A First Step Towards a Dependability Framework for Smart Environment Applications

Ehsan Ullah Warriach, Tanir Ozcelebi, Johan J. Lukkien
Department of Mathematics and Computer Science
Eindhoven University of Technical
Eindhoven, The Netherlands
{e.u.warriach, t.ozcelebi, j.j.lukkien}@tue.nl

***Abstract*—Smart environments will consist of a large number of heterogeneous devices that communicate to collaboratively perform various tasks for users. We propose a novel dependability framework to increase availability and reliability of smart environment applications. We argue that the key step in achieving high dependability is to predict faults before they occur. Many statistical fault prediction techniques have been proposed for smart environment applications. Selecting the best one among these techniques involves performance assessment and detailed comparison on given metrics. We present a linear regression-based prediction model to predict the remaining battery lifetime of a device to prevent faults due to low battery. Further, we discuss the proposed dependability framework, the basic approaches and the corresponding mechanisms to achieve our long-term research goal. We envision that dependability framework will reduce maintenance costs of large-scale smart environments and increase the dependability of smart environment applications.**

Keywords-smart environments*; *dependability*; *fault-prediction*; *battery fault-prediction model*; *linear regression.

I. INTRODUCTION

A smart environment is a physical space enriched with embedded Information Communications Technology (ICT) and adequate software modules that can communicate their local states, which are adaptive. From a technology point of view, sensor and actuator technologies, as well as communication standards are the main drivers for the development of today's smart environments. A convergence of these technologies raised interest in the smart environment research and its applications such as smart buildings (homes or offices), intelligent lighting, and remote health monitoring [1]–[3].

In smart environments, low capacity sensor and actuator nodes play an important role as they provide the bridge between the digital world and the physical world. A smart environment application relies first and foremost on sensory data acquired from multiple sensors in various locations of the real-world [4]. Sensor nodes are typically small, inexpensive, wireless, and battery-powered devices, prone to faults due to internal and external influences, such as low battery, miscalibration, hardware or software failures, environmental interferences and sensor aging. We define a *fault* as a deviation of at least one characteristic property or parameter of the system from normal operation. Faulty sensors deliver incorrect information to the application and this may lead to incorrect conclusions and consequently application failures, since sensors are usually left unattended for long periods of time in the field. Therefore, the adoption of smart environments is largely hindered by the fact that there is constant need for human (or even expert) intervention and the cost of maintenance of such systems is very high. Thus, dependable systems are required, evolving at runtime to maximize the availability and reliability of their applications. We identify two levels of dependability mechanisms, i.e., proactive mechanisms in the absence of faults and reactive mechanisms in the presence of faults. Systems that have the ability to identify faulty behaviors and make the necessary alterations to restore normal operation without human intervention [5] by means of a reactive dependability mechanism, such as fault tolerance through hardware redundancy, are said to be *self-healing*. On the other hand, systems that utilize proactive dependability mechanisms aim to predict and prevent faults before they occur, or at least delay them. There are two goals of this: *i)* to maintain application functionality as long as possible, and *ii)* graceful degradation of application performance. Fault prediction is required for proactive dependability and is the focus of this paper. It is a key mechanism of the dependability framework proposed in this paper, along with other mechanisms for fault monitoring, adaptation, fault tolerance, fault healing and fault notification.

Several fault prediction models for smart environment applications have been proposed in the literature. However, further research is needed to assess the quality and the resource requirements (e.g., memory, Central Processing Unit (CPU)) of these models. This paper describes work in progress for comparing various models for fault prediction against our proposed linear regression model. Linear regression analysis is one of the most widely used multivariate analysis methods, which assumes linear relationships between independent and dependent variables [6].

Faults can occur due to many reasons. For example, the battery is a critical resource of a battery-powered sensor, and it is one of the most common sources of faulty behavior [7] [8]. A battery powered node may start transmitting faulty values due to low battery [9] [10]. Predicting the remaining battery lifetime can help to use it more efficiently and to predict when a fault is likely to occur, allowing to take actions to prevent it. The need for reliable and accurate battery lifetime prediction

 ISBN: 978-1-61208-341-4

models have been expressed repeatedly in the literature. A battery depletion prediction model was introduced by Kevin et al. [11], where the Received Signal Strength Indicator (RSSI) value is monitored to predict the battery lifetime of sensor because RSSI becomes very low shortly before the depletion of the node battery. Profiling of the battery usage offline to make online predictions was proposed by Wen et al. [12], where the history of average energy consumption rate is used to predict the remaining battery lifetime of mobile devices. Takahashi and Ide [13], proposed a prediction model that uses previous battery usage pattern in the regression function as a trajectory in a feature space to predict the remaining battery lifetime.

In our approach, the discharge rates of running applications on a sensor network are modeled offline and this model is then used to predict the remaining battery lifetime online. We identify the Battery Dependent Components (BDCs) of a sensor that affect energy discharge rate. For example, radio (Transmitting (TX)/Receiving (RX)), Light-Emitting Diodes (LEDs), CPU, and sensor board are prominent BDCs. According to a regression model, remaining battery lifetime is a dependent variable, and energy consumption states of BDCs are independent variables. Our goal is to predict the remaining battery lifetime and take actions to avoid upcoming faults. In our future work, the considered fault prediction models will be evaluated based on a comparison of their resource requirements as well as *precision* and *recall* performance metrics.

The remainder of this paper is structured as follows. We propose the dependability framework for smart environments applications in Section II. The fault-prediction model is presented in Section III to predict the remaining battery lifetime of a sensor node using a linear regression model. The metrics that enable measuring the performance of a fault-prediction model are reviewed in Section IV. Finally, conclusion and future work are presented in Section V.

II. Dependability Framework for Smart Environments

Our long-term research goal is the development of a detailed dependability framework and to evaluate reliability and availability of smart environment applications as a result of self-x (self-protection, self-adaptation, self-healing) properties of the framework. Fig. 1 shows the high-level architecture of the proposed dependability framework, consisting of three main states.

The *fault prevention* state tries to predict faults and prevent the faults (proactive) by adapting the system and the applications based on available resources. The *failure prevention* state attempts to keep the system and the applications functioning with the help of fault tolerance and fault healing mechanism in the presence of detected faults (reactive). The *application failure* state notifies the system administrator to take mandatory actions against the detected fault to bring the system back into a safe state. Main mechanisms of the dependability framework are *i)* fault prediction, *ii)* adaptation, *iii)* fault monitoring, *iv)* fault tolerance, *v)* fault healing and *vi)* fault notification. Fault prediction looks to the future. It is based on monitoring the current state of a system in terms of resource attributes and also considers a history of such state information. Adaptation goal is to prolong the time before either a system or an application reaches a faulty state. Fault monitoring is responsible for monitoring and detecting violations of regular operating constraints (fault) in all hardware, software, and network configurations, as well as identifying the fault type. Fault tolerance refers to the ability of a system to avoid application failures in the presence of faults. Fault healing is the ability of a system to repair, update, or replace the faulty part. After healing, the system returns to a safe state. Fault notification is responsible to notify the user about the fault in a way that it causes minimal disruption to the user activity when it is not healed.

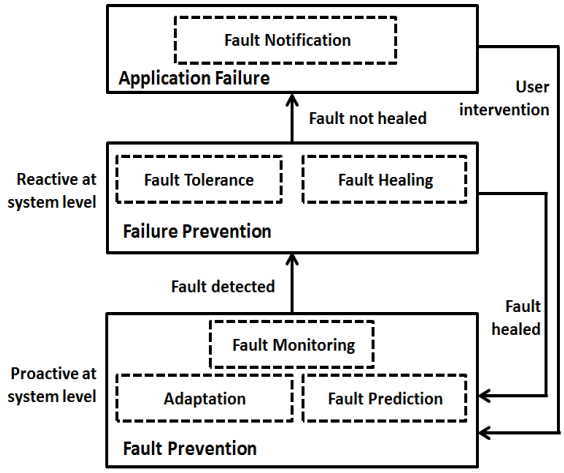

Figure 1: Dependability Framework for Smart Environments

III. Fault Prediction

In this initial phase of our work, we concentrate on the fault prediction model of the proposed framework. In general, given a fault prediction model, the system periodically monitors and logs the current state of the system at run-time and predicts the next state(s). In a smart environment, a set of resource attributes $Y = \{Y_1, ..., Y_K\}$ (e.g., battery levels or memory statuses of devices) are monitored and logged. Further, a number of statistical features $f \in F = \{F_1, ..., F_M\}$ are extracted from the history and the current value of $y \in Y$. The elements of f (e.g., minimum, maximum, expected value, gradient, mean, median, variance) are used in the fault prediction analysis. When a particular resource attribute of a device or the statistical features that correspond to the device are beyond the acceptable range (e.g., defined by thresholds of battery level), the fault prediction invokes the adaptation mechanism.

A fault prediction is defined by the triple $\{F_{type}, t_{PF}^{min}, t_{PF}^{max}\}$, where F_{type} refers to the type of the predicted fault, and $[t_{PF}^{min}, t_{PF}^{max}]$ refers to the interval in which the fault is expected. This is visualized in Fig. 2, where time t refers to a monitoring instance and $\triangle t_{WS}$ indicates how much into the past the fault

Figure 2: Fault Prediction Model Quality

prediction looks. $\triangle t_{LPF}$ is the time until the predicted fault interval. The prediction indicates that the fault of type F_{type} will occur in a time interval of length $\triangle t_{PF} = t_{PF}^{max} - t_{PF}^{min}$. A smaller value of $\triangle t_{PF}$ indicates a more accurate fault prediction model. If $\triangle t_{PF}$ is too large, it is very likely that a predicted fault falls within this determined interval. However, in this case the fault prediction is not useful since the fault can happen anywhere in the large interval.

A. Battery Fault Prediction Model

In this section, we introduce a prediction model for the remaining battery lifetime of a device. As shown in Fig. 3, the prediction of remaining battery lifetime can be divided into two main parts, namely, *offline modeling* and *online prediction*.

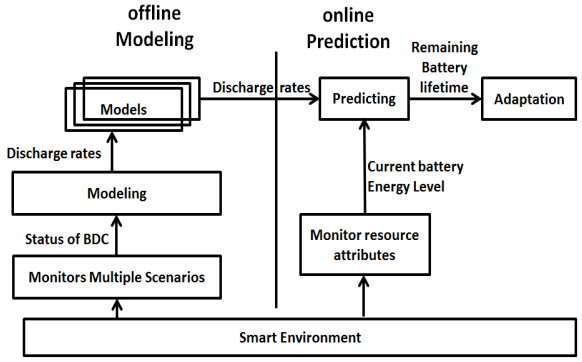

Figure 3: Battery Lifetime Prediction Model

Since the battery discharge rate varies according to the energy consumption of running applications (of BDCs that take a role), battery lifetime is application dependent. We identify a number of BDCs and their possible states, e.g., CPU (active, idle, standby), LEDs, sensor board and radio (TX/RX). The radio component can have different settings of TX and RX modes based on either available battery of a device or application specific. We quantify the relation between BDC states and the battery discharge rate using a linear model, resulting in a multiple linear regression model that employs application specific battery discharge rates. Whenever the application behaviour changes, the fault prediction model needs to be revised using a multiple linear regression model to calculate the current battery discharge rate. These are then used during the *online prediction* process together with the current battery energy level to predict the remaining battery lifetime.

B. Linear Regression Model

Linear regression model [6] has been successfully used for forecasting and prediction in various fields and we consider this model to predict the remaining battery lifetime. Multiple linear regression models the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data [6].

Consider a set of tasks $\tau = \{\tau_1, ..., \tau_N\}$ running on a device, where each task has a battery discharge rate $R = \{R_1, ..., R_N\}$ based on the BDCs energy consumption states $X = \{X_1, ..., X_P\}$ while running that task. The external or internal events can influence the running task and change the state of a BDC. For example, a task can dynamically pick one of the data sampling periods T_1 and T_2, specifying that its battery discharge rate also varies dynamically. Therefore, we need to identify the quantitative relationship between different states of the task and their discharge rate. We consider the states of BDCs as independent variables (X for explanatory) and battery discharge rate as dependent variables (R for response). Linear regression is used to predict the values of the response variables, $(r_1, ..., r_N) \in R$, given a set of explanatory variables $(x_1, ..., x_P) \in X$ (states of BDCs). The relationship between the explanatory variables and the response variables is given by the following equation 1:

$$r_N = \beta_0 + \beta_1 X_{11} + \beta_2 X_{12} + ... + \beta_p X_{1p} + \varepsilon_i \quad (1)$$

where,

$$R = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{pmatrix}, X = \begin{pmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1P} \\ x_{21} & \cdots & x_{2P} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NP} \end{pmatrix}$$

$$\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_P \end{pmatrix}, \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_P \end{pmatrix}$$

- R is a $(N \times 1)$ dependent variable matrix, where N is the number of tasks (discharge rates).
- X is a $(N \times P)$ matrix of independent variables, where $x_{n,p}$ is the state of the p^{th} BDC while task τ_n is running.
- β is a $(P \times 1)$ vector of regression coefficients.
- ε is a $(N \times 1)$ vector of additive random error. We assume that the error ε_i is a statistical error, which is normally distributed with mean zero and variance σ^2, abbreviated as $N(0, \sigma^2)$ [6].

The linear regression function of the battery discharge and BDCs is described using the equation 2:

$$E(t) = e_0 - \beta * t \quad (2)$$

where, e_0 and $(-\beta)$ are the intercept and slope of the line respectively. $\bar{e} = \frac{\sum_{i=1}^{n} e_i}{n}$, $\bar{t} = \frac{\sum_{i=1}^{n} t_i}{n}$, $\beta = \frac{\sum_{i=1}^{n}(t_i - \bar{t})(e_i - \bar{e})}{\sum_{i=1}^{n}(t_i - \bar{t})^2}$ and $e_0 = \bar{e} - \beta * t$, β measures the change in the mean of E for a unit change in t, which is the discharge rate of the battery [6]. In order to explain the battery lifetime prediction model, let us suppose we have a set of data samples $(t_i, e_i), i = \{1, 2, ..., n\}$, as shown in Fig. 4.

It shows the prediction of the battery lifetime of a sensor, where e is the value of battery and t is the time. Suppose,

 ISBN: 978-1-61208-341-4

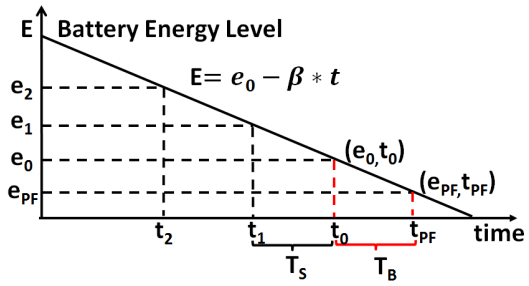

Figure 4: Estimation Curve of a battery discharge rate

a system monitoring module measured the battery energy level e_0 at the time t_0, which is beyond the acceptable range. Then, battery lifetime prediction model is invoked to determine the remaining battery lifetime against the current status of BDCs states and using the current battery energy level e_0. Consequently, we can have an estimation of remaining battery lifetime using the discharge rate. We assume that the battery energy level is e_{PF} at time t_{PF}. Then, by observing Fig. 4 and by applying following Equation 3, we can measure the remaining battery lifetime:

$$T_B = t_{PF} - t_0 = \frac{e_0 - e_{PF}}{\beta} \quad (3)$$

IV. Evaluation of Fault Prediction Model

In order to investigate the quality of fault prediction model and to compare various fault prediction models against our proposed linear regression model it is required to identify suitable metrics. The goal of a fault prediction model is to predict faults accurately, efficiently and in a timely manner. An accurate fault prediction model would accomplish a one-to-one matching between predicted and true faults. A fault prediction is a *True Positive* (TP), if a fault occurs within the predicted period. If no fault occurs and a fault is predicted, the prediction is a *False Positive* (FP). If the model misses to predict a true fault, it is a *False Negative* (FN). If no true fault happens and no fault notice is given, the prediction is a *True Negative* (TN). Further, we consider precision and recall based on above metrics. *Precision* is defined as the ratio of correctly identified faults to the number of all predicted faults $precision = \frac{TP}{TP+FP}$. *Recall* is the ratio of correctly predicted faults to the number of true faults $recall = \frac{TP}{TP+FN}$ [14]. *Accuracy* is the number of correct predictions over the total number of predictions made. Further, we will investigate the *computational* requirements of fault prediction models, e.g., memory and CPU.

V. Conclusion and Future Work

We presented a high-level architecture of a dependability framework for smart environment applications. In this context, we consider the problem of predicting faults in smart environments. As a first step, we presented a prediction model based on the multiple linear regression model for the remaining battery lifetime of a device. In order to develop an accurate and efficient fault prediction model, we must understand the trade-offs among the metrics defined in Section IV for each prediction model and choose the best tradeoff for a given dependable framework for smart environments. Our future work will focus on implementing this architecture on top of an operational platform in a real smart environment in order to guarantee the availability and reliability of applications. The ability of applications in a system to survive free of faults depends on adaptations supported by the dependability architecture. We recognize that the system may be affected by many types of faults. Thus, we will investigate optimal application adaptation mechanisms as well as fault prediction models against other types of faults, e.g., low memory, link quality, hardware or connection failures, miscalibration. Our ultimate goal is to develop a dependability framework for smart environments to provide users with services, which are highly reliable and available.

Acknowledgment

This work has been supported by the ProHeal project (n. 10017751) funded by the Information Technology for European Advancement (ITEA2).

References

[1] D. Cook and M. Schmitter-Edgecombe, Assessing the quality of activities in a smart environment, Methods of Information in Medicine, vol. 48, no. 5, 2009, pp. 480-500.

[2] P. Yu, X. Ma, J. Cao and J. Lu, Application mobility in pervasive computing: A survey, Pervasive and Mobile Computing, vol. 9, no. 1, 2013, pp. 2-17.

[3] S. Bhardwaj, T. Ozcelebi, O. Ozunlu, and J.J. Lukkien, Increasing reliability and availability in smart spaces: A novel architecture for resource and service management, IEEE International Conference on Consumer Electronics (ICCE), 2012, pp. 439-440.

[4] F. L. Lewis, Wireless Sensor Networks: Smart Environments, John Wiley & Sons, Inc., 2005, pp. 11-46.

[5] D. Ghosh, R. Sharman, H. Raghav Rao, and S. Upadhyaya, Self-healing systems - survey and synthesis, Decision Support System, vol. 42, no. 4, 2007, pp. 2164-2185.

[6] L. Wasserman, Lecture Notes for Linear Regression, 2010, Retrieved 08-04-2014. [Online]. Available: http://www.stat.cmu.edu/ roeder/stat707/lectures.pdf.

[7] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, Lessons From A Sensor Network Expedition, European Conference on Wireless Sensor Networks, 2004, pp. 307-322.

[8] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, Wireless sensor networks for habitat monitoring, 1st ACM international workshop on Wireless sensor networks and applications, vol. 2, 2002, pp. 88–97.

[9] E.U. Warriach, M. Aiello, and K. Tei, A Machine Learning Approach for Identifying and Classifying Faults in Wireless Sensor Network, IEEE 15th International Conference on Computational Science and Engineering (CSE), 2012, pp. 618-625.

[10] N. Kevin et al., Sensor Network Data Fault Types, ACM Transaction Sensor Network, vol. 5, no. 3, 2009, pp. 25:1–25:29.

[11] I. H. Yano, V. C. Oliveira, E. A. de Mello Fagotto, A. A. Mota, and L. T. M. Mota, Predicting battery charge depletion in wireless sensor networks using received signal strength indicator, Journal of Computer Science, vol. 9, no. 7, 2013, pp. 821-826.

[12] Y. Wen, R. Wolski, and C. Krintz, Online Prediction of Battery Lifetime for Embedded and Mobile Devices, Power-Aware Computer Systems - Lecture Notes in Computer Science, vol. 3164, 2005, pp. 57-72.

[13] T. Takahashi, and T. Ide, Predicting battery life from usage trajectory patterns, 21st International Conference on Pattern Recognition (ICPR), 2012, pp. 2946-2949.

[14] F. Salfner, M. Lenk, and M. Malek, A Survey of Online Failure Prediction Methods, ACM Computer Survey, vol. 42, no. 3, 2010, pp. 10:1–10:42.

 ISBN: 978-1-61208-341-4