Lifetime Evaluation of Wireless Sensor Networks

Wg Cdr Varghese Thattil (Retd)[1] and Dr. N Vasantha[2]
[1] CVR College of Engineering, Department of ECE, Ibrahimpatan, R.R.District, A.P., India
Email:mailthattil@gmail.com
[2] Vasavi College of Engineering, Hyderabad, A.P., India
Email: vasanthavasantha@rediffmail.com

Abstract—**Wireless Senor Networks (WSN) are coming of in age and are being installed in many applications. Some of the common monitoring applications are seismic, volcano eruption, tsunami, structural, intruder detection, health, habitat etc. Most of these applications of WSN are event driven wherein the system should be able to sense an occurrence of an event at an unknown future instant of time when it occurs. If the WSN system fails to detect the event at the time of occurrence the entire deployment of WSN fails to achieve its purpose. WSN has a specified life time and therefore in order to continue the operation of monitoring; it is essential to know the operational life period of WSN system. Knowledge of this will help in scheduling and planning the required redeployment. WSNs are highly resource constrained systems and most of the research in the WSN has been carried out to improve the performance under high resource constraints. The estimation of useful life of Wireless Sensor Network can be carried out by preparing a fault model of the WSN system. The system fault model can be used to predict the system survivability. The paper discuses the WSN fault model.**

Index Terms—**Wireless Sensor Networks, Quality of Service, Fault model**

I. INTRODUCTION

As the Internet has revolutionized our life via the exchange of diverse forms of information readily among a large number of users, Wireless Sensor Networks (WSNs) is expected to revolutionize to provide "ambient intelligence" where many different devices will gather and process information from many different sources to both control physical processes and to interact with human users. WSNs will also be equally significant by providing information regarding the physical phenomena of interest and ultimately being able to detect and control them or enable us to construct more accurate models of the physical world. With the recent advances in the Micro electromechanical Systems (MEMS) Technology, sensors are becoming smaller and affordable. More and more applications are being introduced using WSNs to monitor environment, industrial process, battlefield, seismic, health, habitat etc.

While a lot of research has been done on some important aspects of WSNs such as architecture and protocol design, energy conservation, routing, localization etc.; not much work has been carried out to generate a system model for WSN and thus estimate the survivability of the WSN system. This is mainly because WSNs are very different from traditional networks. In the Internet, the network and transport layer protocols ensures end-to-end reliability where as in the case of WSNs this will not be optimal because of the unique constraints like energy, memory, computational power etc. Further sensor networks as a whole has a specific task to be carried out depending upon the application and therefore the WSN system models will be application specific. These models then can be utilized to carry out various performance evaluations.

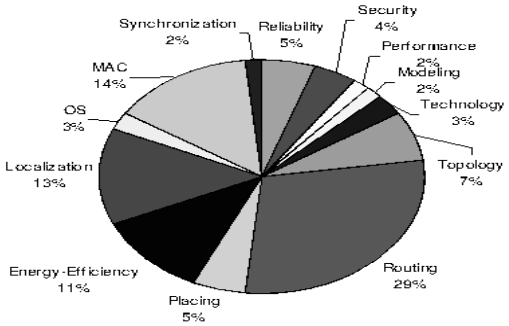

Figure 1. Classification of areas of research in WSN

Figure 1 shows a classification of remarkable papers on WSNs, published on several leading IEEE and ACM journals and conference proceedings [1]. As one could expect, our study evidences that the 67% of the research efforts have been carried into routing protocols (29%), MAC protocols (14%), localization strategies (13%), and energy efficiency (11%). Only the 5% of the considered literature is related to WSNs reliability issues, and none of them explicitly addresses fault forecasting issues. The reliability of wireless networks has been addressed primarily in the context of quality of service (QoS). The main considerations have been routing and the overhead taking care of energy consumption and broken communication paths. However, a survey of literature shows that hardly any attempt has been made to estimate whether the WSN, as a system will be able to detect an event if it occurs within a specified period of time, if so with what confidence level.

In order to study and address Quality of Service (QoS) issues in service-oriented systems, we need a model of the system in question. Such a system-model allows us to study important properties of the system. Systems can be modelled at various levels of abstraction, ranging from abstract mathematical frameworks such as stochastic processes or queuing networks to system testbeds, i.e. physical systems equipped with measurement and experimentation infrastructure.

 DOI: 10.32377/cvrjst0107

Ideally, models at different abstraction levels should be used, as different models can often complement each other. A queuing-network model of a system, for instance, may be used to efficiently study a large space of parameters, and thereby arrive at general conclusions. A testbed-model of the same system, in contrast, enables measurements under realistic conditions, which serve to validate the more abstract model, and to improve the quality of the conclusions by providing realistic model parameters.

Irrespective of their abstraction level, all system-models allow us to study properties of the system. When studying QoS issues we are particularly interested in the behaviour of the system under various common faults or disturbances. For instance, in a queuing-network model we may compute job completion times, while in a testbed we may measure response-times. Fault-models are considered as parameters to a system model that influence the modelled system's QoS.

In order to use a system-model to study the effect of faults on a system, we must be able to introduce models for these faults into the system-model. Since with some model classes the system-model may change significantly when a model for a fault is introduced into it; we required to obtain the same measures with the same interpretation from the system-model, regardless of the employed fault-model. That is, using the terminology of functional and non functional behaviour, we require that the system-model maintains the same functional behaviour and the same system structure when we change the fault-model. This clear distinction between the fault-model and the system-model is common in fault-injection experiments for dependability benchmarking, where one explicitly describes a fault-load that the system is subjected to.

Fault Models: Many different types of faults have been defined, some having orthogonal properties [2]. For example, failstop behavior implies that the faulty system ceases operation and alerts other processors of this fault. Crash faults, on the other hand, assume that the system fails and looses all of its internal state, e.g. the processor is simply down. One speaks of omission faults when values are not delivered or sent, e.g., due to a communication problem. If outputs are produced in an untimely fashion, then one speaks of a timing fault. Transient faults imply temporary faults, e.g. glitches, with fault free behavior thereafter. If transient faults occur frequently, one speaks of intermittent faults. This set of fault types is by no means complete and serves only as a basic introduction. The definition of faults seems to change with the application domain. For instance, fault models suitable for computer dependability may not necessarily match the behavior of network and computer security applications.

The behavior of the faults with respect to other processors can be described in simpler models which have been used with in replication and agreement algorithms. Specifically, fault models have been considered whose main behavior types are benign, i.e., globally diagnosable, symmetric (faulty values are seen equal by all non-fault processes) and asymmetric or malicious, i.e., there are no assumptions on the fault behavior [3].

The faults are generally divided into following three types:

Bernoulli Trials: Fault is described as a Bernoulli-distributed random variable. That is, the occurrence of the fault is defined by a probability p. The typical examples for this type are fault-models that reflect service availability/unavailability,

General Random Variables: Fault occurrence in this case is described by a random variable with a distribution that is more general than Bernoulli. The distribution is described by a distribution function (cumulative distribution function, CDF), a complementary CDF (CCDF) or a probability density function (PDF).

Stochastic Processes: A stochastic fault is a fault whose occurrence or non-occurrence is predicted by one or more random variables. It is not possible to show the occurrence or non-occurrence of a stochastic fault by a logical argument based on the design of the component. That is, we cannot apply fault prevention. What we can do with a stochastic fault is apply the laws of mathematical probability to predict its likelihood.

II. FAILURES IN WIRELESS SENOR NETWORKS

Wireless sensors Network may have many nodes deployed with each node having different sensors. Each service running on node is expected to periodically send the measurements of its sensors to an access point. If the camera in one of the sensor node stops scanning and if the node has not been designed to detect it and overcome the situation; it has reached an erroneous state. The sensor node thus not able to send the accurate data to the access point causing a failure at the node as observed from the access point. Here the defect in the camera is a fault, the nonavailability of scanning is the incorrect state and Access point observing the stationary camera is the failure.

Sources of Faults in WSNs: Data delivery in sensor networks is inherently faulty and unpredictable. Failures in wireless sensor networks can occur for various reasons [4].

1. Sensor nodes are fragile, and they may fail due to depletion of batteries or destruction by an external event. In addition, nodes may capture and communicate incorrect readings because of environmental influence on their sensing components.
2. As in any ad hoc wireless networks, links are failure-prone, causing network partitions and dynamic changes in network topology. Links may fail when permanently or temporarily blocked by an external object or environmental condition. Packets may be corrupted due to the erroneous nature of communication. In addition, when nodes are embedded or carried by mobile objects, nodes can be taken out of the range of communication.
3. Congestion may lead to packet loss. Congestion may occur due to a large number of nodes' simultaneous

transition from a power-saving state to an active transmission state in response to an event-of-interest.

4. Faults also occur because of the multihop nature and mobility of the nodes which can cause link failures

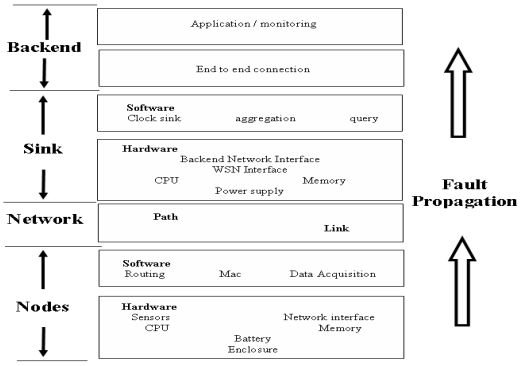

Figure 2. Fault propagation in WSN

Faults in Wireless Sensor Networks can occur at several protocol layers of the system. Effect of the fault in a particular layer can propagate to other layers in the system. Faults normally propagate from the nodes through the network to the sink. The figure 2 shows how the fault propagates through various protocol layers and components. It can be seen that in a sensor node; the sensors, processor (CPU), memory, Network interface, battery, enclosure for the node all of them can initiate a fault. Another source of faults in sensor node is software programmes that carry out routing, data acquisition, Medium Access Control etc. In the network layer the faults can be due to parameter variations in link, path, environment, location, etc.

Power management in wireless networks is an essential factor for their smooth function. Wireless nodes, especially sensors, use small batteries for energy supplies that in many cases cannot be replaced. Therefore, energy conservation is a vital factor in a sustained network lifetime.

The battery: lifetime determines how long one can use a device. Battery modeling can help to predict, and possibly extend this lifetime. Battery models are combined with a workload model to create a more powerful battery model. WSN devices rely on battery energy to work. The energy stored in these batteries is limited. So, it is important to use this energy as efficiently as possible, to extend the battery lifetime. The lifetime of the battery as the time one can use the battery before it is empty. Note that, for rechargeable batteries, this is not the same as the time one can use the battery before it stops working properly.

The battery lifetime mainly depends on the rate of energy consumption of the device. However, lowering the average consumption rate is not the only way to increase battery lifetime. Due to nonlinear physical effects in the battery, the lifetime also depends on the usage pattern. During periods of high energy consumption the effective battery capacity degrades, and therefore the lifetime will be shortened. However, during periods without energy consumption the battery can recover some of its lost capacity, and the lifetime will be lengthened.

Energy consumption of wireless devices has been studied using performance models. These models describe the various states a device can be in, and the energy consumption rate in these states. However, typically these models only take the energy consumption into account and do not deal with the effects of the usage pattern on the battery lifetime. To be able to do this we have to extend the model, by combining it with a battery model.

In stochastic models of the battery it is described in an abstract manner where the discharging and the recovery effect are described as stochastic processes.

III. FAULT SCENARIOS IN WIRELESS SENOR NETWORKS

The node faults can be classified into two types: *permanent* and *potential*. The permanent fault completely disconnects the sensor node from other nodes and brings eternal impact on the network performance like in the case of hardware faults within a component of a sensor node. A permanent fault once activated remains effective until it is detected and handled. The impact of this failure is usually measured when assessing the network performance. On the other hand, a potential fault usually results from the depletion of node hardware resource, i.e. battery energy. Such fault might cause the node sudden death, and eventually threaten the network life time. When the battery depleted, a node is useless and cannot share in sensing or data dissemination. Potential failure can be detected and treated before it causes the sudden death of a node e.g. sensor node with low residual energy can be send to sleep mode before it completely shuts down and disrupt network operation. Faults can be further classified into: *node level fault* and *network level fault*. Node level fault represents the potential and permanent failure of a node while "network level" describes the network faults caused by either potential or permanent failure of one or a set of sensor nodes. These are shown in figure 3.

Individual node level fault usually results from: application software misbehaviour, hardware failure and external impact of harsh environmental conditions (direct contact with water causing short circuit, node crash by a falling tree etc). The network level faults are as a result of either the potential or permanent failure, and are usually related to the network connectivity, and sensor coverage rate.

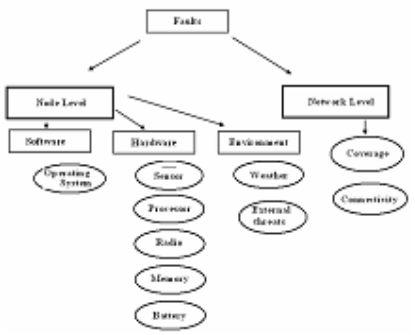

Figure 3. WSN Fault scenario

Nodes measure some physical quantities close-by them and transmit the information to the base station. Nodes can both transmit and receive information. The final target of the information transmitted by nodes is the base station.

IV. MODELLING TOOLS

Performance and dependability modeling is an integral part of the design process of many computer and communication systems. A variety of techniques have been developed to address different issues of modeling. For example, combinatorial models were developed to assess reliability and availability under strong independence assumptions; queuing networks were developed to assess system performance; and Markov process-based approaches have become popular for evaluating performance with synchronization or dependability without independence assumptions. Finally, simulation has been used extensively when other methods fail.

In order to harness full strength of any software modeling tool to solve for measures of interest of stochastic discrete event one must first understand what the modeling process is. That is once specification of the real system is known; one must know how to specify the model in a particular formalism. That requires knowledge of both the system to be modeled and also the formalism in which the system is to be specified. Petri nets are widely used to model and analyze the system behavior which provides graphical representation of the system state changes. The modeling requires knowledge of probability measure, conditional probability, continuous random variables, discrete random variables, PDF (probability density function), CDF (cumulative density function), Stochastic processes, including Markov processes, continuous time Markov chains (CTMC), discrete time Markov chains (DTMC), state transition rate matrices, generator matrices etc.

As solving models became more and more complex; different formalisms (or formal languages for expressing models) were also developed. Each of this formalism has its own merits. Some formalisms afford very efficient solution methods; like BCMP [5] queuing networks admit product-form solutions, Other formalisms, such as Stochastic Petri Networks (SPN) were developed provide a simple elegance in their modeling primitives, while superposed generalized stochastic Petri nets (SGSPNs) and colored GSPNs (CGSPNs) [6] yield state-space reductions. A number of extensions, such as stochastic activity networks (SANs) [7], were developed for compactly expressing complex behaviors.

Along with formalisms, modeling tools also have been developed. A tool is generally built around a single formalism and one or more solution techniques, with simulation sometimes available as a second solution method. Some of the tools developed are DyQN-Tool+ which uses dynamic queuing networks as its high-level formalism; GreatSPN which is based on GSPNs, UltraSAN, which is based on SANs, TANGRAM-II, which is an object- and message-based formalism for evaluating computer and communication systems. While all of these tools are useful within the domains for which they were intended, they are limited in that all parts of a model must be built in the single formalism that is supported by the tool. Thus, it is difficult to model systems that cross different domains and would benefit from multiple modeling techniques [8].

Performance and dependability modeling software tools have become increasingly powerful in recent years. Engineers have the ability to model increasingly complex systems using only a moderate amount of computing resources. However, despite the technological advances in system modeling, there remain several obstacles hindering the prediction of system behavior. Two facts contribute to these obstacles: the fact that system models have grown in both scale and intricacy of detail, and the fact that modeling software tools do not provide the appropriate feedback to the engineer during the design process.

As system models become more elaborate, the number of variables that can be parameterized increases rapidly. Difficulties arise from having such a large number of model parameters. The main problem with modeling large systems is in deciding how to make the best use of the computing resources available. The first step in optimizing use of available resources is reducing the number of model parameters to vary. Varying model parameters that do not contribute to the reward variables being measured wastes experimentation time. Selection of the parameters to vary requires detailed knowledge of the underlying model, and often relies on the designer's intuition and experience with similar systems. Another way to optimize use of computing resources is to reduce the number of values assigned to the model parameters. Increasing the number of values for a particular model parameter requires more experiments if all values are to be tested. If there are several parameters being varied, then the number of experiments needed to test each combination can grow to an unmanageable number. The engineer should focus on a range of parameter values over which the reward variables are expected to change significantly. Again, the art of modeling requires detailed knowledge of the model, which is best gained from previous experience. Unfortunately, software tools cannot automatically grant an engineer intuition, but they can

provide valuable information that, over time, can be used to develop a knowledge base useful for solving future problems.

The second problem contributing to the engineer's difficulty with efficiently predicting system behavior is the fact that today's modeling tools do not provide the engineer with the necessary feedback during the design process. The *design space* consists of all possible system configurations. Each configuration consists of unique values assigned to each system parameter. The experimentation process is often iterative, consisting of several sets of experimental runs, each producing results requiring analysis. This process ends when the desired system configuration is obtained.

In general, it is desirable to minimize the amount of experimentation time needed to determine the desired configuration. After running a set of experiments, the engineer may find that the results do not meet the desired specification. Further experimentation is necessary to find which model parameter values are acceptable. It would be useful if the modeling software could analyze the results and provide information suggesting which parameter values to choose for future runs. This iterative feedback would help the engineer to efficiently arrive at a desired model configuration. Without such feedback, the engineer may incorrectly guess which values cause the model to converge to the desired specification, resulting in a waste of experimentation time. The feedback obtained during this efficient navigation of the design space can also be used to reveal less expensive model configurations that meet the specifications, rather than exceed them.

CONCLUSIONS

The paper has brought out various issues concerned with the system modeling with special reference to wireless sensor networks. The system modeling is essentially required for the estimation of the survivability of mission critical applications. It can be seen that a modeling engineer not only need to have in depth knowledge of the system under consideration but also very good understanding of statistical and probability measures. The modeling normally uses stochastic processes, including Markov processes, continuous time Markov chains (CTMC), discrete time Markov chains (DTMC), state transition rate matrices, generator matrices etc. Once a satisfactory model of the system is arrived at it is to be verified with test bed or other statistical data and accordingly the system model is modified till a final system model is arrived at.

REFERENCES

[1] Marcello Cinque, Domenico Cotroneo Gianpaolo De Caro, Massimiliano Pelella, "Reliability Requirements of Wireless Sensor Networks for Dynamic Structural Monitoring" www.mobilab.unina.it/projects/StragoResults /WASR-06_Cinque.pdf

[2] A. Avizienis, J.C. Laprie and B. Randell, Fundamental Concepts of Dependability, Information Survivability Workshop (ISW-2000), Boston, Oct. 24-26, 2000.

[3] Axel W. Krings University of Idaho Moscow, Idaho 83844-1010, USA, krings@ uidaho.edu, "Fault-Models in Wireless Communication: Towards Survivable Wireless Networks", *citeseerx.ist.psu.edu*

[4] Lilia Paradis and Qi Han; "Dealing with Faults in Wireless Sensor Networks" http :// inside.mines.edu /~qhan/students / Lilia/docs/ WSN Faults Survey.pdf

[5] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. "Open, closed, and mixed networks of queues with different classes of customers"' Journal of the Association for Computing Machinery, 22(2):248–260, April 1975.

[6] S M. Ajmone Marsan, Dipartimento di Scienze dell', Stochastic Petri Nets: An Elementary Introduction In Proceedings: European Workshop on Applications and Theory in Petri Nets, Year: 1988, Pages: 29

[7] William H. Sanders1 and John F. Meyer, Stochastic Activity Networks: Formal Definitions and Concepts, William H. Sanders1 and John F. Meyer, *citeseerx.ist.psu.edu/viewdoc*

[8] [W. H. Sanders. Integrated frameworks for multi-level and multi-formalism modeling. In Proceedings of the 8th International Workshop on Petri Nets and Performance Models, pages 2–9, Zaragoza, Spain, September 1999.