Modeling the Failure Behavior of Wireless Sensor Networks

M. Cinque¹, D. Cotroneo¹, C. Di Martino¹, S. Russo¹

1: Dipartimento di Informatica e Sistemistica
Università di Napoli Federico II
Via Claudio 21, 80125 - Napoli, Italy
macinque,cotroneo,catello.dimartino,stefano.russo@unina.it

Abstract

The increasing computation and communication capabilities of recent wireless sensors have broadened the possibility of application of Wireless Sensor Networks (WSNs) in a wide class of application scenarios. A single wireless sensor node plays nowadays a complex role into the network, which goes far beyond the traditional “data dissemination” objective: recent goals involve more and more critical tasks when hypothesizing the use of WSNs in health care, object tracking, control of smart buildings, and dynamic monitoring of civil structures. In this context, the dependability becomes a fundamental requirement that cannot be neglected all over the life of a WSN, from the conception to operation. To this aim, it becomes crucial to define novel techniques able to ease the design and validation processes of WSNs, with particular emphasis on dependability aspects. The solution proposed in this technical report is based on the simulation modeling approach. A detailed failure model for WSN is presented by using the Stochastic Activity Networks formalism. The model is derived from the study of the state-of-art, and it is parametric, in that the value of its parameters can be specialized according to the particular network topology and environmental conditions.
1 Introduction

The aim of this report is to describe a failure model of Wireless Sensor Network (WSN), which is able to characterize the behavior of single node and of the whole network under failures. From the existent literature, it is known that both empirical and simulative approaches appear inadequate to analyze the behavior of a WSN under abnormal operating conditions [1]. As demonstrated in [2], the problem of measuring dependability attributes of a Wireless Sensor Network, such as reliability, is NP-hard. This is due to the difficulty of i) discriminating a network failure from the failure of one or more nodes; and ii) analyzing the dependency of a single node failure from the topological and the behavioral characteristics of the infrastructure (such as the used routing algorithm and the nodes positions on the interested area).

In this report we model the failure behavior of a Wireless Sensor Network by relating single contribution of a node failure to the entire topology of the network, including the routing algorithm being used. The model is parametric in the sense that its parameters are taken by field measurement campaigns and/or by external simulative tools. The model is modular and structured in order to address all issues, separately. Such a model takes into account behavioral characteristics (e.g. software application installed on the nodes) and structural ones (e.g. topology) as well as hardware nature (e.g. hardware failure model, batteries natures) and the installed software (e.g software failure model, software induced energy consumption, etc.).

Figure 1: Arranged approach to the modelation activity

2 Motivations

The adoption of the failure model allow to use fault-forecasting techniques to evaluate the presence, the creation, and the consequences of faults. However, given the complexity of the considered networks, the only fault-forecasting technique is not enough to guarantee
a reliable and accurate estimation of dependability attributes [3]. This uncompleteness would be, however, limited using a parametric approach where a general failure model is populated by experimental data, obtained by behavioral simulations and/or by filed-based campaign. A summarizing schema of the considering model is reported in fig 1.

The dependability measures, obtained in this way, are able to pinpoint bottlenecks in the system reliability, thus leading the network installer to adopt fault-tolerant techniques and, more in general, dependability improvement strategies. Moreover, having a failure model allows to validate and to evaluate the goodness of the used strategies for fault-tolerant, using fault-injection techniques. This report is focused on the third phase of the approach, illustrated in fig 1.

3 Modeling approach

Given the strong complexity of the reality to model, it is important to adopt a structured approach to realize a formal model, such as to allow (Russo et al. [4]):

1. availing the adopted formal model potency;
2. passing to a semi-formal and intuitive (states and events) notation, simple to comprehend and use;
3. allowing a definition of the considered system behavior.

The principal property of the adopted modeling approach is to infer the model in a systematic and modular manner, beginning from a functional and behavioral description of the networks and its nodes. This method aims at specifying the behavior of each component (nodes, sensors board, radio modules, etc) and their interactions separately as event sequences, in which they are involved ([5]).

The method steps are the following:

1. From the reality description in natural language, it is derived a list of components, and for each one of these, a list of events, states and logical and temporal constraints. The reality description in natural language is followed by the model assumptions, or rather a sub-set of all possible states, events and constraints for the considered model.
   A state is defined identifying the events that trigger a transition on it. The logical constrains are typically imposed on the event sequence in which each component is involved. The temporal constrains concern information on the duration of events depended by temporal parameters (for example node wakeup timer) and on their statistic characterization (temporal distribution of event happening).

2. For each identified components A sub-model is realized and the previous step is repeated with a lower abstraction level, until identifying the "‘atomic’" events and components.\(^1\)

\(^1\)The atomic term is used to identify components and/or events that is not no more important or possible to investigate, because the results will not be useful for the model.
3. For each obtained sub-model, a list of proper functional bonds and "interfaces" is identified.

4. A schema for each sub-model is inferred, translating the information states, events and functional requirements.

5. Each sub-model schema is transformed in the correspondent stochastic network, considering timing constraints as statistical characterization of the modeled events (rate and timing distribution).

6. The overall model is achieved by composing "atomics" sub-models by means of their own interface identified at point 2. Overall model solution come out from single models solutions conveniently arranged by means of the interfaces.

4 General assumptions

From the functional and behavioral description proposed in [7], we derive the following modeling assumptions on the reality under study:

- each node can be considered as composed by:
  - a processor board;
  - a sensor board composed by one or more sensors (e.g. temperature, humidity etc.);
  - a radio board;
  - a set of batteries.

- Initially all the nodes have the same capabilities and characteristics;

- every node’s position is stationary;

- every node is equipped with an unique identification (id) which characterize the node network wide;

- several nodes clustered in the same area can create a region or group;

- every node belongs exactly to a single region;

- every node is connected only to his own neighbors or, to be more precise, to all the nodes that lies into its transmission range.

5 The model

A Wireless Sensor Network is here modeled as a set of homogeneous entities (nodes) which constitute vertex of an oriented weighted graph $G(V, E, W)$, where:

- $V=\{v_1, v_2, ..., v_n\}$ is the set of all vertex of the graph (network nodes). The footer of the element $v_i$ is related to the identification of the node which is associated.

---

2In that paper, interface is meant as a set of state, shared between sub-models, in accord to the semantic of Stochastic Activity Networks [6].
• E is a square unitary matrix \( nxn \), where the element \( e_{ij} = 1 \) if a path from node i to node j exists, 0 otherwise.

In this context:

• W is a square matrix \( nxn \), where the generic element \( w_{ij} \) reports the quality of direct link (lqi) between nodes i a j;

**Definition 5.1** A direct path from node i to node j is defined:

- **bad** ⇔ a probability of sending one packet from i to j successfully is greater than 0.3 and less than 0.7, or in other words:
  \[ 0.3 < P(\text{successfully} \mid \text{sending } i \rightarrow j) < 0.7; \]
- **good** ⇔ a probability of sending one packet from i to j successfully is greater than 0.7, , or in other words:
  \[ P(\text{successfully} \mid \text{sending } i \rightarrow j) \geq 0.7. \]

A link between node i and node j exists ⇔ there is at least a direct bad path bad which connect them;

**Definition 5.2** Consider two ”‘neighbor”’ nodes, i and j, a ”‘quality link estimator”’ indicated by \( lqi_{ij} \) can be defined as a function of the number of failures which are detected during the packet transmission, compared to the total number of attempts \( ^{3} \).

More in detail:

\[-\log_{10}(1-P(\text{failure sending} \mid \text{transmission } i \rightarrow k))\]

**Definition 5.3** An asymmetrical link can be defined as a direct link between two nodes i and j such as \( lqi_{ij} \neq lqi_{ji} \).

Fig. 2 shows a simple wireless network and associated graph model which contains some asymmetrical links.

It is worth noting that the probability graph models are largely used in the literature to study RBN \(^{4}\), such as the study conducted in [8].

### 5.1 WSN failure assumptions

From the studies presented in [9, 10, 11] as well as on the field experiments, it is possible to classify WSN failures as:

• Network failures (e.g. net partition or single node isolation from sink);
• Coverage failure;
• Measures synchronization failures \(^{5}\);

---

5
Figure 2: WSN schema and associated graph: colored discs are representative of single node’s broadcast range.

Figure 3: Packet loss patterns within the deployed network during a week in August’04 relative to the studies presented in [12]. Y-axis represents time divided into virtual packet slots (note: time increases downwards). A black line in the slot indicates that a packet expected to arrive in this time slot was missed, a white line means that a packet was successfully received.
Network failures encompass all those failures that involve more than one nodes, caused by connectivity degradation. This kind of failures depends on environmental factors like interferences and reflections, as well as on the same traffic present in the various sections of the network. In Fig. 3 the analysis of the data of the experiences presented in [12] within the activity of the famous project Great Duck Island is reported. In this diagram it is shown packet losses pattern. If all packet loss was distributed independently, the graph would contain a random placement of black and white bars appearing as a gray square. Visual inspection reveals patterns of loss: several black horizontal lines emerge, spanning almost all nodes, e.g. midday on August 6, 7, and 8. Looking at the packet loss on August 7, we note it is the only time in the sample window when motes 45 and 49 transmit packets successfully.

Figure 4: Example of bad deployed network that could easily exhibits node isolation or partitions behavior.

A net partitioned failure can be observed when a subset of the network remains isolated. The fault from which this kind of failure depends on, consists in the failure of all the nodes belonging to one of cut set of the network.

An example of a net partition failure is provided in Fig. 4. Consider as hypothesis the sink node with the id equal to 0, and indicated in green in Fig. 4. Because nodes 13 and 21 represent a cut set, if a failure of such nodes takes places in the network, this will generate a net partition failure: nodes 14,15,16,17,18,19 would remain isolated because a single nodes: the total effect on a node that tries to transmit is same of an absent channel (from the point of view of the sink), since the MAC layer protocol will always turn out in back-off because of the elevated traffic, thus inhibiting the transmissions.

A node or a set of nodes is isolated if and only if it is not able to reach the sink. Considered a line dividing in two parts the network, the cut set is the set of nodes touched by the line
path to the sink will not exist anymore. The network exhibit a coverage failure behavior when the number of nodes that compose a WSN is less than a prefixed threshold. Such threshold depends on the kind of information sent to the sink in order to satisfy the application requirements. An example can be the monitoring of an environment where the main task is to trigger fire alarms. The fire alarm is raised when a pre-established number of nodes, belonging to a same group, detect the fire (agreement between nodes): in case of the group contains an insufficient number of nodes which do not verify the condition of the fire detection alarm, the network fails.

The synchronization failure occurs when the network is not able to guarantee synchronization between the activities of the single nodes. These failures are critical for the event triggered WSNs, widely used for the dynamical structures monitoring. Such failures are often due to the bad quality of the network (e.g. heavy traffic or network badly proportioned) and/or to the anomalous behaviors of single nodes. A plenty of studies in literature proposed techniques for tolerating synchronization failures, such as the one presented in [13, 14, 15]. Fig. 5 shows an example in which synchronization failure leads to a network failure. The considered WSN is an example of dynamical monitoring structures: as it detects a solicitation of the structure, nodes are activated and a collection of measures from sensors are transmitted (e.g. acceleration measures), in order to calculate the F.F.T. If one transmission is delayed, the algorithm fails because the measures are temporarily skewed, and the global measure does not have sense.

![Figure 5: F.F.T. on a WSN for dynamical structures monitoring](image)

6 WSN failure model

To preserve the generality of the adopted approach, a UML schema of the failure model is outlined in Fig. 6. As it can be noticed from the figure, the model accounts for two main components:

1. "network dynamics" 10

2. "WSN Failure model".

The first provides all the information regarding:

- the network traffic profile;

---

9 Fast Fourier Transform
10 Evidenced in blue in Fig. 6.
- the routing tree;
- the power profile related to software and hardware components;

The Network dynamics components then supplies a set of input to the failure model. Stochastic Activity Networks (SAN) have been adopted as the formalism for the failure model, thanks to their flexibility and simplicity when used to model complex systems’ dependability [6].
6.1 **Failure model**

From the definition provided in sec. 5, it is immediate to consider a WSN failure model as composed model, those components are:

- n *Single node failure models*;\(^{11}\)
- one *Network level failure model*.

As illustrated in Fig. 6, the formal model is characterized by a set of parameters which are related to the network dynamics. The particular values assumed by the parameters drive the evolution of the failure model in accordance with actual environmental conditions. In Fig. 7 the interaction between real-word parameters and the failure model is outlined.

The overall failure model has been implemented as a set of interconnected *Stochastic Activity Networks* sub-models, which has then been resolved via simulation by means of the *Mobius* tool [16].

In Fig. 8 a network of fifteen nodes is depicted: the Mobius schema and the relative WSN model are shown: the advantage of the approach lies into the model generation process, which ease the management of the complexity. The correspondences between a particular

![Figure 8: WSN composed model](image)

WSN’s failure and the component/sub-model which models that failure are reported in the following:

- *net partitioned failures* are modeled by the *routing library* of the *network dynamics* component;
- *coverage failures* are considered by the *network state* sub-model (reported in blue in Fig. 8).

6.2 **Network state**

The main task of this sub-model is to manage the network state information and, as mentioned before, to model *coverage failures*.

The information handled by the sub-model is:

\(^{11}\)Described later in sec. 7.
1. number_of_running_motes;
2. number_of_failed_motes;
3. number_of_death_motes\textsuperscript{12};

Fig. 9 shows the corresponding SAN network of the \textit{network state} sub-model.

![Network State Sub-model](image)

**Figure 9: The sub-model ”network state”**

To model \textit{coverage failures}, a (\texttt{coverage\_granted}) variable has been used. Its value expresses the minimum level of coverage (percentage of running nodes) that can be tolerated according to the specific WSN requirements. A \textit{coverage failure} occurs when:

\[
\text{number_of_running_motes} \rightarrow \text{Mark}() < (\text{number_of_motes} \times \text{coverage_granted})
\]

where \texttt{number_of_motes} indicates the total number of nodes in the network. The \texttt{fail} and \texttt{die} actions are enabled by the \textit{input gates}, according to a predicate which specifies the presence of a \textit{token} in the places \texttt{someone\_fails} and \texttt{someone\_dying} \textsuperscript{13}. The release of the action makes the token flow from the place \texttt{number_of_running_motes} to one of the places \texttt{number_of_failed_motes} and \texttt{number_of_death_motes}. The same applies for the recovery actions (\texttt{recovering}, \texttt{revive}). They are enabled from the presence of a \textit{token} in:

1. \texttt{someone\_reborn};
2. \texttt{someone\_recovering};

that belong to the model’s ”interface”.

**Model interface**

The objective of the model interface is to allow the single node failure sub-models to communicate. For this reason, the interface of \textit{network state} is visible from all the sub-models of the net (Fig. 6).

In Fig. 10 the interface of the model is shown. The interface is implemented using the \textit{shared places}, according to the SAN net semantics [16]. The places representing ”input” parameters (reported in yellow) are shared between all the nodes of the net, allowing every sub-model \texttt{single\_node} (Fig. 8) to raise events when failures and recovery actions occur. More in detail the events provided by the interface are:

\textsuperscript{12}nodes whose batteries are exhausted
\textsuperscript{13}In the context of the present failure model, a node is said to be \textit{death} when it has exhausted its own batteries
Figure 10: interface of the sub-model "net_manager"

- **someone_fails**: event raised when a node fails;
- **someone_dying**: event raised when the battery of a node is exhausted;
- **someone_reborn**: event raised when the substitution of the batteries of a node is performed;
- **someone_recovering**: event raised when a node recovers from a transient failure.

The places represented in blue in the Fig. 10 represent instead the output parameters of the sub-model interface. In the example reported in Fig. 8 these parameters are not used. Fig. 11 represents an example where such output places are considered, and it is relative to a WSN composed of three clusters.

In this case, the parameters are used so that the health state of every cluster member can be discerned. The information produced from this sub-model and codified in the value of the token of the output places are:

Figure 11: WSN composed by three cluster
• \textit{number\_of\_runnings\_motes}: number of the still running nodes belonging to the sub-network;

• \textit{number\_of\_failed}: number of the failed nodes belonging to the sub-network;

• \textit{number\_of\_deaths\_motes}: number of the failed nodes due to energy exhaustion of the batteries and belonging to the sub-network;

• \textit{coverage\_failure}: the coverage of the area to monitor is under the minimum threshold.

A more compact representation for the net illustrated in Fig. 11 it is provided in Fig. 12 where the mechanism of model composition is used.

Figure 12: Clustered WSN represented by sub-model composition

7 Single node failure model

7.1 Failure modes assumption

The causes of failures affecting a single node of the network lie in the hardware and the software of the node. Such failures can be classified as follows:

• Hardware failures:
  
  – \textit{Battery failure}: failure due to batteries energy exhaustion;
  – \textit{Sensors failure}: Sensor board failure (less or no measures at all can be gathered);
  – \textit{Cpu failure}: CPU board failure (micro-controller and/or radio module);

• Software failures:
  
  – \textit{Operating system failures}
    
    * \textit{Receiving buffer allocation failure};
    * \textit{Timer fire event failure};
    * \textit{component initialization failure};
  – \textit{Software reset failure};
  – \textit{Routing failure};

The behavior due to node failures is derived from the experiments performed in [10]:

\pagebreak
• Stuck at zero [17] (lock-up or halting failure): The device is out-of-order and it’s not able to answer to environmental inputs and to produce any output;

• Frozen reading of a sensor (value failure) the device provides always the same output values;

• Null reading of a sensor (value failure) the device provides null output values;

• Out of scale reading of a sensor (value failure) the device provides no meaningful values;

• Instable behavior (erratic failure [18]) the device exhibits a behavior which deviates from the application specifications, e.g., it begins to send continuously the same packet until the radio channel results overloaded.

A node fails in a permanent manner when:

• All its sensors are out-of-order.

• Battery energy is exhausted;

• Permanent stuck at zero manifests;

A node fails in a transient manner when:

• transient sensor board failures occur;

• transient stuck at zero manifests;

• transient failure of neighbor belonging to the path to the sink occur.

As a general assumption, the failures relative to the cpu board and to the operating system will not be considered in the following. CPU failures will be ignored because of them negligible MTTF (mean time to failure) estimate, according to manufacturers data-sheets. Operating system failures will be not considered due to the lack of available information.

### 7.2 Single node SAN net

The SAN model of the failures of a single node is composed by the following sub-models (Fig. 13):

• sensor board failures (sensor_failure);

• battery failures(battery_failure);

• isolation failures (isolation_failure);

• network failures (net_node_failure);

• node state (node_state);

The last one is responsible to keep the information of the global node state. In addition, it represents an interface to the real world in that it allows to specify the actual values of the model’s parameters.

Coherently with the adopted modeling approach, following sections describes the sub-models separately. Then it will be depicted how the sub-models can be joined together to achieve the overall model of the node and of the whole network.
7.2.1 **Node.state**

*Node.state* model carries out the main activities:

1. "'start'" (boot) of the whole single node failure model;
2. collection of detailed information from every sub-model;

The aim of the first activity is to initialize the *struct* containing all the information coming from the real world. To this extent, the interface of the component *network dynamic* is used, and the value of probabilistic transitions are specified. Moreover, a unique identifier (ID) is generated for each node, thus enabling the whole model.

The aim of the second activity is concerned with the collection of detailed information on the state of the node, so as to allow the communication between the sub-models. The *node.state* model can be seen as an *event bus* that transmits events generated from a component to all the other components which manifested the interest in such information. The interest of a member in receiving an event, is manifested through its interface.

In Fig. 14 is illustrated the sub-model *node.state*. Its structure can be seen as:

- parts for the initial transient;
- parts for steady state functioning;

Tab. 1 reports all the fields of the record returned from invokation of the interface of *network dynamic* and contained in the place *node.state*. The action *booting* is timed: the timing distribution used is a Normal with average 0s and standard deviation 2s, so as avoid false synchronization between events relative to different nodes.

The place *running* represents an enabling gate for the functioning of the whole model: every event (e.g. battery exhaustion) which makes the token flow from *running* will cause the total stop of the model.

The place *running_msg* and in general all the places with the postfix _msg constitute an interface to the other failure sub-models. In steady state, the sub-network used is that shown in Fig. 15(a): in order to make the model more comprehensible, the picture has been colored with three colors evidencing the three macro-state of operation wherein a node can be. The places *failed, running*, and *death* represent "'stable'" states for the model of the node. It is possible to transit to them through the events risen by the input parameters of the model’s interface.
Figure 14: sub-model node_state
Figure 15: Net used at steady state in the sub-model node state (a), principal state of functioning of a node (b)

An other part of the model which needs more attention is the one of the branch going out from the place reset_after_battery_change. In such a part of the model we consider the case wherein an operator replaces the battery of a node, due to a precedent battery failure or due to normal maintenance. The action with probabilistic choice reset_sensor_board considers how many failures for the sensors board have been detected until this time. If at least a failure has manifested itself, a probability for the node to be not able to recover from failures happened before, after battery replacing, will exist. Moreover, if node failures due to a transient failure of the sensor board have been detected, a probability of permanent failure after battery replace will exist.

7.2.2 Model interface

As it can be evinced from Fig. 6, node state depends on all the other sub-models: the dependencies are explicited through its own interface (shown in Fig. 16) together with the places containing the data shared between the remaining sub-models. The input parameters of the interface (in yellow in Fig. 16) can be cataloged with respect to the sub-model they belong:

- **Sensor_board_failure:**
  - sensorBoard_notOperating;
  - transient;
Table 1: Fields of the record contained in the place node.state

<table>
<thead>
<tr>
<th>Record fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>tot_sent_pkt;</td>
</tr>
<tr>
<td>tot_fwd_pkt;</td>
</tr>
<tr>
<td>tot_bcast_pkt;</td>
</tr>
<tr>
<td>tot_myown_pkt;</td>
</tr>
<tr>
<td>tot_received_pkt;</td>
</tr>
<tr>
<td>sent_rate;</td>
</tr>
<tr>
<td>fwd_rate;</td>
</tr>
<tr>
<td>fwd_prob_rx;</td>
</tr>
<tr>
<td>fwd_prob_tx;</td>
</tr>
<tr>
<td>myown_send_prob;</td>
</tr>
<tr>
<td>bcast_send_prob;</td>
</tr>
<tr>
<td>receive_rate;</td>
</tr>
<tr>
<td>receive_prob;</td>
</tr>
<tr>
<td>tot_energy_for_sending;</td>
</tr>
<tr>
<td>tot_energy_for_receiving;</td>
</tr>
<tr>
<td>avg_cons_radio_per_sent_pkt;</td>
</tr>
<tr>
<td>avg_cons_per_rcv_pkt;</td>
</tr>
<tr>
<td>tot_energy_for_myown_sent_pkt;</td>
</tr>
<tr>
<td>avg_cpu_per_pkt;</td>
</tr>
<tr>
<td>cpu_cons_per_sec;</td>
</tr>
<tr>
<td>receive_error_rate;</td>
</tr>
<tr>
<td>sink_ID;</td>
</tr>
</tbody>
</table>

- permanent;
- stuck;
- sensor_failure_recovered;

- battery_failure:
  - battery_failed;
  - reset_after_battery_reset;

- isolation_failure:
  - isIsolated;
  - net_recovered;

The output parameters of the interface, shown in blue in Fig. 16, are used from node.state to rise events on the sub-model network.state before analyzed, according to what described to sec. 6.1.

The places Num_sensors and num_of_motes represent respectively the number of the sensors on the sensor board still operating and the total number of nodes the network is furnished.

7.3 Sensor board failure model

During the analysis performed in [12], for the Mica2 platform [19] some anomalous behaviors have been identified. Such anomalous behaviors present an high correlation with node failures. The attention has been focused on a particular sensor board for the Mica2 platform: the weather board produced by crossbow, due to the high success of this board on the WSN market, and to the plenty of work which has been published on the behavior
of this platform (Mica2 and weather board).

The sensors of such a board can jeopardize the correct operation of the node due to the failures they are subjected to. For a correct synthesis of the failure model it is therefore necessary to classify the failures of the sensors and their mutual correlations.

From the experiment presented in [19] the light sensor seems to be the most reliable one: this thanks to its simplicity (it is a simple photo resistance). Only two failure modes have been analyzed for this sensor: out of scale (e.g. 100% light over night) and null reading (e.g. 0 % light in the morning). However a failure of this component has a strong and critical impact on all the node: it has been quantified that 85% of its failures are accompanied by failures of the entire board (and therefore of the node, sec. 7), while the remaining 15% can cause failures to the humidity and temperature sensors as well.

For the humidity sensor, the anomalous behaviors recorded are mainly of two types: practically null and out of scale readings. Some peaks in the humidity reading (e.g. due to the condense) cause transitory increased drains of current from the batteries and therefore a strong drop of voltage on the supply stage. When this event is detected, a transient stuck at zero\textsuperscript{14} of the node is recorded. In 55% of the cases\textsuperscript{10} of null reading for the humidity sensor, the node fails as soon as two days. From a statistical fitting on the analyzed data, the temporal distribution of the failures is demonstrated to follow a Weibull distribution with shape parameter equals to 2. Moreover, when practically null humidity readings take place, simultaneous failures of the temperature sensor can be observed.

In the considered experiment that have manifested a null readings of humidity, only one node out of 43 affected by null reading\textsuperscript{15} has survived until the end of its mission due to the battery discharge.

The temperature sensor presents two main failure modes: practically null or frozen reading\textsuperscript{15}; however only the first one can also cause the failure of the humidity sensor.

In Fig. 17 is shown the cumulative distribution function (CDF) of node failure in presence of an anomalous reading from humidity sensor, compared with the whole population of the network. Summarizing what exposed so far, the failures contemplated in the context of this report are:

- light sensor failure

\textsuperscript{14}in reference to the taxonomy introduced to the Par. 7.

\textsuperscript{15}to a fixed value
Figure 17: Cumulative distribution of the probability of failure for a node

- out of scale and/or null reading failure;
  * failure correlated to humidity sensor;
  * failure correlated to temperature sensor;
- failure correlated to the failure of the sensor board;

- humidity sensor failure:
  - practically null reading failure:
    * failure correlated to temperature sensor;
    * failure as soon as two days of the whole board;
  - out of scale failure:

- temperature sensor failure:
  - frozen reading failure;
  - practically null reading;
    * failure correlated to humidity sensor failure (out of scale and/or practically null reading);

Each one of the failure cited can be either permanent or transient: in this last case, the actions of self-recovery are distributed as a Weibull distribution with parameters dependent on the nature of the failure.

### 7.4 Sensor failure model

As previously outlined in Fig. 15(b), a node can be in one of the three working state:

- **running**;
- **failed**;
- **death**.
These considerations are worth also for the present sub-model. The main objective of sensor failure is to model the events of failure and the recovery actions of sensors board. To this aim the model is composed by three main sections:

1. fault "extraction" section;
2. section responsible of generating failure events;
3. recovery section.

In Fig. 18 is introduced the SAN model of the sensor board. The hardware failures are modeled with respect to an exponential distribution [20]. The parameters, initialized to a value of default, are altered during the simulation, according to the failures that take place. Every time the action (Sensor.fail) fires, the choice of the sensor responsible for the fault is made, with respect to the statistics obtained by field data analysis. Moreover, after this event occurs, its probability of occurrence is set to zero so that it cannot take place anymore. This operation demands a re-normalization of remaining probabilities\(^{16}\) of the failures still considered for the working sensors. Dually, after an action of recovery, the probability of failure for the sensor (through the actions with the post-fix recover) is restored to a value different from zero, demanding a new action of re-normalization. A failed and then restored sensor will exhibit a bigger failure probability than before, so as to model the hardware aging effect.

In the schema of Fig. 18, the dependencies between the failures previously analyzed, are explicited. Every time a failure of one of the sensors of the board takes place, the number of working sensors is decremented: this is modeled through the mark of the place Num_sensors. When this mark becomes null due of repeated failures of the sensors, an event relative to a failure of the node it is raised making a token flow in the the place "Stuck", connected to the interface of the model node state (see the section 7.2.1). Moreover the place Stuck also models possible stuck at zero caused by a failure of the humidity sensor. In particular, the transition will_stuck is used: such a transition is timed and distributed as a Weibull\(^{17}\) with shape parameter \(\alpha = 2\)\(^{18}\), since a failure of the type practically null reading on this sensor can carry in 45% of the cases to a failure of the node within two days; in terms of parameters of the distribution, that is translated in:

\[
P(x < t^* + 172800s \mid x > t^*) = 0.55
\]

\[
P(x < t^* + 172800 \mid x > t^*) = \frac{P(t^* < x < t^* + 172800)}{P(x > t^*)}
\]

\[
\frac{F_x(t^* + 172800) + F_x(t^*)}{1 - F_x(t^*)} = \frac{e^{-\lambda(t^*)^2} - e^{-\lambda(t^*+172800)^2}}{e^{-\lambda(t^*)^2}}
\]

\[
1 - \frac{e^{-\lambda(t^*+172800)^2}}{e^{-\lambda(t^*)^2}} = 1 - e^{-\lambda \cdot 172800 \cdot (172800 + 2(t^*)^2)} = 0.55
\]

\(^{16}\) In the way that their sum is always equal to 1.
\(^{17}\) \(F(x) = 1 - e^{-\lambda x}\) is the temporal distribution of probability
\(^{18}\) The shape equal to 2 is translated in a failure rate \(h(t)\) of the distribution, that it increases linearly with the time
\[ \lambda = -\frac{\ln(0.45)}{172800^2 + 345600t^*} \]  

where \( t^* \) indicates the time in which the failure of the sensor has taken place.

The parameters of the other probabilistic transitions instead have been gained from several studies (in particular [12]). The assumption that has been made regarding the actions of recovery consider that they are related as a Weibull distribution with shape parameter equal to 2 and \( \lambda \) in such way that within 1 day the 90% of the transient failures are recovered. The interface of sensor failure is outlined in Fig. 19.

### 7.5 Battery failure model

The maximum temporal horizon of life for a node of a WSN is determined from the time-to-failure of its batteries.

The behavioral description of batteries is a complex problem because of the non-linearity between the voltage and remaining charge. This non-linearity is strongly emphasized especially when the discharge current is not constant (as it happens for a node of a WSN).

For an ideal generator of voltage, the voltage \( V(t) \) of the battery is constant on all the period of discharge and is equals to \( V_{oc} \) (open circuit voltage) until the energy of the batteries is exhausted; from this point on a discontinuity in the voltage to the terminals of the batteries is generated, making the voltage drop from \( V_{cutoff} \) to zero.

In some electronic devices, the batteries are not directly connected to the load: a so called DC-DC converted is usually used to stabilize the voltage to the clamps of the load, but they reduce the overall live of the device.

An other factor that generates non-linearity in the characteristic of discharge of the battery is the charge "recovery effect". This effect is closely dependent on the nature of the battery.

#### 7.5.1 Mathematical model

To the aim of the modeling, we will consider the stage of power supply as composed by the batteries and DC-DC converter, in a unique black-box so that the characteristic of voltage of the batteries can be approximated as constant in the interval \( (V_{oc}, V_{cutoff} - dc) \).

In this model the battery is considered as a "tank" of known capacity, from which it is possible to drain energy. The maximum capacity of the battery is assumed to be constant, without considering the effects capacity lessening due to the current request: this choice is reasonable due to the negligible current absorption for the nodes (about 5-15 mA).

The energy supplied by the batteries, represents a starting point in the model and it is calculated as follows:

\[ E_{batt} = V_{cc} \cdot 3600 \frac{s}{h} \cdot C_{effBatt} \]  

\(^{19}\)It is the voltage in which the battery does not succeed to distribute current on any load, included its inner resistance

\(^{20}\)They rise the \( V_{cutoff} \).
Figure 18: Mobius schema of the failure model of the sensor board
Figure 19: Sensor board failure model interface

where:

- \( V_{cc} \) = voltage;
- \( C_{\text{eff Batt}} \) = Total battery capacity really usable by a load and expressed in Ah\(^{21}\).

The remaining capacity of the batteries after a \( t^* \) seconds long operation, measured in Ah, is expressed by the following equation:

\[
U = U' - \int_{0}^{t_{0}+t^*} I(t)dt + R(i, t)
\]  
(3)

where:

- \( U' \) is the remaining capacity as result from the previous calculation step, and measured in Ah;
- \( I(t) \) is the instantaneous current drained from the batteries at time \( t \), measured in A;
- \( R(i, t) \) is the non linear function of the charge recovery effects of the batteries.

The equation above 3 can be also expressed in terms of the energy requested from the batteries, expressed in Joule, or in other words:

\[
U = U' - 3600 \frac{\sum}{h} \int_{0}^{t_{0}+t^*} \frac{E(t)}{V_{cc}} dt + R'(e, t)
\]  
(4)

where for the energy \( E \):

\[
E = E' - \int_{t_{0}}^{t_{0}+t^*} E(t)dt + R''(e, t)
\]  
(5)

7.5.2 Power consumption assumption

The batteries model can be easily achieved from the 3, assuming that:

- the integration interval is small enough to consider the energy consumption constant;

\(^{21}\)Experimentally it has been measured as the 80% of total capacity
• the voltage of the battery is constant, since we suppose that the load (the node) does not absorb excessive current to cause drop of voltage on DC-DC converter;

• The recovery effects described by $R'(e, t)$ can be thought rounded to:

$$\int_{t_0}^{t_0+t} \frac{Q(t) \cdot V_{cc}}{t_{idle}} dt \approx Q \cdot V_{cc}$$

where:

- $Q =$ quantity of charge expressed in Coulomb involved with the phenomena of recovery;
- $t_{idle} =$ idle time for the node

The approximation reported in 6 does not excessively compromise the validity of the model, making it usable only for some kinds of batteries $^{22}$.

Under these assumptions the 5 becomes:

$$E = E' - E \Delta t + \epsilon$$

From the previous assumptions it is correct to consider that the "status of charge" of the battery can be considerable as the remaining energy. That allows to model the above equation by means of the automata outlined in Fig. 20. In the proposed schema, every request for discharge of $i$ energy unit causes a transition in the state correspondent to the level of remaining energy. Dually the phenomena of recovery causes transitions in the opposite sense.

![Figure 20: Finite state automata relative to the battery model](image)

7.6 Battery failure

Fig. 21 shows the SAN net correspondent to the mathematical model characterized by the 7.

A node of a WSN alternates moments of activity with moments of wait in a low power state: this alternation is modeled through the mark in the place sleep. If the node is not sleeping ($\text{sleep} \rightarrow \text{Mark}=0$) then calculation of the consumption of the node has to be executed. The awakening of the node is modeled through a timed deterministic action ($\text{wake_up}$) (for

$^{22}$e.g. alkaline batteries or lithium. It does not constitute a problem for the considered scene since batteries AA used from the node satisfy this condition
networks with periodic characteristics). Once the node is awakened, the model calculates the consumption through the output gate "charge_drawn": every awakening, the model subtracts the energy requested by the node, from the remaining energy. The amount of drawn energy is known as power profile, thanks to the component network dynamic, indicated previously in the Fig. 6.

![SAN net of the model "battery failure"](image)

The remaining energy is modeled through the value of the mark of the place battery_charge_status: initially the mark is initialized to the value expressed through equation 2.

For applications with periodic characteristics, the consumption in the period of activity is known and it is expressed in millijoules-for-second. Consequently in correspondence of every awakening, amount of energy equals to that supplied from the profile and multiplied for the interval of activity of the cpu (also provided from the component network dynamic) is drained from battery_charge_status.

If the remaining charge is sufficient to satisfy the request, then the cpu can carry out its operations in order to return in the sleeping state. Otherwise the node fails: in this case the token is eliminated from the place running and a new one is inserted into the place battery_failed.

The mark contained in the places humidity_failed and light_failed respectively indicate that the failures of the humidity sensor and/or the light sensor have been taken place. That influences the consumption, increasing it of a percentage that depends on the detected failure.

The actions relative to the recovery effect is modeled through the timed action battery_recovery that it is activated in mutual exclusion with wake_up. An action of recovery of charge will cause an increase of the remaining charge equal to an amount dependent on the adopted battery technology 23.

The branch from the place death, is activated only when the charge of the batteries is exhausted. The timed action batt_replace models an action of maintenance executed by of an operator. This action consists in the substitution of the exhausted battery. The result of this operation is a restoration of the charge of the battery, and a forced reset of the

---

23It is interesting to notice that using batteries to lithium rather than others, this effect is eliminated, thanks to the characteristics of the employed cells
node in accord to sec. 7.2.1. The place *isIsolated* only contains a mark if failures of *isolation* (modeled in *isolation failure*) have taken place: in this evenience, the power consumption of the node will be diminished for the lack of all the forwarding activity of the packets coming from ”children” nodes. In *nodeState* is contained the value of the consumption share to subtract to the nominal power profile, in order to calculate the power consumption correctly.

Fig. 22 outlines the interface of the model.

![Interface of the model "battery failure"](image)

**Figure 22: Interface of the model "battery failure"**

### 7.7 communication failures model

The *communication failures* can be detected during the operations of sending and receiving a packet. In this context becomes important to divide the traffic seen from a node in three shares:

- outcoming traffic;
- incoming traffic;
- forwarding traffic;

The division of the traffic in these three shares permits to the model to be able to carry out a more detailed analysis on the behavior of a node that is about to send or to receive a packet: in this way we can differentiate the power consumption ([21, 22]) and the failures for each of these operations.

The local traffic of a node is relative to the produced packet on behalf of the application in execution. For nets with periodic characteristics, the forwarding rate for the the local packet is already knew at compilation time. For *event triggered* networks, this analysis is not useful: the production of packet depends on the solicitations supplied from the sensed environment.

The forwarding traffic is relative to the position assumed from a node in the routing tree [7]. Often in these networks particulars strategies are adopted by the MAC and the network layers, e.g. dynamically modifying the transmission range (to make always it be the smallest possible) to diminish the energy consumption. In this way many collision domains of small exvoltage are created, increasing the productivity of the network and at the same time uses the node in order to arrive to the sink
time, pulling down the consumptions (since the transmissive power is proportioned to the square of the energy demanded from the radio module). With these strategies is necessary to guarantee the connectivity between peripheral nodes and the sink adopting opportune routing algorithms. Since meticulously modeling of the activity of the radio module would require an excessive particularization of the model for a given couple of MAC and network layers, we choose to rely synthetic information. The failures contemplated for the transmission operations only regard the mechanisms to access the transmissive medium. The failures contemplated for the reception operations regard instead:

- receive buffer failure;
- failure relative to packet crc corruption;

7.8 Communication failure model

In Fig. 23 is shown the SAN model of communication failures. The model is triggered only when the node is working (running), connected to the net (connected) and the radio module is idle (radio_idle). The two action pkt_pend and pkt_to_send are enabled at the same time: the activation condition is eliminated when one of the two fires. For nets with the periodic character, these actions are timed and deterministic: the value of the rates that characterize them respectively is called pkt_pending_rate and own_packet_rate children_packet_rate . The previous values are taken from certain fields of the record contained in the place node_state, as outlined previously in Tab. 1.

When pkt_pend fires, the model determines if the received packet is destined to itself or to forward: the difference between the flow correspondent to the first or the second choice is in the mark of the place children_forward that equals 1 for packet forwarding. If there is the token in children_forward, this event also generates the activation of the branch responsible of forwarding: this is achieved by inserting a token in the place ready_to_send thanks to the output gate done3, that also decrements the remaining energy of the batteries (in battery_charge_status).

As for the actions relative to failures:

- buffer_fail implements the server of a M/D/1 queue with loss: the receiving buffer is modeled through the value of the mark in place pkt_pend that cannot contain more than a number buffer_size of packet. The server introduces a deterministic
time of service of value equal to \( t_{\text{getting packet}} \). The queue is memory less because the traffic is modeled as the sum of exponential distributions.

- **corruption**: instantaneous action with probabilistic choice based on the probability that a packet is corrupted.

The sending branch models the access to the transmissive channel, with possible waits due to collisions or to the temporal multiplication. The success on the transmission of a packet is modeled incrementing the mark in the place \( \text{sent packet} \) by using the output gate \( \text{done1} \), that also updates the remaining level of energy.

### 7.8.1 Model interface

![Diagram](image)

Figure 24: Interface of the model "communication failure"

Fig. 24 outlines the interface of the communication failure model evidencing that in the interface is composed only by input parameters; thus demonstrating that the failures contemplated in this model do not represent critical failures for the node\(^{25}\), but they are considered only because they could alter the power profile, making the node lifetime shorter on the long period.

*Connected* is an input parameter for the interface. Its goal is to propagate the isolation failures to the communication failure sub-model.

### 7.9 Isolation failure model

An isolation failure manifests when a node results isolated from the sink due to problems that does not depend from its status. In these problems, we can consider:

- failures of nodes that act as forwarders along the path to the sink;
- environmental interference;

This kind of failures is critically dependent on how many alternative paths exist between the nodes. In Fig. 25 an example is shown, in which the failure of a node generates a "domino" effect on all the nodes that depend on it.

Node 5 uses node 7 in order to reach the sink. Node 7 uses instead node 4, and both do not have alternative pathz due of their position. if node 4 fails (e.g. it has exhausted the batteries), a failure is propagated to node 7 and then to node 5. In this case nodes 7 and 5

\(^{25}\)There are no external event risen by this sub-models
are still working, but they are isolated. In this case, the behavior of the node depends on the routing adopted in the application. In the case of the common multihop routing, once that the node recognize that it does not have neighbors, it suspends its main activity (e.g. monitoring environment) and begins to execute the routine for the constitution of a new routing table, typically by sending hello messages on the network. If node 4 return working (e.g. substitution of the batteries), its presence is assessed from node 7 that inserts it as next hop in order to reach the sink. Consequently, node 5 notices that node 7 succeeds to reach the sink (the 7 answers to the hello packet sent) and resumes the suspended activity.
7.9.1 Isolation failure model

Fig. 25 outlines an example of isolation failure. In the case in which a failure of a node takes place in the network, it is probable that some other nodes remain isolated. Consequently, when an isolation failure of has been detected, it is possible to divide what happens in the following phases:

1. Failure of a network node;
2. Computation of the routing path from the node that used the failed node as forwarder to the sink;
3. If a node results isolated, its leaf nodes will again calculate the routing as point 2.

Instead, when the node recovers from the isolation failure, it happens:

1. the node previously failed recovers and constructs its routing table;
2. if it succeeds to reach the destination, it answers to hello packets of isolated nodes;
3. If an isolated node succeeds to reach the destination, it propagates this information in the routing tree, answering to hello packets of other isolated nodes.

The overall isolation failure model is outlined in Fig. 26. The most part of the analyzed routing algorithms, periodically update the routing table. In the present model, this operation is considered through the timed action routing_timer that verifies the presence of “neighbor” connected to the sink. This operation can be implemented through the methods of network dynamic whose invocation is shown in In Tab. 2.

The place isolated is filled up with a token only when the call to the booleana function data.amIConnected(myid->Mark()) returns a false value: in this case the mark is canceled from the place connected and is inserted in the place isolated_msg in order to propagate the information to the network_state model. Else if data.amIConnected(myid->Mark()) returns a true value, then the mark remains in the place connected. Finally if after an isolation data.amIConnected(myid->Mark()) returns a true value, a token is inserted in connected and in net_recovered_msg (the dual of isolated_msg), and the method data.propagateReconnect(myid=Mark()) is invoked.

Finally in Fig. 27 the interface of the model of the isolation failure is outlined.
8 Conclusion

This report described a failure model of Wireless Sensor Network (WSN), which is able to characterize the behavior of single node and of the whole network under failures. The failure model has been realized by relating single contributions of a node failure to the entire topology of the network, including the routing algorithm being used. The model took into account behavioral characteristics (e.g. software application installed on the nodes) and structural ones (e.g. topology) as well as hardware nature (e.g. hardware failure model, batteries nature) and the installed software. The model has been parametrized for the Mica2 platform and for the TinyOS.
References


