# RESILIENCY ASSESSMENT OF WIRELESS SENSOR NETWORKS: A HOLISTIC APPROACH

By
Catello Di Martino

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
"FEDERICO II" UNIVERSITY OF NAPLES
VIA CLAUDIO 21, 80125 – NAPOLI, ITALY
DECEMBER 2009

# Table of Contents

# List of Tables

# List of Figures

# Introduction

Wireless Sensor Networks (WSNs) are emerging as a promising technology to foster the design and the implementation of self-configuring, self-healing, and cost-effective monitoring infrastructures. In the last decade, they have been used in several research applications, such as detection of fires [1], object tracking [2, 3], security monitoring [4], supply chain monitoring [5] and stability monitoring of civil engineering structures, such as buildings [6], bridges [7], railroad tunnels [8], and dams [9, 10]. The commercial use of WSN is expected to grow dramatically in the next few years. However, industries in the field of wired sensing and monitoring infrastructures are still questioning the adoption of WSN in critical applications, despite attracted by their interesting features and by the possibility of reducing deployment and management costs of more than one order of magnitude [8]. This gap between research achievements and industrial development is mainly due to the little trust that companies repose in the resiliency of WSNs. Other than for toy applications, the design of WSNs is hardened by the lack of effective approaches able to provide a confident feedback on critical figures of merit such as WSN lifetime or data delivery efficiency, especially when aiming to design WSNs able to perform with a persistent level of dependability in spite of manifesting changes, i.e., able to perform with a given level of resiliency [11]. Hence, assessing the resiliency is a crucial task in designing dependable WSNs, since it could help to i) anticipate critical choices e.g., concerning node placement, running software, routing and MAC protocols, ii) mitigate risks, e.g., by forecasting the time when the WSN will not be able to perform with a suitable level of resiliency, and iii) prevent money loss, e.g., providing a criteria to plan and schedule maintenance actions effectively.

Resiliency assessment of WSNs plays a central role in raising the level of trust of WSNs for critical applications. WSNs are exposed to several faults due to wireless medium, limited energy budget they are equipped with, harsh environment [12], and cheap adopted hardware. Even if digital signals are less prone to electromagnetic interference, packets can be lost or delivered with errors, sensors may be frozen to wrong fixed values and nodes may periodically reset due to malfunctioning. In these systems, data sensed by WSN nodes has to be properly delivered to the sink node (i.e., the node responsible of data collection), in spite of ”changes” introduced during WSN operation (e.g., a node failure). As discussed in Chapter 2, past research efforts have been devoted to define the concept of connection (or network) resiliency for computer networks [13] and ad-hoc networks [14], i.e., the number of “changes”, in terms of node failures, that can be accommodated while preserving a specific degree of connectivity in the network. However, while these concepts still apply to WSNs, they are not enough to characterize the data-driven nature of WSNs. The service delivered by the WSN does not encompass only the connection, but also the computation, i.e., even when sensor nodes are potentially connected to the sink, data losses can still occur for instance due to packet loss, hampering the final computation. It is intuitive to figure out that, other than for the available definition of connection resiliency, the resiliency assessment of WSNs is dramatically complicated by the complexity of potential changes that may take place at runtime. The workload impacts on the number of packets sent on the network. The path followed by packets depends on the routing algorithm, on the topology, and on the wireless propagation profile (packets can be lost). The energy profile is affected by the workload, by the number of forwarded packets, and by the battery technology. All above factors impact on the failure behavior, e.g., a node can fail due to battery exhaustion. A node can also fail independently, due to faults in the sensing hardware. In turn, a failure of a node may induce a partition of the network into two or more subsets, involving a large set of nodes to be unavailable, i.e., isolated, since they are no more able to deliver data to the sink. Clearly, such high degree of inter-dependence complicates the assessment task by dramatically increasing the number of variables and dynamics to encompass. The situation is further exacerbated by the highly dynamic nature of WSN and by their proneness in manifesting transient failures [12], and self-reconfigurations. Finally, but not less important, different power consumptions and failure rates are indeed experienced when varying the underlying platforms, such as sensing hardware, radio chip and node operating system.

To overcome the mentioned issues, this thesis i) *proposes a novel and holistic approach for the resiliency assessment of WSNs*, and ii) *qualifies the concept of WSN resiliency as a non functional properties composed by both connection resiliency and data delivery resiliency*, defined in this thesis as a) *the number of changes in terms of node failure that a WSN can accommodate*, and b), relating to the time, *the greatest interval of time a WSN can survive*, both *while preserving a packet delivery efficiency greater than a threshold.*

While the concept of connection resiliency relates to the WSN topology, i.e., the degree of path redundancy in the network, the concept of data delivery resiliency is related to i) the computational load on nodes which may causes packet losses due to buffer overrun, ii) routing and MAC protocols impacting on the data delivery features and packet error rate, and iii) radio interferences and packet loss/corruption phenomenon on the propagation medium.

Key focus of the proposed approach is the *holistic assessment*, i.e., the comprehensive assessment performed by taking into account all subsystems and inter-related factors concurring to the behavior of the WSN. It is clear that a single node failure may impact on the behavior of the overall network in an unmanageable number of ways. Conversely, different user choices (e.g., the node workload and the routing algorithm) influence the nominal behavior( i.e., the expected behavior of running workload or routing protocols) as well as the failure behavior of every single node. Hence, an important step toward the resiliency assessment of a WSN is to evaluate: i) how the node workload, hardware platforms, topology and routing protocols impact on the failure proneness of nodes and of the network, and, vice-versa, ii) how node and network failures impacts on the nominal behavior of the WSN.

State-of-art techniques for the assessment of WSNs are mostly based on behavioral simulators and analytical models, as deeply discussed in Chapter 2.

WSN Behavioral simulators, such as ns-2 [15] or TOSSIM [16], are close to real WSNs. They typically belong to the the final user (e.g., the deployer) domain of knowledge and allow to reproduce the expected behavior of single WSN nodes on the basis of the real application planned to execute. However, they are not designed to express and to evaluate dependability attributes and hence the resiliency. Such an analysis requires to evaluate statistical estimators and hence it needs several simulations runs in order to achieve results with an acceptable confidence. This in turn increases the time needed for the simulation by order of magnitudes, given the low-level of detail of these approaches.

Analytical models, such as Petri nets and Markov chains, are the reference for resiliency assessment techniques. They have been successfully used for decades for the assessment of computer systems, including WSNs [17, 18]. However, the highly dynamic nature of WSNs requires the definition of detailed and complex models which are difficult to develop and hardly re-usable for different scenarios For instance, if a modeling team would invest for a fine grain model of a WSN, taking into account software, routing issues and hardware platforms, even a single change in the design parameters would probably require a modeling phase *ex-novo*, incurring in unaffordable design costs. As matter of fact, the assessment of WSN resiliency following a mere analytical approach requires strong simplifying assumptions that often lead to rather abstract results.

To master the complexity in the assessment process, and to overcome the limitation of the mentioned approaches, the proposed approach separates the assessment of the failure behavior from the evaluation of the nominal behavior by considering i) a set of parametric analytical failure models, and ii) a WSN behavioral model, respectively, interacting as detailed in the following.

Initially, the behavioral model is exploited to configure the WSNs in terms of hardware platform, topology, routing and MAC protocols, and to study the nominal behavior of the software, included the OS, and the power consumption of the nodes. Evaluations performed with the behavioral models are used to gather values for failure model parameters of the WSN under study, such as the packet forwarding rate of each node. Then the power of the analytical failure model is exploited to evaluate a set of metrics of interest such as the resiliency. However, it is not difficult to realize that some parameters are *dynamic* over time, i.e., their values need to be dynamically updated during the assessment, driven by the failure model. To exemplify consider a node that stops working, due to battery exhaustion. After this failure, the routing tree needs to be updated, and traffic patterns in the network change consequently. Different traffic patterns in turn cause a different nodes battery discharge rate which finally affects the lifetime of individual nodes, and likely, of the WSN. A possible solution would be to stop the failure model at each change event, and to step back to the behavioral simulation in order to re-compute network parameters coherently with new working condition, however, at the price of unaffordable simulation costs. For this reason, the proposed approach delegates the effort of computing the variation of dynamic parameters to an additional component, here referred as External Engine which orchestrates the evolution of the failure model. The External Engine can be regarded as a supervision entity encapsulating and managing aspects that are generally difficult to express at the level of abstraction of analytical models. Hence, the engine is essential to keep models simple, general and reusable.

The use of the External Engine decouples analytical models from “changes“ management issues, allowing to simplify the failure model which can adapt to each manifesting change transparently. Moreover, the assessment is more realistic since it encompasses all network/application related parameters which are likely to change during WSN lifetime.

The proposed approach is also conceived to reduce the modeling effort of final users by automating the creation of failure models, metrics to be estimated, and experiments to be performed by means of a *Failure Model Template Library.* Failure Model Templates are skeletons of parametric failure models, and are produced *una tantum* by a domain expert. Models of the *Model Template Library* are composed of i) a well defined interface, ii) a part depending on the specific WSN and on the node hardware/software , and iii) a fixed part. Well defined interface are used to compose complex models by joining different sub-models together. Template parts depending on the specific WSNs are the objective of the automated failure model generation since they need to be generated according to the considered WSN.

After the behavioral simulation, a model is generated by specializing for each sensor node a set of failure models from the Template Libraries with respect to the adopted radio chip, routing algorithm, sensor board and topology. For instance, given a specific topology provided by the user, a model is generated for each node with as many output links to other node failure models as its neighbors, in order to model actual communication links in the WSN. Generated models are populated with parameters which values reflect the WSN studied in the behavioral simulation, e.g., by specializing communication link model with evaluated link packet loss rate. The generation phase ends by producing a XML description for each generated models, which are completed with a XML description of metrics and experiments of interests to perform (e.g., selected by the user, consistently with his/her interests). Finally, a parser translates the XML descriptors in a format compliant to the selected analytical model formalism, i.e., specializing the produced XML for the modeling framework chosen for performing experiments of interest.

Relying on an automated modeling phase, the proposed approach allows final users (i.e., WSN developers) to work within their knowledge domain, without requiring specific modeling and/or programming skills. In other terms, developers interact with artifacts that are related to their domain, such as behavioral simulators. Finally, interested industries may release failure model libraries upon the release of WSN hardware, following the same approach as for HDL libraries.

In the context of this thesis, Stochastic Activity Networks (SAN) formalism [19] and the Mobius [20] framework are adopted to develop and simulate WSNs failure models, due to their flexibility and extensibility features.

The effectiveness of the approach is shown by means of a resiliency assessment campaign based on a set of realistic WSNs. As it will detailed later, the approach allows to anticipate design choices by evaluating the resiliency under different failure conditions and scenarios, workload behavior, and adopted routing algorithms. The approach can be adopted by a hypothetic user, who can exploit simulation results to fine-tune his applications, for instance, selecting an appropriate routing algorithm and/or application workload which make the WSN able to fulfill given requirements, e.g., in terms of resiliency. The proposed approach may help also in the case of already deployed WSN, for instance, by forecasting the time when the WSN will exhibit a degraded behavior by deviating from its specifications helping to schedule maintenance actions in advance.

This thesis is organized in 7 chapters as it follows. Chapter 1 provides a brief overview of WSNs and their applications, stressing their requirements and the importance of resiliency in the considered scenarios. Chapter 2 analyzes the state of the art in the field of WSN simulation, and in the field of dependability assessment. Chapter 3 provides the definition of both connection and data resiliency. Chapter 4 is focused on the holistic approach, objective of this thesis, and it presents challenges and solutions for the orchestration of the behavioral and analytical simulation and for the automated failure model generation. Chapter 5 presents the behavioral models and the parameters needed to generate the failure model. Chapter 6 presents the failure models and the followed modeling approach. Chapter 7 finally provides a set of case studies with the objective showing the need of the proposed definition of WSN resiliency, and the effectiveness of the approach concerning different WSN deployments.

# Chapter 1

# Dependability of Wireless Sensor Networks

*Wireless Sensor Networks (WSNs) are emerging as one of the most compelling research areas, with profound impact on technological development. WSNs have been used with success into more and more critical application scenarios, such as structural monitoring of civil engineering structures, where the dependability of WSNs becomes an important factor, discriminating the success of large-scale industrial applications. However, unreliable hardware, installed software, energy consumption and topology are the major constraints affecting Wireless Sensor Networks (WSNs) resiliency. For this reason, dependability evaluation of WSNs is gaining popularity since it could help to reduce risks and money losses by forecasting the resiliency of a WSN before the deployment. This Chapter briefly introduces this issue, WSNs applications and requirements. Finally, it introduces the dependability requirements of WSN, motivating the approach presented in this thesis.*

## 1.1 Wireless Sensor Networks

Recent advances in wireless communications and electronics have enabled the development of low-cost, low-power, multi-functional sensors, capable of local computation and equipped with short range radio transmitting devices[21]. The main purpose of a Wireless Sensor Network as a whole is to serve as an interface to the real world, providing physical information such as temperature, light, radiation, and others, to a computer system.

WSN are expected to be a breakthrough in the way natural phenomenon are observed: the accuracy of observations will be considerably improved, leading to a better understanding of the monitored environment.

These networks have a simple structure: there are dozens up to 100s of elements, called "sensor nodes" able to sense physical features of their surroundings or to monitoring a set of items. WSN nodes exchange information on environment in order to build a global view of the monitored items/regions which is made accessible to the external user through one or more gateway node(s), named base station or sink node(s) [22]. Sensor nodes are often referred as smart sensors or smart dust because of their processing, power, and memory capabilities [23, 24, 25, 26]. The small size of sensors (about the size of a coin) allows them to be easily embedded into materials [25] or deployed in a mobile scenarios such as remote health care, cars, or floats over water [27].

A WSN typically operates by stepping through the following phases: i) sensor nodes acquire sensed data, ii) data is locally processed, iii) data is routed in a multi-hop fashion, iv) data is delivered to the sink node, and v) data is forwarded by the sink node to a conventional network, e.g. Internet [26].

Sensor networks may be organized in two basic architectures, hierarchical and flat. In flat configurations, all the nodes participate in both the decision-making processes and the internal protocols, like routing. On the other hand, in hierarchical configurations the network can be divided into clusters, or group of nodes, where all the organizational decisions, like data aggregation, are made by a single entity called cluster head.

The main components of a sensor node are its microprocessor, its communication chip, its integrated sensors, and limited mass storage. It is also possible to have support for external components, such as GPS chips or external flash cards, or a better security support, like radio chips with hardware implementations of cryptography mechanisms such as AES. The main drawback of WSN sensor nodes is the restricted resource of energy leading to limited lifetimes. This fact motivates attention and effort the research community has devoted to the development of low power consumption techniques, not only at MAC layer, but also at network and application layers.

## 1.2 WSN Requirements

### Lifetime

In most application scenarios, a majority, if not the totality of the nodes are self-powered, and hence, in the best, they are able to survive for a limited time. The most common adopted lifetime metric is related to the time till a certain percentage of surviving nodes in the network falls below a given threshold [28]. When a WSN should be considered non-functional, however, is application specific.

### Area Coverage

Area coverage is defined as the ratio between the number of up, running, and connected nodes at a given instant of time, over the number of initially deployed sensors. Due to the aging and wear out process of nodes, the area coverage is a decreasing function of the time. WSN applications define the minimal level of area coverage to assure so that the observed phenomenon can be monitored with acceptable confidence.

**Timeliness**

In environment monitoring applications is often required to correlate samples coming across different nodes in order to gather combined measurements. in such WSNs It is a common practice to compute Fast Fourier Transforms across different nodes, for instance upon a vibration event of the monitored structure[7] . In this case, nodes must be synchronized in order to take part to the distributed computation correctly, i.e. by providing samples acquired within a bounded interval of time. In WSNs, nodes clock drifts apart over time due to inaccuracies in oscillators. High-precision synchronization mechanisms must be provided to continually compensate for these inaccuracies, e.g., by means of synchronization protocols, but this is in contrast with lifetime, since such protocols often require periodic exchange of extra radio packets.

**Data Delivery**

The main mission of a typical WSN is to collect environmental data and to send measurements to the sink node. Depending on the specific application, it may be fundamental to collect at least a specific amount of data in order to fulfill application requirements. However, despite at the deployment, the WSN is able to deliver a sufficient amount of data to the sink node, interferences and failures may force topology reconfiguration that may impact on the delivery efficiency. Consequently, it is important to evaluate the efficiency of WSNs in delivering data to the sink in order to detect whether a WSN is still able to behave as expected. Moreover, the evaluation of data delivery features may be used for topology control purposes, for instance, enabling spare nodes or switching to a routing policy able to deal with new working condition.

## 1.3 Critical Applications

The field of WSNs offers a rich, multi-disciplinary area of research, in which a variety of tools and concepts can be employed to address a diverse set of applications. As such, many potentials of this field have been under study both in academia and in the industry. Only recently they have become a technology which is more and more envisioned in real applications, included industrial systems or critical scenarios as a good opportunity to drastically reduce installation, management, and maintenance costs and related times.

According to the European Commission, Critical scenarios consist of "*[...] those physical and information technology facilities, networks, services and assets which, if disrupted or destroyed, would have a serious impact on the health, safety, security or economic well-being of citizens. [...]*" [29].

The number of critical scenarios where sensor networks can be used is incredibly broad. It is not the primary aim of this thesis to overview and detail the large spectrum of existing critical applications in the field; interested readers could refer to good surveys in the literature [21, 30].

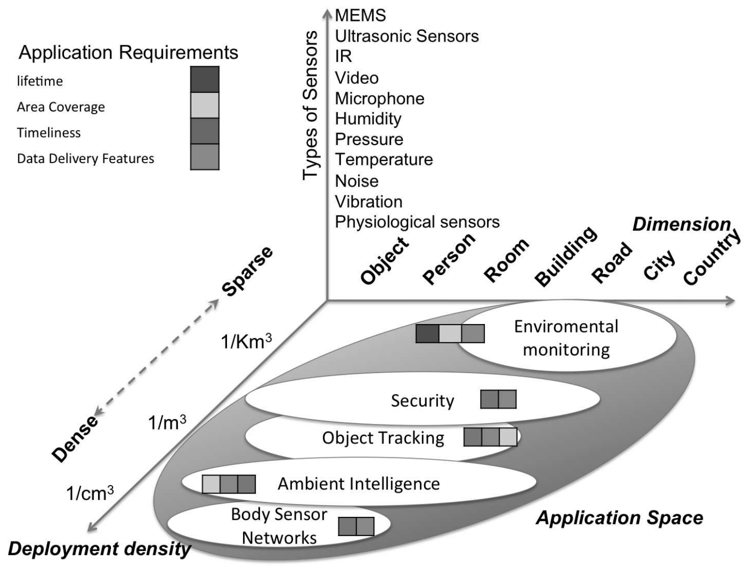


Figure 1.1: WSN applications domains and basic applicative requirements.

Rather, in this section we review five main WSN application classes: environment monitoring, security monitoring, object tracking, ambient intelligence and body network applications. The majority of critical WSN applications will fall into one of these class templates. Figure 1.1 provides a view of the considered WSN application domains which can be classified according to three main parameters: density of the deployment, scale of the deployment, and sensing capabilities of the network. Depending on the objective of the deployment, WSNs may be equipped of a number of nodes spacing from few dozen (e.g. Body Sensor Networks) up to thousands (e.g. Environment monitoring). Nodes may be deployed in a small area with high density or be scattered on a large region. Finally, for each application scenario, different requirements must be met, as reported in Figure 1.1 and as further discussed in the following sections.

### 1.3.1 Environment Monitoring

Environment monitoring applications generally consist of deploying a number of sensors in the field to periodically measure meteorological and hydrological parameters, such as wind speed and direction. Most of them change relatively slowly in time, which allows for sparse sampling (one sample every two to five minutes is most often sufficient). However, as interesting phenomena, such as rock slides or avalanches, occur seldom and are difficult to predict, deployments must last long enough to capture them, and must assure a reliable delivery of gathered data to the sink node. Thus, requirements of an environmental monitoring system are lifetime and data delivery resiliency. Achieving resiliency is difficult because packet losses are more likely to happen during harsh weather conditions (e.g., heavy rain, intense cold) which are at the same time the most interesting episodes for data analysis. Moreover, due to the multi-hop organizations of WSNs, as the network ages, it is very likely to observe node failures that may cause disconnections of nodes to the sink and network partitions.

All these requirements are especially important when deploying a network in remote and difficult-to-access places. For instance, one of the SensorScope deployments [31] occurred in high mountain, in collaboration with authorities. A helicopter was required for carrying hardware and people. Going back to the site a few days later because a battery is depleted or because a station needs to be manually rebooted is obviously inconceivable.

### 1.3.2 Security Monitoring

Differently from environment monitoring, WSN Security monitoring applications do not collect any data, and are classified as "report by exception" networks. The common task of each node is to frequently check the status of its sensors, and to transmit a data report strictly only when an exception is detected, such as security breaches or unauthorized access to an environment. Nodes are typically equipped with both permanent and backup power sources. This has a significant impact on the optimal network architecture and on nodes lifetime that is not as critical as for environment monitoring. The timeliness and data delivery resiliency are the primary system requirements.

### 1.3.3 Object Tracking

The purpose of this class of WSN applications is to provide an effective solutions for the tracking of mobile objects [3] by using a combination of WSNs and radio frequency identification/positioning technologies. With WSNs, objects can be tracked by simply tagging them with a small sensor node. The sensor node will be tracked as it moves through a field of sensor nodes that are deployed in the environment at known locations. Instead of sensing environmental data, these nodes will be deployed to sense the RF messages of the nodes attached to various objects.

Unlike sensing or security WSN, node tracking applications will continually have topology changes as nodes move through the network. While the connectivity among nodes at fixed locations will remain relatively stable, the connectivity to mobile nodes will be continually changing. Additionally, the set of nodes being tracked will continually change as objects enter and leave the system. Timeliness in detecting new moving objects and in managing hand-offs as the tracked objects move is the first requirement of such WSN applications. Moreover, a full coverage of the monitored area and reliable delivery of data related to tracked object is essential.

### 1.3.4 Ambient Intelligence

Ambient Intelligence (AmI) is a paradigm that applies to a number of vertical domains. AmI is a term coined by Philips management to conjure up a vision of an imminent future in which persons are surrounded by a multitude of fine grained distributed networks comprising sensors, and computational devices that are unobtrusively embedded in everyday objects such as furniture, clothes, and vehicles, and that together create electronic habitats that are sensitive, adaptive and responsive to the presence of people [32, 33]. Examples of Ami applications are smart offices and buildings [34]. In smart offices, it is possible to record the movement and meeting patterns of employees, and also answer queries related to the employees (such as their location) and related to the rooms (such as their temperature).

AmI applications are affected by several new treats due to mobility of resources [35, 36] and to the peculiarities of domains [37]. AmI requirements typically encompass data delivery features, timeliness and lifetime.

### 1.3.5 Body Sensor Networks

The term BSN is first coined in [38] in order to bring together scientists from different disciplines such as computing, electronics, bioengineering and medicine for addressing the general issues related to using wearable/wireless and implantable sensors on the human body. The basic structure of BSN consists on a set of wireless physiological sensors, such as body temperature, blood pressure and oxygen saturation, cardiac activity (ECG), and encephalic activity (EEG). Sensors are used jointly to measure and monitor remotely the status of a patient. It is also possible to use BSN in helping assisted-living and independent-living residents by continuously and unobtrusively monitoring health-related factors such as their heart-rate, heart-rhythm, and temperature.

Requirements of BSN consists of limited and predictable time latency and reliable transmission of sensed data and alarms. Moreover, area coverage is of paramount importance, since BSN are constituted by a very limited number of sensor nodes, without any overlap between sensed area or sensed parameters.

## 1.4 Challenges

Although the technology for WSNs is relatively mature, and WSN have been employed in several pilot research applications, real large scale applications are completely lacking. This is in part due to a number of still unsolved problems afflicting WSNs.

A number of smart sensor prototypes have been designed and implemented by the academic research community. The most famous of such prototypes are probably the Berkley Motes [39] and Smart Dust [25]. Later on, many academic interdisciplinary projects have been funded (and are currently being funded) to actually deploy and utilize sensor networks. One such example is the Great Duck Island project, in which a WSN has been deployed to monitor the habitat of the nesting petrels without any human interference with animals [12].

Smart sensor nodes are also being produced and commercialized by some electronic manufacturer, such as Crossbow, Philips, Siemens, STMicroelectronic.

There is also a considerable standardization activity in the field if WSNs. The most notable effort in this direction is the IEEE 802.15.4 standard which defines the physical and MAC layer protocols for remote monitoring and control, as well as sensor network applications. ZigBee Alliance is an industry consortium with the goal of promoting the IEEE 802.15.4 standard.

Other than standardization, main challenges related to WSN implementation are reported in the following.

**Energy Conservation**

Because of the reduced size of the sensor nodes, the battery has low capacity and the available energy is very limited. Despite the scarcity of energy, the network is expected to operate for a relatively long time. Given that replacing/refilling batteries is usually impossible or very expensive, one of the primary challenges is to maximize the WSN lifetime while preserving acceptable performances.

**Low-quality communication**

WSN are often deployed in harsh environments, and sometimes they operate under extreme weather conditions. In these situations, the quality of the radio communication might be extremely poor and performing the requested collective sensing task might become very difficult.

**Operation in hostile environments**

In many scenarios, WSN are expected to operate under critical environmental conditions, which translates in an accelerated failure rate of sensor nodes. Thus, it is essential that sensor nodes are carefully designed, and the WSN assessed under real failure assumptions. Furthermore, the protocols for network operation should be resilient to sensor fault, which must be considered in these scenarios a norm rather than an exception.

**Security Attacks**

As networks grow, the vulnerability of network nodes to physical and software attack increases. Attackers can also obtain their own commodity sensor nodes and induce the network to accept them as legitimate nodes, or they can claim multiple identities for an altered node. Once in control of a few nodes inside the network, the adversary can then mount a variety of attacks, for instance, falsification of sensor data, extraction of private sensed information from sensor network readings, and denial of service attacks. Therefore, routing protocols must be resilient against compromised nodes that behave maliciously. Ensuring that sensed information stays within the sensor network and is accessible only to trusted parties is an essential step toward achieving security. Data encryption and access control is one approach. Another is to restrict the network is ability to gather data at a detail level that could compromise privacy. For example, in a healthcare environment, storing data in an anonymous format and removing any personal referencing information. Another approach is to process queries in the sensor network in a distributed manner so that no single node can observe the query results in their entirety. In this case security comes at the price of a reduced lifetime due to the extra overhead induced in the network.

**Maintenance Cost**

The initial deployment and configuration is only the first step in the WSN lifecycle. In WSN where deployment is expected to surpass the lifetime of batteries, the total cost of management for a system may have more to do with the maintenance cost than the initial deployment cost. Throughout the lifetime of a deployment, nodes may be relocated or replaced due to outages, and discharged batteries. In addition, reintegrating the failed nodes adds further labor expenses. An approach to limit interventions would be to increase the lifetime by adopting a trigger-based sampling strategy: sensors start to acquire data only when given conditions are met. However, this approach introduces a further coordination problem among sensors, e.g. ,nodes monitoring the same area must agree on the triggered event, synchronize their clock, and start to sample data coordinately. Since access costs are dominant over in-situ costs, it is important, therefore i) to identify sources of maintenance related costs and to reduce them, and ii) to schedule maintenance so that once the network is accessed, a convenient number of nodes are maintained and hence, the overall maintenance cost optimized.

**Lack of easy to commercialize applications**

Nowadays, several chip makers and electronic companies started the production of sensor nodes. However, it is much more difficult for these companies to commercialize applications based on WSN. Selling applications, instead of relatively cheap sensors, would be much more profitable for industry. Unfortunately, most sensor network application scenarios are very specific, and companies would have little or no profit in developing very specific applications, since the potential buyers would be very few.

## 1.5 Resiliency in WSN Critical Applications

The commercial use of WSN is expected to grow dramatically in the next few years, however, industries in the field of wired sensing and monitoring infrastructures are still questioning the adoption of WSN in critical applications, despite attracted by their interesting features and by the possibility of reducing deployment and management costs of more than one order of magnitude [8].

This gap between research achievements and industrial development is mainly due to the little trust that companies repose in the resiliency of WSNs. Resiliency as been recently defined as [11]: the persistence of dependability when facing ”changes”. This change of perspective leads to new requirements of modern fault tolerant systems, such as the ability of accommodating unforeseen environmental perturbations or disturbances.

WSNs are exposed to several faults due to both wireless medium characteristic, the battery exhaustion of a sensor, harsh environment [12] and cheap hardware. Even if digital signals are less prone to electromagnetic interference, packets might be lost or delivered with errors, sensors may be frozen to wrong fixed values and nodes may periodically reset due to malfunctioning. In these systems, data sensed by WSN nodes has to be properly delivered to the sink node, in spite of ”changes” introduced during WSN operation (e.g., a node failure, or a route update). This situation is further exacerbated considering the the fail silent behavior of a sensor, that makes difficult the detection of malfunctioning or failed nodes. As a result, the utilization of a WSN for critical applications introduces specific resiliency requirements, which relate to those reported in Figure 1.1.

**Connection Resiliency**

Path redundancy and self-organization are cost-free features provided by WSNs. Sensor nodes are arranged in an ad hoc manner providing a number of potential redundant paths toward the sink, depending on the number of their neighbors. However, real installations are often subjected to specific deployment constraints such as number and density of sensor nodes. For instance, WSNs designed for bridges, towers and buildings monitoring are typically organized following an in-line, grid or two chain topology due to site constraints, limiting the number of redundant paths in the network ( a node may reach only a very low number of neighbors) [17] . Consequently, a failure of an inner node in the topology is likely to cause the isolation of a set of nodes. Hence, judicious selection of node position and density is crucial to optimize the path redundancy and to enforce the connectivity resiliency of the WSNs. Referring to the requirements showed in Figure 1.1, connection resiliency relates to lifetime and area coverage.

### Data Delivery Resiliency

In order for fulfill application requirements, such that for structural monitoring and analysis or security monitoring, at least a minimum amount of measurements have to be gathered during each measurement step, despite communication and/or sensor faults. For example, if n sensors, forming a cluster, are used to cover a pillar of a bridge, at least k-out-of-n sensor readings have to be delivered to the sink node in time, at each measurement step. The value of k have to be selected with respect to the physical characteristics of the structure's section. Moreover, the k correct sensors cannot be arbitrary chosen among the n available sensors. In other terms, other than providing path redundancy to the sink, data delivery reliability is essential if not necessary in most of WSN applications. Indeed, it may be possible that, despite existing paths reaching the sink, a node is not able to deliver its measurements due to buffer overflows, packet corruptions and application errors. Typically, to face this issue, acknowledgement based routing protocol or epidemic routing protocols such as flooding are used to decrease the probability of packet loss throughout the network. However, such approaches strongly influence the WSN lifetime and time latency in data delivery due to the overhead they cause in the forwarding nodes. Referring to the requirements showed in Figure 1.1, data delivery resiliency relates to timeliness and data delivery features.

### 1.5.1 The Need of a Holistic Approach

Resiliency assessment of WSNs by means of fault forecasting represents an intriguing research issue indirectly related to the defined requirements. Its proper application could help to i) anticipate critical choices e.g., concerning node placement, running software, routing and MAC protocols, ii) mitigate risks, e.g., by forecasting the time when the WSN will not be able to perform with a suitable level of resiliency, and iii) prevent money loss, e.g., providing a criteria to plan and schedule maintenance actions effectively. However, it is easy to figure out that resiliency assessment of WSNs is dramatically exacerbated by the complexity of potential changes that may take place at runtime. The workload, included the use of aggregation/fusion algorithms, impacts on the number of packets sent on the network. The path followed by packets depends on the routing algorithm, on the topology, and on the wireless medium (packets can be lost). The energy profile is affected by the workload, by the number of forwarded packets, and by the battery technology. All above factors impact on the failure behavior, e.g., a node can fail due to battery exhaustion. A node can also fail independently, due to faults in the sensing hardware. In turn, a failure of a node may induce a partition of the network into two or more subsets, involving a large set of nodes to be unavailable, i.e., isolated, hence, unable to send acquired data to the sink. Clearly, such high degree of inter-dependence complicates the assessment task, by dramatically increasing the number of variables and dynamics to encompass. Finally, but not less important, resiliency assessment must also take into account actual hardware/software platforms features and the sensing hardware being used:

different power consumptions and failure rates are indeed experienced when varying the underlying platforms, such as sensing hardware, radio chip and node operating system.

Resiliency assessment of WSN cannot deviate from the use of models. Although various interesting dependability evaluation techniques and tools have been developed in the last decades, still a little attention is devoted to define approaches able to master the intrinsic complexity of WSN resiliency assessment. State-of-art techniques for the assessment of non-functional properties, such as power consumption or dependability attributes are mostly based on behavioral simulators and analytical models, as deeply discussed in Chapter 2.

WSN Behavioral simulators, such as ns-2 [15] or TOSSIM [16], are closer to real WSNs. They allow to reproduce the expected behavior of every single WSN node on the basis of the real application planned to execute. However, they are not designed to express and to evaluate non-functional properties. Such analysis would require to evaluate statistical estimators and hence several simulation runs in order to achieve results with an acceptable confidence. This in turn would increase the time needed for the simulation by order of magnitudes, given the low-level of detail of these approaches.

Indeed, analytical models, such as Petri nets and Markov chains, are the reference for resiliency assessment techniques. They have been successfully used for decades for the assessment of computer systems, including WSNs [17]. However, the highly dynamic nature of WSNs requires the definition of detailed and complex models which are difficult to develop and hardly re-usable for different scenarios For instance, if a modeling team would invest for a fine grain model of a WSN to design, taking into account software, routing issues and hardware platforms, even a tiny change in the design parameters of the considered WSNs, such as the software or the topology, would probably require a modeling phase *ex-novo*, incurring in unaffordable design costs. As matter of fact, the assessment of WSN resiliency following a mere analytical approach requires strong simplifying assumptions that often lead to rather abstract results.

# Chapter 2

# WSN Assessment: Models, Tools and Related Work

*While measurement is a valuable option for assessing an existing system or a prototype, it is not a feasible option during the system design and implementation phases. Model-based evaluation has proven to be an appealing alternative. Several modeling paradigms, various techniques and tools for model evaluation are currently used in the field of WSNs. This Chapter revises modeling approaches and tools currently used in the field of dependability modeling, network simulation and WSN assessment, including international project and related studies.*

## 2.1 Dependability Modeling and Modeling Formalisms

Research in dependability analysis has led to a variety of models, each focusing on particular levels of abstraction and/or system characteristics.

Dependability modeling and analysis assumes that the dynamics of the system can be described by temporal random variables. Through dependability modeling it possible to perform an evaluation of the system behavior with respect to fault occurrence or activation (fault-forecasting) which is of paramount importance to evaluate a set of metrics of interests such dependability attributes (See Appendix A). Hence, models can give immediate feedback to the designers who can timely improve the design. Models parameters are however based on past experiences on same systems, and these parameters can be often invalidated as the modeled system ages during the simulation.

Traditional model formalisms used in the dependability analysis of stochastic systems can be divided in two broad categories [40]: 1) combinatorial models (e.g. Reliability Block Diagrams[41], Fault-tree [42, 43]), 2) state-space based models (Markov chains [44], Petri nets [45], Stochastic Activity Networks [46]).

## 2.2 Combinatorial Modeling Formalisms

Combinatorial models assume the parts of the system to be statistically independent and achieve high analytical tractability combined with a low modeling power. Combinatorial methods are quite limited in the stochastic behavior that they can express. Despite several extensions that have been made to combinatorial models, they do not easily capture certain features, such as stochastic dependence and imperfect fault coverage.

### 2.2.1 Reliability Block Diagrams

Reliability Block Diagrams (RBD) consists of a graphical structure with two types of nodes: blocks representing system components and dummy nodes for connections between the components. Edges and dummy nodes model the operational dependency of a system on its components. At any instant of time, if there exists a path in the system from the start dummy node to the end dummy node, then the system is considered operational; otherwise, the system is considered failed. A failed component blocks all the paths on which it appears. RBDs thus map the operational dependency of a system on its components and not the actual physical structure of the system. Series-Parallel RBDs are useful not only because they are very intuitive, but also because they can be solved in linear time [47]. Such RBDs are quite frequently used in reliability and availability modeling [48, 49] and many software packages exist that support construction and solution of RBD models [47, 50].

### 2.2.2 Fault Trees

Fault Trees are acyclic graphs with internal nodes that are logic gates (e.g., AND, OR, k-of-n) and external nodes (leaves or basic events) that represent system components. The edges represent the flow of failure information in terms of Boolean entities (TRUE and FALSE or 0s and 1s). Typically, if a component has failed, a TRUE is transmitted; otherwise, a FALSE is transmitted. The edge connections determine the operational dependency of the system on the components. At any instant of time, the logic value at the root node determines whether or not the system is operational. If shared (repeated) nodes (nodes that share a common input) are not allowed, then the acyclic structure is a rooted tree. Fault trees without shared nodes are equivalent to series-parallel RBDs [49], but when shared nodes (or repeated events) are allowed, fault trees are more powerful [51].Fault trees have been extensively used in reliability and availability modeling [52, 53, 54, 55], safety modeling [56], and modeling of software fault tolerance[57].

Evolution of FTs are Dynamic Fault Trees [58] which are based on the definition of new gates (Priority-AND, Functional Dependency and Warm Spare) that induce temporal as well as statistical dependencies. The quantitative analysis of the DFT consists in exploding minimal modules of dynamic gates into their state-space representation and computing the related occurrence probability by means of a CTMC (Continuous Time Markov Chains) [44].

## 2.3 State Space Based Modeling Formalisms

State-space methods are much more comprehensive than combinatorial methods and rely on the enumeration of the whole set of possible states of the system and on the specification of the possible transitions among them. The main drawback of these techniques is the well known state explosion problem, due to the circumstance that the dimension of the state space grows exponentially with the number of components.

### 2.3.1 Markov models

Markov models consider the system behavior being modeled as a Markov process, on a discrete state space. Markov Chains, Discrete Time Markov Chains are example of formalisms bases on Markov models.

Markov models have been extensively used for dependability analysis of hardware systems [48, 49, 47] real-time system performance in the presence of failures [59, 60], combined analysis of hardware-software reliability [61, 62], system performance and performability analysis [47, 63].

For complex systems with large numbers of components, the number of system states can grow prohibitively large. This is called the largeness problem for Markov models. A major objection to the use of Markov models in the evaluation of performance and dependability behavior of systems is the assumption that the sojourn (holding) time in any state is exponentially distributed. The exponential distribution has many useful properties that lead to analytic tractability, but it does not always realistically represent the observed distribution functions.

### 2.3.2 Stochastic Petri nets

Stochastic Petri nets (SPNs) [64] and extensions have been developed as extensions to Petri nets (originally introduced by C. A. Petri in 1962) with timed transitions for which the firing time distributions are assumed to be exponential. SPNs have been extensively used in the area of dependability evaluation [48] due to the small size of their descriptions and their visual/conceptual clarity.

The firing of transitions is assumed to take an exponentially distributed amount of time. Given the initial marking of an SPN, all the markings as well as the transition rates can be derived, under the condition that the number of tokens in every

In the last two decades many extensions to the basic SPN model have been proposed to enhance its modeling power and flexibility of use. The most popular model of this type is called generalized stochastic Petri nets (GSPNs) [65]. More flexible firing rules have also been proposed, most notably the introduction of gates in stochastic activity networks (SANs)[66].

### 2.3.3 Stochastic Activity Networks

Stochastic activity networks [66] have been used since the mid-1980s for performance, dependability, and performability evaluation. This formalism is more powerful and flexible than most other stochastic extensions of Petri nets such as SPNs and

GSPNs. SANs permit the representation of concurrency, timeliness, fault-tolerance and degradable performance in a single model [66].

Informally SANs are generalized Petri nets and provide a graphical representation consisting of places, timed and instantaneous activities, input and output gates. Timed activities represent the activities of the modeled system whose durations impact the system's ability to perform. The amount of time to complete a timed activity can follow a specific distribution, such as Exponential, and Weibull. Instantaneous activities, on the other hand, represent system activities that, relative to the performance variable in question, are completed in a negligible amount of time. Cases associated with activities permit the realization of two types of spatial uncertainty. Uncertainty about which activities are enabled in a certain state is realized by cases associated with intervening instantaneous activities, represented by circles on the right side of an activity.

Uncertainty about the next state assumed upon completion of a timed activity is realized by cases associated with that activity. Gates are introduced to permit greater flexibility in defining enabling and completion rules.

SANs provides two different types of gates: input and output gates. input gates, each of which has a finite set of inputs and one output. Associated with each input gate are an n-ary computable predicate and an n-ary computable partial function over the set of natural numbers which are called the enabling predicate and the input function, respectively. The input function is defined for all values for which the enabling predicate is true. Output gates have finite set of outputs and one input. Associated with each output gate is an n-ary computable function on the set of natural numbers, called the output function. The use of gates permits a greater flexibility in specifying enabling conditions and completion rules than simple SPN, by embedding C++ code into the model.

SANs allow to define custom metrics by means of reward variables. The evaluation of the reward variables involves specifying a performance (reward) variable and a reward structure which associates reward rates with state occupancies and reward impulses with state transitions, namely, a "reward" is accumulated into a reward structure every time a set of events of interests take place during the simulation of the model.

SANs have been used as a modeling formalism in three modeling tools (METASAN [67], UltraSAN [68], and Mobius [20]), and have been used to evaluate a wide range of systems.

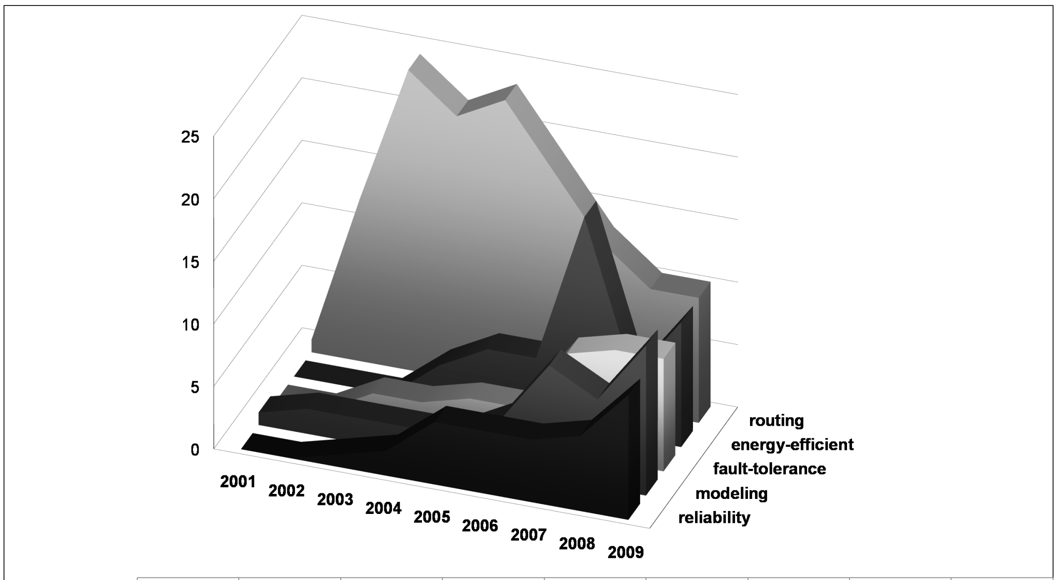
**Multi-formalism approaches**

As the system under study grows in complexity and heterogeneity a single modeling formalism reveals almost always inadequate. Current research in the area of dependability modeling tends to exploit the best from the different approaches by combining them in some hierarchical way. Multi-formalism [69, 70, 71] allows to adapt the modeling formalism to the nature and level of abstraction of the subsystem to be modeled and provide the modeler with a single cohesive view of the entire system. Modularity and compositionality ease modeling and also allows for the reuse of components. Model complexity, is tackled by a heterogeneous combination of multi-formalism modeling techniques and related multi-solution analysis. Resorting to a hierarchical approach brings benefits under several aspects, among which: i) facilitating the construction of models; ii) speeding up their solution; iii)favoring scalability; iv) mastering complexity by handling smaller models that hide at one hierarchical level some modeling details of the lower one.

Examples of applications of multi-formalisms approaches comes from safety analysis. Safety problems usually requires to account for some critical continuous variables that exceed acceptable limits. Thus, even if the property called safety is considered to be an attribute of the dependability, it often requires autonomous and specific modeling techniques. Two main modeling approaches have been recently proposed to deal with hybrid systems, i.e. systems modeled with a multi-formalism approach: Hybrid Automata and Fluid Petri Nets. Fluid Petri Nets (FPN) [72] are an extension of standard Petri Nets, where, beyond the places that contain a discrete number of tokens, a new kind of place is added that contains a continuous quantity(fluid). The fluid flows along fluid arcs according to an instantaneous flow rate. The discrete part of the FPN regulates the flow of the fluid through the continuous part, and the enabling conditions of a transition depend only on the discrete part. Hence, this extension is suitable to be considered for modeling and analyzing hybrid systems.

## 2.4 Approaches for Assessing WSNs

Figure 2.1 presents a per year trend growth analysis concerning the number of published works on several topics related to WSN. In particular, a number of papers,



| | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 |
|---|---|---|---|---|---|---|---|---|---|
| ■ reliability | 0 | 0 | 1 | 2 | 5 | 5 | 5 | 6 | 10 |
| ■ modeling | 1 | 2 | 2 | 2 | 2 | 4 | 9 | 7 | 12 |
| ■ fault-tolerance | 0 | 0 | 2 | 2 | 3 | 3 | 8 | 9 | 9 |
| ■ energy-efficient | 0 | 0 | 0 | 3 | 5 | 5 | 17 | 5 | 10 |
| ■ routing | 1 | 13 | 24 | 21 | 23 | 18 | 13 | 10 | 10 |

Figure 2.1: Per year trend growth analysis concerning the number of published works on several WSN topics.

published on various leading IEEE and ACM journals and conference proceedings[1] over the last nine years have been considered. The recent scientific production on WSNs dependability reliability, assessment and modeling is growing due to the proliferation of critical application scenarios where WSNs are starting to be adopted. In the last three years, the attention of the community is focusing more and more on

[1]A Set of considered venues and journals are: IEEE International Conference on Distributed Computing Systems; IEEE International Conference on Sensor Technologies and Applications; The Sensors Journal; International Symposium on Mobile Ad Hoc Networking & Computing; IEEE Transaction on Reliability; IEEE Transaction on Parallel and Distributed Computing; IEEE Transaction on Wireless Communications; IEEE Transaction on Computers; IEEE Micro; International Journal of Sensor Networks; IEEE International conference on Dependable Systems and Networks; International Symposium on Reliable and Distributed Systems; Journal of parallel and distributed systems; International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing; World of Wireless, Mobile and Multimedia Networks, 2008; Journal Wireless Networks; International Conference on Mobile Computing and Networking; European Conference on Wireless Sensor Networks; ACM Transaction on Sensor Networks; ACM Conference on Embedded Networked Sensor Systems Networks; IEEE computer communications; IEEE Journal in Selected Areas in Communications. International Conference on Parallel Processing.

i) modeling, showing a 200% of increment in the number of published work (2006 - 2009) , ii) on reliability studies with an increment of the 100% from 2006, of which 73% of the publications are related to evaluation of real-world setups of WSNs, giving evidence on the dependability threats that can be experienced in practice, and iii) on fault tolerant solutions (200% of increment from 2006), which are giving more and more attention to security (about the 80% of the analyzed work of this set). It is interesting also to note that in the last three years, well established and mature topics, such as researches on WSN routing shown a decreasing trend in the number of published work, arriving to be comparable with minor topics such as modeling and reliability assessment.

Current adopted approaches to evaluate WSN dependability attributes can be categorized according to two classes: experimental, and model based approaches.

### 2.4.1 Experimental approaches

Experimental test-beds are used in several works such as [73, 74, 75, 76, 77, 78]. One of the most relevant studies dealing with an experimental test-bed is the Great Duck Island project [12]. This project focuses on a in-depth study of applying WSN to real-world habitat monitoring on Great Duck Island (Maine, USA). The paper provides important analysis regarding the experimental evaluation of node failure cumulative probability, demonstrating that, in harsh environment, the 50% of the nodes becomes unavailable within only four days out of three months of the deployment.

Experimental setup of WSNs is thus useful to gain insight in the actual failure behavior of WSNs, and to come up with proper failure mode assumptions. However, results are difficult to reproduce and are too specific for the given application scenario. Hence, they can hardly been used to validate design choices through dependability assessment at development time.

For this reason, recent research studies on WSNs adopt either analytical or simulative approaches to validate proposed solutions and anticipate deployment choices at design and development time.

In [77] a prototype WSN composed of 27 Crossbow Mica2 motes [23] is deployed in a real coal mine. System errors, detection latency, packet loss rate, and network bandwidth were measured. Based on the collected data, authors conducted a large-scale simulation to evaluate the system scalability and reliability.

In [78] authors claim that i) despite software algorithms can be tested through simulations and syntax checking, it is difficult to predict or test for problems that may occur once the WSN has been deployed, due to the high complexity of the WSN dynamics, and ii) the requirement for testing is not limited at the design phase. Hence, they propose a framework for in-situ testing and validation which instruments the final applications, using the environmental data and stimuli as real input to the testing and validation process.

### 2.4.2 Model Based Approaches

The work in [4] faces the design and implementation of a complete running WSN for surveillance missions, called VigilNet. They show that common reliability assumptions in much current research on WSN, do not hold well in practice, leading to wrong evaluation of critical parameters, e.g., network lifetime. For instance, they found that the packet loss in the MICA2 [23] platform can be as large as 20%.

A number of analytical approaches are presented in [79, 80, 81, 82, 83, 84, 85, 86], which allow to formally express the main characteristics of WSNs in order to assess their properties. Energetic consumption modeling and lifetime analysis of a WSN are presented in [79, 80, 87, 88]. In [79] authors provide an analytical model used to forecast the power consumption and thus the lifetime of the network, only related to the communication activity. In [80] a network state model is presented to forecast the network residual energy. In [87] an on-line battery model for estimating the remaining energy of a node battery in WSNs is presented. In [88] author studied the effect of using mobile sinks for data gathering in wireless sensors networks, to mitigate energy holes issues in proximity of fixed sink nodes. All the work in [80, 87, 88] consider the network divided in several zones in order to detect those critical due to energetic aspects.

In each of these work, the lifetime is concerned as dependent both on power consumption and on how many packets have been processed. In [82] authors propose a generic definition of sensor network lifetime for use in analytic evaluations as well as in simulation models. They claim that available definitions of network lifetime are unable to reflects all the application demands and environmental influences. The provided definition of lifetime is composed in a modular way, enabling the incorporation of different application requirements, such as i) number of alive nodes, ii) time latency in the delivery process, ii) delivery ratio, iii) connectivity, iv) coverage, and v) availability. Finally they define the WSN lifetime as the time in which application requirements on i)-v) are fulfilled.

The work in [81] focused on reliability and delay measures for a WSN. In this work, a probabilistic graph is adopted to model the network behavior, associating an operational probability to each node, achieved by means of field data analysis on real sensors. Authors assume that failures are caused by component wear out, power failures and in some cases by natural catastrophes, and they are randomly distributed in the network. They showed that evaluating the reliability of an arbitrary WSN is a #NP-hard problem for arbitrary networks. Then they present two algorithms in order to compute the reliability, and the expected message delay for arbitrary networks. Finally authors provide numerical results on the proposed algorithms.

In [86] authors investigate node aging process phenomena. They examine the general node aging problem by unfolding the energy consumption rate and the failure rate. To this aim, authors provides for each node a survivor function in terms of a *Weibull* distribution. Then, the energy consumption rate in a data gathering tree is presented with and without data aggregation strategies. It is shown that the node aging process has a significant impact on the connectivity as the hop distance increases. In particular the simulation process evidences that nodes at first hop without data aggregation consume their energy much faster than the case with aggregation. Thus the consumption is proven to be dependent on the number of children nodes. The work ends evidencing that data aggregation scheme enhances the node lifetime and thus the network reliability.

The work in [84, 85] focus on sensor network where nodes send their data to a sink node by using multi-hop transmissions. In in [84] authors develop an analytical to investigate the trade-offs existing between energy saving and system performance, as the sensors dynamics in sleep/active mode vary. The proposed model allowed authors to derive several performance metrics, among which the distribution of the data delivery delay. The work in [84] is one of the first work adopting analytical model specifically representing the sensor dynamics in sleep/active mode, while taking into account channel contention and routing issues. However, they assume that the WSN can be model using Markovian techniques, which may result in overestimated metrics. Finally, they do not take into account how the energy consumption and failure behavior impact on evaluated performance metrics. In [85] the problem of multi-hop lifetime aware routing is addressed. Authors provide a linear programming model to compute the minimum cost arborescence for reaching the sink node, preserving the maximum lifetime of nodes.

All mentioned works show that analytical approaches are suitable to assess non-functional properties of WSNs, such as lifetime, coverage, reliability, and so on. However, it has been proven that the analytical evaluation of non-functional properties of an arbitrary WSN is a *NP-hard* problem [81]. Similar observation is provided in [89] in which authors show that maximizing WSN lifetime, while maintaining coverage and connectivity simultaneously, without any sensing or communication range restrictions. is a NP-hard problem.This is partially due to the potentially unlimited number of deployment choices that can affect WSN applications (e.g., adopted hardware, routing algorithm, network topology, etc.). In addition, the potential of analytical models is also affected by the high level of abstraction that does not allow to consider detailed deployment aspects, leading to approximate and unrealistic results.

**Fault/Failure Models**

Experimental studies have contributed to the definition of failure mode assumptions for WSNs. They provide valuable understanding on how WSNs fail in practice. Following this wave, [90] introduces a taxonomy for WSN faults. A fault is concerned with respect to inconsistent measurements provided by a sensor, such as measurements offset bias, frozen reading, death of a sensor. The work in [91] classifies hardware components of a sensor node into two groups: i) computation engine, storage subsystem, power supply infrastructure; ii) sensor and actuators. The latter is assumed to be failure prone due to calibration error, random noise error, and complete malfunctioning. [92] evaluates energy/reliability trade-off of multipath schemes, assuming two widely different failure model: independent and geographically correlated (patterned failures).

Simulative approaches aims at evaluating WSN fault/failure models are provided in [93, 94, 91, 95]. In [93] authors evaluate the reliability of the communication infrastructure of a WSN. To the best of our knowledge, this is one of the few work that does not assume WSN failures randomly or uniformly distributed on the network area. The paper reports also consideration of common cause failures [96] in the provided reliability analysis. A sensor node can fail $s$-independently[2] due to component wear-out, power failure or depletion, and natural catastrophes according to an exponential distribution with a rate of $\lambda = 1E - 7$ failure per second. The network can fail due to collision, interference and jamming. However, no assumption on the fault nature are provided.

In [95] author study the connectivity properties of large-scale wireless sensor networks and discuss their implicit effect on routing algorithms and network reliability. They assume a network model of n sensors which are randomly distributed over a field. The sensors may be unreliable with a probability distribution, which possibly depends on n and the location of sensors. Two active sensor nodes are connected with probability p e (n) if they are within communication range of each other. Author prove a general result relating unreliable sensor networks to reliable networks. These results are shown through graph theoretical derivations and are also verified through simulations. The work in [92] evaluates energy/resilience tradeoff of multipath schemes, assuming two widely different failure model: independent and geographically correlated. Authors show that multipath routing (e.g. random walk routing) can preserve WSN lifetime by means of its path-resiliency. They finally provide useful insights on the relationship between the probability of node isolation from the sink, and the path length. In [97] simulations are used to evaluate the impact of failures and routing protocols on achievable sensing coverage. They start from the observation that different routing protocols lead to different values of achievable sensing coverage when some

[2]$s$ node can fail due to a common cause.

nodes are no longer available. Finally authors propose coverage-preserving routing protocols for randomly distributed WSNs that is shown by simulation to be able to substantially improve the performance of network sensing coverage, if compared with the original ones.

In [94] a framework for modeling reliability of data transport protocols in WSN is presented. Here, faults are categorized in i) communication failures ii) Node failures. These are further subdivided in accidental damage, sensing devices, energy depletion and transient failures.

Fault model used in fault detection strategies are presented in [91]. The work in [91] defines a network and a fault model. Hardware components of sensor motes are here divided into two groups: i) computation engine, storage subsystem, power supply infrastructure; ii) sensor and actuators: these are assumed to be the only components prone to failure. The assumed faults are only calibration error, random noise error, and complete malfunctioning. Nodes are assumed to be still capable of receiving, sending, and processing when they are faulty.

Work that deal both with WSN dependability evaluation and network dynamics are [98, 99, 100]. In [98] authors give evidence of the network dynamics awareness as a fundamental concept to evaluate/develop faut tolerant WSN solutions. In [99] is evidenced the dependency of WSN dependability on the base station position. In [100] authors make their point on the paramount importance that environment, sensing hardware and network related aspects have in a pragmatic evaluation of WSN lifetime, outlining the basic structure for a WSN evaluation framework.

### 2.4.3 Discussion

All the analyzed work presented interesting methods and/or techniques which are agnostic for the target application being used. This is a limiting factor since that nodes' lifetime is strictly dependent upon the activities performed by the installed software applications (e.g., different aggregation schemes [101], or different computational loads ). In addition, most of the existing failure models, stem from strong assumptions on network topology (e.g., random topology, which is most often unrealistic) and on power consumption figures (e.g., infinite energy or ideal battery cells). Finally each analyzed work defined its own fault model, making it difficult to generalize the results. The lack of a realistic fault model is also due to the fact that the majority of results (over the 80% of the papers considered in our study) are drawn by means of rather abstract simulations. Finally, None of the presented works take into the account the nature and the objective of the considered WSN, i.e. the envisioned mission.

## 2.5 Available WSN assessment Frameworks

In this section we analyze the main characteristics of the most adopted WSNs simulation frameworks. Detailed aspects of frameworks are out of the scope of this thesis, and can be found in the referenced papers. Several simulation engines for WSNs have been recently proposed. A subset of the most referenced/used simulation/emulation

Table 2.1: Characteristics of the analyzed simulators

| | Characteristics | | | | | |
|---|---|---|---|---|---|---|
| **Simulator** | **Routing** | **Propagation** | **Failures** | **Energy** | **Workload** | **Topology** |
| **SHAWN** | YES (programmed in C++) | YES | Partial (packet loss & corruptions) | NO | YES (programmed in C++) | YES |
| **ALGOSENSIM** | YES (described in XML) | YES | Partial (packet loss) | YES (simple model | YES | YES (random & uniform) |
| **NS-2** | YES (choice of 4 algorithms) | YES | Partial (packet loss) | YES | YES (programmed in TCL) | YES |
| **GLOMOSIM** | YES (choice of several algorithms) | YES | Partial (packet loss) | NO | YES (programmed as modules) | YES |
| **J-SIM** | YES (choice of several algorithms) | YES | Partial (packet loss) | YES | YES (programmed in TCL) | YES |
| **TOSSIM** | YES (programmed in nesc) | YES | Partial (packet loss & corruptions) | YES | YES (programmed in nesc) | YES |
| **EMSTAR** | YES (programmed in nesc) | YES | Partial (packet loss) | NO | YES (programmed in nesc) | YES |
| **ATEMU** | YES (programmed in nesc) | YES | Partial (packet loss & corruptions) | NO | YES (programmed in nesc) | YES |
| **AVRORA** | YES (programmed in assembly) | NO | NO | YES | YES (programmed in assembly) | YES |

environment of WSNs are reported in Table 2.1 which relates the considered assessment frameworks to the provided capabilities in terms of the possibility to reproduce/take into account i) routing protocol behavior, ii) medium propagation, ii) failures, iv) energy consumption, v) workload and vi) customizable topology. Let us analyze the table by columns.

Examples are TOSSIM [16], Avrora [102], Ns-2 [15], J-SIM [103], Glomosim [104] and EMstaremstar. TOSSIM [16] is an event-based simulation environment for WSNs based on the TinyOS operating system [105]. The user can simulate his applications written with the NesC programming language [106]. The simulator permits to observe the behavior of nodes, even in terms of energy consumption, under different conditions in terms of topology and wireless propagation models. TOSSIM simulates the execution of nesC code on TinyOS/MICA hardware, aims to study the behavior of TinyOS and its applications rather than analyzing the performance of new protocols.

Avrora is a WSN emulator which helps developing sensor network with clock-cycle accurate execution of microcontroller programs [102]. Avrora attempts to find a middle ground between TOSSIM and microcontroller emulator, such as ATEMU [107]. Avrora is implemented in Java, unlike the other two emulators, which are written in C. Similar to many of the object-oriented simulators, Avrora implements each node as its own thread. However, it still emulates actual Mica code. Avrora runs code in an instruction-by-instruction fashion. However, the simulator attempts to achieve better scalability and speed than TOSSIM by avoiding synchronization of all nodes after every instruction.

Ns-2 [15] is a discrete event networks simulator that began in 1989 as a variant of an even earlier network simulator. It is written in a combination of C++ and OTcl, an object oriented scripting language. Support for wireless networks was added in 1997; it is designed to simulate wireless LAN protocols, though later expanded to mobile ad-hoc networks. A project at the Naval Research Laboratory produced an extension to NS-2 for sensor networks [108].

This extension adds a channel module for modeling physical phenomena such as sensor nodes and the environment. Although NS-2 has been used to evaluate wireless sensor networks, the accuracy of results are questionable since the MAC protocols, packet formats, and energy models are very different from those of typical WSN platforms.

J-Sim (formerly known as JavaSim) is a general purpose Java-based simulation environment [103] and is based on the ns-2 network simulator. It permits to describe the WSN in terms of network components and TCL scripts and considers correlated aspects, such as energy depletion and network topology. It has been built upon the notion of the autonomous component programming model. The main benefit of J-Sim is its considerable list of supported protocols, including a WSNs simulation framework with a very detailed model of sensor networks, and a implementation of localization, routing and data diffusion algorithms. J-Sim provides a GUI library for animation, tracing and debugging support and a java scripting interface Jacl.

GloMoSim [104] is a simulation environment for wireless and wired network systems. It employs the parallel discrete-event simulation capability provided by Parsec (PARSEC: Parallel Simulation Environment for Complex System) which is a C-based simulation language. GloMoSim source and binary code can be downloaded only by academic institutions for research purposed. Commercial users must use sQualNet (sQualnet: A Scalable Simulation Framework for Sensor Networks), which is the commercial version of GloMoSim. While effective for simulating IP networks, it is not capable of simulating any other type of network. This effectively ensures that WSNs cannot be simulated accurately.

EmStar [109] is a software framework to develop WSN applications on 32-bit platforms as Microservers, running Linux, as well as for conventional 8-bit platforms running the TinyOS operating system. The EmStar environment contains a Linux micro kernel extension, libraries, services and tools. The most important tools are: (1) EmSim: A simulator of the microservers environment where every simulated node runs an Em-Star stack, and is connected through a simulated radio channel model. EmSim is not a discrete event but a time-driven simulator, which means that there is no virtual clock. (2) EmCee: An interface to real low-power radios, instead of a simulated radio model, obtaining radio emulation. Additionally, the UCLA staff has developed (3) EmTOS: An extension of EmStar that enables nesC/TinyOS applications to run in an EmStar framework. OMNeT++ [110] is a public source component-based discrete event network simulator. The simulator mainly supports standard wired and wireless IP communication networks, but some extensions for WSN exist. Like NS-2, OMNeT++ is popular, extensible and actively maintained by its user community in the Academia who has also produced extensions for WSN simulation. OMNeT++ uses C++ language for simulation models. Simulation models (modules) are assembled with high-level language NED into larger components to represent greater systems. OMNeT++ is capable of running most TinyOS simulations by NesCT application that converts TinyOS source to simulator compatible C++ code [111].

SENS (Sensor, Environment and Network Simulator) [112], is a platform-independent and has a modular, layered architecture which is capable of modeling the application, networking and physical environment. The ability to model physical environments by defining them as a grid of interchangeable tiles is a core strength of SENS. Three modeling implementations with different signal propagation characteristics including concrete, grass and walls are currently available. However, the existing power model needs an improvement to include a battery model. No details of routing protocols behavior are available.

Shawn [113] is open source simulator designed to to support large-scale network simulation. Shanw is a algorithm oriented simulator [5][6] and it provides a support to the user for writing algorithms to simulate in C++ language. In addition, the simulator allows to decide simulation characteristics such as topological model (which can be selected among a set of already available models) and medium propagation model which can be used to reproduce delivery delay and packet loss.

Similarly to Shawn, Algosensim is a algorithm oriented simulator [114]. Algosensim provides powerful means for the development of routing and positioning algorithms for WSN. In addition, Algosensim includes the simulation of sensor nodes hardware aspects, such as battery discharge process. Mobile scenarios are also envisioned, despite not implemented yet. Packet corruption and loss is simulated and the simulator allow to select nodes to be considered as failed, but it does not allow to reproduce the failure behavior by itself.

### 2.5.1 Discussion

The failure behavior is faced partially, i.e., only with respect to network failures, such as packet loss and packet corruption (corruptions are modeled only by SHAWN, TOSSIM, and ATEMU; AVRORA does not model network failures). As for node failures, only node switch offs (due to battery exhaustion) are simulated, and only

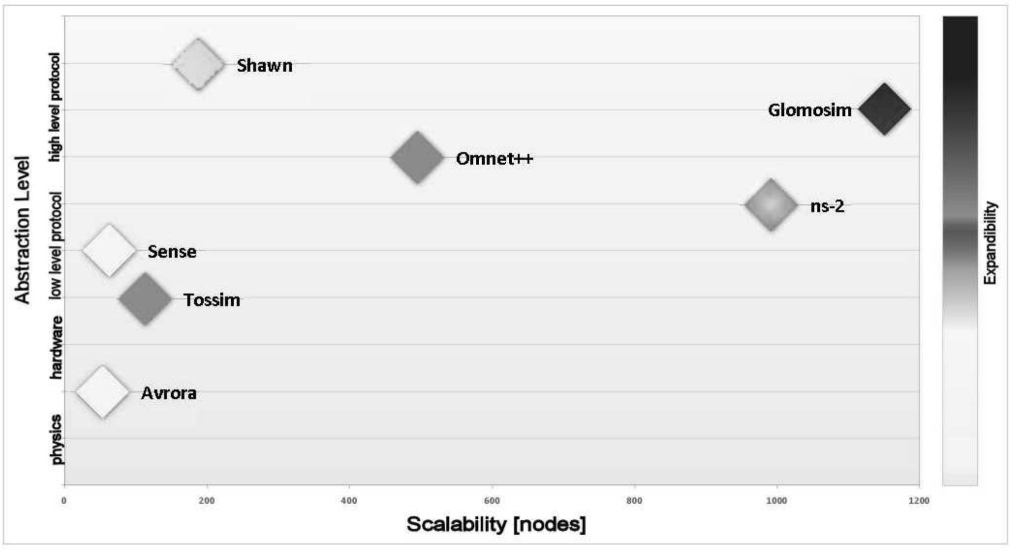


Figure 2.2: Classification of analyzed simulator: the darker the color, the more extensible the simulator

by a subset of simulators, such as, ALGOSENSIM , NS-2, J-SIM, and AVRORA. Node failures due to the sensing hardware or to the software running on nodes are not considered at all.

The energy consumption is an important aspect affecting (and affected by) WSN applications, network dynamics and routing algorithms. However, several frameworks do not model this aspect, and other frameworks simplify it. For instance, TOSSIM estimates the energy consumed by nodes, by means of an energy profile of the software running on the emulated nodes, but it does not simulate battery exhaustion (WSN nodes run without interruption during the simulation).

A further qualitative classification of a subset of the considered simulator is provided in Figure 2.2, considering scalability, extendibility and level of abstraction provided by the considered simulator/emulator environment.

If compared to analytical approaches, simulative approaches are closer to real WSN settings, with greater attention to low-level aspects. They allow to reproduce the expected behavior of every single WSN node on the basis of the real algorithms that are planned to be run on nodes. Despite this, WSN simulators still miss to consider several real-world aspects that may affect the results, such as detailed failure models and their link to network dynamics (e.g., how a failure of a node in the network affects the behavior of the remaining nodes). In addition, it is not always possible to express and hence evaluate non-functional properties trough simulative approaches. In addition, such evaluation would require several simulation runs in order to achieve results with adequate confidence. This in turn would increase the time needed for the simulation by order of magnitudes, given the low-level of detail of these approaches. As another issue, the analyzed frameworks present a not negligible difficulty of managing the inter-dependence between all aspects impacting on WSN behavior (routing, energy, failures, etc.), which is driven by state changes (e.g. topology update due to failures/recoveries). The deriving complexity has to be explicitly managed by users. This brings to another limitation: the difficulty of use. Users have to program the workload and the routing algorithm. In some cases, the routing algorithm can be chosen from a library, but the inclusion of new algorithms (e.g., facing change management) in the library is not a simple task.

# Chapter 3

# Resiliency of WSNs

*Available definitions of network resiliency encompassing only the connectivity to the sink or between nodes, are not enough to characterize the data-driven nature of WSNs. The service delivered by the WSN does not encompass only the connection, but also the computation, i.e., even when sensor nodes are potentially connected ( a path exists between nodes and sink node), data losses can still occur.*
*This chapter introduces the concept of WSN resiliency as compound non-functional property, composed of Connection Resiliency and Data Delivery Resiliency, which not interrelated. Data Delivery Resiliency is defined in this thesis as the persistence of the delivery efficiency of node measurements/computation to the sink, against manifesting changes. Then, it presents the holistic approach proposed in this thesis for assessing the resiliency of WSNs.*

## 3.1 System Assumptions

The system under study is a wireless sensor network made out of $N$ nodes. Each node is composed of a processor board, a sensor board equipped with one or more sensors (e.g., humidity sensor, thermistor, photo resistance, etc.), a radio board enabling the wireless communication between nodes, and a power supply unit, including batteries. We assume that initially all nodes have the same characteristics and capabilities. The network is stationary, i.e., nodes do not move during the WSN lifetime, which is typical for environmental and structural monitoring applications. The network includes a higher-level node (usually a laptop or a set-top-box), called “sink” node, which is responsible of gathering data and of controlling WSN nodes. Sink node failures are not considered in this thesis. This is reasonable due to the more reliable hardware/software equipment a sink is typically made of, if compared to a WSN node.

The mission of the considered WSN is to gather data from nodes (e.g. environmental measurements) and to forward it to the sink node. Typical application requirements for this types of WSNs are i) to deliver of a given amount (or fraction) of measurements to the sink node, and ii) to keep a given connectivity degree of the network, avoiding that significant portions (or sub-networks) of the WSN are completely isolated from the sink. Both the requirements have to be met within a given time horizon, i.e., the considered mission time.

## 3.2 WSN Resiliency

Resiliency has been recently defined as *the persistence of dependability when facing "changes"*[11]. Changes are related to mutations in the topology, workload, link quality etc. due to failure/recovery of nodes. For instance, a node failure in a WSN has the effect of modifying the system topology by the removal of a communication node and its corresponding links. Since a real WSN configuration is not generally a fully connected graph, successive failures may result in a disconnection of the system, namely a disconnection failure, and therefore prevent a set of nodes from reaching the sink (i.e. isolated nodes). Hence, the concept of connection resiliency is related to the WSN topology, i.e. the degree of path redundancy in the network. However, the service delivered by the WSN does not encompass only the connection, but also the computation, i.e., even when sensor nodes are potentially connected (a path exists between nodes and sink node), data losses can still occur.

To overcome this limit, *this thesis defines the concept of data delivery resiliency and qualifies the concept of WSN resiliency as a non functional properties composed by both connection resiliency and data delivery resiliency, which are not interrelated.*

The concept of data delivery resiliency relates to i) the computational load on nodes which may causes packet losses due to buffer overrun, ii) application requirements, e.g. at least a given amount of produced measurements must be delivered to the sink node iii) routing and MAC protocols impacting on the data delivery features and packet error rate and iv) radio interferences and packet loss/corruption phenomena on the propagation medium. The variation in the amount of useful data received by the sink due to disconnection failures that can be tolerated by the WSN depends on the requirements of the application.

Hence, assessing the data delivery resiliency as well as the connection resiliency is a crucial if not essential task in designing dependable WSNs, since it could help to i) anticipate critical choices e.g., concerning node placement, running software, routing and MAC protocols, ii) mitigate risks, e.g., by forecasting the time when the WSN will not be able to perform with a suitable level of resiliency, and iii) prevent money loss, e.g., providing a criteria to plan and schedule maintenance actions effectively.

The idea of complementing connection resiliency with data delivery resiliency stems from the observation that connection resiliency can be misleading for characterizing the overall resiliency level of a WSNs. Let us discuss this claim with an example. Let's consider a WSN deployed so that the sink node is connected only to a single node of the WSN, and all the remaining nodes rely on that node for reaching the sink. The network is connected, i.e. every node of the WSN can reach the sink. However, the only node connected to the sink may experience higher packet loss and failure rate than other nodes, since the more stressful forwarding activity it have to face. Hence, the WSN, despite presenting a high connection resiliency (a high number of nodes fails in the network but not the only node directly connected to the sink), manifests low level in data delivery resiliency, since measurements produced by all nodes are funneled to the sink through a single node that, due to the induced overhead, is not able to accept all incoming packets. This way, not all the produced data is delivered to the sink. Depending on application requirements this may lead to a WSN failure or not. For instance, structural health monitoring and security monitoring applications (See Chapter 1) require all data from every node to be delivered to the sink node. For structural health monitoring, typically acceleration samples down to 500$\mu$G have to be acquired at a frequency higher than 1KHz and synchronously at all nodes. Failing in delivering synchronization commands, or data then an imperfect picture of the monitored structured will be achieved, making analysis in the best inaccurate. Nevertheless, optimizing the WSN topology is not the panacea to the mentioned scenario, and the evaluation of only connection resiliency may anyway produce misleading results.

As a further example, figure 3.1 anticipates some of the results of Chapter 7, and in particular it sketches the results we obtain from two hypothetical WSNs,

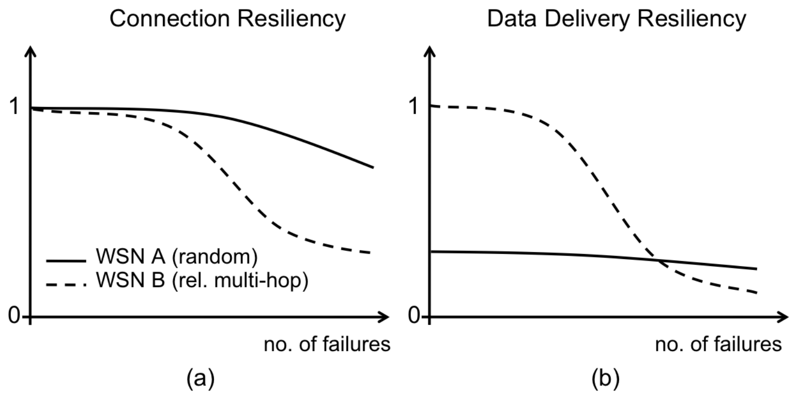


Figure 3.1: WSN Resiliency (failure domain) (a) connection resiliency, (b) data delivery resiliency

$A$ and $B$, both of them connected and with same topology (the sink is reachable through multiple nodes). In this example, the only difference between the considered WSN regards the routing algorithm being adopted. $A$ adopts a random walk routing algorithm (which randomly selects one of the neighboring nodes to forward a packet to the sink), and $B$ uses a reliable multi-hop routing algorithm (which builds a routing tree and uses acknowledgments and retransmissions to reduce data losses).

Looking at figure 3.1.(a), it seems that random routing has a greater capacity to withstand failures, keeping a higher level of connection resiliency, if compared to reliable multi-hop routing (this result is confirmed under several different network conditions, in Chapter 7). Random routing is a very light weight protocol, which better preserves the life of WSN nodes. Hence, at a first sight, a developer would be induced to choose random routing for a given WSN. However, figure 3.1.(b) shows that data delivery resiliency of WSN $B$ outperforms WSN $A$, due to the higher capacity of reliable multi-hop at delivering a higher amount of useful data to the

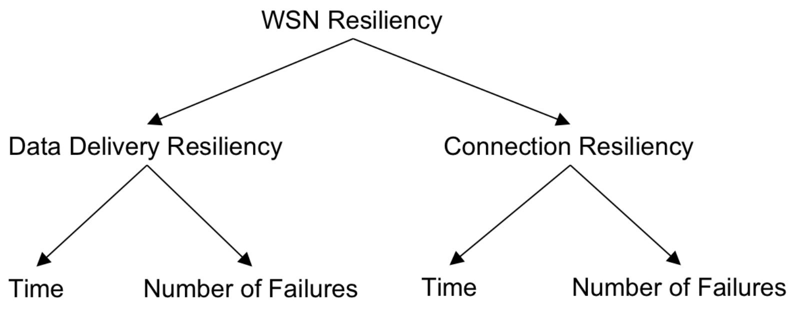


Figure 3.2: Scheme for the definition of the resiliency of WSNs

sink, even in the presence of failures. Hence, depending on application requirements, data delivery resiliency add one further element to discriminate between different solutions. Also, results are affected by other factors, such as the workload running on nodes and the failure rate. This variable factors complicate the evaluation task, and motivate the need for an holistic approach for resiliency assessment.

### 3.2.1 Definitions

Figure 3.2 shows the taxonomy of WSN resiliency proposed in this thesis. We conceive the Resiliency of a WSN as a compound property composed of connection resiliency and data delivery resiliency. The former is the *persistence of the amount of data delivered to the sink* , the latter is the *persistence of network connectivity*, both in spite of WSN changes. Data delivery resiliency and connection resiliency can be estimated in the time domain, i.e. as a function of time, or against the number of failures manifested in the WSN. Estimating the resiliency in the time domain is useful to forecast non functional properties (e.g. provided level of dependability) in order to drive design choices, at design time, or to schedule maintenance actions, during the operational phase. On the other side, the evaluation of the resiliency against manifested failures is of paramount importance to assess the behavior of the WSN when dealing with fault tolerant routing algorithms, clustering strategies, and redundant nodes.

The concept of connection resiliency for computer networks and ad-hoc networks has already been defined in the literature. In particular, we qualify this concept for WSN, extending the definition provided in [13] and defining the probability of disconnection $P(n)$ as follows:

$$P(n) = Q(n) \prod_{j=1}^{n-1} (1 - Q(j)) \tag{3.1}$$

where $Q(j)$ is the probability that after the $n_{th}$ failure, a set of $K$ nodes is isolated from the sink, given that for $n-1$ failures, they are connected. In this work we assume $K = 1$ for the sake of simplicity. This does not impact on the quality of provided results neither on the followed approach. Consistently with data delivery resiliency, from Equation 3.1, we define the connection resiliency in the time domain (Equation 3.2) the longest time interval in which the WSN is able to survive while preserving a disconnection probability lower or equal to a given threshold. $\rho$, namely:

$$\int_0^{t^* \wedge t^* \in [0,T]} p(t)\, dt \leq \gamma \tag{3.2}$$

where $[0, T]$ is the observation interval, $t^*$ the instant of time : $\forall t > t^*$ Equation 3.2 is not verified. p(t) is the probability density function of WSN failures in the observation interval.

Similarly, we define the Connection Resiliency with respect to manifested failures, as the greatest number of failures $NF$ that the WSN can accommodate while preserving a disconnection probability lower or equal than a threshold. In other terms:

$$\sum_{i=1}^{NF} P(i) \leq \gamma \tag{3.3}$$

In addition to connection resiliency, we define the concept of data delivery resiliency for WSNs. To this aim, we start from the definition of probability of data delivery failure to the sink as:

$$D(k, n) = R(k, n) \prod_{j=1}^{n-1} (1 - R(k, j)) \tag{3.4}$$

where $R(k, n)$ is the probability that after the $n_{th}$ failure, a given amount of data ($k$) computed by nodes is not received by the sink, given that for $n - 1$ failures, it was delivered. More in detail, $R(k, n)$ is defined as:

$$R(k, n) = \int_{T(n-1)}^{T(n)} r(k, t|\text{corruptions})dt + \int_{T(n-1)}^{T(n)} r(k, t|\text{duplications })dt \tag{3.5}$$

Where i) the first term takes into account the delivery probability of a given amount of data when no corruption caused the discard of the packet (e.g. the MAC layer was able to correct a bit error or no bit inversion manifested during the transmission), ii) the second term takes into account the delivery of non duplicated packets, hence relating to routing protocol efficiency. Equation 3.5 relates to the manifested failures by means of $T(n - 1)$ and $T(n)$. More specifically, when computing 3.5 with respect to the number of manifested failures, $T(n - 1)$ is the time when the $n - 1th$ failure manifested and $\Delta T(n)$ is the time interval between the $n - 1th$ and the $n - th$ failure. When computing 3.5 with respect to time, $T(n)$ is the instant of time of the $n-th$ evaluation of the Data Delivery resiliency over an observation interval of $T(n) - T(n-1)$.

From Equation 3.5 we define the data delivery resiliency in the time domain as the longest time interval $[0, t^*]$in which the WSN is able to contain data delivery failures hence delivering an acceptable amount of useful data to the sink, i.e.:

$$\int_0^{t^* \wedge t^*} d(t)\, dt \leq \rho \tag{3.6}$$

where $[0, T]$ is the observation interval, $t^*$ the instant of time : $\forall t > t^*$ Equation 3.2 is not verified. Analogously, we define the data delivery resiliency with respect to manifested failures, as the greatest number of manifested failures $NF$ that can take place in the WSN, while preserving an acceptable amount of data delivered to the sink, i.e. :

$$\sum_{i=1}^{NF} D(k, i) \leq \rho \tag{3.7}$$

It is worth noting that the provided definitions of Date Delivery Resiliency relate to the size of the amount of data delivered to the sink by means of equation 3.4. This way it is possible to compute the Data Delivery Resiliency with respect to application requirements, e.g. at least on k computed measurements, a fraction of them must be correctly received by the sink node.

## 3.3 Routing issues influencing WSN Resiliency

It is not the primary aim of this thesis to overview and detail the large spectrum of existing routing algorithms for WSNs; interested readers could refer to good surveys in the literature [21, 115]. Rather, in this section we review and classify the main underlying characteristics, common to several routing solutions, affects the WSN resiliency. The first characteristic we consider is the selection mode of forwarding nodes. This is a basic function of each routing algorithm: each sensor node has to select one (or more) node, among its neighboring nodes, to be used as a forwarder to the sink node. According to this characteristic, routing algorithms can be classified as random or based on a selection function.

Flooding and Gossiping [116] are well-known examples of the first class of algorithms, where no selection function is used. Nodes simply forward they packets to all their neighbors (flooding) or to a random subset of them (gossiping), with the assumption that data will eventually reach the sink. These algorithms are very simple to implement, but are inefficient in terms of the number of packets actually delivered to the sink, in a time unit. The second class of algorithms can be broken into several sub-classes, depending on the selection function. A non-exhaustive list of selection criteria is reported in the following. For each criterion, we indicate the name generally used in the literature to refer to the corresponding routing solutions. Clearly, different criteria have a different impact on performance metrics. i) Select the neighbor on the path that minimizes the number of hops to reach the sink (flat and/or hierarchical multi-hop routing); ii) select the neighbor on the path that minimizes the overall energy to reach the sink (energy-aware routing); iii) select the neighbor on the path that satisfies a set of quality metrics (other than energy), such as delay, reliability, bandwidth, etc. (QoS-based routing); iv) select the neighbor geographically closest to the sink (location-based routing); and v) select the neighbor based on the type of information contained in the packets (data-centric routing). All these solutions require sensor nodes to maintain a routing table to perform the selection. The update of such tables requires extra operation, impacting on performance metrics. Hence, the second characteristic to be considered is the routing table update mode. Without loss of generality, three update modes can be considered: i) reactive update, i.e., tables are updated upon changes (such as node crashes, residual energy depletion, etc.), ii) proactive update, e.g., the table is periodically updated, and iii) application driven update, i.e., tables are computed on-demand, only when required by applications (this is typical of query-based solutions, often used in conjunction with data-centric routing).

Another key characteristic is the overhead introduced by routing solutions in terms of extra packets. For instance, the implementation of a reliable multi-hop protocol requires acknowledgments packets. Another example is data-centric routing, such as SPIN (Sensor Protocols for Information via Negotiation) [117]. SPIN requires to flood advertisement packets (to announce the presence of new data), and to send back request packets (to command the data transfer and to update routing tables on demand). The last characteristic we consider is the use of data aggregation techniques. The main idea of data aggregation is to combine the data coming from different sources en route (in-network aggregation) by minimizing the number of transmissions, thus prolonging network lifetime. Directed Diffusion [118] is a well-known example of routing algorithm using data aggregation as its foundation.

Chapter 4

# Orchestrating Behavioral and Analytical Simulation: the Holistic Approach

*This chapter presents the holistic approach proposed in this thesis. The objective of the approach is to evaluate holistically how inter-dependent factors such as workload and failure behavior interacts, and hence, to assess how the WSN behaves in whole. To this aim, the approach combines the expressiveness of analytical models with the capability of behavioral simulators to estimate detailed figures on the system behavior.*
*In order to avoid behavioral simulations upon each change manifesting during the simulation of the failure model, the approach relies on an external component. Such component is here referred as External Engine and orchestrates the failure model simulation recomputing model parameters values upon triggered changes. The External Engine decouples analytical models from change management issues, achieving higher modularity and simplifying the structure of the failure model, while taking into account all needed details. Moreover, the External Engine is in charge of generating the analytical model with respect to behavioral simulation results and user preferences, avoiding a modeling phase performed by the final user.*

## 4.1 Holistic Approach for Resiliency Assessment

The proposed approach is depicted in Figure 4.1 and it is organized in 7 steps. In step 1 the user provides the needed inputs to configure the real WSN scenario in terms of: i) the number and type of nodes, ii) the network topology, iii) the workload (i.e., the user application) to be run on each node, iv) the radio communication technology, v) the adopted routing algorithm, vi) the battery technology for each node, and vii) the

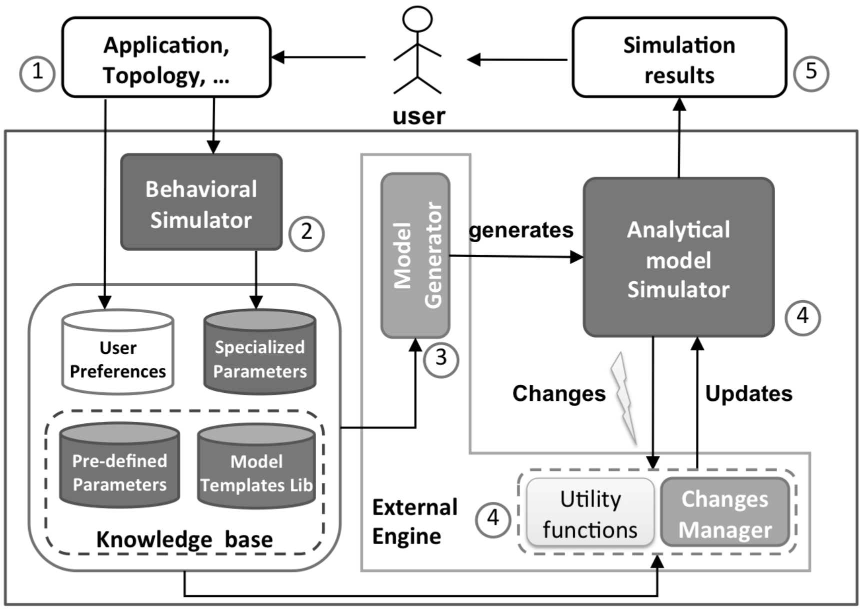


Figure 4.1: The proposed approach

sensing hardware technology for each node. Inputs from i) to v) are used to setup the behavioral simulator (we adopted TOSSIM, as motivated in next section). All the inputs not explicitly considered by the adopted behavioral simulator (vi and vii in our case) are stored as user preferences.

Step 2 concerns the behavioral simulation of the WSN under study. Simulation results consists of a set of network parameters which are specialized for the WSN under study and which will be used by the external engine to feed SAN models and to handle changes. Examples of computed parameters are the per-node energy consumption profile, the link-by-link loss probability, and workload characteristics, such as the duty-cycle (e.g., the average percentage of useful work performed by each node when woken up), and the average size of sent/received packets.

The automatic generation of analytical models and related metrics (e.g. resiliency) is performed in the third step by the Model Generator component of the External Engine.

The Model Generator produces SAN models starting from a predefined library of model templates (stored in a knowledge base and developed una tantum by a domain expert). The number and type of models to be generated depends on user inputs. For instance, N node models will be generated for a WSN composed of N nodes. Each node model will be then specialized depending on the topology (which determines the neighbors of each node), on the hardware platform (which impact on the failure model of the sensor board), on the energy profile, and so on. Initial values for model parameters are configured starting from the results of the behavioral simulation (e.g., the link loss probability of each link is set depending on the values computed by the behavioral simulation) and from a set of pre-defined parameters (e.g., the failure rates of hardware components) which are provided una tantum by domain experts and stored in the knowledge base.

Step 4 concerns the simulation of generated analytical models by means of a SAN simulator engine, such as Mobius [20]. To deal with changes, models are programmed to notify changes to the External Engine (i.e., to the Changes Manager sub-component) and to react to consequent updates propagated by the engine. The external engine can be regarded as a supervision entity encapsulating all the coordination functions (among analytical models) that are generally difficult to express at the same level of abstraction of models (i.e., through the same modeling formalism). Hence, the engine is essential to keep models simple, general and re-usable. We can think at the external engine as an inter-model interface facility, which is used to generate and bootstrap the analytical failure model by parsing behavioral simulation logs (i.e., containing behavioral simulation results). However, it is not difficult to realize that some parameters are *dynamic* over time, i.e., their values need to be dynamically updated during the simulation, driven by changes. To exemplify let us consider a case for a node X. Let us assume that, due to a change in the workload behavior, the node increases the amount of work to be committed. The change is intercepted by the engine, which processes it and propagates an energy consumption update to node X model, which adapts itself to the change (the node starts to consume more energy). To complicate the picture, let us assume that, later in time, a neighboring node Y starts to send more packets to node X. This change (managed by the engine) results in a further increase of energy consumed by node X. As a result, at a given point in time, node X stops working, due to battery exhaustion. The failure is notified to the engine which re-computes the routing tree (according to the routing algorithm chosen by the user), and propagates a routing tree update to the models of all involved nodes.

From the above example, it is clear how the behavior of a single node impacts on the behavior of the overall network in an unmanageable number of ways. Conversely, different user choices (e.g., on the routing algorithm) influence the behavior of every single node. To master this complexity, we use parametric and re-usable models, which are autonomous and capable of adapting to changes induced by other models.

During the simulation, resiliency metrics are evaluated and results are finally delivered to the user in step 5.

It is worth noting that the proposed approach allows final users to work within their knowledge domain by interacting only with the behavioral WSN simulator and by providing his preferences as inputs. In other terms, developers interact with artifacts that are related to their domain, such as number and type of nodes, nodes placement, the program/algorithm running on each node, and so on. At the end, they get required resiliency figures which are the results of the SAN simulation.

Finally, it is important to note that the proposed approach is general enough to be extended to other classes of systems and to assess other classes of metrics.

## 4.2 The External Engine

The External Engine (EE) aims at i) automating the generation of SAN failure models with initial parameters values, and ii) at handling changes on behalf of the SAN models, computing dynamic parameters values, adapting the behavior of SAN models to network changes. The EE is composed of two main components, namely Model Generator and Changes Manager, whose details are provided in the following. The Changes Manager also includes a set of utility functions, designed to keep SAN models simple.

### 4.2.1 Model Generator

Objectives of the Model Generator component is to reduce the effort a hypothetic have to face in order to use the proposed approach. More in detail, the Model Generator component is designed to automate the creation of SAN models, metrics to be estimated, and experiments to be performed, according to preferences collected by the user interface during step 1. Hence, the Model Generator component can be seen as the "'interface"' of the generated model for parametric data: trough this component, the SAN model can be easily generated and fed with data gathered from the behavioral simulation and user preferences. It works in two phases: i) generation of the SAN models, and ii) specialization of the generated models. During i), information collected by the user interface (see Chapter 5 ) concerning adopted node and sensing platform, radio chip, batteries and workload, are exploited by the Model Generation component to select a specific set of templates from the model template library, as reported in Figure 4.1. Model templates are skeletons of SAN models, described by means of XML files. They are produced *una tantum* by a domain expert they are organized in a way so that all the details that strictly depend on the considered WSN (e.g. nodes interconnections, parameters), have to be generated once the behavioral simulation has terminated. For instance, if the model template T represents the network layer of the WSN, its structure has to be generated according to the topology considered during the behavioral simulation, i.e. modeling only the links that actually are present. Hence, if node X is a neighbor of node Y and Z, then the model of node X will present two distinct output branches toward the model of node Y and Z. This way, the Model Generator component greatly reducing the user modeling effort, giving flexibility to the approach, so that different WSN can be studied without requiring a new modeling phase.

During the phase ii), the Model Generator components populates the generated models parameters with values reflecting the considered WSN. Such values are gathered from behavioral simulation and from user preferences on metrics and experiments of interests. For instance, for each of the modeled links, probabilities of packet loss need to be specialized according to behavioral simulation results. Therefore, at the end of the specialization phase, a XML description of all of the generated models is produced. Then, a parser, translates the XML description of models to a format understandable by the tool chosen for the simulation of SAN models. In the rest of this thesis, we refer to the Mobius modeling environment [119] to simulate the SAN model[1]. In particular, we designed the XML parser to i) translate the XML files to the XML format used in Mobius to describe SAN model, metrics and experiments, ii) to exploit the facilities provided by Mobius for compiling XML descriptions of SAN models to C++ and executable files, and iii) to make it possible to visualize the generated model through the Mobius User Interface. Hence, at the end of the generation process of SAN models the user may decide to either visualize/modify/simulate the SAN model within the Mobius environment or to proceed independently, ignoring implementation details of the generated SAN model. Table 4.1 reports a subset of the facilities provided by the Model Generator Component.

[1]Other simulation frameworks may be taken into account by implementing different parsers, without interfering with the proposed approach.

Table 4.1: Facilities for the automated generation of the model.

| Facilities | Methods | Descriptions |
|---|---|---|
| Model Generation | `generateRoutingModel(topology)` | It generates the SAN model for the routing. |
| | `generateOverallModel(sensorBoard)` | It generates the XML description of the submodel that are joined to form the single node model. |
| | `generateComposedWSNModel(topology)` | It generates the XML description of the WSN model, joining together the whole set of single node models. The junction is made considering the actual network topology. |
| | `generateRewardVariables(rew1,rew2..)` | Depending on the user choice, it adds to the model, the XML description of the desired reward variables. |
| | `generateExperiments(Parameters)` | It generates the set of sensitivity studies to be performed on the WSN model depending on the user preferences and on behavioral parameters. |
| Parameter Specialiation | `getParent(nodeID,routingType)` | It returns the parent of nodeID on the current routing three. |
| | `AmIConnectedTo(nodeID,destID,routing Type)` | It returns true if nodeID is connected to destID in one hop when using the routing algorithm routingType |
| | `getLossData(nodeID)` | It returns a record with all the probability of packet loss on all the output link of node nodeID |
| | `getStaticParameters(nodeID)` | It returns a record containing all the stati parameters for node nodeID |

### 4.2.2 Changes Manager

This component is in charge of handling changes that occur during the simulation of the SAN model. Changes management is needed due to the occurrence of node failures or reconfiguration actions which may cause topology changes (e.g. the network is partitioned due to the failure of a node). Once a topology change has occurred, it is important to update all parameters coherently, so that the SAN model simulation can compute updated figures, such as power consumption (and hence nodes lifetime) due to the different traffic they have to forward toward the sink. Hence, after each change in the network topology, the routing three must be re-computed for each node. This is accomplished by running on the updated topology the routing algorithm(s), as indicated by the user at step 1 of the proposed approach. To this aim, the Changes Manager holds an abstract model of the network layer of the WSN: a weighted connectivity graph representing the topology of the considered WSN is built starting from the packet loss matrix (see Chapter 5) stored in the behavioral simulation logs (step 2 approach in Figure 4.1). A weight on an edge (i,j) in the graph represents the packet loss probability of the wireless link between nodes i and j. Then, the change manager build the routing three on the updated topology.

Once the routing three is updated, the Changes Manager allows the SAN model to update the energy consumption of the running nodes. This is accomplished by using parameters values about energy consumption, estimated from the behavioral simulation. For instance, a leaf node X (e.g. a node that is not forwarding packets to other nodes) after a node failure, may be selected by other nodes as a forwarder to the sink, due to the updated routing tree. Consequently, node X may experience a higher energy consumption rate that is dependent to all the packets that now it forwards to the sink.

The Changes Manager component is implemented as an external library linked to the SAN model. In particular, the Model Generator instruments the SAN model with explicit calls to the methods of the Changes Manager. For instance, methods are invoked during the simulation of the SAN model upon topology changes (e.g. failure of a node) with the objective of propagating network changes to the graph managed by the Changes Manager. Using an external library to handle such changes in the network preserves the generality of the failure model making it possible, for instance, to simulate different routing algorithms, without the need of different SAN models.

A subset of the methods provided by the Changes Manager is shown in Table 4.2. For instance, upon each failure event, the function `Failing` (Table 4.2) is invoked by the SAN model, in order to update the graph of the available nodes held by the Changes Manager component. Consequently, a running node may be notified of the change in the topology after the manifested failure. If the failed node was the only parent (i.e. the gateway currently used to reach the sink node) of a running node, then that node is temporary isolated from the sink. To signal this event, the function `propagate_isolation,`is invoked by the isolated node with the objective of propagating this information to all its children nodes (e.g. all the nodes sending packets through the isolated node). The involved nodes will then select alternative paths to the sink. Once the new routing tree is computed, and if alternative paths to the sink are found, the function `propagate_reconnect` (Table 4.2) is invoked by the SAN model in order to update the list of online nodes.

Table 4.2: A subset of the facilities provided by the Changes Manager

| **Methods** | **Descriptions** |
|---|---|
| `Failing(nodeID)` | It updates the topology and the routing three upon a node failure |
| `Recovering(nodeID)` | It updates the topology and the routing three upon a node recovery |
| `propagateIsolation(nodeID)` | It notifies asynchrounsly all nodes that nodeID is isolated from the sink |
| `propagateReconnect(nodeID)` | It notifies asynchrounsly all nodes that nodeID is connected to the sink |
| `ComputeRoutingTable(nodeID)` | It computes the routing table of nodeID. It is invoked by the former methods |

### Accounting for Routing Behavior

The `computeRoutingTable(nodeID)` method is invoked to request an update of the routing table of a node. This is accomplished by implementing this method consistently with the selected routing algorithm. Different sub-functions are included in `computeRoutingTable,`making it possible to reproduce the behavior of different routing algorithms without changing the SAN model.

More specifically, the Changes Manager component takes into account only the characteristics of routing algorithms that have a direct impact on the resiliency of the considered WSN. The first characteristic we consider is the selection function of forwarding nodes. This is a basic function of each routing algorithm: each sensor node has to select one (or more) nodes, among its neighboring nodes, to be used as a forwarder to the sink node. According to this characteristic, routing algorithms can be classified as random or based on a selection function. The former simply forwards they packets to all their neighbors (**flooding**), to a random subset of them (**gossiping**), or to a random neighbor (**random path routing**) with the assumption that data will eventually reach the sink. The latter selects the neighbor on the path that minimizes/maximize a cost function (e.g. quality of traversed links, overall energy to reach the sink, QoS-based routing ). For instance, concerning a **multi-hop** routing algorithm, a function is computed maximizing the quality of the overall path to be traversed to reach the sink from each node.

The last characteristic we consider is the use of **acknowledgments** and **data aggregation** techniques. The first are used to signal to the sending node that the recipient node received the packet, hence providing a feed-back for reliable transmissions. The latter consists in waiting a specific interval of time before forwarding a packet, so to combine different packets coming from different sources en route, hence reducing the number of transmissions and increasing network lifetime. As will be later described in Chapter 6, we take into account data aggregation and acknowledgments in the routing SAN model.

It is worth noting that the approach pursued in the design of the change manager component enables the simulation of a large number of routing algorithms by replacing graph weights, node selection function or both, without any change to the structure of the SAN model. For instance, it is possible to simulate energy aware routing algorithms by replacing multi-hop graph weights with figures related to the remaining energy available on nodes, and by implementing a routing function finding paths from all nodes to the sink maximizing WSN lifetime.

### 4.2.3 Utility functions

The EE includes a set of utility functions used to simplify the structure of SAN models. For instance, a set of functions is included to manage packets, including their structure (e.g., the sender and receiver, the time to live field, the signature). This is useful to evaluate the number of useful (i.e., not duplicated) packets reaching the sink. Another set of functions is used to handle acknowledgment packets (only when needed, e.g., required for a given routing algorithms, such as **reliable multi-hop**). In particular, the EE implements a linked list for each node in the network for storing the packets sent by a node and waiting for an acknowledgement. Packets in the list can be thus re-sent if an acknowledgement is not received within a given timeout.

## 4.3 User interface

The use of an interface makes possible to decrease the semantic gap between the WSN and the model used to assess the resilience, allowing potential users to interacts with artifacts that are closer to their domain, such as behavioral simulators.

The user interface is used to collect static information which is not computed by the behavioral simulator, such as node position, the adopted hardware platform (sensing hardware and micro-controller), battery technology (e.g. lithium ions or nickel metal hydrate) and the software to be run on nodes. In addition, the interface allows to explicitly specify intervals of variation for some parameters (e.g., intervals for the workload duty cycle, the packet size, the number and type of routing algorithms) in order to ease the conduction of sensitivity analysis studies.

# Chapter 5

# Behavioral Models

*In the presented approoach, a behavioral model is exploited to configure the WSNs in terms of hardware platform, topology, routing and MAC protocols, and to study the nominal behavior of the software, included the OS, and the power consumption of the nodes. Evaluations performed with the behavioral models are used to gather values for failure model parameters of the WSN under study, such as the packet forwarding rate of each node. This Chapter presents the considered behavioral model and how it is realized in the TOSSIM simulator.*

## 5.1 Requirements and Rationale

The objective of the behavioral simulation is to evaluate the parameters needed by the external engine to specialize the SAN model, during steps 3 and 4 of the approach (see Figure 4.1 in Chapter 4). Three classes of parameters are evaluated: i) the energy consumption profile of each node, ii) the loss probability of each link, and iii) the workload characteristics of each node. These parameters are crucial to initialize and drive the SAN model simulation. To exemplify, when a packet transmission event is triggered for node X during the SAN simulation, the SAN model has to drain an amount of energy from the battery of node X, which depends on the energy consumption of the radio board and on the packet size. Then, the SAN model will either simulate a packet delivery or a packet loss depending on the link loss

Table 5.1: Behavioral model parameters needed by the SAN models

| **Parameters** | **Platform** | **Network** | **Application specific** |
|---|---|---|---|
| **Component/ function** | | | |
| **Power supply** | Battery technology | N/A | Application duty-cycle |
| | Battery depeltion profile | | |
| | Initial battery charge | | |
| **Communication** | Radio communication technology | Packet loss rate per link | Packet generation rate |
| | | Topology | Packet size |
| | | | Routing algorithm |
| **Sensorboard(s)** | Sensing hardware technology | N/A | N/A |
| | Failure rates | | |

probability. On the other hand, the number of packets transmission events triggered for each node varies over time, and depends on the current configuration of the WSN (e.g., number of failed and isolated nodes, network topology, routing tree) which is updated by the Change Manager during the simulation.

Such parameters can be static or dynamic, and are summarized in Table 5.1. Static parameters are related to aspects that do not mutate during the simulation of the SAN model, such as the hardware platform, the battery technology, and the radio communication technology. Dynamic parameters depend on the current configuration of the WSN (e.g. number of failed nodes, number of isolated nodes, transmission rate of each link) and need to be re-computed in the face of every change triggered during SAN model simulation. The change manager component (step 4 Figure 4.1 ) is in charge of computing updated values for dynamic SAN model parameters, avoiding a behavioral simulation upon each manifested change.

The evaluation of model parameters is based on a behavioral simulator able to

i) simulate or emulate commercial hardware (e.g. commercial sensor boards, micro-controllers), ii) to mimic the behavior of running applications with a good level of details, and iii) be extended with custom facilities in order to be integrated within the proposed approach.

As reported in Chapter 2 (see Section 2.5), a number of the most cited/used environments for the simulation of WSNs have been analyzed. The performed analysis shows that the TOSSIM simulator [105] provides a convenient trade-off between the level of details and extensibility. TOSSIM provides a set of facilities that make it possible to emulate real node hardware and to simulate WSN applications (programmed in NesC [106] and running on TinyOS) with a realistic level of details. We also extended TOSSIM with facilities to collect user preferences on the considered WSN by means of a customized user interface, used in step 2 of the approach depicted in Figure 4.1.

More details on parameters evaluation and on the user interface are provided in the following.

## 5.2 Energy Consumption Model

We consider the overall energy consumption of a generic node as the sum of three main components, namely: energy needed for sensing, for computing and for sending/receiving packets. More specifically, the energy needed for sensing is dependent on the adopted sensing platform. Information about energy needs of sensors is gathered from data-sheets. The energy needed for computing activity is dependent on the running software and on the workload duty-cycle. Finally, the aliquot of energy spent for managing packets depends on i) type of radio chip used and programmed transmissive level of power, ii) length of packets in bytes, iii) packet loss rate, iv) topology (e.g. per node incoming and outgoing links).

We instrumented TOSSIM simulation engine in order to profile the energy consumption of node subsystems. In particular, we hijack simulation events from the TOSSIM simulation engine that are responsible for the energy consumption, considering i) cpu power transition (sleep to active and vice-versa), ii) radio events (reception and sending of packets), iii) application events, and iv) hardware events (e.g. sensing hardware events). Each time one of the mentioned events take place, an overall energy consumption figure is computed as the sum of all the energy requests taking place in that instant of time. In turn, such figures are periodically averaged on a specific interval of time, obtaining, for each considered subsystem of the node an energy consumption rate. This computation is performed during the whole behavioral simulation that is forced to terminate when the computed figures reach an interval of confidence of 95%. At the end of the behavioral simulation, values for energy consumption parameters are stored to a set of simulation logs that are later used by the model generator and Changes Manager component shown in Figure 4.1.

## 5.3 Radio Model

We collect this information in a matrix where the element $a_{ij}$ is the packet loss probability on the link between node i and node j. Such a matrix is stored in the simulation logs available to the model generator component, depicted in Figure 4.1. In order to evaluate packet loss figures of WSN links, it is necessary to adopt an accurate mathematical model for radio propagation. We extended TOSSIM simulator by integrating a well know model for the estimation of the node-by-node Received Signal Strength Indicator (RSSI) [120] in its radio modeling facilities. More in detail, we use information on nodes mutual distances, specified in the user interface, to evaluate the RSSI from a node to all the others by means of the following equation[1].

$$Rx_{pow} = Tx_{pow} + Tx_{gain} - Tx_{loss} - FSL - X_\sigma + Rx_{gain} - Rx_{loss} \tag{5.1}$$

where:

$$FSL = -27.55_{dbm} + 20log_{10}(Freq_{MHZ}) + n * 20log_{10}(d_m) \tag{5.2}$$

- FSL is the free space loss factor, expressed in dBm (decibel milliwatt) .
- $d_m$ = is the transmissive distance between the source and the destination expressed in meters
- $Tx_{pow}$, $Tx_{gain}$ $Tx_{loss}$ are the transmissive power, antennas gain and transmission losses factors, respectively, all expressed in dBm;
- $Rx_{pow}$, $Rx_{gain}$ $Rx_{loss}$ are the reception power, antennas gain and reception losses factors, respectively, all expressed in dBm;
- $X_\sigma$ is a random variable modeling environmental reverb, e.g. multipath, fading;

[1]Further details on the propagation model may be found in [121]

- $n$ is the path loss exponent depending on the environment and on the node deployment.

Values for $Tx_{pow}$, $Tx_{gain}$, $Tx_{loss}$, $Rx_{gain}$, $Rx_{loss}$ are gathered from the data-sheet of the radio chip selected by the user for the simulation. $X_\sigma$ and $n$ are selected depending on the user preferences on the nodes deployment (e.g. indoor or outdoor).

After computing Equation 5.1 for each node, we figure out about the lower bound of the RSSI after that there is a bit inversion in the transmission. Finally depending on the packet length and adopted MAC protocols[2], we achieve the relationship between bit error rate and packet loss.

## 5.4 Workload model

The periodic workload models a typical monitoring sensor network in which readings are generated at fixed time intervals. Deployments exhibiting this traffic pattern are quite common in practice [122, 123, 12]. In this workload, each sensor sources traffic at some offered load, and helps to forward other sensors' traffic to a sink. Depending on the amount of data gathered from sensors, on the number of computed samples, and on the local performed computation, one or more packets of a given size are produced periodically. Data may be also stored in the local non volatile memory for being sent later. Hence, we opted to model such a type of workload by considering the following parameters: i) application duty-cycle, i.e., the time interval of the application, ii) the average size of generated packets, iii) the total time

[2]Information collected by the user interface.

spent in local computation, iv) the application packet generation rate, v) the time spent in idle state by the application, e.g. while waiting for data being sampled by sensors. Similarly to energy consumption evaluation, these parameters are computed by intercepting TOSSIM events responsible of i) cpu power transition, ii) radio events, and iii) application events [16].

# Chapter 6

# WSN Failure Model

*In this chapter 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. We adopt a two-steps modeling approach. First, we perform a classification of failures as they have been observed on actual WSNs and as they have been indicated in the existing literature. In particular, failures are classified according to Failure Mode and Effect Analysis (FMEA), i.e., from functional components, down to failure modes and possible causes, as better detailed in section 6.1. Second, starting from the conducted FMEA, we infer a detailed failure model, according to a bottom-up approach: a detailed SAN model is built for each of the classified failure mode, then the models are combined to form the failure model of a single sensor node, and finally the failure model of the overall network).*

## 6.1 Failures and Failure Modes Assumptions

In this section the results of the FMEA is presented. The most frequent failure occurrences have been derived from past experiences on real testbed prototypes and from the existing literature, trying to relate failure occurrences with potential causes (faults). The use of a FMEA makes it possible to evidence the Critical Item List (CIL) of the network. Each CIL element represents a critical component/function: a failure of one of these components hardly impacts on system dependability. Thus, CIL information can be used i) to drive the implementation of the failure model (FM), and ii) to provide failure mode assumptions that reflects real world.

The following high-level assumptions have been made when performing the FMEA and when building the model:

1. Each node is composed by a processor board, a sensor board with one or more sensors (e.g. temperature, humidity etc.), a radio board, and a set of batteries.
2. Initially all the nodes have the same capabilities and characteristics and are stationary.
3. Sink node failures are not considered. This is reasonable since a sink is typically realized with more reliable hardware/software equipment (e.g., laptop or a linux embedded device).

In this thesis, both single wireless node and whole network failure modes are encompassed. Failures due to physical damage of nodes (e.g. physical crashes due to accidents or very adverse weather conditions), malicious activities (e.g. manual, and unexpected, node withdrawal or substitution), and security threats are excluded.

FMEA results are summarized in Table 6.1: six components/functions have been identified for the node: the sensor board, the power supply unit, the radio board, the communication function, the CPU, and the operating system. For each component/function of a node or of the overall network, failure modes, potential effects and possible causes are reported.

### 6.1.1 Node failures.

From the prospective the mission of the WSN, a node is failed when i) it is no longer able to deliver its measurements to the sink, and ii) it is not longer able to provide

Table 6.1: FMEA results.

| Component / Function | Sub-component / Sub-function | Potential Failure Mode | Potential Effects of Failure | Potential Causes of Failure |
|---|---|---|---|---|
| **Node** | **Sensor board** | Stuck at Zero | The device is out-of-order: it does not deliver any output to inputs | Sensing hardware |
| | | Null Reading | The device delivers null output values | Sensing hardware |
| | | Out of Scale Reading | The device delivers no meaningful values | Sensing hardware |
| | | Stuck at N | The device provides the same value to different inputs | Sensing hardware |
| | **Power Supply** | Stuck at Zero | The device is out-of-order: it does not deliver any output to inputs | Natural energy exhaustion |
| | | Reset | The node resets itself to its initial conditions. | Anomalous current request that cannot be supplied by batteries |
| | **Routing** | Packet Loss | The radio packet is not delivered | Packet Corruption |
| | | | | Buffer Overrun |
| | **Communication** | Isolation | The node is not longer connected to the sink node | failure of all forwarding nodes |
| | **CPU** | Stuck at Zero | The device is out-of-order: it does not deliver any output to inputs | micro-controller |
| | **OS** | Software Hang | The device is powered on, but not able to deliver any output | operating system's corrupted state |
| **Network** | | Coverage | The monitored area is covered with a number of nodes below a threshold | The number of isolated nodes is greater than a certain treshold |
| | | | | The number of failed nodes is more than a given threshold |
| | | Data Delivey Failures | The network is not able to deliver the required amount of measurements | The number of isolated nodes is greater than a certain threshold |
| | | | | The number of failed nodes is more than a given threshold |
| | | Net Partition | The network results partioned into more than one subnets | Failure of all the nodes belonging to a "cut set" of the network |

meaningful measurements. This can be due the malfunction of one of the components of the node, as detailed in the following.

*Sensor Board*: we assume the sensor board can fail according to four failure modes: *stuck-at-zero*, *null reading*, *out-of-scale reading*, and *stuck-at-N*. A stuck-at-zero of the sensor board produces the effect of a out-of-order device, which does not deliver any outputs to external inputs. Potential causes lay into faults of the sensing hardware (e.g., as can be observed in [124], the humidity sensor produces a short circuit, causing a high current drain which turns off the overall node). Potential causes lay into faults of the sensing hardware (e.g., as can be observed in [124], the humidity sensor produces a short circuit, causing a high current drain which turns off the overall node).

Null readings cause the sensor to deliver null output values, for a certain interval of time. This may be caused by temporary short circuits that also cause the node to drain excessive power from batteries, hence shortening the overall lifetime of the node [124]. Out-of-scale readings and stuck-at-N cause the sensor board to respectively provide no meaningful outputs and frozen readings, for a certain interval of time. These failures may be due to faults of the analog-to-digital converter or due to error in the sensing device firmware.

*Power Supply*: the power supply component may exhibit *stuck-at-zero* as well as *reset* failure modes (i.e., the node shutdowns and restarts itself). The former is due to battery energy exhaustion. The latter can be caused by anomalous power requests that cannot be supplied by batteries, e.g. the residual charge is not sufficient to provide the required amount of power.

*Radio Board*: it can exhibit *packet losses*, i.e., the radio packet is not delivered to its intended destination for instance due to packet corruption.

*Communication*: the communication function can be compromised by so-called *isolation* failures: a well behaving node X, which does not fail itself, can manifest an isolation failure when it is no longer connected to the sink node. This is due to the failure of all the nodes which act as forwarders for the node X.

It is important to note that the characteristics of all considered node failures depend on the network configuration. For instance, the isolation failure strictly depends on the actual network topology, whereas sensor board failure rates depend on the adopted hardware, and battery failures on the running software and routing algorithm.

### 6.1.2 Network Failures.

At the network level, we encompass two failure modes, which are consistent to the mission of the WSN and to defined requirements: *coverage failures*, *data delivery failure* and *disconnection failure*. Coverage failures occur when the number of nodes in a specific area of the network drops below a given threshold. Data delivery failures occur when the network is not able to deliver the required amount of measurements to the sink. Disconnection failures occur when all the nodes belonging to a cut set of the network fail. As a result, the network is partitioned into two or more isolated sub-networks. In both cases, the number of failures that can be tolerated strictly depend on the application requirements.

## 6.2 WSN Failure Model

Starting from the classification obtained with the FMEA, we define a the WSN failure model which reproduces the failing behavior of WSNs and sensor nodes. We build the model according to a bottom-up approach: for each identified component/function of a node, we defined a detailed failure model (FM), adopting the Stochastic Activity Networks (SANs) formalism. The failure model of a single node is achieved as interconnection of atomic models. The interconnection of the failure models of

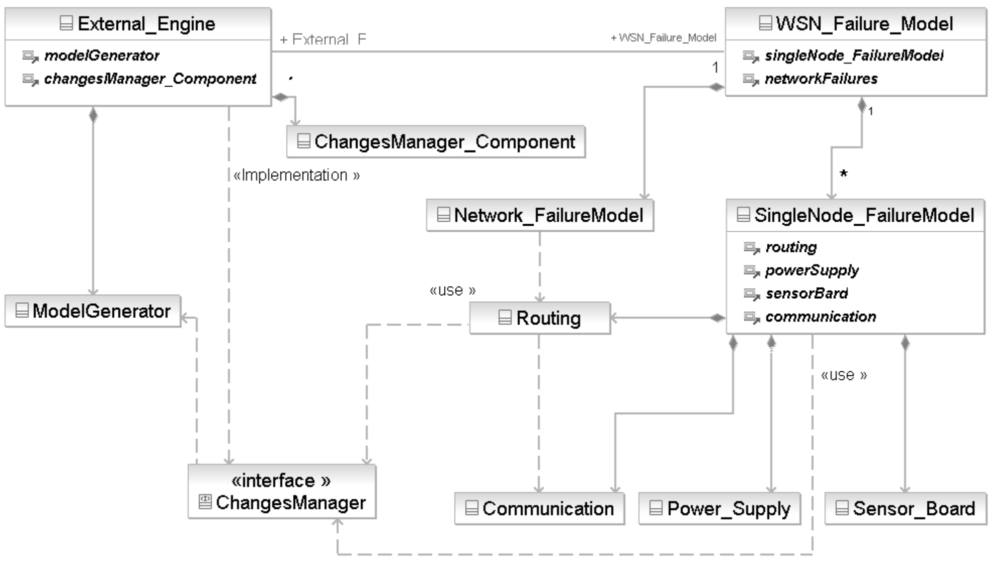


Figure 6.1: Overall WSN failure model

all the nodes determine in turn the failure model of the overall WSN, which also embodies network failures (e.g., coverage and net partition), and network dynamics (e.g., current network topology, traffic and computational load). The model is then solved by means of simulation, using the MOBIUS tool.

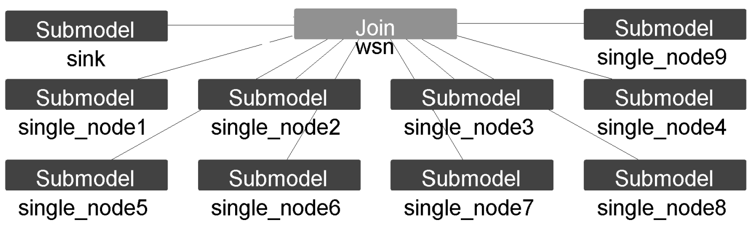
The structure of the proposed model is shown in Figure 6.1 in terms of a UML class diagram. Each class in the lower part of the diagram corresponds to the SAN failure model of the component/function defined by the FMEA. We can thus find the sensor board, the power supply, the routing and the communication function. The *sensor board* FM encompasses all the hardware-related failures of the sensor board. The *power supply* FM reproduces the battery discharge process, taking into account stuck-at-zero and reset. The *routing* FM reproduces the transmission/reception/forwarding activity of the node, reproducing the behavior of a specific routing algorithm (selected by the user) and the effects of packet losses. The *communication* FM takes into the account isolation failures.

Note that we opted to not consider CPU and OS failures. CPU failures are neglected since the mean time to failure (MTTF) of micro-controllers is greater by orders of magnitude than the MTTF of other components[1]. As for the OS, we assume the bug incidence to be usually low with respect to other failure dynamics, due to the extreme simplicity and reduced size of node operating systems [105].

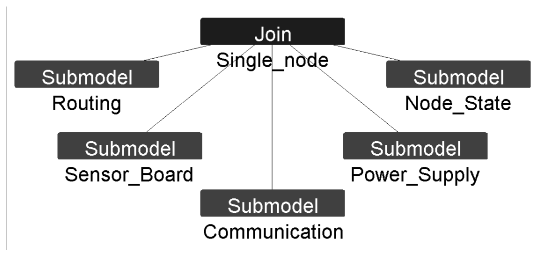
As shown in Figure 6.1, the failure model of a single node is composed of the above mentioned failure models. The network level failure class embodies the failure model of the network component, i.e., coverage, data delivery and disconnection failures, depending on the information provided by the routing and communication failure model. A coverage failure is indeed strictly related to the communication state of all the nodes. The failure model of the overall WSN is then obtained as the composition of $n$ single node failure models and of the network level failure model, where $n$ is the total number of nodes.

It is wort noting that the *External Engine* interacts with the whole WSN model by means of the `ChangesManager` interface, which is used by single node models as well as by the routing models. As will discussed in Section 6.3.2, the routing model exploit

[1]According to manufacturers, typical order of magnitude for a micro-controller MTTF is around ten thousand hours.



(a)



(b)

Figure 6.2: (a) SAN model of the overall network; (b) SAN model of a single node.

the facilities of the Changes Manager and of the External Engine to reproduce the behavior of the routing protocol being simulated. The `ChangesManager` interface is also dependent on the Model Generator component, since it is generated jointly with the SAN Failure model, depending on the specific modeling framework being used (See Chapter 4). Following sections will provide example on the interactions between the model and the External Engine.

Figure 6.2.(a) and (b) shows the compositional schema of the WSN and single node failure mode generated for a hypothetical WSN composed of 9 sensor nodes and 1 sink node. Both schemes are automatically generated at the end of step 2 (see section 4.1) as a set of MOBIUS XML descriptors. The number and, as will be detailed in Section 6.3.2, the interconnections of single node models are generated according to the topology provided by the user for the behavioral simulation.Single node models are interconnected to each other (via the join operator - realizing the WSN model - and the use of places which are shared[2] between linked models) according to the topology of the WSN to simulate.

The models are composed accordingly to the items in figure 6.1. In addition, a node coordinator model, called *nodeState* (see Figure 6.2.(b)) is added to coordinate single node sub-models.

The proposed model has been validated adopting a trace validation approach [125], i.e., the behavior of different types of specific entities in the model is traced to determine if the model logic is correct. To accomplish this task, we used the Traviando framework [126], a software tool capable of analyzing and visualizing simulation traces. In particular, by means of Traviando we validated the causal order of events, assuring that the model was behaving like expected.

### 6.2.1 The *Network* Failure Model

The *Network* FM, shown in Figure 6.3 model is in charge modeling coverage, data delivery and disconnection failures. To this aim, the *Network Failure* model holds information on the number of working nodes in the WSN by means of the shared place *running_nodes* Each time a failure of a node takes place, a token is moved from the place *running_ nodes* to the place *number_of_ failed_nodes.* Upon this transition, the input gate *areaCoverage* invokes a method of the Changes Manager (see section 4.2.2) to test if the density of running nodes is still able to produce an amount

[2]In the SAN terminology, a model can provide an "interface" to other models by sharing places with other sub-models by means of the join operator.

of data compatible with the WSN mission. For instance, it may happen that due to disconnection failures, a subset of the WSN is isolated from the sink. Hence, at the sink, the area monitored by the disconnected set of nodes is not producing measurements. In this case, despite the remaining nodes are working correctly, a data delivery failure manifests due to a coverage failure. The Changes Manager allow also to specify for each node, the area where it is deployed, hence enabling the evaluation of the density of nodes, area by area.

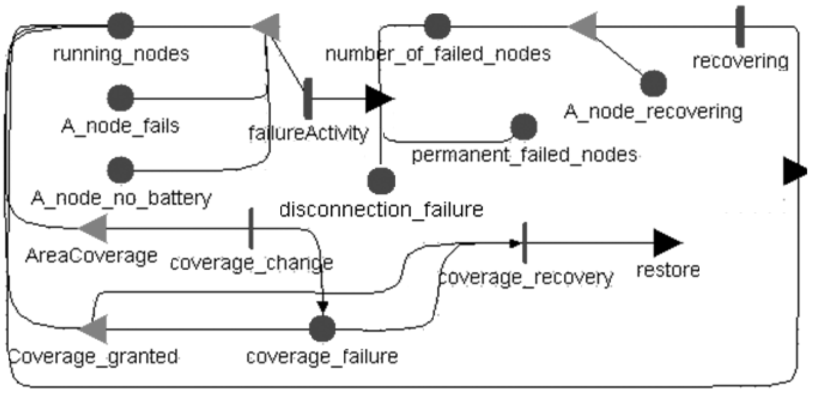


Figure 6.3: The *network* SAN mode

Disconnection failures are modeled by the *Network Failure* model upon each manifesting failure, by means of the methods provided by the change manager. Such methods permits to evaluate if, eliminating the failed node by the current topology, other nodes than it result isolated. In this case, a Disconnection failure is triggered and a token is moved in the place *disconnection_failure.*

Upon a node recovery (signaled by the shared place *A_node_recovering*) a place is moved from *number_of_failed_nodes* to the place *running_nodes* by means of the action *recovering*. In this case, if the former failure caused a data delivery failure, after a recovery the input gate *Coverage_granted* evaluate if the new configuration of the network is able to fulfill application requirements on the area coverage and on the data to produce. If this condition is met, then a token is moved out from the place *coverage_failure* by enabling the action *coverage_recovery*.

## 6.3 Single Node Model

The single node model, which Mobius scheme is shown in Figure 6.2.(b) is composed by a failure model (referred as *failure_model_node*), a routing model and a node state model (referred as *nodeState*). Consistently with the assumptions provided in Section 3.1, the failure model is composed by i) the *sensor board* failure model, which encompasses all the hardware-related failures of the sensor board (e.g., stuck-at-zero, null reading, etc.), ii) the *power supply* failure model, which reproduces the battery discharge process, taking into account stuck-at-zero and reset, and iii) the *communication* failure model, modeling isolation failures. This last model is of particular importance to measure the connection resiliency.

The routing model describes the packet generation and delivery process of the node, according to the profiled workload and the chosen routing algorithm respectively, taking into account packet loss. Modeling the packet generation and delivery process is fundamental to measure the data delivery resiliency as a function of both simulated failures and simulated time.

The failures triggered by failure models are used to update the overall state of the single node in the *nodeState* model (i.e., the node is up, it is temporary failed, it is permanently failed, or it is isolated). It is worth noting that all node sub-models are correlated to each other, since a failure triggered in a sub-model can in turn affect another sub-model. For instance, a null reading of a sensor (sensor board failure model) can cause an excessive drain of power, hence shortening the life of the sensor (power supply failure model).

In the MOBIUS tool, SAN models and their compositions are represented by means of XML descriptors. The Model Generator component of the External Engine generates the descriptors and populate them with proper parameters values (e.g., link-by-link loss rate) according both to the results of the behavioral simulation and to user preferences.

### 6.3.1 The Node State Model

*Node_state* model accomplish to two main tasks:

1. "start" (*boot*) of the whole single node model;
2. collection of detailed information from every sub-model, modeling the global state of the node;

The objective of the first task is to initialize the data structure used by the single node model and containing all the information achieved from the behavioral simulator by means of the External Engine (see Chapter 5). The timed action *gettingParameters* is used for this task and a Normal timing distribution is used to avoid false synchronization between the boot of different nodes. The place *running* enable the single node model. All the events (e.g. battery exhaustion) which makes the token be moved out from the place *running* cause the total stop of the node model.

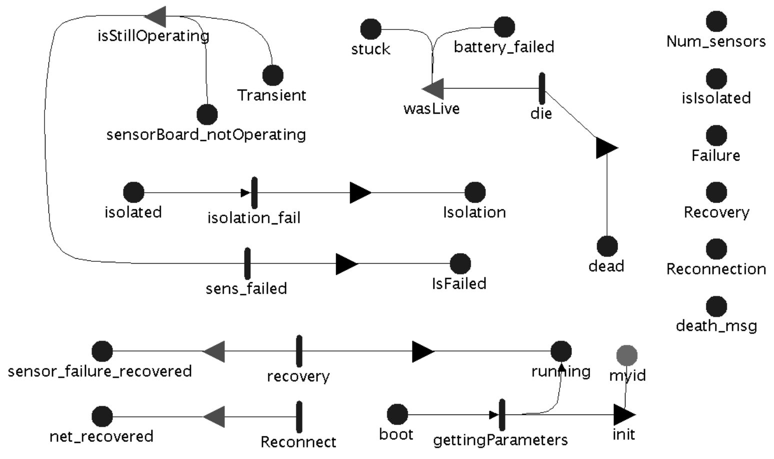


Figure 6.4: sub-model node_state

The objective of the second task is to collect and propagate information on the state of the node sub-systems. The model *node state* depends on and manages all the other node sub-models. It 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. To this aim,*node state* model interface, shown in Table. 6.2, is exploited by other models upon every failure or recovery event. The places used as input parameters of the interface are in charge of collecting the state of node sub-models (e.g. a failure triggered by a sub-model) in order to update the node state. The output parameters of the interface are used from *node_state* to rise events on the sub-models, such as to align the mark of the *running* place shared with all other models, e.g., after a failure.

The places *Num_sensors* and *num_of_motes* represent respectively the number of the sensors on the node still operating.

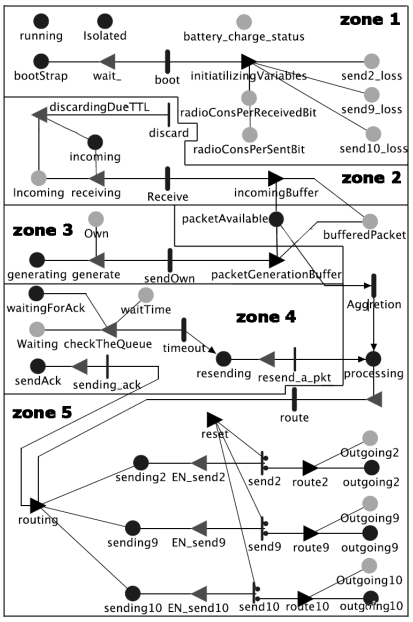
Table 6.2: *NodeState* model interface

| Places | IN | OUT | Description |
|---|---|---|---|
| `Failure` | | ✓ | Propagates node failures to netManager model |
| `Recovery` | | ✓ | Propagates Recoveries to netManager model |
| `Reconnection` | | ✓ | Propagates Reconnections to netManager model |
| `Isolation` | | ✓ | Propagates Isolation failures to netManager model |
| `running` | | ✓ | Enable the whole model – shared with all submodels |
| `myid` | | ✓ | ID of the node - shared with all submodels |
| `sensorFailureRecovered` | ✓ | | Triggered by the sensor board failure model |
| `netRecovered` | ✓ | | Triggered by the communication failure model |
| `isIsolated` | ✓ | | Triggered by the communication failure model |
| `Transient` | ✓ | | Triggered by the sensor boardfailure model |
| `SensorBoard_not_operating` | ✓ | | Triggered by the communication failure model |
| `stuck` | ✓ | | Triggered by the communication failure model |
| `Battery failure` | ✓ | | Triggered by the communication failure model |
| `Num_sensors` | ✓ | | Updated by the sensor board failure model |

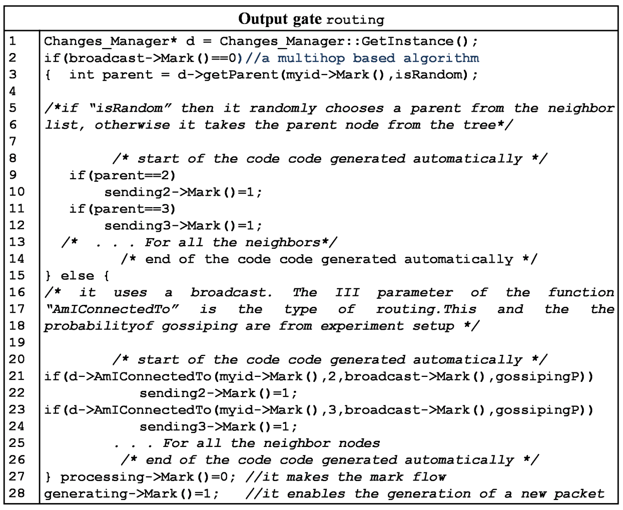
### 6.3.2 Routing Failure Model

Figure 6.5 reports the SAN model in charge of modeling the packet generation and delivery process of a single node. The model reported in the Figure 6.5 is related to an exemplary node X with three neighbors (nodes 2, 9, and 10 in the example, as reported in Figure 6.6), and it is divided in 5 zones, each of them responsible for a specific task.

Zone 1 is in charge managing packet loss ratio parameters for each link between node X and its neighbors (places *sendY_loss* in Figure 6.5, zone 1, where Y is the ID of the neighboring node). It also manage radio energy consumption parameters for each sent or received bit (places *radioConsPerReceivedBit*, *radioConsPerSentBit*), and the initial energy of batteries (place *battery_charge_status*, shared with the *power supply* failure model). Specific values for the parameters are gathered from the results of



(a)



**Output gate** `routing`

```
Changes_Manager* d = Changes_Manager::GetInstance();
if(broadcast->Mark()==0)//a multihop based algorithm
{  int parent = d->getParent(myid->Mark(),isRandom);

/*if "isRandom" then it randomly chooses a parent from the neighbor
list, otherwise it takes the parent node from the tree*/

        /* start of the code code generated automatically */
   if(parent==2)
       sending2->Mark()=1;
   if(parent==3)
       sending3->Mark()=1;
  /*  . . . For all the neighbors*/
         /* end of the code code generated automatically */
} else {
/*  it  uses  a  broadcast.  The  III  parameter  of  the  function
"AmIConnectedTo"  is  the  type  of  routing.This  and  the  the
probabilityof gossiping are from experiment setup */

        /* start of the code code generated automatically */
if(d->AmIConnectedTo(myid->Mark(),2,broadcast->Mark(),gossipingP))
           sending2->Mark()=1;
if(d->AmIConnectedTo(myid->Mark(),3,broadcast->Mark(),gossipingP))
           sending3->Mark()=1;
        . . . For all the neighbor nodes
         /* end of the code code generated automatically */
} processing->Mark()=0; //it makes the mark flow
generating->Mark()=1;   //it enables the generation of a new packet
```

(b)

Figure 6.5: SAN of the Routing model.

the behavioral simulation.

Zone 2 models the reception of packets from other nodes. A packet is received when a token is placed in the place *Incoming*. The received packet can be discarded if its TTL is zero, by means of the action *discard*, otherwise the packet is made available and processed (places *packetAvaliable* and *processing* in Zone 5). In both the cases, when a packet is received, a quantity of energy is subtracted from the *battery_charge_status place*, depending on the adopted radio chip (in terms of energy per received bit) and on the packet size.

Items included in Zone 3 model the packets generation process. The timed action *sendOwn*, models the duty-cycle profiled during the behavioral simulation. Once that the *sendOwn* activity fires, a packet of given length is generated through the utility

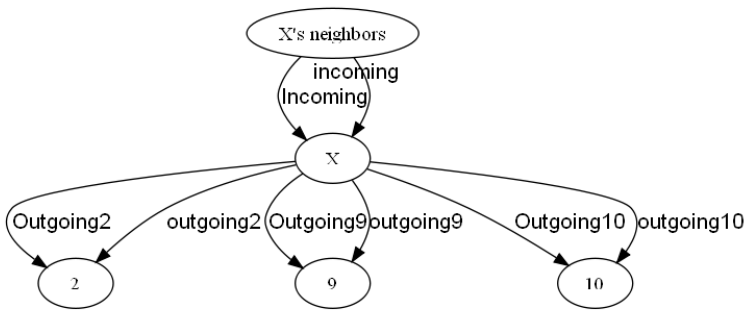


Figure 6.6: Topology considered in Figure 6.5. Labels of edges represent the places used in the model.

functions of the Changes Manager (see Chapter 5) and stored in the place *Own*. At the same time, a token flows from the place *generating* to the places *packetAvailable* and *processing*.

Zone 4 is in charge of reproducing the acknowledgment mechanism, if employed in the simulated routing algorithm. This includes the retransmission of missing packets. The modeling is simplified by the use of the Changes Manager, which implements a linked list for each node, storing the packets waiting for an acknowledgment. Figure 6.7 shows the portion of code of the Routing output gate responsible for managing acknowledgments, acting on the list, providing an example of interaction with the Changes Manager component.

All the items in Zone 5 of Figure 6.5 are used to model the routing of received or generated packets. The output gate *routing* is in charge of modeling the selection of the destination node for the packet stored in the *processing* place. Figure 6.5.(b) reports the code used for this task, providing a further example of the interaction between the SAN model and the Changes Manager. Depending on the routing algorithm being simulated, the output gate *routing* invokes a specific parent selection

**Output gate** `routing`

```
Changes_Manager* d = Changes_Manager::GetInstance();
if(Sending_an_ack->Mark()==0 &&isACK->Mark()!=0)
// the packet being sent needs an acknowledgement
{
/*It creates a new object of type packet and initializes its fields
with those of the packet being sent
Then it enqueues the packet in the "waiting for ack" queue. If the
queue is empty then it programs the timeout */
   if(waitingForAck->Mark()==0)
   {
         waitTime->Mark()=timeOut;       //it sets The timer
        //it updates the number of packets waiting for ack
         waitingForAck->Mark()=d->getSizeOfTheQueue(myid->Mark());
    }
}
. . . . . . . .
/*This is the code responsible for routing the acks when receiving
a packet that requires an ack*/
if(sendAck->Mark()!=0) {//it means that an ack is needed
   Sending_an_ack->Mark()=1;
/* if an ack is needed than the node will send an ack to the
sender. to keep track of the sent packet, i.e. NODE "myid->Mark()"
is sending an ACK to "receivedPacket->getSenderID()" for packet
receivedPacket->getSignature() */
/* start of the code code generated automatically */
    if(sender==0)
         sending0->Mark()=1;
    if(sender==2)
         sending2->Mark()=1;
. . . Forall the neighbornodes
/* end of the code code generated automatically */
      sendAck->Mark()=0;
}
```

Figure 6.7: Output gate *routing*: extract of the code responsible for managing acknowledgements

function of the Changes Manager to select the destination node. If the algorithm employs a routing tree (e.g., multi-hop routing), the gate will select only one parent, e.g., the node 10, from its neighbor list by means of the method `getParent()` of the Changes Manager. Then, a token is inserted in the *sending10* place, reproducing the sending of the packet toward node 10. On the other hand, if a flooding (or gossiping) algorithm is being simulated, a broadcast is simulated and a token is inserted in all (or a subset) of the places of the neighbor nodes. The IDs of the neighboring nodes are gathered by the method `AmIconnectedTo()`. The code responsible for the forwarding is generated automatically by the Model Generator component of the External Engine, as highlighted in Figure 6.5.(b).

When a token reaches one of the *sendingY* places (where Y represents the ID of the destination node), the packet is forwarded to the specific node by means of the corresponding *sendY* activity and the *routeY* output gate. All *sendY* activities are characterized by *cases*, used to model packet losses. A packet is lost or forwarded, depending on the loss rates contained in the *sendY_loss* places (Zone 1). When a packet is sent, a specific quantity of energy is drawn from the batteries, depending on i) the current settings of the radio hardware of the node (e.g. transmissive power, gathered from behavioral simulation ), and ii) the packet size. After updating the remaining energy (place *battery_charge_status*) the packet is transferred to the corresponding place *OutgoingY*. A token is placed also into the shared place *outgoingY* to signal to the destination node that a new packet is available. Such shared places represent the model interface towards the models of neighboring nodes. In particular, the *outgoingY* place is shared with the *incoming* place of the Routing model of node Y.

It is worth noting that the routing model represents a good example of model template. It is clear that the number and the names of all the items present in Zones 1 and 5 of Figure 6.5 strictly depend on the topology of the network. Hence they need to be generated by the Model Generator (places, gates, C++ code, and activities) consistently to the results of the behavioral simulation and to user preferences. For instance, in the case node X had another neighbor node, node 5, then the routing model of node X would have been generated with another output branch linked to the *route* output gate, and ending in the places *outgoing5*, and *Outgoing5*.

### 6.3.3 Power Supply 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; after this point, a discontinuity in the voltage to the terminals of the batteries is generated, making the voltage drop from $V_{cutoff}$ [3] to zero.

An other factor that generates not-linearity in the characteristic of discharge of the battery is the charge "recovery effect". This effect is due to transient current requests: a rapid current request causes the depletion of the electrical charges from the electrode, with a consequent drop in the voltage. If the request is massive, then the voltage may drop below the $V_{cutoff}$, causing the temporary unavailability of the batteries, and hence a temporary shutdown of the node. Nevertheless, after a given amount of time, the charge will again move from the electrolyte to the electrode,

[3]It is the voltage in which the battery does not succeed to distribute current on any load, included its inner resistance

thus, making the node again available. This effect is closely dependent on the nature of the battery, and for lithium battery it can be neglected.

In WSN nodes, 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 at the price of a reduced overall life of the device due to conversion inefficiencies[4].

**Mathematical model**

In this thesis, power supply stage is assumed to be composed by batteries and by the DC-DC converter ,in a unique *black-box*, so that the voltage of the batteries can be considered constant in the interval ($\mathrm{V}_{oc}$, $\mathrm{V}_{cutoff-dc}$).

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:

$$E_{batt} = V_{cc} \cdot 3600\frac{s}{h} \cdot C_{effBatt} \tag{6.1}$$

where:

- $V_{cc}$=voltage;

[4]They rise the $\mathrm{V}_{cutoff}$.

- $C_{effBatt}$=Total battery capacity really usable by a load and expressed in Ah[5].

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

$$U = U' - \int_{t0}^{to+t^{*}} I(t)dt \tag{6.2}$$

where:

- $U'$ is the remaining capacity as result from the previous calculation step, and measured in Ah;
- I(t) is the current drained from the batteries at time t, measured in A;

The equation above 6.2 can also be expressed in terms of the energy requested from the batteries, expressed in Joule, or in other words:

$$U = U' - 3600\frac{s}{h}\int_{t0}^{to+t^{*}} \frac{E(t)}{V_{cc}}dt + R'(e) \tag{6.3}$$

where $R(e)$ is the non linear function of the charge recovery effects of the batteries, represented as a decreasing exponential function of the state of charge of the battery [127].

**Power consumption assumption**

The batteries model can be easily achieved from the 6.2, assuming that:

[5]Experimentally it has been measured as the 80% of total capacity

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

Under these assumptions the Equation 6.3 can be expressed with respect to the Energy $E$ as:

$$E = E' - E\Delta t + \epsilon \tag{6.4}$$

with $\epsilon$ being the contribution of the charge recovery effect function. From the previous

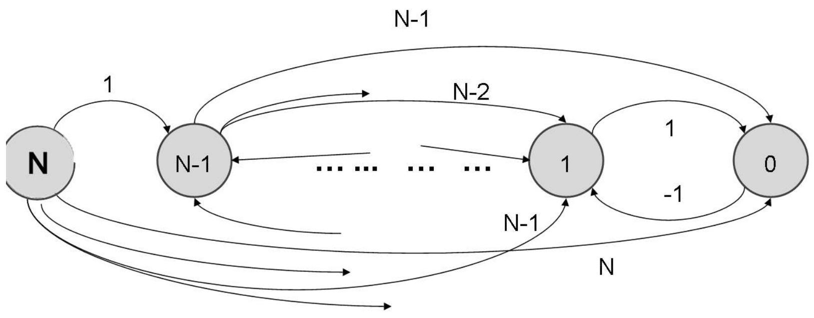
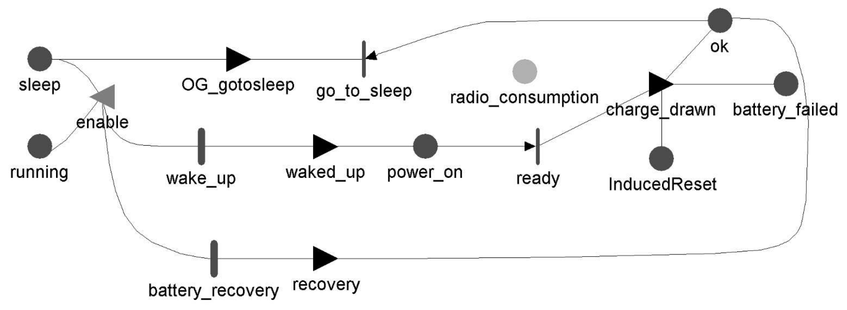


Figure 6.8: Finite state automata relative to the battery model

assumptions it is correct to consider that the "state of charge" of the battery can be considered as the remaining energy. That allows to model the above equation by means of the automata outlined in Fig. 6.8. $N$ is assumed to be the number of charge units that can be drained from the battery under constant discharge. 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



(a)

**Output gate** `Charge_drawn`

```
Float prob_reset;
Distribution reset;
Changes_Manager* n = Changes_Manager::Istance();
Double energyConsumption = n->get_cpu_sensors_energyRequest();
Double alpha = n->get_reset_distribution_value(energyConsumption);
Battery_charge_status->Mark() -= energyConsumption;
Prob_reset = reset.Weibull(alpha,beta);
If(Prob_reset>n->get_MinResetProb(energyConsumption))
      inducedReset->Mark() = 1;
If(battery_charge_status->Mark() <= minEnergy)
             battery_failed->Mark() = 1;
```

(b)

Figure 6.9: (a) SAN model of the Power Supply FM; (b) Output gate definition of *charge_drawn*.

recovery causes transitions in the opposite sense.

**San Model**

Figure 6.9.(a). shows the SAN model of the power supply component. It models stuck-at-zero and reset failures of a single node, by considering the natural battery discharge process, charge recovery effect and anomalous energy requests due to hardware faults.

Node's natural battery discharge process can be thought as the sum of two main contributions: i) processing activity, including CPU, I/O devices (e.g., leds), and sensing hardware, and ii) radio activity. T Node activity is considered as the alternation between two states: "running" and "sleep". The CPU awakening is modeled through a timed deterministic action (*wake_up*). At each awakening, the output gate *charge_drawn* drains the energy requested by the processing activity (see Figure 6.9.(b)).

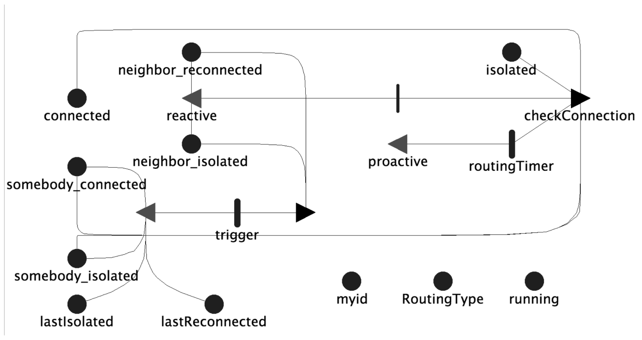
The other contribution (radio activity) is modeled in the Routing model. In particular, a proper amount of energy is drained every time a packet is sent, received or forwarded, depending on the current radio power being used, and on the size of the packet.

As reported in Figure 6.9.(b), the *charge_drawn* output gate is in charge of modeling stuck-at-zero and reset failures. A stuck-at-zero occurs when the residual charge is not sufficient to satisfy the processing request. Hence a token is placed in the *battery_failed* place. A reset is instead caused by anomalous energy requests due to temporary hardware faults. As shown in [86], these faults can be modeled as a Weibull stochastic process. When this activity fires, a token is moved in the *inducedReset* place. Figure 3.b also shows how model parameters (such as the energy consumption aliquot) can be gathered from the Changes Manager.

Finally, the model also takes into account nonlinear phenomena due to the so-called energy "recovery effect" [128], which is typical of several battery technologies (e.g., nickel metal hydride), apart from lithium batteries. The effect is modeled through a timed action (*battery_recovery*). When the action fires, the remaining charge is increased of a quantity which value depends on the adopted battery technology.

We validate the battery model by simulating a single node model and no failures. In this test, the Mica2 platform [23] is considered, achieving a 170 hours node lifetime, against the 172 hours lifetime measured on a real stand alone node running a simple broadcast counter application, as reported in [129, 16].

### 6.3.4 Communication Failure Model



(a)

**Outuput gate** `checkConnection`

```
Changes_Manager* d = Changes_Manager::GetInstance();
d->computeRoutingTable(); //computed if there are changes of
                          //interest for the current node
If(!d->isConnected(myid->Mark()) && (connected->Mark()==1){
      connected->Mark() = 0;
      isolated->Mark() = 1;
      d->propagateIsolation(myid->Mark());
              //it triggers the isolation failure
}
Else if(d->isConnected(myid->Mark() &&connected->Mark() == 0) {
      connected->Mark() = 1;
      isolated->Mark() = 0;
      d->propagateReconnect(myid->Mark());
//it triggers the reconnection of the node to the current topology
}
```

(b)

Figure 6.10: (a) Connection failure SAN model; (b) Output gate definition of *checkConnection.*

Figure 6.10.(a) shows the SAN of the Communication failure model. It models the behavior of the node when isolation failures occur. The model reproduces different behaviors depending on the nature of the routing tree update policy, namely if a reactive or proactive routing algorithm is being simulated [115]. If the proactive routing is considered, the model checks for isolation failures periodically (*routingTimer* activity). In the reactive case, the model checks for isolation failures when needed, i.e.

before sending a packet. In particular, the *reactive* input gate enable the execution of the gate *check connection* just before sending a packet. In order to simplify the model, the route update is not executed every time, but only when there is a change in the topology. This is accomplished exploiting the places *somebody_isolated* and *somebody_connected* that contain the ID of the last node manifesting a failure or the ID of the node that re-connected to the network, respectively. If the ID of the node contained in one of the mentioned places is present on the routing tree, or is a neighbor of a checking node, then the route is updated. In the case of proactive routing, this check is executed periodically. The code of the gate *check_connection* is reported in Figure 6.10.(b)., The function `computeRoutingTable()` provided by the Changes Manager (line 2 of Figure 6.10.(b)) is in charge of recomputing the routing tree of the node toward the sink node, i.e. it evaluate the route to the sink, if any. If there no route to the sink, (the checking node is isolated), a token is moved from the *connected* to the *isolated* place (lines 9-12). Similarly, when a node becomes connected after the computation of the routing table (e.g., nodes on the path to the sink recover), a token is moved from the *isolated* place to the *connected* place (lines 4-8). Then an isolation or reconnect event is notified to the Changes Manager, in order to propagate the event to all the interested nodes, and to compute new parameters for the model consistently with the updated topology. Note that the token from the *running* place is not removed when the node becomes isolated, modeling the common situation of a node which is up and running, but no more connected to the network. In this case, the node continues to produce packets, hence continuing to discharge its batteries.

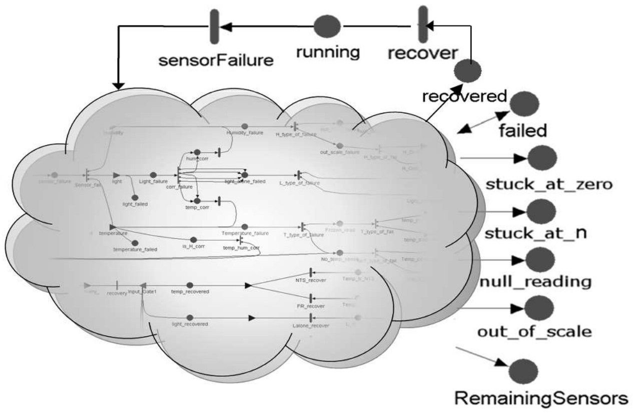


Figure 6.11: Internal structure of the "'Sensor"' FM

### 6.3.5 Sensor Board Failure Model

Sensor board failures are strictly dependent on the sensing hardware being used. Hence, we implemented a number of SAN libraries concerning the most common used sensor boards in WSN applications. Information collected by the user interface (see Section 5.1) indicates which sensor board is used for the WSN nodes, and which template among those available has to be specialized for the node model.

Figure 6.11 illustrates the conceptual structure of the sensor board model template. The hardware dependent part of the model is shown in the cloud, and need to be specialized according to user preferences. The places out of the cloud are the "interface" of this sub-model. Through the interface, the model generates and receives events related to other sub-models. The "sensorFailure" activity models the failing behavior of the board, according to an exponential distribution with constant rate [130], which depends on the adopted hardware. The part of the model reproducing specific hardware behavior is modeled starting from a FMEA approach (see section

6.1) conducted on several sensor boards, and added to the template by the model generation component (see Chapter 4, Section 4.2.1) . For example, to model the Crossbow Weather board [124], humidity, temperature and light sensor failures have been considered. The specific failure modes, and their rates, have been specified according to [124], where several failure dynamics related to this board have been observed. For the Crossbow sensor board, after the generation process we achieve the following failure model.

**Crossbow Weather Board Failure Model**

Figure 6.12 shows the SAN model of the Mica2 weather sensor board. The model is composed by two main sections:

1. failure event section;
2. recovery section.

The exponential distribution activities *fail_sensor* action and represents the times to failure of the sensor board. After the *ith* failure, which is represented by the number of tokens in the places *permanent* and *transient*, the time to the next failure is reduced by using a distribution with a mean equal to the original one divided by $2^i$. When the number of tokens of the places *Num_sensors* (initialized at a value equal to the number of the sensors on the board) becomes zero, the token is extracted from the place *running*, and the node becomes unavailable. The situation is the same when critical failures, such as those causing the stuck of the node (place *stuck*), such as failures of the humidity sensor. The place *stuck* is connected to the interface of the model *node state* (see the Section 6.3.1)

The exponential activities Normal and Burst represent the alternation of normal periods whose expected duration is indicated by the parameter $T_N$, where the failures occurs with rate $\lambda_N$, and of abnormal periods, having expected duration $T_B$, characterized by a higher rate $\lambda_B$. The rate of the exponential activity *fail_sensor*, is $\lambda_N$ or $\lambda_B$, depending on the marking of the places *NormalHWF* and *burstHWF*.

Each time the action (*Sensor_fail*) fires, the choice of the sensor responsible for the fault is made, with respect to the statistics obtained by failure data analysis presented in [12]. The sensor is put as unavailable after its failure. This operation demands a re-normalization of remaining probabilities relating to the sensors still operating. Dually, after an action of recovery of a sensor, 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.

The action *will_stuck* models the time to failure of the node, when given conditions are met, as observed in [12]. It follows a **Weibull**[6] distribution with shape parameter $\alpha = 2$[7]. Considering the statistics provided in [12] for the considered sensor board, *null reading* failures are likely to cause in the 45% of the cases permanent node failures within two days, and the following value for the distribution parameters are achieved:

[6] $F(x) = 1 - e^{-\lambda t^{\alpha}}$ is the temporal distribution of probability

[7] The shape equal to 2 is translated in a hazard rate $h(t)$ that it increases linearly over time

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

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

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

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

$$\Rightarrow \lambda = -\frac{\ln(0.45)}{172800^2 + 345600t^*} \tag{6.5}$$

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

Similar consideration is made regarding recovery actions, modeled as a lognormal distribution with shape parameter equal to 2 and $\lambda$ in such way that within 1 day the 90% of the transient failures are recovered.

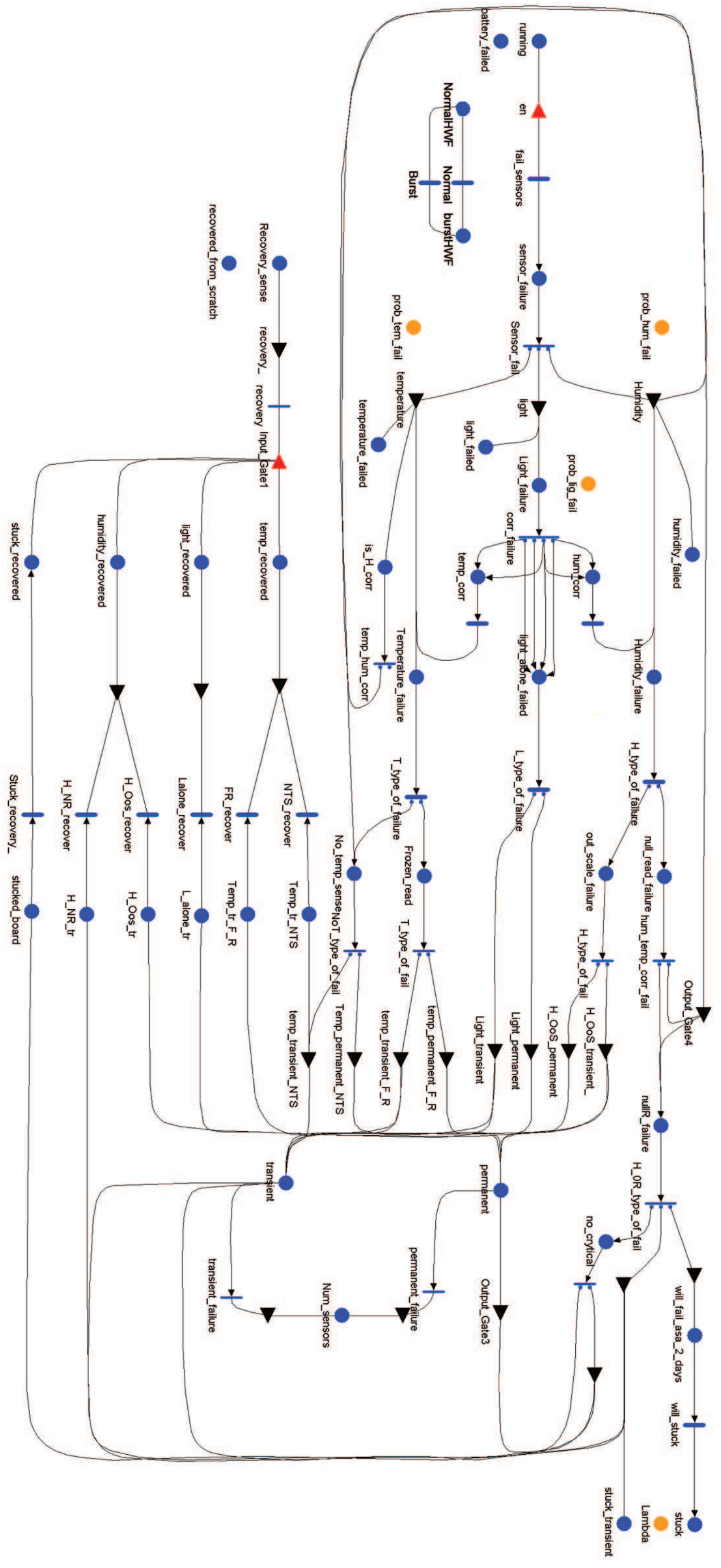


Figure 6.12: Mobius schema of the failure model of the Crossbow Weather sensor board

# Chapter 7

# Case Studies

*This chapter describes the results obtained following the approach proposed in this thesis, and it focuses on three different WSN deployments, namely a 10 nodes in line WSN, a 50 nodes randomly deployed WSN, and a 30 nodes WSN deployed on a hypothetical bridge. The simulations aim at showing how the approach proposed in this thesis can be adopted to evaluate significant resiliency measures for a specific network, when varying application configuration and routing protocols.*

## 7.1 Selecting realistic case studies

In order to consider realistic scenarios, we analyzed the experiences reported in the field of WSN for environmental and structural monitoring.

Table 7.1 reports a number of real world case studies on bridges and tunnels ([131, 132, 133, 134]), buildings and infrastructures ([135, 6, 136, 137, 138]), and environment ([12]) monitoring. In particular, for each work, we report information about number of nodes, size of radio packets, application duty-cycle, topology, routing algorithm being used and, where specified, measured network lifetime. From Table 7.1 we can see that the typical number of sensor nodes used in such applications is around to few tens which are typically organized in a in-line, 2-lines or grid topology. For such installations, radio packet sizes are always around tens of Bytes. Reported

Table 7.1: Analysis of existing studies

| **Work** | **Nodes** | **Packet [Byte]** | **Duty-cycle [s]** | **Topology** | **Routing** | **Measured lifetime** |
|---|---|---|---|---|---|---|
| [136] | 25 | 80 | 0.5 | grid | Reliable m-hop | several weeks |
| [131] | 14 | 6 | 600 | 2 lines | Regular m-hop | 6 months |
| [131] | 13 | 6 | 600 | 2 lines | custom | NA |
| [132] | 12 | NA | NA | 2 lines | Reliable m-hop | NA |
| [134] | 18 | NA | NA | 2 lines | NA | NA |
| [139] | 48 | NA | NA | 2 lines | NA | NA |
| [6] | 16 | NA | 4 | custom | Regular m-hop | 3.2 months |
| [133] | 64 | 36 | 43200 | 1 line | Regular m-hop | NA |
| [135] | 8 | NA | NA | 1 line | Regular m-hop | NA |
| [137] | 15 | 50 | NA | grid | custom | NA |
| [138] | 14 | 16 | 0.5 | 2 lines | Regular/ Reliable m-hop | NA |
| [12] | 43 | 32 | 300 | custom | Regular m-hop | 4 months |
| [12] | 92 | 32 | 1200 | custom | Regular m-hop | 2.5 months |

duty-cycles, instead, are different since they depend on application requirements. It ranges from 1 or more packets generated per second, up to a packet generated each 12 hours. Finally, adopted routing algorithms are multi-hop and reliable multi-hop (see Section 3.3), other than custom implementations. Details on mentioned routing protocols can be found in [115].

Table 7.2 reports the details of all the case studies performed in the following sections. In particular, to easily show the capabilities of the proposed approach, we consider three case studies related to different platform, topology, applications, and routing algorithms.

Table 7.2: Performed Experiments

| Platform | Case Study | Nodes | Packet [Byte] | Duty-cycle [s] | Topology | Routing | Metrics |
|---|---|---|---|---|---|---|---|
| Xbow | 1 | 10 | 25 | 2,4,20 | in line | Regular m-hop | MTTF, uptime |
| Mica 2 | 1* | 13 | 25 | 2,4,20 | hybrid in line | Regular m-hop | MTTF, uptime |
| Xbow Mica Z | 2 | 50 | 25 | 120, 600, 1800 | random | Regular m-hop Reliable m-hop , random walk | Connection Res. data delivery Res. lifetime |
| Xbow Mica Z | 3 | 30 | [25,250] | 360,3600 | Bridge | Regular m-hop Reliable m-hop random walk, gossiping flooding | Connection Res. data delivery Res. lifetime, overhead |

## 7.2 Experiments on simple in line topology

Figure 7.1 depict the topology composed of 10 nodes considered in this set of experiments. The inter node distances are set so that nodes are able to communicate only with their 1-hop neighbors. Despite its simplicity, this deployment is often adopted in many real-world applications as shown in Table 7.1. Moreover, this topology can be thought as a single branch of a more complex topology.

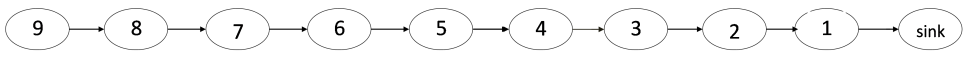


Figure 7.1: The considered linear topology.

### 7.2.1 Evaluated Metrics

The metrics we evaluate are: i) node uptime, and ii) node Mean Time To Failure (MTTF), referred to isolation failures (see Table 6.1). The node uptime is defined as the availability of a node (probability that a node is running) multiplied by the overall observation period. They have been defined as reward variables in the Mobius tool [140].

For several applications, it is interesting to evaluate such metrics per cluster of nodes. A cluster of nodes can be considered as a group of nodes which lays in the spatial proximity of each other and which shares the same neighboring nodes. The cluster uptime can be defined as the maximum of uptime values of all the nodes belonging to the cluster. In the same way, the cluster MTTF is the maximum of MTTF values of all cluster nodes. We chose these definitions because they are consistent with the mission of the WSN as a whole, instead of the mission of single nodes: nodes belonging to the same cluster are indeed usually adopted to monitor the same environmental phenomena. Moreover, it is sufficient that a single node per cluster is available to let produced measurements be forwarded to the sink

It is worth noting that other metrics can be simply defined using the same formalism. Hence the measurements can be tailored for the specific objectives of the analysis.

Sensitivities analysis are conducted on these metrics as a function of both the workload and the failure rate. The workload can be considered as the set of activities periodically performed by each node (measuring, computing, transmitting, receiving and forwarding) every X seconds, where X is the so called "duty-cycle" or "idle period".

In this set of experiments, attention is focused on measures related to links instead of nodes because, as reported in Section 6.1, the WSN fails when ( see Table 6.1): a) the amount of data delivered to the sink is less than a given threshold (data delivery failure), and/or b) when a subset of the network is not able to reach the

Table 7.3: Parameters used for the simulation process (their values are provided by the External Engine)

| Variable | Type | Range type | Range |
|---|---|---|---|
| isReactive | bool | Fixed | 1 |
| bandwidth | int | Fixed | 19600 |
| batteryEnergyJoule | int | Fixed | 11700 |
| CoverageThreshold | double | Fixed | 0.7 |
| CpuEnergySeconds | double | Fixed | 2,00E-06 |
| **dutyCycle** | **int** | **Manual** | **[2,4,20]** |
| numberOfNodes | int | Fixed | 10 |
| **packetSize** | **int** | **Manual** | 50 |
| defaultRXRadioConsumption | double | Fixed | 4.0E-5 |
| defaultTXRadioConsumption | double | Fixed | 6.5E-5 |
| routingType | int | Fixed | 0 |
| TimerCheckConnection | int | Fixed | 20 |
| **sensorFailureRate** | **double** | **Manual** | **[1E-7, 5E-7, 9E-7, 1.1E-6]** |
| simulationTime | int | Fixed | 30 days |
| timeOutACK | int | Fixed | 20 |

sink (disconnection failure). Node oriented metrics could return a finer-grain analysis however not useful and not cost-effective when evaluating the whole network. Moreover considering only the coverage failures will results in a too much coarse-grain analysis. Metrics i) and ii) are a good trade-off between detailed analysis and evaluation costs.

### 7.2.2 Simulation Setup

Table 7.3 summarizes most of the parameters of the SAN model, along with the values we adopted for the experiments. Actual values for these parameters have been gathered from the related work (see section 2). The last three parameters are set as *variable*. In particular, *sensor_failure_rate* and *idle_period* parameters are used to conduct sensitivity analysis, hence their values change during the simulation.

Each simulated node is a Crossbow Mica2, which has an Atmel ATmega128L microcontroller with 4 KB of RAM, 128 KB of ash, and a CC1000 radio. The radio operates at 968 MHz, transmits at 38.4 Kbps, and uses Manchester encoding. Each node was attached to a Crossbow Weather board [39, 124, 23] equipped with light, humidity and temperature sensors. We assume every node to run a typical TinyOS multi-hop application which senses and sends light and temperature values to a sink node periodically, in multi-hop fashion, without stand-by periods, i.e. periods spent by node in a low power state.

The model has been numerically solved using the Mobius tool, by performing a transient analysis in a period of 30 days. The assumed observation period is longer than the maximum expected lifetime of a single node [129] for the considered platform and application i.e. 200 hours, considering the nature of the proposed application (always-on and periodic WSN).

### 7.2.3 Results

Figure 7.2 shows the results of the sensitivity analysis conducted on the node uptime and MTTF as a function of the idle period (i.e., the workload) and the failure rate. In particular, Figures 7.2.a and 7.2.b show the results obtained with a failure rate fixed to $5E-7$ and a idle period varying from 20 seconds down to 4 and 2 seconds. Figures 7.2.c and 7.2.d are relative to a fixed idle period (4 seconds) and a varying failure rate ($1E-7$, $5E-7$, $9E-7$, $1.1E-6$ failures per second - fps). The plotted values are obtained within a relative confidence interval of 95% of their mean and variance.

As one could expect, the longer the idle period, the higher the uptime. As a

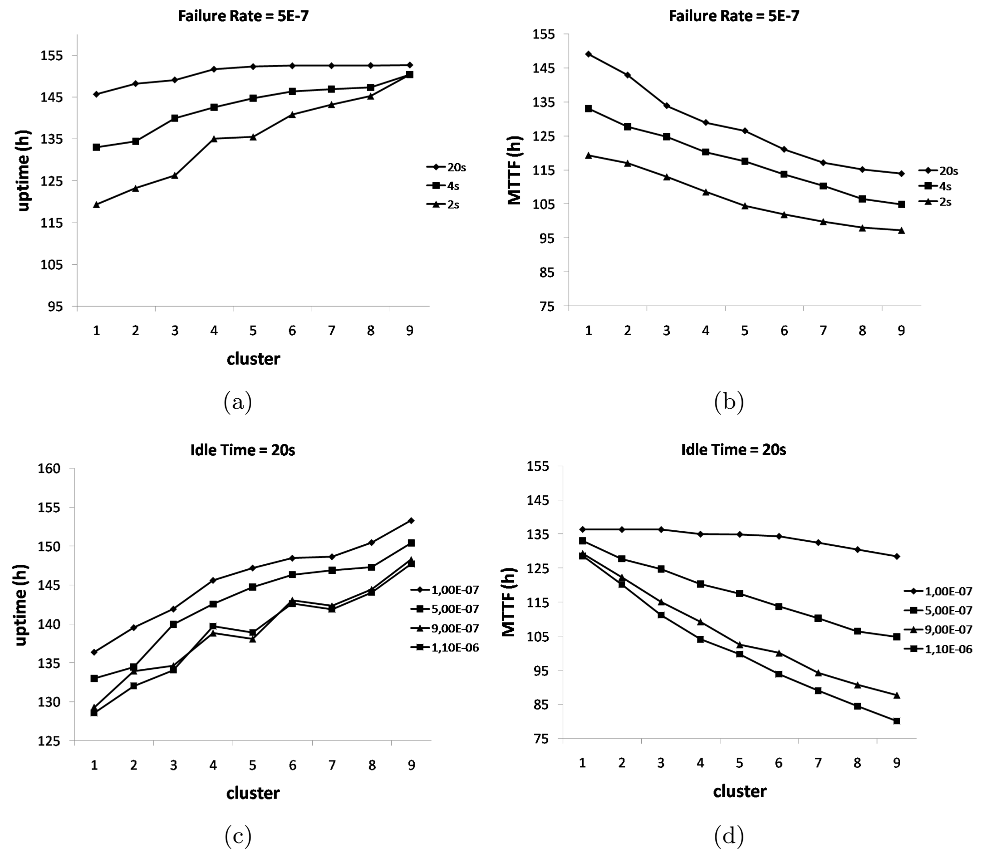


Figure 7.2: Uptime (a),(c) and Mean Time To Isolate (b),(d) both in hours, for topology of fig.7.1: (a) and (b) for different idle period of the application with constant failure rate of $5E - 7$, (c) and (d) for different failure rate for a idle period of 4s.

general result, the estimated slope of the uptime increases as the idle period decreases, for the considered topology. More in detail, for the 20s idle period experiment, the uptime assumes almost the same value for the last 6 nodes, then it starts to decrease. This is due to the critical value that the aggregated traffic assumes, starting from node 3 down to 1. In other terms, the more a node/cluster is close to the sink, the more traffic it has to manage, the faster it will discharge its battery. This effect “shifts” to the right as the idle period decreases. In particular, for idle periods equal to 4s and 2s, the uptime starts to significantly decrease from nodes 6 and 8, respectively (short idle periods involve more traffic to be produced).

Moreover, for the last experiment (duty cycle of 2s) more than one knee can be noted on the plot due to the great traffic that flows into the network: on the fourth and sixth 'hop'.

Figure 7.2.b outlines the values obtained for the MTTF. This plot evidences the occurrence of isolation failures and how they propagate into the network. Consistently with intuition, the MTTF is a decreasing function of the distance to the sink. For instance, the failure of the node 2 implies that all other nodes, from 3 to 9, are isolated. Moreover, the plots translate down as the idle period decreases. It is interesting to relate MTTF estimates to uptime estimates. For example, with reference to the 20s idle period, node number 9 is alive for 153.2 hours (Figure 7.2.a). However, it is able to reach the sink for only 113.3 hours on average, wasting about 40 hours of its lifetime.

As for Figures 7.2.c and 7.2.d, the relationship between the plots is dual to the one observed in the previous case. When the failure rate increases, the uptime plots translates down (Figure 7.2.c), whereas the MTTF plots slope increases (Figure 7.2.d). This last behavior is intuitive: the more often a node fails, the more often its child nodes will result isolated. Note that the uptime plots obtained in correspondence of failure rates equal to $9E-7$, $1.1E-6$ failures per second exhibit a more irregular profile, due to the big impact of node failures: if a node fails, its one-hop neighbor will be subject to less traffic to be forwarded, hence it will live longer.

### 7.2.4 Improving the linear Topology

Measurements obtained for the linear topology show that the more a node is next to the sink, the more it is prone to fail. Although peripheral nodes are less stressed, they also fail prematurely, due to the isolation from the sink. A resiliency improvement would thus be the MTTF improvement of peripheral nodes. To this aim, an intuitive approach is to enforces the network topology duplicating the nodes in proximity of the sink. Specifically, node 1 is duplicated with node 10, forming a pseudo-node 1', node 2 with 11, forming 2', and node 3 with node 12, forming 3' as shown in Figure 7.3.a. for a 30% cost increase. This enforcement introduces redundant paths into the network, so as to tolerate the failure of one of the nodes belonging to one of the mentioned groups (pseudo-nodes).

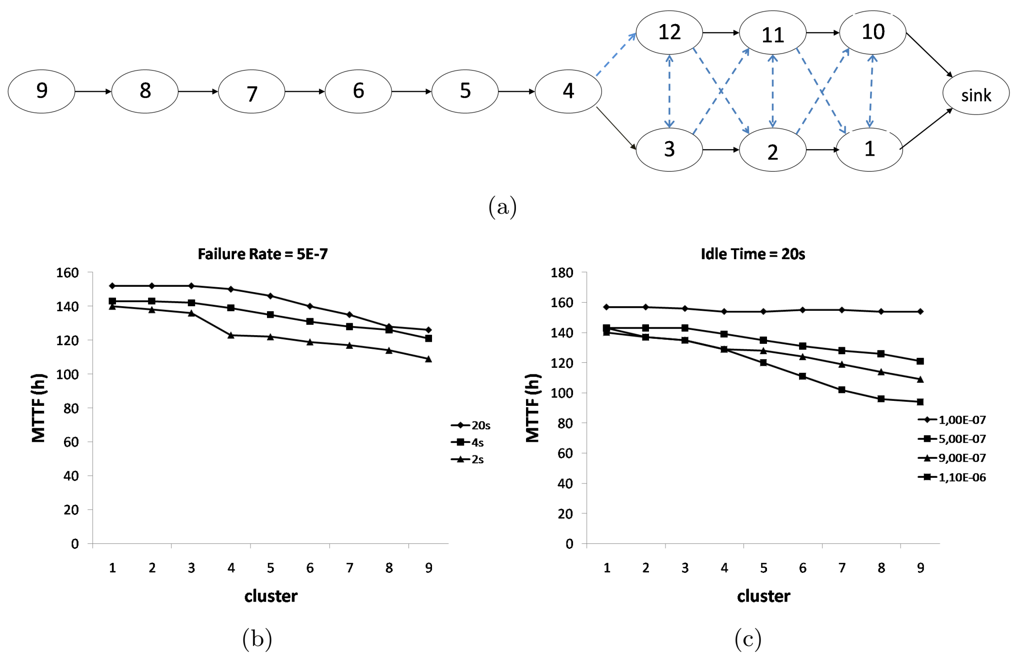


Figure 7.3: (a) Improved topology: alternative paths are outlined with dotted arrows; (b) and (c) MTTF achieved measures.

Let us describe how the proposed approach helps to quantify the resiliency improvement we expect for this enforced topology. The MTTF benefit introduced by this simple improvement is shown in Figures 7.3.b and 7.3.c. The Figure report the MTTF as a function of the distance of the replicas to the sink, when varying the idle period and the failure rate. We observe an overall improvement of the MTTF for all the nodes, especially for the peripheral ones. With reference to Figure 7.3.b, the last node results isolated after 127h, 121h and 110 h respectively, instead of 114h, 105h and 96h obtained with the linear topology. A rapid change of slope can be observed at the third cluster. The change is justifiable by the fact that node 4 acts as a bottleneck for all the network traffic, hence failing prematurely and causing the isolation of all the nodes that depend on it. From Figure 7.3.c it is interesting observe that, for a low failure rate (1E-7 fps), a almost constant MTTF is obtained (almost equals to 155h). In this case, we obtain a MTTF increment for the last node of about 27 hours (21% of improvement). This also applies to other nodes. It is also clear that even in this case node 4 still represent a resiliency bottleneck for the given topology.

## 7.3 Experiment on a Random Topology

This section presents results of the simulation of a WSN of 50 nodes randomly deployed. The topology has been generated achieving a distribution of number of neighbors per node reported in Figure 7.4 In our analysis, we consider 3 distinct routing algorithms, namely i) multihop, ii) TinyOS Reliable Multihop, iii) Random Parent Selection, here referred as random routing. For each considered algorithm, we evaluate a set of metrics to pinpoint both dependability bottlenecks and, for the considered topology, the most appropriate routing algorithm among those considered. The same approach can be followed by a hypothetic user of the model, who can exploit simulation results to fine-tune his applications.

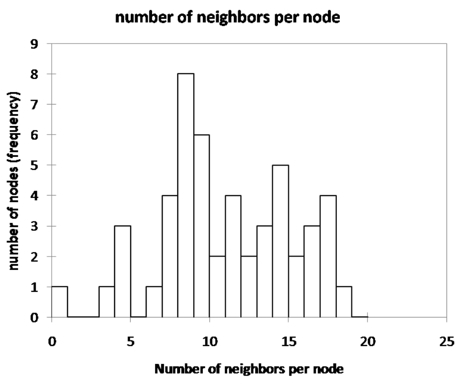


Figure 7.4: Initial Routing tree for the considered topology

### 7.3.1 Evaluated Metrics

The following metrics (or reward variables) are used to evaluate system performances:

- **Connection Resiliency** as a function of **time**, defined as the average number of alive and not isolated nodes, evaluated on a daily basis. The metric can be used to estimate the maximum mission time after which the WSN is no more able to fulfill the required coverage.

- **Relative Data Delivery Resiliency** as a function of **time**, defined as the average number of useful packets delivered to the sink over the number of produced packets, evaluated on a daily basis. It is useful to estimate the maximum mission time after which the WSN is no more able to deliver the required fraction of data to the sink.

**Node lifetime** defined as the summing of all the interval of times where the node is not failed.

Consistently with the former set of performed experiments, the workload is considered as the set of activities periodically performed by each node (measuring, computing, transmitting, receiving and forwarding) every X seconds, where X is the "duty cycle". We assume every node to run a typical TinyOS multi-hop application which senses and sends measurements to a sink node periodically (node 0) according to the adopted routing algorithm. We focus on the Mica2 platform, equipped with the Chipcon CC1000 radio devices at 968MHz and weather sensor board.

### 7.3.2 Simulation setup

Table 7.4 summarizes most of the parameters of the SAN model, along with the values adopted for the experiments. Actual values for these parameters have been gathered from real settings [12]. Sensitivity analyses are conducted by computing the mentioned metrics as a function of i) node duty cycle (120s, 600s, and 1800s), and ii) failure rate (1E-7, 5E-7, 9E-7, 1.1E-6 failures per hour). The model has been numerically solved using the Mobius tool, by performing a transient analysis in a period of 60 days. It is worth noting that in this set of experiments we consider a larger time horizon for the simulation than in the former case studies, since the simulated application use a larger duty-cycle, and hence nodes are likely to survive for a larger amount of time.

Table 7.4: Values for parameters used in the simulations.

| Variable | Type | Range type | Range |
|---|---|---|---|
| numberOfRetries | int | Fixed | 5 |
| isReactive | bool | Fixed | 1 |
| TTL | short | Fixed | 50 |
| ackSize | int | Fixed | 1 |
| aggregationTime | int | Fixed | 10 |
| bandwidth | int | Fixed | 39600 |
| batteryEnergyJoule | int | Fixed | 11700 |
| CoverageThreshold | double | Fixed | 0.7 |
| CpuEnergySeconds | double | Fixed | 2,00E-06 |
| IsDataAggregation | bool | Fixed | 0 |
| **dutyCycle** | **int** | **Manual** | **[120, 600, 1800]** |
| gossipingProbability | double | Fixed | 0.3 |
| numberOfNodes | int | Fixed | 50 |
| **packetSize** | **int** | **Manual** | 25 |
| defaultRXRadioConsumption | double | Fixed | 0.00004 |
| defaultTXRadioConsumption | double | Fixed | 6.5E-5 |
| **routingType** | **int** | **Manual** | **[0,1,2,3,4]** |
| TimerCheckConnection | int | Fixed | 60 |
| **sensorFailureRate** | **double** | **Manual** | **[1E-7, 5E-7, 9E-7, 1.1E-6]** |
| simulationTime | int | Fixed | 7776000 |
| timeOutACK | int | Fixed | 20 |

### 7.3.3 Results

Figure 7.5 sketches the routing tree computed on the considered topology at the startup of the simulation by the multihop algorithm (both regular and reliable). As shown in Figure 7.5, there are several nodes that act as leaf nodes, i.e. nodes without children, while there are other nodes, such as node 47, acting as router node for a conspicuous amount of nodes (for node 47, we account for 24 nodes). Consistent with the intuition, such router nodes are more stressed by the forwarding activity and hence more prone to fail. A failure of one of such nodes requires the re-computation of the routing tree since new router nodes have to been selected in order to reach the sink. For instance, we measured for node 47 a lifetime of only about 15 days over the 60 simulated. Consequently, after the failure of node 47, a new router is selected . In this case, the analysis showed that node 48, a leaf node in the former topology, starts to act as a router node, replacing node 47. Figure 7.4 reports the distribution of the number of neighbor per node.

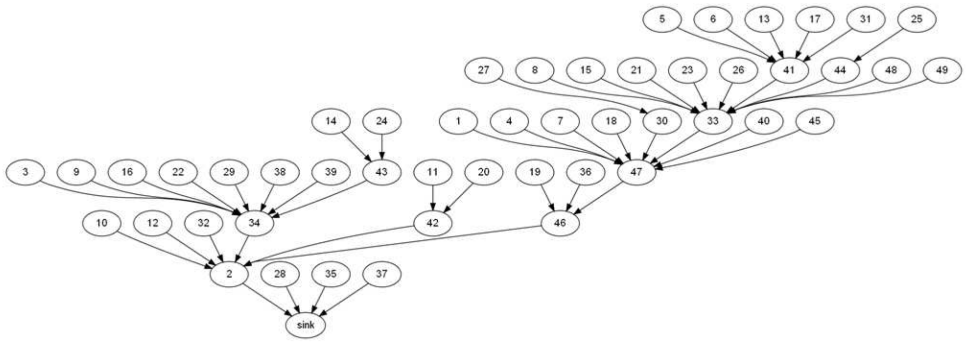
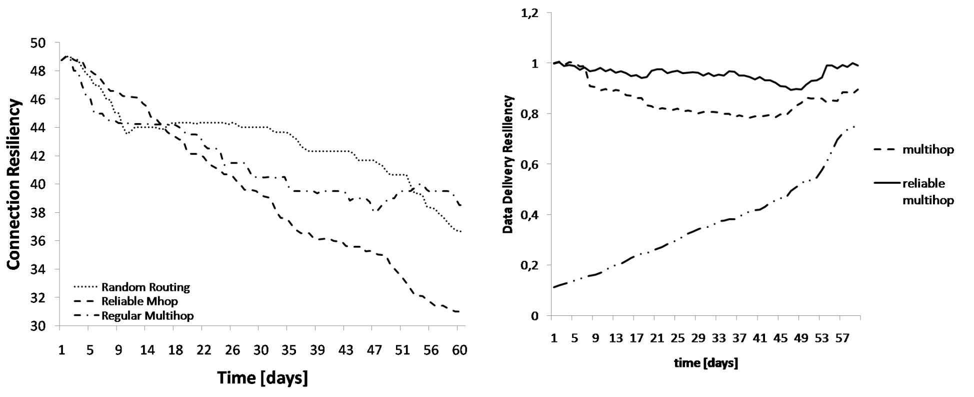


Figure 7.5: Initial Routing tree for the considered topology

This effect manifests more frequently when adopting the multihop and the reliable multihop algorithms, since they tend to use the same set of routes while available, hence overloading always the same set of nodes. This is very unlikely to happen when using the random routing algorithm. This algorithm acts selecting randomly one of the neighbors from the neighbor list, for each packet sent, thus distributing the load in the network. While this could seam an acceptable solution, its main drawback is that often packets transit following non optimal routes, in turn increasing the average hops to pass to reach the sink, causing a low level of packets delivered to the sink.

Figure 7.6.(a) provides an example of estimation of the connection resiliency. In the first days, the coverage of the two multihop based algorithms is higher than that delivered by the random routing. However, as nodes start to fail (both due



(a) (b)

Figure 7.6: (a) Connection Resiliency for the considered routing protocols, and for duty cycle = 600s, failure rate = 5e-7 f/h, packet length = 25B; (b)Data Delivery Resiliency for packet size of 25B, duty cycle 600s and failure rate 5E-7 f/h

to hardware failures, network partition or battery depletion), the random routing algorithm tends to deliver better connection resiliency than the others. As shown in Figure 7, this is due to the uniform discharge induced by selecting a random parent in place of a fixed one, with the final effect of increasing the average number of surviving nodes, and consequently, the overall connection resiliency.

Figure 7.6.(b) shows the metric of data delivery resiliency of the considered routing algorithms, for a packet size of 25Bytes, a failure rate of 5e-7 f/h and a duty-cycle of 1 packet generated per 600s. Coherently with the intuition, the reliable multihop is capable of delivering a larger amount of useful data to the sink (packet that are not duplicated) - 95% on average of the generated packets. On the other side, the random routing, while showing interesting characteristics for the coverage, is not capable of delivering acceptable performance, since it is only capable of delivering 36% of the packets, on average.

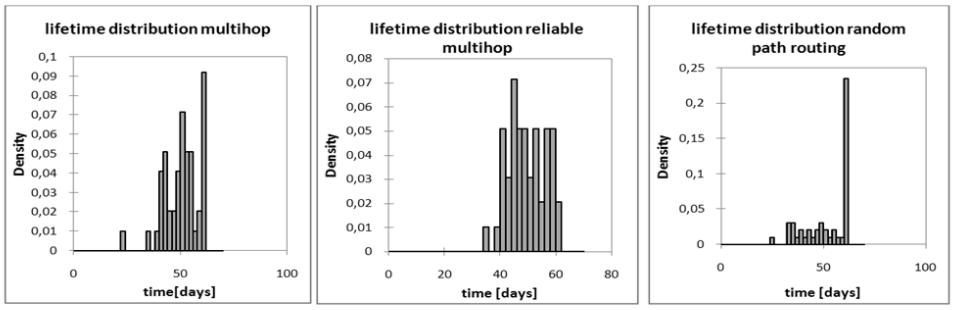


Figure 7.7: Data Delivery Resiliency for packet size of 25B, duty cycle 600s and failure rate 5E-7 f/h

Figure 7.7 shows the lifetime distributions related to the considered routing algorithms.

Table 7.5 shows statistics for the distribution of Figure 7.7. As reported, the random routing enables 23 nodes out of 50 nodes to reach the end of the simulation without failing (maximum value for the lifetime, i.e., 60 days), while the multihop and reliable multihop allow only 9 and 2 nodes to reach the maximum lifetime, respectively. The distributions of Figure 7.7 shows also that the lifetime distributions related to the reliable multihop routing has a small range of variance (26 days), tightly concentrating values for the lifetime of nodes around the mean of 49 days.

This translates in a more foreseeable behavior of the WSN due the small interval for node lifetime values. Moreover, the reliable multihop is capable of delivering a higher rate of information to the sink node than the regular multihop, however at the price of an additional overhead due to the acknowledgement, and extra transmissions of lost packets. Oppositely, while the random algorithm delivers higher coverage and smaller overhead, it is not capable of delivering the same amount of information to

Table 7.5: Statistics for the lifetime distributions shown in Figure 7.7

| statistics | Multihop | Rel. Multihop | Random |
|---|---|---|---|
| No. Of observations | 49 | 49 | 49 |
| Minimum | 23 | 34 | 24 |
| Maximum | 60 | 60 | 60 |
| Freq. Of Minumum | 1 | 1 | 1 |
| Freq. Of Maximum | 9 | 2 | 23 |
| Range | 37 | 26 | 36 |
| 1st Quartile | 44 | 45 | 43 |
| Median | 51 | 49 | 57 |
| 3rd Quartile | 55 | 55 | 60 |
| Mean | 49,857 | 49,184 | 51,163 |
| Standard Deviation | 7,962 | 6,435 | 10,502 |

the sink as the multihop.

As final analysis, for the considered topology, platform and for the considered parameters, the simulation pinpoints the reliable multihop as the most convenient routing algorithm to be used in order to achieve a good trade-off between the lifetime of the network and the amount of data delivered to the sink. Analysis for different value of parameters revealed similar insights showing that the reliable multihop is an effective solution when dealing with large networks with non negligible failure rates. It also pointed out that the random routing is a viable solution for smaller networks and short duty-cycle where the overhead of the reliable multihop overweighs the provided benefits.

## 7.4 Experiment on a Real Topology

The objective of the following analysis is to demonstrate how the proposed approach may be employed during the design of a WSN for infrastructure monitoring.

Consistently with the size of the real world case studies reported in Table 7.1, we

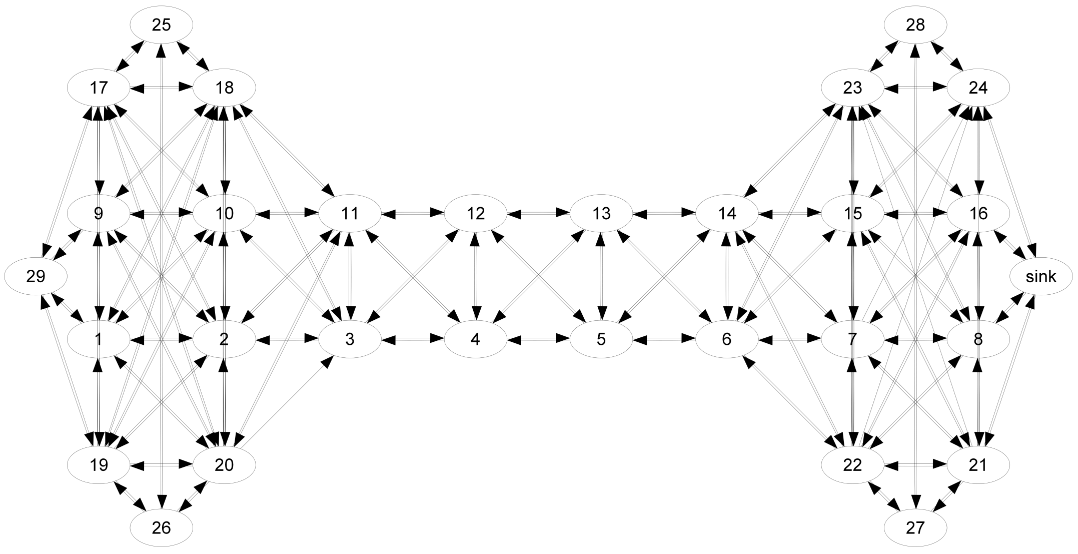


Figure 7.8: Topology of the studied WSN

focus on a 30 nodes WSN covering a hypothetic bridge composed of a 250 meters suspended roadway and of two 50 meters high towers. The 2-dimensional topology of the considered WSN is depicted in Figure 7.8.

### 7.4.1 Evaluated Metrics

According to the definitions provided in Section 3, the following reward metrics are evaluated:

**- Relative Connection Resiliency** as a function of the **number of manifested failures**, defined as the fraction of alive and not isolated nodes over the number of alive nodes. This metric is useful to measure the WSN ability to keep the network connected in the presence of failures.

**- Absolute Connection Resiliency** as a function of **time**, defined as the average number of alive and not isolated nodes, evaluated on a daily basis. The metric can be used to estimate the maximum mission time after which the WSN is no more able to fulfill the required coverage.

**- Relative Data Delivery Resiliency** as a function of the **number of manifested failures**, defined as the fraction of useful packets delivered to the sink over the number of produced packets. It is used to measure the WSN ability to keep the required data delivery efficiency in the presence of failures.

**- Relative Data Delivery Resiliency** as a function of **time**, defined as the average number of useful packets delivered to the sink over the number of produced packets, evaluated on a daily basis. It is useful to estimate the maximum mission time after which the WSN is no more able to deliver the required fraction of data to the sink.

**- Overhead**, defined as the ratio between the number of non useful packets (duplicated and corrupted) and useful packets delivered to the sink in $\Delta T$, i.e.

$$Overhead(\Delta T) = \frac{Dup\_Packets(\Delta T) + Corr\_Packets(\Delta T)}{Useful\_Packets(\Delta T)} \tag{7.1}$$

**- Node Lifetime** defined as the time after which a node stops working.

### 7.4.2 Evaluation Strategy

We aim to assess the WSN resiliency as a function of routing algorithms and several application parameters values. We focus on the following routing algorithms: i) multihop, ii) TinyOS reliable multihop, iii) random parent selection[1], iv) gossiping, and v) flooding. As for application parameters, we focus on packet sizes (from 25 up to 250 Byes) and application duty-cycles (from 3600 down to 360 seconds). Finally,

[1]here referred as random routing

since there is no agreement in the literature on the hardware failure rate of these networks, we repeat each analysis for different failure rates. The rate of the hardware faults is 1 each 115 days (1E-7 failures per second - fps) and 11,5 days (1E-6 fps).

In the following, results of three out of eight possible experimental campaigns are reported:

**Experiment 1**: low workload (packet size of 25B and duty-cycle of 3600s) and low failure rate (1E-7 fps);

**Experiment 2**: high workload (packet size of 250B and duty-cycle of 360s) and low failure rate, to analyze the impact of a more stressful application;

**Experiment 3**: low workload and high failure rate (1E-6 fps), to analyze the consequences due to a harsher environment.

We use the Mobius modeling environment [119] to simulate the SAN model. A simulation on a period of 6 months is used. The confidence level is 95%. Unless otherwise specified, the parameter values are as in Table 7.6.

According to the proposed approach, experiments consist of the following phases.

**behavioral simulation.** The topology is built and the network is simulated by the TOSSIM simulator. Values for static parameters are gathered and stored in simulator logs. As for the sensor node, we focus on Crossbow MicaZ platform, which is the evolution of the Mica2 [23], which has an Atmel ATmega128L microcontroller with 4 KB of RAM, 128 KB of ash, and a CC2420 radio. The radio operates at 2.4GHz using the Zig-Bee standard, and it transmits at 256 Kbps. Each node was attached to

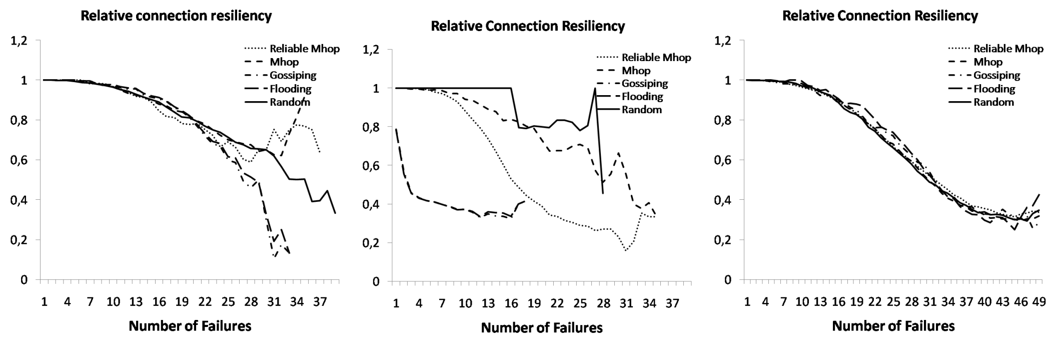
Table 7.6: Simulation parameters

| Parameters | Description | Default Value |
|---|---|---|
| Energy_PerReceivedBit | Energy consumption for radio activity, per received bit | 1.36E-6J |
| Energy_PerSentBit | Energy consumption for radio activity, per sent bit | 2,80E-06J |
| NumberOfRetries | Number of retries when sending a packet, after that if no ack is received, the packet is discarded | 3 3 |
| TimeToLive | Time to Live of sent packets | 15 |
| Vcc | Batteries Voltage | 3V |
| Ack_Size | Size of the acknowledgement in Byte | 10B |
| Bandwidth | Available bandwidth | 19200 bit/s |
| Batt_energy | Total battery energy in Joule | 10800J |
| Cpu_Consumption | CPU active Energy Consumption in mA | 0.0087mA |
| Cpu_IdleConsumption | CPU idle Energy Consumption in mA | 2.16E-4mA |
| Duty_Cycle | Application duty cycle | [3600.0s, 360.0s] |
| Gossiping_Prob | Probability of gossiping - used in gossiping routing | 0.03 |
| Sensor_number | Number of sensors composing the sensor board | 3 |
| Nodes_number | Number of nodes | 30 |
| Packet_Size | Size of the packets in Byte | [25B, 250B] |
| Routing_Type | Type of routing algorithm. 0 multihop 1 reliable multihop, 2 random, 3 gossiping, 4 flooding | [0-4] [0-4] |
| Routing_time | Time period for sending route check packets | 120s |
| Sensors_fault_rate | Rate of sensor board hardware faults | [1.0E-7fps, 1.0E-6fps] |
| Simulation_End | Simulation time | 15552001s |
| Ack_timeout | Time-out after that a packet is resent | 20s |

a Crossbow Weather board equipped with light, humidity and temperature sensors [39, 124].We assume every node to run a typical WSN application which senses and sends light and temperature values to a sink node, periodically. We assume that all nodes are initially equipped with two AA lithium ions batteries capable of providing 10800 Joules of total energy (see Table 7.6).

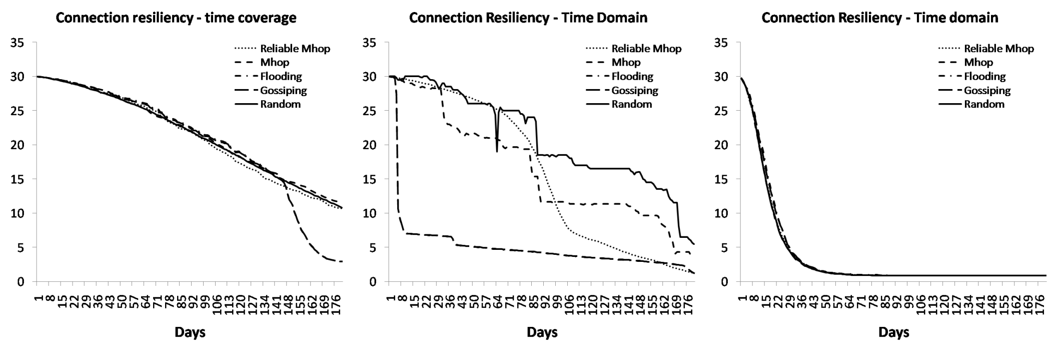
**Model generation.** The Model Generator component is used to generate the SAN model by using the result of the behavioral simulation. Single node models are joined together forming the WSN failure model, according to the simulated topology. The Changes Manager component is added to the model as an external library and the overall model is compiled into a single executable file.

**SAN model simulation.** The SAN model is simulated and the reward variables are estimated.



(a) (b) (c)

Figure 7.9: Relative connection resiliency against the number of manifested failures. (a) Experiment 1, (b) Experiment 2, (c) Experiment 3. Comparing (a) and (b), the difference between algorithms is more evident as the workload increases. Comparing (a) and (c), an increasing slope can be observed when the failure rate increases, and no differences are observed between routing algorithms, since hardware faults become the predominant cause of node failures.

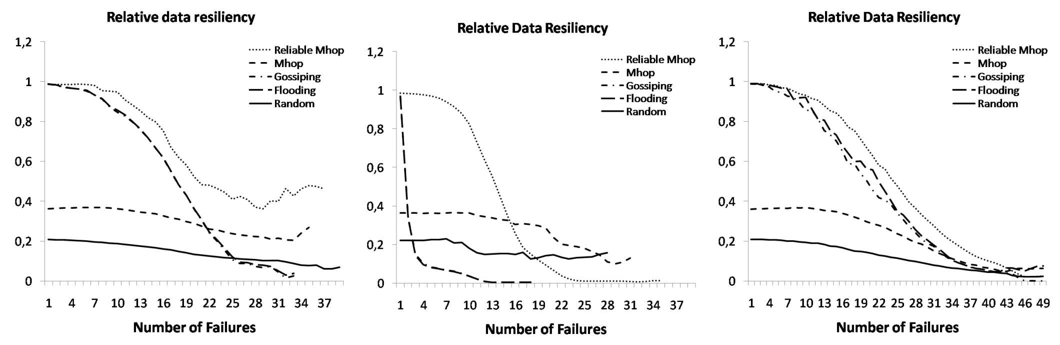


(a) (b) (c)

Figure 7.10: Absolute connection resiliency against time. (a) Experiment 1, (b) Experiment 2, (c) Experiment 3.

### 7.4.3 Results from Low Failure Rate Scenario

Figures 7.9-7.13 show the results of the performed analysis.

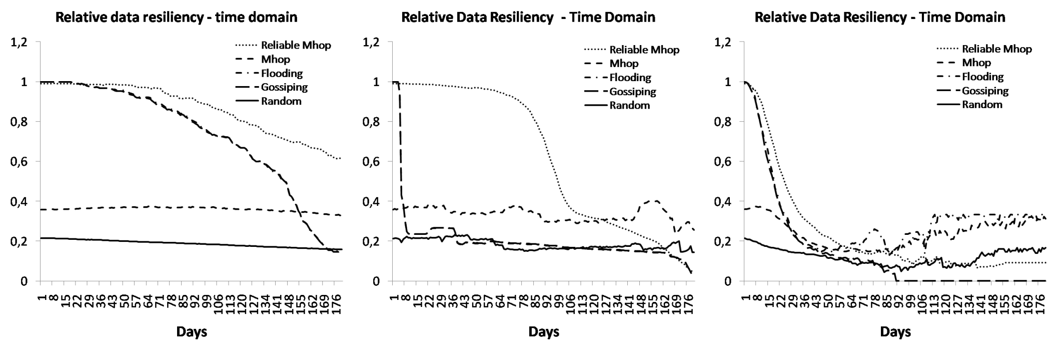


(a) (b) (c)

Figure 7.11: Relative data resiliency against the number of manifested failures. (a) Experiment 1, (b) Experiment 2, (c) Experiment 3. Comparing (a) and (b), flooding, gossiping and reliable multi-hop show a sharp drop, which is increased and anticipated in the case of higher workload. Random and regular multi-hop are less sensitive to workload changes, at the cost of lower resiliency. Comparing (a) and (c), Differences between flooding, gossiping and reliable multi-hop are mitigated by the higher failure rates. At the end in fig. (c), after several failures, all the algorithms tend to the same resiliency level.

Figure 7.9.(a) reports the relative connection resiliency against the number of manifested failures in the network[2] for Experiment 1. Initially, all the considered routing algorithms rebound to failures in a similar way, despite their different behavior. Differences start to be noticed after the 25th failure (connection resiliency of about 0.7) when the resiliency of gossiping and flooding start to decrease more sharply. This is mainly due to the higher overhead these algorithms cause in the network (see Figure 7.13.(a)), which cause a *collapse* of the WSN at a given point. The positive slope shown for multi-hop and reliable multi-hop algorithm after the 30th failure is indicative also of transient isolation failures, i.e. induced by transient hardware failures of inner nodes. This effect is however not observable if looking

[2]note that both transient and permanent failures are considered in this work, hence the number of observed failures can be greater than the number of nodes.

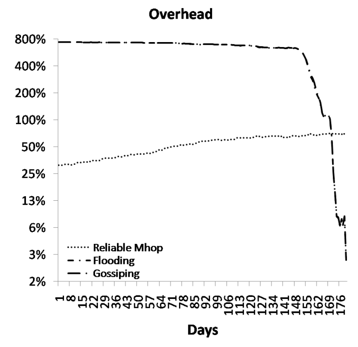


(a) (b) (c)

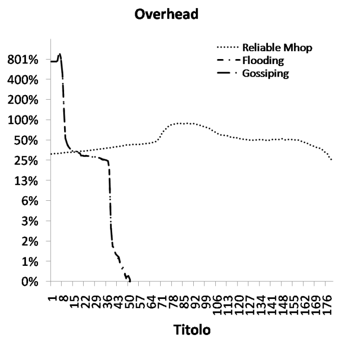
Figure 7.12: Relative data resiliency against time. (a) Experiment 1, (b) Experiment 2, (c) Experiment 3. The differences observed in sub-fig. (c) after 64 days are not significant, since at that time almost all nodes are failed, as shown in Fig. 7.10.(c).

at Figure 7.10.(a), which reports the absolute connection resiliency as a function of time. This is due to the different scale: the time-to-failure process is not uniform with respect to time, and several failures tend to concentrate at the last days of exercise. On the other hand, 7.10.(a) is useful to estimate the maximum mission time after which the WSN is no more able to fulfill the required connectivity degree. Hence, it might be useful to schedule maintenance actions in advance (e.g., battery replacement and damaged sensor nodes substitution). The collapse is also observable from Figure 7.10.(a), which reports the absolute connection resiliency as a function of time. From this Figure we can also estimate that the collapse occurs after about 145 days of exercise. This is useful to guide developer choices or to schedule maintenance actions in advance. For instance, for a WSN with a mission time of 120 days (i.e., before the collapse) and a 15 nodes minimum coverage, gossiping and flooding may still represent a reasonable choice, for the given workload and failure rate.

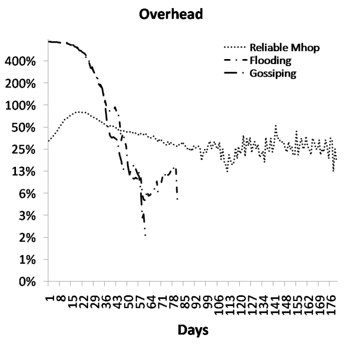
The situation changes when considering a more stressful workload (Experiment 2,



(a)



(b)



(c)

Figure 7.13: Network overhead (log scale) induced by routing algorithms over time; regular multi-hop and random are not reported, since they do not introduce overhead. (a) Experiment 1: flooding and gossiping present almost the same large overhead; reliable multi-hop overhead increases as the network ages, due to the reduction of available nodes and (good quality) paths to the sink. (b) Experiment 2: flooding and gossiping overhead drops earlier, due to the shorter lifetime of the network; reliable multi-hop overhead has a peak value at 78 days of exercise, corresponding to the sharp drop in both data and connection resiliency (Figs. 7.10.(b) and 7.12.(b)). (c) Experiment 3: the peak for reliable multi-hop overhead shifts left (at 15 days), justifying the earlier drop in data and connection resiliency (Figs. 7.10.(c) and 7.12.(c)).

Figures 7.9.(b) and 7.10.(b)). In this case, flooding and gossiping are not a suitalble choice since they tend to overload the network, shortening the lifetime of forwarding nodes that exhaust their energy budget sooner than when running random, multihop, and reliable multihop routing algorithms. In particular, as shown in Figure 7.9.(b), the random routing outperforms the others in terms of the connection resiliency. However, looking only at connection resiliency may lead to incorrect conclusions. As a matter of fact, random routing is not able to deliver more than the 24% of the produced data (see Figure 7.11.(b) reporting the relative data delivery resiliency as a function of the number of failures). This result is confirmed also for Experiment 1 (see Figure 7.11.(a)). Hence, a judicious choice may be to adopt reliable multihop that enables the network to deliver a larger amount of data to the sink thanks to its transmission control policy.

This result was expected due to the presence of lossy links and to the absence of retransmission mechanisms, as anticipated in sections 3.2 and 5.3. Random routing performs even worse than regular multi-hop, due to the simplicity of its parent selection function which does not always select the best node over the path to the sink. However, this situation changes as network ages, especially in the case of a more stressful workload. Looking at Figures 7.11.(b) and 7.12.(b) it is worth noting that after 127 days, corresponding to 16 failures, reliable multi-hop is outperformed by regular multi-hop, and, after 162 days (19 failures), it is outperformed even by random routing. In other terms, as the network ages, and the number of alive sensors starts to decrease, it is more resilient to send packets randomly, rather than consuming resources for reliable transmission (the rate of retransmitted packets increases due to the worse quality of available paths). This effect can also be observed in figure 7.13.(a): the overhead induced by reliable multi-hop increases over time. This suggests the adoption of adaptive routing strategies, able to change the adopted routing policy according to the current resiliency level.

### 7.4.4 Results from High Failure Rate Scenario

All mentioned observations are mitigated when the WSN manifests higher failure rates (Experiment 3). In this case, the connection resiliency (Figures 7.9.(c) and 7.10.(c)) does not help for selecting the best routing algorithm, since hardware faults become the predominant cause of node failures, and the different behavior of routing algorithms does not impact on the failure dynamics. On the other hand, data delivery resiliency (Figures 7.11.(c) and 7.12.(c)) is still useful to observe significant differences, suggesting that i) redundant (flooding and gossiping) and reliable algorithms are the best choice to face the higher number of failures, as expected, and ii) the difference between redundant and reliable algorithms is not as relevant as in the former experiments, hence a reasonable choice is to select flooding or gossiping, since they are simpler to implement.

Table 7.7: Nodes with the shortest evaluated lifetime, for each experiment and routing algorithm

| | Flooding | | Gossiping | | Multi-hop | | Reliable multi-hop | | Random | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Exp** | **node ID** | **lifetime days** | **node ID** | **lifetime days** | **node ID** | **lifetime days** | **node ID** | **lifetime days** | **node ID** | **lifetime days** |
| | 12 | 120,7 | 12 | 119,6 | 21 | 132,6 | 11 | 129,2 | 20 | 136,7 |
| | 6 | 123,8 | 6 | 124,4 | 23 | 136,8 | 22 | 131,2 | 22 | 138,1 |
| **1** | 4 | 124,7 | 7 | 124,7 | 29 | 137,0 | 24 | 132,0 | 28 | 138,2 |
| | 7 | 125,1 | 4 | 124,9 | 17 | 137,6 | 1 | 132,6 | 13 | 138,8 |
| | 15 | 125,6 | 13 | 124,9 | 4 | 137,9 | 7 | 133,6 | 15 | 138,8 |
| | 13 | 126,6 | 10 | 126,6 | 25 | 138,4 | 4 | 134,2 | 1 | 139,7 |
| | 12 | 4,9 | 12 | 4,9 | 4 | 59,2 | 16 | 68,1 | 29 | 41,4 |
| | 6 | 5,1 | 13 | 5,1 | 19 | 70,4 | 15 | 73,3 | 27 | 95,4 |
| **2** | 13 | 5,1 | 14 | 5,2 | 11 | 71,7 | 14 | 86,9 | 15 | 97,5 |
| | 14 | 5,3 | 6 | 5,3 | 7 | 80,7 | 6 | 88,5 | 3 | 98,8 |
| | 3 | 5,3 | 3 | 5,4 | 12 | 91,1 | 5 | 97,0 | 18 | 100,1 |
| | 4 | 5,4 | 4 | 5,4 | 15 | 96,0 | 4 | 104,8 | 25 | 102,9 |
| | 29 | 21,6 | 19 | 22,3 | 5 | 23,0 | 1 | 22,5 | 26 | 23,0 |
| | 6 | 22,8 | 8 | 22,4 | 7 | 23,3 | 5 | 23,5 | 13 | 23,2 |
| **3** | 22 | 23,2 | 1 | 22,9 | 27 | 23,3 | 25 | 23,8 | 9 | 23,3 |
| | 20 | 23,4 | 10 | 23,3 | 21 | 23,5 | 26 | 24,0 | 2 | 23,4 |
| | 8 | 23,5 | 20 | 23,5 | 29 | 23,5 | 27 | 24,0 | 14 | 23,5 |
| | 9 | 23,7 | 23 | 23,6 | 11 | 23,8 | 18 | 24,1 | 19 | 23,6 |

### 7.4.5 Discussion

Table 7.7 reports lifetime results. In particular, the table lists the nodes with the shortest evaluated lifetime for each experiment and routing algorithm. This is useful to highlight dependability bottlenecks and to gain more insight on the failure behavior. For instance, in experiments 1 and 2 flooding and gossiping tend to discharge *inner* nodes first, e.g., nodes number 12 and 6 which are deployed on the suspended roadway and which are forwarders to the sink for the other nodes. This causes network disconnection earlier than other algorithms. In particular, concerning experiment 2, the estimated lifetime of the first failing nodes is about 5 days, against 61 days of average network lifetime (see Table 7.8). Hence, after only 5 days, the majority of nodes result isolated, which justifies the collapse observable in Figure 7.11.(b) and 7.12.(b). This effect is not observed for the remaining algorithms. For instance, reliable multi-hop tend to stress the nodes which are closer to the sink (e.g., nodes 16, 15 and 14). In other terms, the more a node is close to the sink, the more traffic and retransmissions it has to manage, the faster it will discharge its battery. At the same time, both data and connection resiliency are better than in the previous case, since: i) the most stressed nodes for reliable multi-hop exhibit a lifetime of one order magnitude greater than the one estimated for flooding and gossiping (e.g. 68 days for node 16 with reliable multi-hop against 5 days for node 12 with flooding), and ii) being closer to the sink, these nodes have a larger set of redundant paths to the sink. More detailed results can be achieved by looking at the traces of the experiments. For instance, we are able to evaluate how many transient failures occur on each node, when they occur, and what consequences they imply, e.g., in terms of lost packets. Also, we can study the path followed by each packet flowing in the network, assessing if it reaches the sink, if (and where) it is lost and why (e.g., due to a node failure, a link failure, or a TTL expiration), and if it is retransmitted.

Table 7.8: Summary of experiments

| Routing Algorithm | Exp | Connection | | Data Delivery | | Average node lifetime (days) | Average isolation time (days) (%lifetime) | Simulation time (seconds per batch) |
|---|---|---|---|---|---|---|---|---|
| | | Days to 75% | #Failues to 75% | Days to 75% | #Failues to 75% | | | |
| Multihop | 1 | 87.3 | 13 | always less than 37% | | 141.5 | 13.6 (9.6%) | 122 |
| | 2 | 41 | 9 | always less than 38% | | 129.8 | 33.5 (25.8%) | 187 |
| | 3 | 9.8 | 14 | always less than 37% | | 24.8 | 7.56 (30.5%) | 35 |
| Reliable mhop | 1 | 84 | 13 | 134 | 16 | 140.5 | 15 (10.7%) | 200 |
| | 2 | 76.3 | 13 | 88 | 11 | 122.6 | 29 (23.7%) | 387 |
| | 3 | 9.5 | 14 | 16 | 18 | 24.5 | 7.2 (29.4%) | 48 |
| Random | 1 | 87 | 12 | always less than 22% | | 141 | 14 (10.1%) | 71 |
| | 2 | 87.5 | 17 | always less than 24% | | 137.5 | 13.3 (9.6%) | 499 |
| | 3 | 9.5 | 14 | always less than 21% | | 24.5 | 7.7 (31.4%) | 21 |
| Flooding | 1 | 91 | 12 | 100 | 13 | 134.7 | 12.7 (9.4%) | 576 |
| | 2 | 5.7 | 2 | 5.7 | 2 | 61.6 | 36.4 (59.1%) | 1385 |
| | 3 | 9.8 | 14 | 11 | 13 | 25.3 | 7.2 (28.4%) | 144 |
| Gossiping | 1 | 90.1 | 13 | 101 | 13 | 134.7 | 13.5 (10.0%) | 511 |
| | 2 | 5.8 | 2 | 5.9 | 2 | 61.7 | 36.3 (58.8%) | 1192 |
| | 3 | 10.6 | 14 | 12.8 | 14 | 25.4 | 6.6 (26.0%) | 132 |

Table 7.8 reports a summary of the experimental results achievable with the proposed approach. In particular the Table reports the connection and data delivery resiliency obtained for each experiment in terms of the days and number of failures after which the resiliency becomes lower than 75% In addition the table includes the average node lifetime, the average time spent in isolation by the nodes, and the simulation time needed to perform one batch of simulation for each experiment.

The table is useful to have a quick understanding of the results. For instance, it is clear that the random routing exhibits the best performance in terms of connection resiliency, especially for experiment 2. This is confirmed by lifetime and isolation results: nodes equipped with random routing usually live longer (e.g., 137.4 days for experiment 2 against 122.6 days of reliable multi-hop) and are connected to the sink (e.g., only 13.3 days of isolation for experiment 2, against 29 days of reliable multi-hop). On the other hand, nodes are not able to deliver their packets to the sink, since data delivery resiliency equals 0.24 at most. This again confirms that data delivery resiliency is very useful to countermeasure traditional metrics, such as connection resiliency and nodes lifetime. Reliable multi-hop appears to be best trade-off in terms of connection and data delivery resiliency results, and it also leads to lifetime and isolation results which are comparable to regular multi-hop. As for flooding and gossiping, it is interesting to look at the results obtained for experiment 2. After only 2 failures, and after about 6 days of operation, both connection and data delivery resiliency becomes lower than 75%. This is consistent with the short nodes lifetime (i.e., about 62 days) and with the long isolation time: nodes spend about 60% of their time, on average, being isolated from the sink. This is due to the fact that flooding and gossiping tend to discharge inner nodes first, leaving a large set of nodes isolated from the sink after only few days of operation.

The simulation time for each experiment is reported to give evidence of the practical feasibility of the approach. In our settings the time needed for one batch is less than 200 seconds in most of the cases. From the table is however evident that the time needed changes depending on the complexity of the particular experiment. For instance, more time is needed for experiment 2, which implies more events to be processed (due to the more stressful workload). On the opposite, experiment 3 require less time, due to the higher failure rate which shorten nodes lifetime. Also, the simulation of flooding and gossiping requires more time due to the larger amount of packets to be simulated.

Chapter 8

# Conclusions

This thesis addressed the problem of the resiliency assessment of WSN. Assessing the resiliency of WSNs is a crucial task in designing dependable WSNs, since it could help to i) anticipate critical choices e.g., concerning node placement, running software, routing and MAC protocols, ii) mitigate risks, e.g., by forecasting the time when the WSN will not be able to perform with a suitable level of resiliency, and iii) prevent money loss, e.g., providing a criteria to plan and schedule maintenance actions effectively. Nowadays, the lack of effective approaches for the resiliency assessment of WSNs is the major cause of distrust of industries that are still questioning the adoption of WSN in critical application scenario, despite WSNs represent a good opportunity to reduce the installation and maintenance cost of more than one order of magnitude. However, in order to assess the resiliency of WSNs, two main challenges must be overcome.

1. Definition of Resiliency. Resiliency has been defined as *the persistence of dependability when facing "changes"*[11], and past research efforts have been devoted to define the concept of connection (or network) resiliency, i.e., the number of "changes", in terms of node failures, that can be accommodated while preserving a specific degree of connectivity in the network. However, while this concept still applies to WSNs, it is not enough to characterize the data-driven nature of WSNs. The service delivered by the WSN does not encompass only the connection, but also the computation, i.e., even when sensor nodes are potentially connected ( a path exists between nodes and the sink node), data losses can still occur.

2. Complexity of the assessment. It is easy to figure that, even assessing only the connection resiliency of WSN is dramatically exacerbated by the complexity of potential changes that may take place at runtime. The workload, included the use of aggregation/fusion algorithms, impacts on the number of packets sent on the network. The path followed by packets depends on the routing algorithm, on the topology, and on the wireless medium (packets can be lost). The energy profile is affected by the workload, by the number of forwarded packets, and by the battery technology. All above factors impact on the failure behavior, e.g., a node can fail due to battery exhaustion. A node can also fail independently, due to faults in the sensing hardware. In turn, a failure of a node may induce a partition of the network into two or more subsets, involving a large set of nodes to be unavailable, i.e., isolated, hence, unable to send acquired data to the sink.

> Clearly, such high degree of inter-dependence complicates the assessment task, by dramatically increasing the number of variables and dynamics to encompass. Hence, important questions to answer are : i) how the node workload, hardware platforms, topology and routing protocols impact on the failure proneness of nodes and of the network, and, vice-versa, ii) how node and network failures impacts on the nominal behavior of the WSN (e.g., how the failure of a node mutates the behavior of running workload or routing protocols.

This thesis defined the concept of data delivery resiliency and qualifies the concept of WSN resiliency as a non functional properties composed by both connection resiliency and data delivery resiliency. Data delivery resiliency is defined as the number of failures (or the longer time interval) the WSN can sustain (a WSN can survive) while delivering an amount of data to the sink node greater than a threshold. The concept of data delivery resiliency relates to i) the computational load on nodes which may causes packet losses due to buffer overrun, ii) application requirements, e.g. at least a given amount of produced measurements must be delivered to the sink node iii) routing and MAC protocols impacting on the data delivery features and packet error rate and iv) radio interferences and packet loss/corruption phenomena on the propagation medium. The variation in the amount of useful data received by the sink due to disconnection failures that can be tolerated by the WSN depends on the requirements of the application.

In order to assess the resiliency mastering the intrinsic complexity, this thesis proposed a novel and holistic approach for the resiliency assessment of WSNs. Key focus of the proposed approach is the holistic assessment, i.e., the comprehensive assessment performed by taking into account all subsystems and inter-related factors concurring to the behavior of the WSN.

To master the assessment complexity, the approach separates the assessment of the failure behavior from the evaluation of the nominal behavior by considering i) a set of parametric analytical failure models, and ii) a WSN behavioral model, respectively. The behavioral model is exploited to configure the WSNs in terms of hardware platform, topology, routing and MAC protocols, and to study the nominal behavior of the software, included the OS, and the power consumption of the nodes. Evaluations performed with the behavioral models are used to gather values for failure model parameters of the WSN under study, such as the packet forwarding rate of each node. Then the power of the analytical failure model is exploited to evaluate a set of metrics of interest such as the resiliency. The approach delegates the effort of re-computing dynamic parameters which vary during the evaluation of the failure model, to an external component, namely the External Engine which orchestrates the evolution of the failure model. The external engine can be regarded as a supervision entity encapsulating and managing aspects that are generally difficult or onerous to express at the level of abstraction of analytical models, such as routing protocols and topology.

Another goal obtained following the proposed approach is the automated failure model generation. By decoupling the analytical models from "changes" management issues, the External Engine allows to simplify the failure model which can adapt to each manifesting change, transparently, e.g., by invoking facilities provided by the External Engine. The extreme simplification obtained, allowed the implementation of a *Failure Model Template Library* which is used to generate the final WSN failure model consistently with the WSN observed in the behavioral simulation, thus avoiding manual modeling phases when changing the structure or the features of the WSN under study.

Relying on an automated modeling phase, the proposed approach allows final users (i.e., WSN developers) to work within their knowledge domain, without requiring specific modeling and/or programming skills. In other terms, developers interact with artifacts that are related to their domain, such as behavioral simulators.

Other than providing specific considerations on the presented case studies, this chapter summarizes the general lessons which have been learned and that can be reasonably taken into account when assessing the resiliency of WSN or developing resilient WSN.

### 8.0.6 Lessons Learned

The approach has been experimented considering a set of realistic case studies, relating to different hardware platform, routing protocols, topology and applications. A Quantitative evaluation concerning MTTF, lifetime, network overhead, data delivery resiliency and connection resiliency (both against the time and the number of manifested failures), has been provided. Performed simulations shown how the approach can be used to pinpoint critical nodes. Moreover, achieved results also shown to be useful to i) guide possible resiliency improvement strategies, for instance to quantitatively and judiciously configure the topology and software of nodes in terms of duty-cycle, size of radio packet and adopted routing algorithm, and ii) to evaluate possible cost-dependability trade-offs. In particular, we demonstrate that, concerning a simple in line topology, we obtained an overall MTTF improvement (e.g., 21% for the last node in the topology for a given setting), by increasing the cost of 30% (three more nodes in the network).

Achieved results allowed us also to justify the need of complementing the concept of connection resiliency with the concept of data delivery resiliency. Experiments showed that connection resiliency can be misleading for characterizing the overall resiliency level of a WSNs and that data delivery resiliency is very useful to countermeasure traditional metrics, such as connection resiliency and nodes lifetime, adding one further element to discriminate between different solutions. For instance, experiments demonstrated that the random routing exhibits the best performance in terms of connection resiliency, especially for stressing workload. This is confirmed by lifetime and isolation results: nodes equipped with random routing usually live longer and are longer connected to the sink wasting a short time as isolated. However, it is not able to deliver a sufficient amount of data to the sink showing a data delivery resiliency equals 0.24 at most.

Interesting results come also across the evaluation of the sensitivity of the resiliency against varying failure rates and routing algorithms. Results show that in the case of high failure rate, the resiliency of a WSN is not sensitive to workload parameters and routing algorithms, which on the other side became a tricky task for low failure rates. In the case of low failure rate, and low workload, epidemic and reliable routing protocols are able to deliver the same level of connection and data delivery resiliency, while in the case of low failure rate and high workload the use of acknowledgment-based multi-hop is the best choice for preserving acceptable levels in data delivery and connection resiliency in a large time horizon, since epidemic algorithms tend to discharge inner nodes first, leaving a large set of nodes isolated from the sink after only few days of operation.

However, WSN aging phenomena and network degradation suggest that, from a given instant of time (or after a given number of manifested failures ), random walk routing represents a good backup solution to reliable multi-hop, enabling a longer lifetime and higher network resiliency due to its null overhead and uniform node discharge features. Consequently, performed analysis showed that the WSN resiliency may benefit of a routing policy switching at a given point.

This thesis demonstrated that in order to assess the resiliency of a complex system, a holistic assessment strategy is needed. Moreover, the proposed approach poses also an intriguing challenge for interested industries in the field of (and not limited to) sensor networks, consisting in the chance of releasing failure model libraries upon the release of WSN hardware, following the same approach as for HDL libraries. This way, following the presented approach, the resiliency assessment may play a central role in making the vision of the successful and trustworthy adoption of WSNs in critical scenarios come true.

# Appendix A

# Appendix A

## A.1 Basic Notion of Dependability and Resiliency

In [141] computer system dependability was defined as " *the quality of the delivered service such that reliance can justifiably be placed on this service*". This notion has evolved over the years. Recent effort from the same community defines the dependability as "*the ability to avoid service failures that are more frequent and more severe than is acceptable*" [142]. This last definition has been introduced since it does not stress the need for justification of reliance.

The dependability is a composed non-functional attribute, that encompasses the following sub-attributes:

- **Availability**: readiness for correct service. A system is said to be available at a the time t if it is able to provide a correct service at that instant of time. The availability can thus be thought as the expected value E(A(t)) of the following A(t) function:

$$A(t) = \begin{cases} 1 & \text{if proper service at } t \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

In other terms, the availability is the fraction of time that the system is operational;

- **Reliability**: continuity of correct service. The reliability R(t) of a system is the conditional probability of delivering a correct service in the interval [0; t], given that the service was correct at the reference time 0 [RIF]:

  $$R(0;t) = P\,(\text{no failures in } [0;t] \mid \text{correct service in } 0) \tag{A.2}$$

  Since reliability is a function of the mission duration T, mean time to failure (MTTF) is often used as a single numeric indicator of system reliability [RIF 82]. In particular, the time to failure (TTF) of a system is defined as the interval of time between a system recovery and the consecutive failure.

- **Safety**: absence of catastrophic consequences on the user(s) and the environment;

- **Confidentiality**: absence of improper system alterations;

- **Maintainability**: ability of a system to be easily repaired after the occurrence of a failure. A commonly adopted indicator for the maintainability is the mean time to recover (MTTR). In particular, the time to recover (TTR) can be defined as the time needed to perform a repair, that is, the interval of time between a failure and its consequent recovery.

Recently, the notion of dependability is shifting more and more to the concept of *resiliency* [11]: the persistence of dependability when facing "changes". This change of perspective leads to new requirements of modern fault tolerant systems, such as the ability of accommodating unforeseen environmental perturbations or disturbances. At same time, the concept of resiliency dictates the need for new evaluation approaches, since, resiliency assessment approaches now must deal with i) dependability assessment when ii) changes manifests in driving and environmental variables. However, as will be detailed later in Chapter 2, still a little attention is devoted in this new perspective due to limitation of past approaches.

### A.1.1 Threats

The causes that lead a system to deliver an incorrect service, i.e., a service deviating from its function, are manifold and can manifest at any phase of its life-cycle. Hardware faults and design errors are just an example of the possible sources of failure. These causes, along with the manifestation of incorrect service, are recognized in the literature as dependability threats, and are commonly categorized as failures, errors, and faults [142].

A failure is an event that occurs when the delivered service deviates from correct service. A service fails either because it does not comply with the functional specification, or because this specification did not adequately describe the system function. A service failure is a transition from correct service to incorrect service. The period of delivery of incorrect service is referred as service outage. The transition from incorrect service to correct service is a service recovery or repair action.

An error can be regarded as the part of a system's total state that may lead to a failure. In other words, a failure occurs when the error causes the delivered service to deviate from correct service. The adjudged or hypothesized cause of an error is called a fault.

Failures, errors, and faults are related each other in the form of a chain of threats. A fault is active when it produces an error; otherwise, it is dormant. A failure occurs when an error is propagated to the service interface and causes the service delivered by the system to deviate from correct service. An error which does not lead the system to failure is said to be a latent error. A failure of a system component causes an internal fault of the system that contains such a component, or causes an external fault for the other system(s) that receive service from the given system.

# Bibliography

[1] Philip Levis and David Culler. The firecracker protocol. *Proceedings of the 11th ACM SIGOPS European Workshop*, September 2004.

[2] Lin Chih-Yu, Pen Wen-Chih, and Tseng Yu-Chee. Efficient in-network moving object tracking in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(8):1044–1056, 2006.

[3] Songhwai Oh, Luca Schenato, and Shankar Sastry. A hierarchical multiple-target tracking algorithm for sensor networks. *Proc. of the International Conference on Robotics and Automation, Barcelona, Spain, April 2005.*, April 2004.

[4] Tian He, Sudha Krishnamurthy, Liqian Luo, Ting Yan, Lin Gu, Radu Stoleru, Gang Zhou, Qing Cao, Pascal Vicaire, John A. Stankovic, Tarek F. Abdelzaher, Jonathan Hui, and Bruce Krogh. Vigilnet: An integrated sensor network system for energy-efficient surveillance. *ACM Transaction on Sensor Networks*, 2(1):1–38, 2006.

[5] Leon Evers and Paul Havinga. Supply chain management automation using wireless sensor networks. *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, 0:1–3, 2007.

[6] Matteo Ceriotti, Luca Mottola, Gian Pietro Picco, Amy L. Murphy, Stefan Guna, Michele Corra, Matteo Pozzi, Daniele Zonta, and Paolo Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *IPSN '09: Proceedings of the 2009 International Conference on*

*Information Processing in Sensor Networks*, pages 277–288, Washington, DC, USA, 2009. IEEE Computer Society.

[7] N. Xu, S. Rangwala, K.K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A Wireless Sensor Network For Structural Monitoring. *Proc. of the 2th ACM Conference on Embedded Networked Sensor Systems (SenSys '02)*, November 2004.

[8] M. Cinque, D. Cotroneo, G. De Caro, and M. Pelella. Reliability requirements of wireless sensor networks for dynamic structural monitoring. *International Conference on Dependable Systems and Networks(DSN)*, 2006.

[9] Marcello Cinque, Catello Di Martino, and Alessandro Testa. icaas: interoperable and configurable architecture for accessing sensor networks. In *ADAMUS 09: Proceedings of the 3rd international workshop on Adaptive and dependable mobile ubiquitous systems*, pages 19–24, New York, NY, USA, 2009. ACM.

[10] Marcello Cinque, Catello Di Martino, and Alessandro Testa. icaas: interoperable and configurable architecture for accessing sensor networks. In *To appear in: International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS), Vol 1, issue 2*, New York, NY, USA, 2009. ACM.

[11] Jean Claude Laprie. From dependability to resilience. *38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks, Anchorage, Alaska, June 2008, Sup. Vol., pp. G8-G9*, 2008.

[12] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys '04)*, pages 214–226, 2004.

[13] Walid Najjar and Jean-Luc Gaudiot. Network resilience: A measure of network fault tolerance. *IEEE Trans. Comput.*, 39(2):174–181, 1990.

[14] T. Dimitar, F. Sonja, M. Jani, and G. Aksenti. Connection resilience to nodes failures in ad hoc networks. In *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, volume 2, pages 579–582 Vol.2, May 2004.

[15] S. Mccanne, S. Floyd, and K. Fall. ns2 (network simulator 2). http://www-nrg.ee.lbl.gov/ns/.

[16] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[17] M. Cinque, D. Cotroneo, C. Di Martino, and S. Russo. Modeling and assessing the dependability of wireless sensor networks. *proc. of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS'07)*, 1:33–42, October 2007.

[18] Marcello Cinque, Domenico Cotroneo, Catello Di Martino, and Stefano Russo. ependability evaluation of wireless networks: a hybrid simulation tool. *Supplemental Volume of the 37th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '07)*, pages 428–429, Edinburgh, UK, giugno 2007.

[19] W. H. Sanders and J. F. Meyer. A unified approach for specifying measures of performance, dependability, and performability. *Dependable Computing and Fault-Tolerant Systems: Dependable Computing for Critical Applications*, 4:215–237, 1991.

[20] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster. The Möbius framework and its implementation. *IEEE Transactions on Software Engineering*, 28(10):956–969, 2002.

[21] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, March 2002.

[22] Kirk Martinez, Jane K. Hart, and Royan Ong. Environmental sensor networks. *Computer*, 37(8):50–56, 2004.

[23] J. Hill and D. Culler. MICA: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro*, 22(6):12–24, 2002.

[24] Jason Hill. A software architecture supporting networked sensors. *UC Berkeley Masters Thesis*, December 2000.

[25] B. Warneke, M. Last, B. Liebowitz, and K.S.J. Pister. Smart dust: communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51, Jan 2001.

[26] G. Asada, T. Dong, F. Lin, G. Pottie, W. Kaiser, and H. Marcy. Wireless integrated network sensors: Low power systems on a chip. *European Solid State Circuits Conference*, 1998.

[27] Jun-Hong Cui, Jiejun Kong, M. Gerla, and Shengli Zhou. The challenges of building mobile underwater wireless networks for aquatic applications. *Network, IEEE*, 20(3):12–18, May-June 2006.

[28] Zhao Cheng, Mark Perillo, and Wendi B. Heinzelman. General network lifetime and cost models for evaluating sensor network deployment strategies. *IEEE Transactions on Mobile Computing*, 7(4):484–497, 2008.

[29] Commission of the european communities. communication from the commission to the council and the european parliament: critical infrastructure protection in the fight against terrorism, com (2004) 702 final, brussels, 20 October 2004.

[30] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, Aug 2002.

[31] Stephen J. Walsh Daniel J. Weiss. Remote sensing of mountain environments. *Geography Compass ,Volumr 3, n.1 pages 1-21*, November 2009.

[32] Commissioned by European Commission for the FP6. Study on economic impact of e-health in ambient intelligence.

[33] Ren-Guey Lee, Kuei-Chien Chen, Chun-Chieh Hsiao, and Chwan-Lu Tseng. A mobile care system with alert mechanism. *Information Technology in Biomedicine, IEEE Transactions on*, 11:507–517, 2007.

[34] Daniel Minder, Pedro José Marrón, Andreas Lachenmann, and Kurt Rothermel. Experimental construction of a meeting model for smart office environments. In *Proceedings of the First Workshop on Real-World Wireless Sensor Networks (REALWSN 2005), SICS Technical Report T2005:09*, June 2005.

[35] Sheikh I. Ahamed, Mohammad Zulkernine, and Suresh Anamanamuri. A dependable device discovery approach for pervasive computing middleware. In *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*, pages 66–73, Washington, DC, USA, 2006. IEEE Computer Society.

[36] F. Zhu, M.W. Mutka, and L.M. Ni. Service discovery in pervasive computing environments. *Pervasive Computing, IEEE*, 4(4):81–90, Oct.-Dec. 2005.

[37] Jurgen Bohn, Felix Gartner, and Harald Vogt. Dependability issues of pervasive computing in a healthcare environment. *Proceedings of the First International Conference on Security in Pervasive Computing, volume 2082 of Lecture Notes in Computer Science*, pages 53–70, 2003.

[38] Guang-Zhong Yang. *Body Sensor Networks.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[39] J. L. Hill and D. E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.

[40] Katerina Goseva-Popstojanova and Kishor S. Trivedi. Stochastic modeling formalisms for dependability, performance and performability. In *Performance Evaluation: Origins and Directions*, pages 403–422, London, UK, 2000. Springer-Verlag.

[41] M. Rausand and A. Hoyland. *System Reliability Theory: Models, Statistical Methods, and Applications, 3rd edition.* Wiley-IEEE, New York, 2003.

[42] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. *Fault Tree Handbook.* Nuclear Regulatory Commission, NUREG-0492, Washington DC, U. S., 1981.

[43] W. S. Lee, D. L. Grosh, and F. A. Tillman. Fault tree analysis, methods, and applications - a review. *Reliability, IEEE Transactions on*, R-34(1):194–203, Aug. 1985.

[44] Kishar Shridharbhai Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1982.

[45] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[46] J. F. Meyer, A. M., and W. H. Sanders. Stochastic activity networks: Structure, behavior, and application. *International Workshop on Timed Petri Nets*, pages 106–115, 1985.

[47] R. A. Sahner, K. S. Trivedi, and A. Puliafito. Kluwer Academic Publishers, Norwell, MA, USA, 1996.

[48] Kishor S. Trivedi. *Probability and statistics with reliability, queuing and computer science applications.* John Wiley and Sons Ltd., Chichester, UK, 2002.

[49] Kishor S. Trivedi. *Probabilistic Reliability: An Engineering Approach.* 2nd Edition. R. E. Krieger Publishing Co., John Wiley and Sons Ltd., Malabar, FL, 1990.

[50] Relex. Reliability prediction. http://www.relexsoftware.com/products/relanalysissoft.asp, 2002.

[51] M. Malhotra and K.S. Trivedi. Power-hierarchy of dependability-model types. *Reliability, IEEE Transactions on*, 43(3):493–502, Sep 1994.

[52] J. E. Arsenault and J. A. Roberts. *eliability and Maintainability of Electronic Systems.* Computer Science Press, Rockville, MD, 1980.

[53] R. E. Barlow and F. Proschan. *Statistical Theory of Reliability and Life Testing.* Holt, Rinehart and Winston, New York., 1975.

[54] B. S. Dhillon and C. Singh. *Engineering Reliability: New Techniques and Applications.* Wiley, New York., 1981.

[55] E. Henley and H. Kumamoto. *Reliability Engineering and Risk Assessment.* Prentice-Hall, Englewood Cliffs, NJ,, 1918.

[56] Nancy G. Leveson. *Safeware: system safety and computers.* ACM, New York, NY, USA, 1995.

[57] Jean Arlat, Karama Kanoun, and Jean-Claude Laprie. Dependability modeling and evaluation of software fault-tolerant systems. *IEEE Trans. Comput.*, 39(4):504–513, 1990.

[58] Mark Allen Boyd. *Dynamic fault tree models: techniques for analysis of advanced fault tolerant computer systems.* PhD thesis, Durham, NC, USA, 1992.

[59] Jogesh K. Muppala, Kishor S. Trivedi, and Steven P. Woolet. Real-time systems performance in the presence of failures. *Computer*, 24(5):37–47, 1991.

[60] K.S. Trivedi, S. Ramani, and R. Fricks. Recent advances in modeling response-time distributions in real-time systems. *Proceedings of the IEEE*, 91(7):1023–1037, July 2003.

[61] Sachin Garg, Chandra M. Kintala, Shalini Yajnik, and Yennun Huang. Performance and reliability evaluation of passive replication schemes in application

level fault tolerance. In *FTCS '99: Proceedings of the Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, page 322, Washington, DC, USA, 1999. IEEE Computer Society.

[62] Jean-Claude Laprie and Karama Kanoun. X-ware reliability and availability modeling. *IEEE Trans. Softw. Eng.*, 18(2):130–147, 1992.

[63] Boudewijn R. Haverkort and Ignas G. Niemegeers. Performability modelling tools and techniques. *Perform. Eval.*, 25(1):17–40, 1996.

[64] Michael K. Molloy. Discrete time stochastic petri nets. *IEEE Trans. Softw. Eng.*, 11(4):417–423, 1985.

[65] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, 2(2):93–122, 1984.

[66] J. F. Meyer, A. Movaghar, and W. H. Sanders. Stochastic activity networks: Structure, behavior and application. In *International Workshop on Timed Petri Nets, Torino, Italy, Jul. 1-3, 1985*, pages 106–115. IEEE Computer Society Press, 1985.

[67] W. H. Sanders and J. F. Meyer. Metasan: a performability evaluation tool based on stochastic activity networks. In *ACM '86: Proceedings of 1986 ACM Fall joint computer conference*, pages 807–816, Los Alamitos, CA, USA, 1986. IEEE Computer Society Press.

[68] W. H. Sanders, W. D. Oball, II, M. A. Qureshi, and F. K. Widjanarko. The ultrasan modeling environment. *Perform. Eval.*, 24(1-2):89–115, 1995.

[69] Edward A. Ipser, David S. Wile, and Dean Jacobs. A multi-formalism specification environment. *SIGSOFT Softw. Eng. Notes*, 15(6):94–106, 1990.

[70] Herbert Praehofer and Bernard P. Zeigler. Modelling and simulation of non-homogeneous models. *EUROCAST '89: Selection of Papers from the International Workshop on Computer Aided Systems Theory*, pages 200–211, 1990.

[71] Paul A. Fishwick. Multimodeling as a unified modeling framework. *WSC '93: Proceedings of the 25th conference on Winter simulation*, pages 580–581, 1993.

[72] Marco Gribaudo and András Horváth. Fluid stochastic petri nets augmented with flush-out arcs: A transient analysis technique. *IEEE Trans. Softw. Eng.*, 28(10):944–955, 2002.

[73] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, P. Buonadonna, S. Burgess, D. Gay, W. Hong, T. Dawson, and D. Culler. A macroscope in the redwoods. *Third ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2005.

[74] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler. The effects of ranging noise on multihop localization: an empirical study. *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 10, 2005.

[75] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An empirical study of epidemic algorithms in large scale multihop wireless networks. *Intel Research, IRB-TR-02-003, Mar. 14, 2002*, 2002.

[76] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings (INFOCOM '02).*, 3, 2002.

[77] Mo Li and Yunhao Liu. Underground coal mine monitoring with wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(2):1–29, 2009.

[78] S. Pennington, T. Bauge, and B. Murray. Integrity-checking framework: An in-situ testing and validation framework for wireless sensor and actuator networks. In *SENSORCOMM '09: Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications*, pages 575–579, Washington, DC, USA, 2009. IEEE Computer Society.

[79] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient routing protocol for wireless microsensor networks. *In Proc. of Hawaii International Conference on System Sciences ,HICSS'00*, pages 1–2, 2000.

[80] A. Mini, B. Nath, and A. Loureiro. A probabilistic approach to predict the energy consumption in wireless sensor networks. *4'th Workshop de Comunicao sem Fio e Computao Mvel, So Paulo, Brazil, Oct. 2002.*, 2002.

[81] HMF AboElFotoh, SS Iyengar, and K. Chakrabarty. Computing reliability and message delay for Cooperative wireless distributed sensor networks subject to random failures. *Reliability, IEEE Transactions on*, 54(1):145–155, 2005.

[82] Isabel Dietrich and Falko Dressler. On the lifetime of wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(1):1–39, 2009.

[83] M. Achir and L. Ouvry. probabilistic model for energy estimation in wireless sensor networks. *In Proc. of first International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS '04), Turku, Finland*, 2002.

[84] F. Chiasserini and M. Garetto. Modeling the performance of wireless sensor networks. *In Proc. of the IEEE Conference on Computer Communications (Infocom '04)*, page 310, 2004.

[85] Jeffrey Stanford and Sutep Tongngam. Approximation algorithm for maximum lifetime in wireless sensor networks with data aggregation. In *SNPD-SAWN '06: Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pages 273–277, Washington, DC, USA, 2006. IEEE Computer Society.

[86] J. Lee, B. Krishnamachari, and C. J. Kuo. Impact of energy depletion and reliability on wireless sensor network connectivity. Technical report, departement of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564, USA, 2005.

[87] Joseph Rahmé and Khaldoun Al Agha. A state-based battery model for nodes' lifetime estimation in wireless sensor networks. In *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 337–338, New York, NY, USA, 2009. ACM.

[88] M. Marta and M. Cardei. Using sink mobility to increase wireless sensor networks lifetime. In *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, pages 1–10, June 2008.

[89] Li Liu, Bin Hu, Huifang Miao, Hao Li, Lian Li, and Qinglin Zhao. Achieving energy conservation, coverage and connectivity requirements in wireless sensor networks. In *ICDCSW '09: Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems Workshops*, pages 227–232, Washington, DC, USA, 2009. IEEE Computer Society.

[90] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli. On-line fault detection of sensor measurements. *Proceedings second IEEE international conference on sensors (Sensors '03)*, pages 974– 979 Vol.2, 2003.

[91] J. Chen, S. Kher, and A. Somani. Distributed fault detection of wireless sensor networks. *DIWANS '06: Proc. of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pages 65–72, 2006.

[92] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 251–254, New York, NY, USA, 2001. ACM.

[93] A. Shrestha, L. Xing, and H. Liu. Infrastructure communication reliability of wireless sensor networks. *International Symposium on Dependable, Autonomic and Secure Computing (DASC '06)*, 0:250–257, 2006.

[94] F. K. Shaikh, A. Khelil, and N. Suri. On modeling the reliability of data transport in wireless sensor networks. *The Fifteen Euromicro Conference on Parallel, Distributed and Network-based Processing*, 37(2), 2007.

[95] Hossein Pishro-Nik, Kevin Chan, and Faramarz Fekri. Connectivity properties of large-scale sensor networks. In *Journal of Wireless Networks*, volume 15, pages 945–964. Springer Netherlands, October 2009.

[96] A. Hoyland and M. Rausand. *System Reliability Theory: Models and Statistical Methods*. John Wiley and Sons, 1994.

[97] Yuh-Ren Tsai. Coverage-preserving routing protocols for randomly distributed wireless sensor networks. *Wireless Communications, IEEE Transactions on*, 6(4):1240–1245, April 2007.

[98] I. Saleh, H. El-Sayed, and M. Eltoweissy. A fault tolerance management framework for wireless sensor networks. *Journal of Communications*, 2(4), June 2007.

[99] K. Akkaya, M. Younis, and W. Youssef. Positioning of base stations in wireless sensor networks. *Communications Magazine, IEEE*, 45(4):96–102, April 2007.

[100] Ayman Z. Faza and Sahra Sedigh-Ali. A general purpose framework for wireless sensor network applications. In *COMPSAC '06: Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06),Chicago, September 17-21*, pages 356–358, Washington, DC, USA, 2006. IEEE Computer Society.

[101] Hüseyin Özgür Tan and Ibrahim Körpeoğlu. Power efficient data gathering and aggregation in wireless sensor networks. *SIGMOD Rec.*, 32(4):66–71, 2003.

[102] Ben L. Titzer, Daniel K. Lee, and Jens Palsberg. Avrora: scalable sensor network simulation with precise timing. page 67, 2005.

[103] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang. J-sim: A simulation environment for wireless sensor networks. *Simulation Symposium, Annual*, 0:175–187, 2005.

[104] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim: a library for parallel simulation of large-scale wireless networks. *SIGSIM Simul. Dig.*, 28(1):154–161, 1998.

[105] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. TinyOS: An operating system for wireless sensor networks. *Ambient Intelligence by W. Weber, J. Rabaey, and E. Aarts. 2005. Springer; 1 edition (April 19, 2005)*, 2005.

[106] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesc language: A holistic approach to networked embedded systems. In *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pages 1–11, New York, NY, USA, 2003. ACM.

[107] J. Polley, D. Blazakis, J. Mcgee, D. Rusk, and J. S. Baras. Atemu: a fine-grained sensor network simulator. *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 145–152, 2004.

[108] NAVAL RESEARCH LAB WASHINGTON DC I. T. Downard. Simulating sensor networks in ns-2. http://tcs.unige.ch/doku.php/algosensim, 2004.

[109] Lewis Girod, Jeremy Elson, Alberto Cerpa, Thanos Stathopoulos, Nithya Ramanathan, and Deborah Estrin. Emstar: a software environment for developing and deploying wireless sensor networks. 2004.

[110] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. Klein Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. Simulating

wireless and mobile networks in omnet++ the mixim vision. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[111] OMNeT++ Community. Discrete event simulation system. http://www.omnetpp.org.

[112] Sameer Sundresh, Wooyoung Kim, and Gul Agha. Sens: A sensor, environment and network simulator. *anss*, 00:221, 2004.

[113] Alexander Kröller, Dennis Pfisterer, Carsten Buschmann, Sándor P. Fekete, and Stefan Fischer. Shawn: A new approach to simulating wireless sensor networks. pages 117–124, 2005.

[114] TCSensor. The algosensim simulator. http://tcs.unige.ch/doku.php/algosensim, 2002.

[115] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325 – 349, 2005.

[116] Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18, number 4, 319-349, 1988.

[117] Joanna Kulik, Wendi Heinzelman, and Hari Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wirel. Netw.*, 8(2/3):169–185, 2002.

[118] R. Govindan C. Intanagonwiwat and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, pages 56–67, Boston, MA USA, 2000.

[119] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster. The Möbius framework and its implementation. *IEEE Transactions on Software Engineering*, 28(10):9–969, 2002.

[120] T. S. Rappaport. *Wireless Communications.* Prentice Hall, New York, 1996.

[121] C. Di Martino. Models for resiliency assessment of wireless sensor networks. Technical report - www.mobilab.unina.it, University of Naples Federico II, January 2009.

[122] Intel Corporation. New computing frontiers : the wireless vineyard. http://www.intel.com/labs/features/rs01031.htm., 2006.

[123] Berkeley University of California. New computing frontiers : the wireless vineyard. http://firebug.sourceforge.net/., 2006.

[124] R., J. Polastre, A. Mainwaing, and D. Culler. Lessons from a sensor network expedition. *European Workshop on Sensor Networks (EWSN)*, January 2004.

[125] Robert G. Sargent. Verification and validation of simulation models. In *WSC '98: Proceedings of the 30th conference on Winter simulation*, pages 121–130, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[126] Peter Kemper and Carsten Tepper. Automated trace analysis of discrete-event system models. *IEEE Transactions on Software Engineering*, 35(2):195–208, 2009.

[127] C.F. Chiasserini and R.R. Rao. A model for battery pulsed discharge with recovery effect. In *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, pages 636–639 vol.2, 1999.

[128] R. Rao, S. Vrudhula, and D.N. Rakhmatov. *Battery modeling for energy aware system design.* IEEE, 2003.

[129] Crossbow Technology inc. Mica2 aa battery pack service life test, 2005.

[130] Economou M. The merits and the limitations of reliability predictions. *In Proc. of the Annual Symposium on Reliability and Maintainability (RAMS '04*, pages 352–357, 2004.

[131] Lynch J.P. Wang Y. and Law K.H. Wireless sensing technologies for civil infrastructure monitoring and management. In *Proceedings of the 5th International Seminar for Safety of Infrastructures*, volume 1, pages 1–7, Seoul, Korea, September 14 2007.

[132] Shamim N. Pakzad, Sukun Kim, Gregory L Fenves, Steven D. Glaser, David E. Culler, and James W. Demmel. Multi-purpose wireless accelerometers for civil infrastructure monitoring. Technical report, University of California at Berkeley, Berkeley, Cal, USA, 2008.

[133] Sukun Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 254–263, April 2007.

[134] Guang zhu CHEN, Zhen cai ZHU, Gong bo ZHOU, Chun feng SHEN, and Yan jing SUN. Sensor deployment strategy for chain-type wireless underground mine sensor network. *Journal of China University of Mining and Technology*, 18(4):561 – 566, 2008.

[135] I. Jawhar, N. Mohamed, and K. Shuaib. A framework for pipeline infrastructure monitoring using wireless sensor networks. In *Wireless Telecommunications Symposium, 2007. WTS 2007*, pages 1–7, April 2007.

[136] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24, New York, NY, USA, 2004. ACM.

[137] K. Chintalapudi, T. Fu, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri. Monitoring civil structures with a wireless sensor network. *Internet Computing, IEEE*, 10(2):26–34, March-April 2006.

[138] K. Chintalapudi, J. Paek, O. Gnawali, T.S. Fu, K. Dantu, J. Caffrey, R. Govindan, E. Johnson, and S. Masri. Structural damage detection and localization using netshm. In *Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on*, pages 475–482, 0-0 2006.

[139] Frank Stajano. Smart infrastucture: Monitoring and assessment of civil engineering infrastructure, 2008.

[140] W. H. Sanders and L. M. Malhis. Dependability evaluation using composed SAN-based reward models. *Journal of Parallel and Distributed Computing 15*, pages 238–254, 1992.

[141] A. Avizienis and J.-C. Laprie. Dependable computing: From concepts to design diversity. *Proceedings of IEEE FTCS-15*, 74(5):629–638, May 1986.

[142] A.Avizienis, J.C. Laprie, B.Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. on Dependable and Secure Computing*, 1(1):11–33, January-March 2004 2004.