound, a new sample is added between them. If there exists no point at the exact middle position due to discretization of some design parameters, the nearest unsimulated point is added. After normalizing each design parameter component of the vector to its range, the Euclidean distance between two samples is used to determine the nearest neighbors.

IV.C. Modeling Technique

We consider two types of modeling methods: global polynomial regression and Kriging.

A polynomial model has the form $y = \beta_0 + \beta_1 t_1 + \cdots + \beta_m t_m + \epsilon$, where y is the response variable, variable t_j is either a single predictor variable or a product of multiple predictors, and each t_j can be raised to a positive power. ϵ is a random error with zero mean. The order of a polynomial model is determined by the maximum of the sum of the powers of the predictor variables in each term of the model. Least-squared error minimizing linear regression is used to estimate coefficients β_j.

Kriging [3] is an interpolation method that minimizes the error of estimated values based on the spatial distribution of known values. A Kriging model is defined as $y(x) = \sum_{j=1}^{N} \beta_j B_j(x) + z(x)$, where $B_j(x)$ is basis function over the experimental domain and $z(x)$ is a random error modeled as a Gaussian process. The general formula is a weighted sum of the data, $y(s_0) = \sum_{i=1}^{N} \lambda_i y(s_i)$, where s_0 is the prediction location, $y(s_i)$ is the measured value at the ith location, λ_i is an unknown weight for the measured value at the ith location, and N is the number of measured values.

The above modeling techniques are implemented in R, open-source software for statistical computing. The following functions are used in our technique: *lm* (linear regression), *Krig* (Kriging), and *cv.lm* (cross-validation).

IV.D. Test of Model Adequacy

The prediction error of the model is estimated with 10-fold cross-validation. The sample set is randomly divided into 10 equal-sized groups. Nine are used as training data and one is used as testing data. We run the 10-fold cross-validation 50 times with different random seeds and average the results. The prediction error for a particular set of testing data is computed with the equation $E = \sqrt{\sum_{i \in T} (y_i^p - y_i^s)^2}/|T|$, where E is the estimated error, T is the testing data set, y_i^p is the predicted value for data point i using a model constructed with the training data, and y_i^s is the simulated value for data point i. When the average error of the 50 tests is smaller than the required maximum error, we deem the model adequate.

IV.E. Wireless Sensor Network Simulation

We use the SIDnet-SWANS simulator [19]. Simulator validation is important because the accuracy of the simulator directly affects the accuracies of the models it is used to build. The simulator itself contains a collection of component models that can be separately validated. The following simulation models have major effects on typical system-level performance metrics: radio propagation model, power consumption model, and network protocol models. SIDnet-SWANS uses an SNR-based reception model validated by Halkes and Langendoen [20]. The error rates for delivery ratio and energy consumption are lower than 5%. The IEEE 802.15.4 implementation in SIDnet-SWANS is ported from ns-2 and was validated by Ivanov et al. [21]. The packet delivery ratio, connectivity graph, and packet latencies have average errors of 0.3%, 10%, and 57%. The power consumption model is based on the power states of MicaZ nodes.

Note that our model construction framework can be used with any sensor network simulator. The accuracies of derived models depend on the accuracy of the simulator in use. However, the focus of our work is not on developing accurate simulators for sensor networks. We therefore assume the error of underlying simulator is ignorable for the remaining analysis and focus on system-level modeling accuracy.

V. System Lifetime Modeling

This section describes the use of the proposed technique to generate a model of system lifetime.

V.A. Domain of Applications and Assumptions

Sensor network applications span a wide domain. Different applications may have very different goals (e.g., data collection vs. object tracking) as well as different performance metrics (e.g., data delivery rate vs. event miss rate). Building one model for each specific application is infeasible since there are numerous applications. We therefore propose to divide the application domain into classes with shared characteristics. In order to select a class of applications for which to generate a system lifetime model, we start with the most frequently encountered type of application: periodic data gathering in a stationary network. Our prior survey of 32 sensor network applications provides evidence that this is the most frequently deployed type of sensor network application [22]. We evaluate our model generation technique for this class of applications. Note that the proposed technique is general enough for use in other domains. Longer model construction time is expected for more complex applications because they require more parameters to represent (e.g., mobile sensor networks need extra parameters to describe motion). We now list our other assumptions. (1) Sensor nodes are homogeneous and have the same lifetime fault model. (2) Sensor node temporal fault distributions are modeled by independent Weibull processes. (3) Sensor nodes are uniformly distributed in a 2D field. (4) A node failure disconnects the affected node from the network. (5) Data from the network are gathered at a sink node located in the center of the field. (6) Data from sensor nodes are routed to the sink using a dynamic data gathering tree. When a parent node fails, its children select other nodes in their communication range with the minimum hop count from the root node as their new parent nodes. (7) We consider two data aggregation cases: perfect aggregation and no aggregation. In the case of perfect aggregation, a single unit of data is transmitted up the routing tree regardless of the number of units of data received from children. In

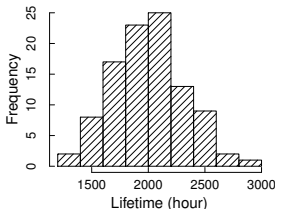

Fig. 5. Histogram of lifetime.

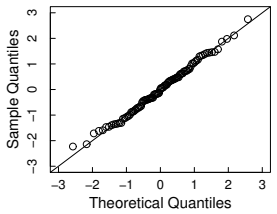

Fig. 6. Quantile-quantile plot of lifetime.

the case of no aggregation, each node transmits a quantity of data equal the sum of received and sensed data quantities. Relaxing these assumptions only requires changing the simulated programs.

V.B. System Lifetime Definition

We define *system lifetime* as the time elapsed since the start of operation until the spatial density of promptly delivered data drops below a threshold specified by the application developer. This definition allows developers to view the system from a data-oriented perspective relevant to their application requirements, while ignoring implementation details such as network structure, communication protocols, and use of redundant nodes. For example, to monitor a field with a large amount of spatial variation in data, the developer may require a higher sampling density. The sampling density criterion cannot be represented with or trivially mapped to other existing criteria. For example, the percentage of functioning nodes or the percentage of connected nodes alone cannot determine the density of data acquisition, because they do not indicate network size, network structure, and packet drop rate.

V.C. Predictor and Response Variables

The system lifetime of a sensor network is affected by many factors, including sensor node reliability, total number of nodes, node positions, node activities, network protocol, battery capacities, power consumptions of components in different power states, etc.

As a case study, we will build a lifetime model for a specific type of hardware platform and assume an outdoor deployment environment. Consequently, some parameters can be assumed to be fixed, e.g., those for radio communication and node lifetime distribution. The proposed technique can be used to build system-level models for various hardware platforms by adjusting the appropriate simulation parameters. Six design parameters are evaluated during simulation: sampling period, network size, distance between adjacent nodes, battery capacity, aggregation, and threshold for desired data delivery density. The predictor variables are independent. They can be separately controlled without affecting each other. However, their impacts on system lifetime are interdependent. We focus on a sub-region of the design space that contains most previously deployed applications. The sub-region further determines the range of each design factor: network size ranges from 9–121 nodes; sampling density threshold ranges from 27–1,000 samples per square kilometer; sampling period ranges from 10 minutes to 1 hour; and inter-node distance ranges from 100–500 feet.

For a specific network design, the system lifetime is best described using a distribution. The network may fail at different times depending on the failure times of individual nodes. Modeling lifetime with a single number, such as mean time to failure, is unnecessarily restrictive. Using a distribution within the model allows application developers to specify confidence levels for lifetime lower bounds.

The Monte Carlo simulation results suggest that system lifetime has a Gaussian distribution. Figures 5 and 6 show the histogram and the quantile–quantile plot of the lifetime for a specific network setting. Results of other network settings show a similar trend and were verified with statistical tests. We therefore assume a Gaussian distribution. We further tested our hypothesis with normality tests, a type of goodness-of-fit test that indicates whether it is reasonable to assume that random samples come from a normal distribution. The average p-value is 0.54 for tests on lifetimes of 100 different design points. According to the test results, we can accept the null hypothesis that the sample data belong to a Gaussian distribution. After determining the distribution of system lifetime, two parameters are sufficient to describe it: mean and standard deviation. These are used as our response variables.

V.D. Monte Carlo Simulation

For each combination of predictors corresponding to a specific network design, we use Monte Carlo simulation to obtain the system lifetime distribution. This procedure is shown in Figure 4. The state of the system corresponds to a particular network topology. A state change in network topology occurs upon each node failure. Each state is associated with a power profile indicating the average power consumption of each node in this state, a residual energy profile indicating the remaining battery energy for each node, and a data delivery ratio indicating the percentage of promptly delivered data. The power profile and data delivery ratio are generated using the SWANS simulator. The remaining battery lifetime of each node is then computed, allowing estimation of the time of the next node failure due to battery depletion. The next battery depletion or node failure event causes a state change. Every time a node fails, it is removed from the network and the updated network placement is used for the next simulation run. Each Monte Carlo trial marches the system through states with decreasing node counts and data delivery ratios. Note that the run does not terminate at a user-specified data delivery ratio. Instead, sufficient data are gathered to build a model that can be evaluated for arbitrary data delivery ratios specified during model evaluation. Trials are repeated (with new, randomized, node fault failure sequences) until the mean lifetime converges.

If it were necessary to do prolonged network simulation for each network state, simulation time would be excessive, rendering the technique impractical. Fortunately, we observe that with a fixed network topology, the power consumption stabilizes within a few sampling periods in the simulated system. Therefore, it is not necessary to run the detailed network simulator until the next node failure. Instead, the network simulator is run long enough to determine average node power consumptions for the current network state. We found that power consumptions converge within three sampling periods for the simulated network. To be conservative, we simulated for five periods.

A Python script coordinates the use of the detailed network simulator for multiple Monte Carlo trials to calculate the system lifetime distribution. Many predictor variable combinations and Monte Carlo trials are required for model construction. Therefore, we run the simulations in parallel on a cluster of machines composed of over 3,500 Opteron cores. The total CPU time required for model construction was approximately 8 weeks, although the task was completed in much less time due to parallelization of the parameter study. The model can be rapidly evaluated on a laptop computer: model use is not computationally demanding.

V.E. Comparison of Modeling Technique Accuracies and Efficiencies

We first compare the performance of polynomial regression and Kriging. Figure 7 shows the relationship between the prediction error and the sample count for applications with and without data aggregation. The x-axis represents the size of the sample set. The y-axis represents the estimated prediction error. The lines labeled “Adaptive regression” and “Adaptive Kriging” represent the errors of a 2nd-order polynomial model and a Kriging model, derived from

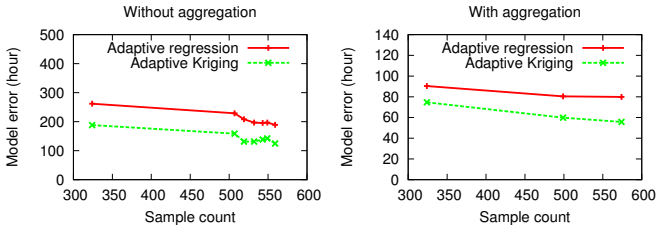

Fig. 7. Model error and sample size.

identical sample sets determined by our adaptive sampling technique. Each point on the lines corresponds to a model generated at the end of a sampling and modeling iteration. Note that the prediction error is estimated with cross validation and is affected by how the data are partitioned. Therefore, the resulting curve is not monotonic. The errors of the polynomial regression models are always larger than those of the Kriging models. On average, the polynomial regression models have 42% larger error than the Kriging models. We conclude that Kriging is more appropriate.

The design space we consider in this case contains 405,790 potential solutions (31 battery capacity levels, 5 network sizes, 11 sampling periods, 17 network densities, 7 thresholds, and 2 aggregation options). Our modeling technique was able to build models with 3.6% average error (absolute error divided by average lifetime) based on approximately 1,100 simulations, i.e., 0.27% of the design space. This demonstrates that the proposed model generation technique is very efficient.

V.F. Comparison with an Analytical Model

To the best of our knowledge, the most relevant is the aging analysis of wireless sensor networks by Lee et al. [13], which focuses on analyzing the degradation in network connectivity due to node-level faults and battery depletion. Their work uses a disc graph model of radio communication and ignores MAC-level behaviors, e.g., contention and collision. Unfortunately, no existing work analyzes system lifetime using our proposed definition. For the sake of comparison, we revert to a definition in past work [13], where lifetime is defined as the time until the percentage of nodes transitively connected to the sink node drops below a threshold. The resulting model has an error of 72 hours (2.1% of average lifetime). In comparison, the average prediction error of the analytical model proposed by Lee et al. is 525 hours (15% of average lifetime).

VI. Conclusions and Caveats

This paper has described an automated technique for generating system performance models for wireless sensor networks, and explained its use to build a system lifetime model for distributed, periodic data gathering applications. We have also proposed a system lifetime definition that captures application-level requirements and decouples specification and implementation. It considers battery lifetimes and node-level fault processes. The proposed adaptive sampling technique allows the generation of lifetime models with only 3.6% error, despite simulating only 0.27% of the solutions in the design space. Taking advantage of more realistic models in sensor network simulators and offline model construction, our modeling technique reduces error by 13% compared with the most advanced analytical model, while supporting rapid model evaluation. Our modeling technique can be applied to other performance metrics. We must comment that the effectiveness of the proposed technique relies on the ability to accurately estimate relevant quality metrics for a number of potential designs, either through simulation or measurement. In future work, we plan to use this modeling technique in automated design of sensor networks.

References

[1] A. Bakshi and V. K. Prasanna, "Algorithm design and synthesis for wireless sensor networks," in *Proc. Int. Conf. Parallel Processing*, Aug. 2004, pp. 423–430.

[2] A. Bonivento, L. P. Carloni, and A. Sangiovanni-Vincentelli, "Platform based design for wireless sensor networks," *Mobile Networks and Applications*, vol. 11, no. 4, pp. 469–485, Aug. 2006.

[3] J. Kleijnen, "Kriging metamodeling in simulation: A review," Tilburg University, Center for Economic Research, Tech. Rep., 2007.

[4] D. E. J. Melo and M. Liu, "Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks," in *Proc. Global Telecommunications Conf.*, vol. 1, Nov. 2002, pp. 21–25.

[5] V. Rai and R. N. Mahapatra, "Lifetime modeling of a sensor network," in *Proc. Design, Automation & Test in Europe Conf.*, Mar. 2005, pp. 202–203.

[6] 2009, http://absynth-project.org.

[7] B. C. Lee and D. M. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," in *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems*, Oct. 2006, pp. 185–194.

[8] B. Ozisikyilmaz, G. Memik, and A. Choudhary, "Efficient system design space exploration using machine learning techniques," in *Proc. Design Automation Conf.*, Jun. 2008, pp. 966–969.

[9] H. Cook and K. Skadron, "Predictive design space exploration using genetically programmed response surfaces," in *Proc. Design Automation Conf.*, Jun. 2008, pp. 960–965.

[10] L. Mounier, L. Samper, and W. Znaidi, "Worst-case lifetime computation of a wireless sensor network by model-checking," in *Proc. Wkshp. on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. ACM, Oct. 2007, pp. 1–8.

[11] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," in *Proc. European Wkshp. on Sensor Networks*, Jan. 2004.

[12] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture," in *Proc. Int. Wkshp. Parallel and Distributed Real-Time Systems*, Apr. 2006, pp. 1–8.

[13] J.-J. Lee, B. Krishnamachari, and C.-C. J. Kuo, "Aging analysis in large-scale wireless sensor networks," *Ad Hoc Networks*, vol. 6, no. 7, pp. 1117–1133, Sep. 2008.

[14] D. Jung, T. Teixeira, and A. Savvides, "Sensor node lifetime analysis: Models and tools," *ACM Trans. on Sensor Networks*, vol. 5, no. 1, pp. 1–33, Feb. 2009.

[15] S. Yamashita, T. Shimura, K. Aiki, K. Ara, Y. Ogata, I. Shimokawa, T. Tanaka, H. Kuriyama, K. Shimada, and K. Yano, "A 15 × 15 mm, 1 μA, reliable sensor-net module: enabling application-specific nodes," in *Proc. Int. Conf. Information Processing in Sensor Networks*, Apr. 2006, pp. 383–390.

[16] C. Park, J. Liu, and P. H. Chou, "Eco: an ultra-compact low-power wireless sensor node for real-time motion monitoring," in *Proc. Int. Conf. Information Processing in Sensor Networks*, Apr. 2005, pp. 398–403.

[17] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Reliability modeling and management in dynamic microprocessor-based systems," in *Proc. Design Automation Conf.*, Jul. 2006, pp. 1057–1060.

[18] D. Linden and T. B. Reddy, *Handbook of Batteries*. MacGraw-Hill, 2002.

[19] O. C. Ghica, G. Trajcevski, P. Scheuermann, Z. Bischof, and N. Valtchanov, "SIDnet-SWANS: a simulator and integrated development platform for sensor networks applications," in *Proc. Int. Conf. Embedded Networked Sensor Systems*, Nov. 2008, pp. 385–386.

[20] G. Halkes and K. Langendoen, "Experimental evaluation of simulation abstractions for wireless sensor network MAC protocols," Delft University of Technology, Tech. Rep., Jan. 2009.

[21] S. Ivanov, A. Herms, and G. Lukas, "Experimental validation of the ns-2 wireless model using simulation, emulation, and real network," in *Proc. on Mobile Ad-Hoc Networks*, 2007.

[22] L. S. Bai, R. P. Dick, and P. A. Dinda, "Archetype-based design: sensor network programming for application experts, not just programming experts," in *Proc. Int. Conf. Information Processing in Sensor Networks*, Apr. 2009, pp. 85–96.