TU WIEN Informatics

Sensor Node Fault Detection in Wireless Sensor Networks

An Immune-inspired Approach

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dominik Widhalm, MSc
Registration Number 11831460

to the Faculty of Informatics

at the TU Wien

Advisor: Priv.-Doz. Mag. DI. DI. Dr. Karl M. Göschka
Second advisor: Ao.Univ.Prof. DI. Dr. Wolfgang Kastner

The dissertation has been reviewed by:

Prof. Andrea Bondavalli	Prof. Davide Quaglia

Vienna, 24th August, 2022

Dominik Widhalm

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Declaration of Authorship

Dominik Widhalm, MSc

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 24. August 2022

Dominik Widhalm

Acknowledgements

I dedicate this thesis to my grandparents, Adelheid and Alfred. They enabled me to have the best childhood one can imagine. Most of all, they supported me in learning basic skills and developing character traits that have significantly shaped my life. For that, I will be eternally grateful to them.

Similarly, I would like to thank my parents, Doris and Helmut, for their everlasting support. They allowed me to live my life freely and pursue my own way. But whenever I needed them, they were there for me and provided support, be it mentally or financially.

The way to a doctoral thesis and the work that goes with it, such as the underlying research, is far from being trivial. In addition to own high expectations and the constant pressure of time, disseminating findings at international conferences and scientific journals is a stressful factor, not least due to significant competition. Therefore, I am unspeakably grateful for the supervision of my two advisors, Karl M. Göschka and Wolfgang Kastner. They enabled me to pursue my ideas and supported me along the way with their expertise and professional experience, both in the good times when my plans worked out and the bad times, such as when papers got rejected. Looking back, I could not imagine better advisors than them.

This work has been supported by the Doctoral College Resilient Embedded Systems, which is run jointly by the TU Wien's Faculty of Informatics and the University of Applied Sciences Technikum Wien.

Abstract

Sensor node faults are a serious threat to wireless sensor networks. They can cause node crashes or lead to the transmission of corrupted data. Especially the latter endangers the quality of subsequent data analyses.

Most related fault detection approaches consider the sensor nodes as black boxes. They neglect vital information available on the node level. Consequently, most of these approaches can not distinguish between (i) irregular but correctly sensed data events and (ii) data corruption caused by soft faults.

In this thesis, we present a fault detection approach that integrates node-level diagnostics with the characteristics of the sensor data. We utilize this node-level diagnostic information to present our fault detection approach, which is inspired by the functioning of dendritic cells in the human immune system.

We used a tripartite experiment setup consisting of simulations, a lab setup, and a practical sensor network testbed to evaluate the correctness and efficiency of the developed approach. The results show that the approach offers a comparably high fault detection rate in combination with a negligibly low false alarm rate. Moreover, it can reliably differentiate between correct data events and fault-induced data anomalies. At the same time, it consumes a reasonably small overhead of resources, especially concerning the sensor node energy. Also, the approach is generally applicable and minimizes the need for parameter adjustment and optimization.

Kurzfassung

Fehler in Sensorknoten sind eine ernsthafte Bedrohung für drahtlose Sensornetzwerke. Sie können zum Absturz der Knoten oder zur Übertragung fehlerhafter Daten führen. Speziell letzteres gefährdet die Qualität nachfolgender Datenanalysen.

Die meisten bisherigen Fehlererkennungsansätze betrachten die Sensorknoten als Black-Boxes. Dabei vernachlässigen sie wichtige Informationen, die auf dem Knoten verfügbar sind. Folglich können die meisten dieser Ansätze nicht zwischen (i) unregelmäßigen, aber korrekt erfassten Datenereignissen und (ii) Datenverfälschungen aufgrund von Soft-Faults unterscheiden.

In dieser Arbeit stellen wir einen Fehlererkennungsansatz vor, der Diagnoseinformationen auf Knotenebene mit den Eigenschaften der gemessenen Sensordaten integriert. Wir verwenden diese diagnostischen Informationen auf Knotenebene zur Erkennung von Sensorknotenfehlern in unserem Ansatz, der von der Funktionsweise dendritischer Zellen im menschlichen Immunsystem inspiriert ist.

Für die Evaluierung der Korrektheit und Effizienz des entwickelten Fehlererkennungsansatzes wurde ein dreiteiliger Versuchsaufbau bestehend aus Simulationen, einem Laboraufbau und einem praktischen Sensor Testnetzwerk verwendet. Die Ergebnisse zeigen, dass der Ansatz eine vergleichsweise hohe Fehlererkennungsrate in Kombination mit einer vernachlässigbar geringen Fehlalarmrate bietet. Darüber hinaus kann dieser zuverlässig zwischen korrekten Datenereignissen und fehlerbedingten Datenanomalien unterscheiden. Gleichzeitig benötigt der Ansatz einen geringen Mehraufwand an Ressourcen, insbesondere betreffend der Sensorknotenenergie. Zudem ist der Ansatz allgemein anwendbar und minimiert die Notwendigkeit einer Parameteranpassung und -optimierung.

Contents

CHAPTER 1

Introduction

Our information society is hungry for data. We gather and analyze an ever-increasing amount of data captured from an expanding number of sources. This data is essential for a plethora of data services that provide us with insights into existing processes or predictions for future events, both being used in industry and academia. Examples include process automation, precision agriculture, or research that leverage the available data for event detection, trend prediction, process analysis, or decision support. The data services heavily depend on the input data's timely availability and fine-grained quality. Inaccurate or false data leads to erroneous information and can ultimately result in incorrect findings and/or wrong (counter-)actions.

In this context, wireless sensor networks (WSNs) have become an essential source of fine-grained data about phenomena or events. Today, they are used in a wide range of services. WSNs consist of wirelessly connected sensor nodes deployed in an area of interest to monitor physical quantities close to their source and, thus, providing data with a high level of detail. In most applications, the sensor nodes perform some (pre)processing of the measurements and forward the data to central services for further processing (i.e., cloud systems). In these central services, the data is eventually fed to statistical, machine learning, or other methods as part of the data services to extract information.

However, during the data analysis, data instances may deviate from an expected or previously learned "normal" behavior. The sensor data can have outliers, show suspicious behavior in the form of offsets and/or drifts, or differ from the data reported by other sensor nodes in the same neighborhood. As such anomalies can potentially reflect an event in the observed area, anomaly detection approaches are widely used to detect data events in WSN applications.

Nevertheless, not all anomalies are related to actual events in the monitored environment; they can also stem from faults in the data chain. Especially sensor node faults have been found to negatively impact the overall quality of the data reported by a WSN. Sensor

nodes are dedicated embedded systems with strictly limited resources that often prevent the use of well-established fault-tolerance concepts such as hardware and/or software redundancy. Most of all, the energy budget available on the sensor nodes is bounded since most nodes are battery-powered. They are expected to operate over long times without the possibility of battery recharging or replacement. Additionally, sensor nodes usually consist of low-cost components that are prone to experiencing faults when being operated under the unpredictable and uncontrollable conditions imposed by outdoor environments. Consequently, proper runtime measures are inevitable to ensure the correctness and accuracy of the data reported by the sensor nodes.

Over the years, numerous approaches and concepts to tackle the problem of fault detection in WSNs have been proposed that meet the requirements of sensor nodes. In this context, concepts inspired by the natural anomaly detection capabilities of the human immune system have gained broad interest from the research community. The immune system offers desirable properties for computing systems, such as its widely distributed and decentralized operation, its ability to perform temporal and spatial correlation, and its high detection rate with a low false alarm rate. Therefore, many researchers took inspiration from immune mechanisms when developing novel approaches for fault detection in resource-constrained systems such as WSNs. However, many of these approaches are bound to certain assumptions and limitations that hinder their generic applicability. Most importantly, most of these approaches suffer from an inability to distinguish between data events and fault-induced data distortion as the sensor nodes are commonly treated as black boxes. The only way to avoid such misinterpretation is to incorporate vital information available on the sensor node (i.e., node-level diagnostics).

This thesis presents a fault detection approach that took inspiration from the so-called danger theory. The danger theory is an immunological theory that highlights the role of dendritic cells in the human immune system. These dendritic cells perform a kind of contextual information fusion in the human body to detect circumstances that pose a threat to the host. Similarly, the presented approach bases its fault detection on contextual information gathered on the node level. This contextual information is acquired by combining statistical metrics of the sensor data with node-level diagnostic data. This immune-inspired fault detection approach improves the detectability of sensor node faults and enables the distinction between data events and fault-induced deviations.

The concept has been implemented on a self-developed sensor node platform incorporating several self-diagnostic capabilities. Its correctness and efficiency have been evaluated using a tripartite experiment setup consisting of (i) simulations using pre-recorded datasets, (ii) experiments in a controlled lab environment, and (iii) practical WSN deployments, both indoor and outdoor. The results show that the proposed approach can reliably detect node faults with high accuracy. It correctly differs between events in the monitored physical phenomenon and the effects of sensor node faults responsible for a high false alarm rate in most related fault detection approaches. The approach requires a reasonable small resource overhead concerning memory and energy overhead. As a result, it does not negatively influence the sensor nodes' operation while providing increased reliability.

1.1 Research Questions

The introduction of fault detection capabilities leveraging node-level diagnostics has only been sparsely discussed in previous literature. Especially immune-inspired techniques such as those based on the danger theory have hardly been applied for node fault detection.

The detection of erroneous data in WSNs is crucial as they can lead to incorrect models, wrong findings, or false (counter-) measures. Previous approaches mainly focused on analyzing the sensor data and neglected valuable information available on the sensor nodes. In this context, particularly the distinction of anomalies in the monitored physical phenomena (i.e., data events) from data distortion induced by hardware and/or software issues (i.e., node faults) is a non-trivial problem.

Consequently, for the adaption and adoption of such immune-inspired principles for the detection of node faults, several research questions (RQs) have to be answered. These RQs describe the course of the present dissertation research and highlight the key issues addressed. A graphical representation of the targeted RQs and their connection is depicted in Figure 1.1. The particular RQs and their intended artifacts are discussed in the following. A summary of the contributions of the present work concerning these RQs is provided in Section 9.1.

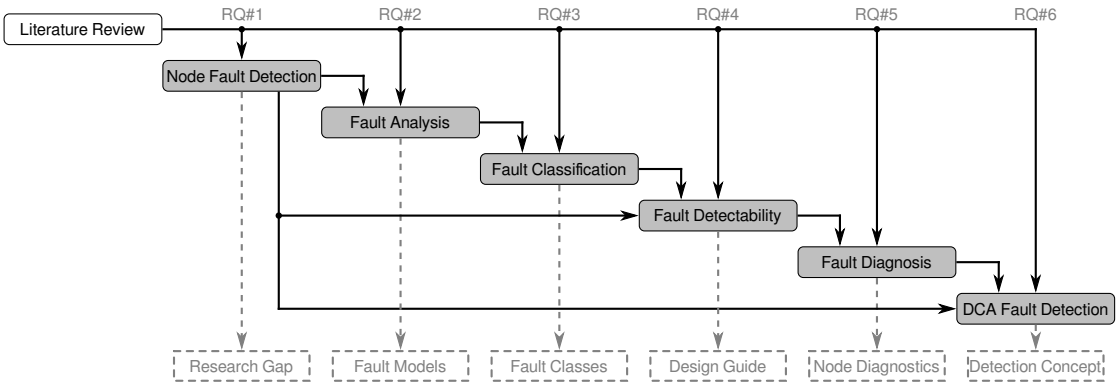


Figure 1.1: Dissertation research questions and dependencies

RQ#1: What are the limits of current node fault detection approaches?

Initially, the advantages and disadvantages of previously proposed fault detection approaches for WSNs were analyzed. In this context, the respective approaches' underlying assumptions, limitations, and characteristics were summarized. Additionally, previous works on fault classes and models were significant for the present work. Identifying the main drawbacks and limits of previous approaches was mandatory to highlight existing research gaps that formed the basis of the research presented in this thesis.

RQ#2: Which node faults are prevalent in WSNs?

The second question answered concerns the node faults commonly occurring in WSNs. Answering this question was mandatory for our fault detection approach to better understand the faults to be detected. In this context, especially the observability of the faults' manifestations was a crucial point to define the information necessary for their detection. The outcome of this RQ was an overview of node faults and served as a basis to model the prevalent faults (i.e., fault models).

RQ#3: How can these node faults be classified?

Answering this question included an analysis of possible root causes for the previously identified node faults and their characteristics. As a result of this, classification of faults depending on their manifestation on the node level was necessary, i.e., whether they (i) lead to a permanent or temporary shutdown of the node, (ii) cause an altered behavior observable by neighbor nodes, (iii) distort the measured value that will be further transmitted over the network, or (iv) offer no significant effect at all. Based on the findings from this RQ, the relevant fault classes were identified, and possible similarities in their root causes and/or manifestations were elaborated.

RQ#4: What effect has the sensor node's design on node faults?

Based on the findings of RQ#2 and RQ#3, we analyzed the influence of the sensor node's software and hardware architecture (i.e., its design) on the previously identified node faults. In this context, the impact of design choices on the probability and observability of sensor node faults was of utmost interest. In answering this question, considerations and guidelines for sensor node hardening were elaborated. This information helps to decrease the probability of node faults. In addition, unpreventable faults were identified, and a better understanding of the mechanisms behind these faults was gained. This information was crucial for defining the diagnostic data required as input for our fault detection approach.

RQ#5: Which data can be used for node-level fault detection?

Crucial for any fault detection approach is the definition of expressive input data and their proper mapping to the biological counterparts in the context of an immune-inspired approach. Previous works have revealed that especially the combination of different input data can improve the detection's correctness, that is, increasing the sensitivity while reducing the false alarm rate (FAR). In this context, the single input data's aggregation and possible weighting had to be examined.

RQ#6: How can immune mechanisms be applied to node fault detection?

Finally, the results of the previous RQs led to the development of our immune-inspired fault detection approach. As a starting point, the core functioning of the dendritic cell algorithm (DCA) (and its variants) was analyzed concerning its suitability for an

immune-inspired sensor node fault detection approach (with due regard to necessary modifications and adaptations). The approach had to be capable of reliably identifying sensor node faults. Thereby, it had to (i) have better overall accuracy than related approaches and (ii) meet the requirements of the sensor node regarding its resource overhead. Concerning the latter, the approach needed to be lightweight in terms of resource requirements and energy consumption. Thus, it had to have a low memory footprint, require moderate processing power, and pose a feasibly small energy overhead. In other words, the advantages of the resulting approach (i.e., enhanced reliability) had to justify possibly negative consequences such as a lowered battery life.

1.2 Methodology

The research presented in this thesis mainly follows the design science research methodology [1] to answer the aforementioned RQs. Design science is a research methodology that focuses on the development and validation of prescriptive knowledge. It is especially suitable for practical engineering problems as it provides guidelines for designing and evaluating solutions to given problems. For this purpose, it provides six activities guiding the decision making in design science research (cf. [1]):

1. Problem identification and motivation
2. Define the objectives for a solution
3. Design and development
4. Demonstration
5. Evaluation
6. Communication

The problem identification and motivation (activity 1) and the objectives for the envisioned solution (activity 2) were discussed in Section 1.1. Both activities are supported by systematic literature reviews (SLRs) [2] to acquire a profound knowledge of the state-of-the-art and related work in this field. Also the artifacts to be designed and developed in the research (activity 3) were described in Section 1.1 and are depicted in Figure 1.1 (dashed boxes at the bottom). Their contribution to the body of knowledge in the field is summarized in Section 1.3.

To demonstrate that the developed approach solves the stated problem (activity 4) and to evaluate the approach's correctness and efficiency (activity 5), a tripartite experiment setup is used that targets (i) theoretical aspects of the underlying fault detection (using simulations), (ii) fault-specific details of the node diagnostics (with directed lab experiments), and (iii) general properties of the developed approach when used in a real-world WSN deployment (by a practical indoor and outdoor WSN testbed). Details on the experiments are presented in Chapter 7.

Finally, the communication of the findings (activity 6) is discussed in Section 9.1. In general, the answer to each of the above stated RQs delivers a purposeful artifact that

is appropriately disseminated as publications at scientific conferences and journals or published as open-source and open-access information on respective online platforms (i.e., Github).

As also visible in Figure 1.1, the design science research methodology has an iterative nature. The activities are not processed linearly from step 1 down to 6. They are repeated throughout the research for each RQ or specific topic, respectively.

1.3 Contribution

This thesis presents a novel immune-inspired node fault detection approach for WSNs. The detection utilizes a modified DCA, an algorithm that abstracts the functioning of dendritic cells in the human immune system. It has shown promising preliminary results for network anomaly detection in computer and sensor networks. However, the original DCA is tailored for centralized network intrusion detection systems.

To leverage the working principle of the DCA for fault detection in WSNs, suitable input data have to be worked out based on which the algorithm can reliably detect sensor node faults. In the present approach, these input data are derived from self-checks and diagnostics performed on the sensor node. The combination of such node-level diagnostics and the DCA for node fault detection has not been addressed in the literature so far.

In contrast to previous fault detection schemes, our immune-inspired approach combines statistical metrics of the sensor data with node-level diagnostic data to enhance its detection rate while, at the same time, significantly lowering the FAR. For the latter, it is crucial to enable the detection to distinguish between data events and fault-induced anomalies. In addition, previous DCA-based schemes required a significant amount of manual intervention, e.g., for the mapping of the required input data and their fine-tuning. As will be presented in this thesis, our approach offers a high degree of generality, hence, removing most of these manual steps.

The research presented in this thesis includes three main contributions and four additional contributions. Our main contributions are:

- the elaboration on expressive node-level fault diagnostics
- the development of a sensor node platform facilitating active node-level reliability
- the implementation of a DCA-based sensor node fault detection system that exploits the combination of sensor data characteristics with node-level diagnostics

Additionally, the contributions include:

- a taxonomy for fault detection schemes in WSNs
- an analysis of the main causes of the prevalent node faults (e.g., undervolting)
- a survey of the characteristics of recent sensor node platforms
- a critical review of the Arduino platform for the sensor node development

However, this thesis does not present original research as most of its contributions were already published in corresponding conference proceedings [3–7] and journal articles [8]. Some parts of this thesis are directly taken from our papers discussing the related topic. In Section 9.1, the published papers concerning the aforementioned RQs are listed, and their specific contributions are summarized. As the design science research methodology involves iterative cycles, some of the RQs are addressed in more than one publication due to new findings and refinements. Additionally, most of the resources developed in the course of the research have been made publicly available on Github under https://github.com/DoWiD-wsn.

1.4 Thesis Outline

The remainder of the thesis is structured as follows. An introduction to wireless sensor networks (WSNs) is given in Chapter 2. Aside from considerations on the fields of applications and the architecture of WSNs, especially the specific characteristics of sensor networks are discussed. Although the chapter mainly summarizes the findings of respective literature reviews, it also presents an extensive overview and comparison of recent sensor node platforms. This overview shows the characteristics and properties of common sensor nodes that are further used for comparison with our self-developed sensor node platform.

An overview of the state-of-the-art for anomaly detection in WSNs is presented in Chapter 3. It highlights the challenges for anomaly detection imposed by the specifics of WSNs, presents standard metrics to assess and benchmark anomaly detection approaches, and presents our comprehensive taxonomy for anomaly detection for WSNs. Based on an overview of related works, especially the limitations and drawbacks of current node fault detection approaches based on anomaly detection schemes are discussed.

Chapter 4 focuses on sensor node faults and their detection in WSNs. It discusses the corresponding terminology and presents our taxonomy for sensor network faults. Additionally, it highlights the danger posed by (node) faults for WSNs. Most of all, related work on fault detection schemes, including their advantages, disadvantages, and limitations, are elaborated in this chapter.

Immune-inspired approaches have shown promising characteristics for developing lightweight fault detection systems. Consequently, an overview of the unique properties of the human immune system (HIS) and their exploitation in artificial immune systems (AISs), or immune-inspired techniques in general, is presented in Chapter 5. In particular, it focuses on the danger theory and its computational counterparts, i.e., the dendritic cell algorithm (DCA). Aside from a summary of related work on DCA-based fault detection, especially the limitations and shortcomings of current approaches are discussed.

Chapter 6 presents our immune-inspired node fault detection approach that forms the core contribution of this thesis. First, the models of prevalent faults in WSN that are targeted by our approach are treated. Then, the developed node-level diagnostics and

their use in our modified DCA are presented. The chapter closes with a discussion on the characteristics and properties of our immune-inspired fault detection approach.

The evaluation of our concept is treated in Chapter 7. It presents the setup of our tripartite evaluation setup and details on the particular experiments (i.e., simulation, lab experiments, and practical deployments).

Chapter 8 summarizes the results of our evaluation experiments. It presents the approach's correctness assessment and efficiency analysis as well as a comparison of our immune-inspired scheme with alternative approaches. Additionally, the main lessons learned in the dissertation research are highlighted.

Chapter 9 concludes this thesis by summarizing the main results and research findings, discusses possible extensions and improvements of our fault detection approach, and presents future research directions for immune-inspired fault detection in WSNs.

CHAPTER 2

Wireless Sensor Networks

One branch of technical systems that emerged from the need for fine-grained data are wireless sensor networks (WSNs). In the last two decades, WSNs have become an active research topic with numerous practical applications in industrial automation, environmental monitoring, and many more (cf. [9]). They mainly consist of sensor nodes that are deployed close to the monitored physical phenomena to sense specific physical quantities and forward their measurements to one or more dedicated services (e.g., cloud systems) for further processing. As a consequence, WSNs provide data with a high level of detail. Such data is essential for precise and correct data analytical services.

However, the sensor nodes are typically low-cost embedded systems with strictly limited resources imposing strict limitations and particular features for the WSNs. These limitations and features impact the processing capabilities of the sensor nodes, their behavior, and their interconnection. The resulting characteristics and properties of WSNs distinguish them from traditional computing networks, hence, making a separate treatment for specific topics (e.g., fault detection) necessary (cf. [10]). Consequently, the research community targets numerous topics such as low-power wireless technology, energy-efficient network protocols, node deployment strategies, or sensor node design.

In Section 2.1, we elaborate on the fields of applications of WSNs and their respective characteristics. The general structure, including standard components and network architectures, are discussed in Section 2.2. Combining the requirements of the intended fields of applications with the architectural and technical properties of sensor networks leads us to the specific characteristics and distinctive features of WSNs (cf. Section 2.3). These characteristics and features are crucial for understanding the significant danger that faults in sensor networks, particularly node faults, pose on the data service quality.

2.1 Fields of Applications

A significant number of fields of application and use cases fall into the realm of WSNs [11], some of which are already well-established and widely used. The main applications of WSNs can be categorized as (i) military applications, (ii) health applications, (iii) home applications, (iv) industrial automation/applications, (v) environmental applications, and (vi) civil structure monitoring (cf. [12, ch. 2]). All WSN deployments have in common that they consist of spatially dispersed sensor nodes usually realized by low-power embedded systems that measure specific physical quantities via attached sensors. Depending on the application requirements, these sensor nodes are interlinked by near-field communication (e.g., Zigbee, Bluetooth low energy (BLE)) [13], or low-power wide-area networks (LPWANs) such as long-range wide-area network (LoRaWAN) [14].

On a general level, most WSN deployments can be categorized into one of two main applications depending on whether they provide continuous sensing (e.g., environmental or process monitoring) or perform event detection (e.g., forest fire detection or surveillance). While both share some common characteristics (such as the network structure), there are differences in their respective requirements, especially concerning the expected lifetimes, the communication patterns, and the amount of data to be transferred. Also, the particular fields of applications differ in the area to cover, the data update intervals, and the environmental conditions in which the sensor nodes are embedded. These differences impact the possibility of detecting faults on the node level. For example, continuous sensing provides the possibility to consider the difference between two consecutive measurements.

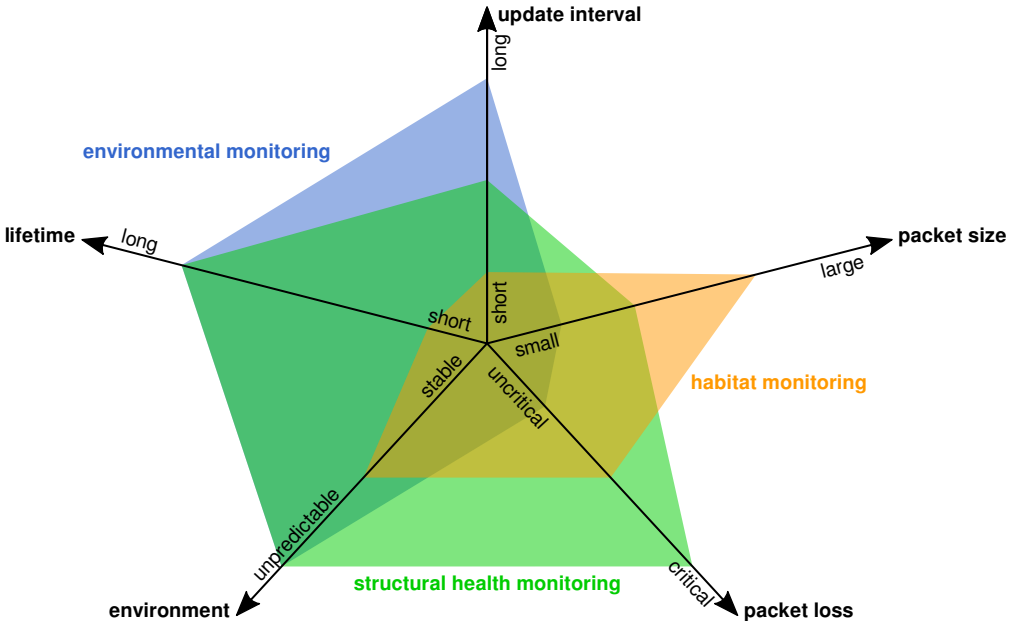


Figure 2.1: Characteristics of different WSN monitoring applications

In this thesis, the focus is laid on monitoring applications that provide continuous sensor data. Even within the field of monitoring applications, the particular characteristics differ significantly as quantitatively shown based on the example of environmental, habitat, and structural health monitoring in Figure 2.1.

Additionally, the advent of the so-called Industry 4.0 has led to an increasing use of sensor networks in industrial monitoring applications (e.g., predictive maintenance [15]). In several industrial applications, harsh environmental conditions caused by the electromagnetic interference of heavy machinery, high ambient temperatures, or strong vibrations challenge the proper operation of both the sensor nodes and their interlinks.

In this work, environmental monitoring applications are primarily targeted for further considerations on the fault models of interest and the immune-inspired node fault detection approach. However, the resulting fault detection approach is generically applicable to most WSN fields of application, even if they do not report continuous sensor data.

2.1.1 Environmental Monitoring

Environmental monitoring [16, 17] was one of the earliest applications of WSNs. It refers to any sensor network that monitors certain environmental phenomena or physical conditions. The intention behind environmental monitoring is to have an additional source of information for (i) the study of certain phenomena (e.g., temperature changes for meteorology), (ii) the analysis of physical conditions that may be used for process optimization (e.g., plant growth for smart agriculture [18, 19]), or (iii) the prediction of future trends, be it for optimization purposes (e.g., smart traffic systems [20]) or for hazard mitigation (e.g., forest fire detection systems [21–23]).

In its early years, most monitoring systems were used for air or water quality monitoring [24, 25]. Many environmental applications are used as pure information sources and require human interaction only in case of certain events. However, more and more applications arise where the monitoring system is part of an automatic or even autonomous system (e.g., for smart traffic applications [20]).

Environmental monitoring applications usually require long battery lifetimes (often more than ten years) and typically operate in harsh and unpredictable environments. The packages transmitted are of comparably small size (few bytes), and the updates happen, depending on the actual application, in a minute or hour granularity. For most environmental monitoring applications, the loss of packets is not critical.

In the present work, the practical considerations focus on environmental monitoring used as part of a smart garden system, that is, a system to monitor and optimize certain physical conditions related to plant growth. The sensor network delivers information on the current state of selected conditions. However, the actuator part (e.g., irrigation system) is not in the scope of this work.

2.1.2 Habitat Monitoring

WSNs are also broadly used for habitat monitoring which refers to systems used to monitor the behavior of animals. Their goal is to track and analyze the health and/or social contacts of certain (groups of) animals. In this context, the two most famous examples are "The Great Duck Island project" [26] and "ZebraNet system" [27].

The Great Duck Island is a famous breading area of seabirds in Maine, USA. Sensor nodes were deployed on this island to analyze the nesting behavior of the local seabirds (i.e., occupancy of nesting burrows) and the impact of microclimatic factors on the habitat selection of these seabirds.

The ZebraNet system, on the other hand, was deployed in Kenya to keep track of two different species of zebras and analyze their natural behavior. With the advent of smart farming (or smart agriculture), such habitat monitoring systems are also increasingly used to monitor the health of cattle or sheep herds [28, 29].

Habitat monitoring applications usually have much shorter battery lifetimes than environmental monitoring deployments. Their environment is partly predictable as the animals usually seek shelter in rough weather conditions. Depending on the actual application, considerably more data are transmitted. Thus, the packet sizes are much larger than in most other monitoring fields of application. Also, the updates usually happen in shorter intervals (seconds to minutes granularity). As with environmental monitoring, the loss of packets is not critical in most cases.

2.1.3 Structural Health Monitoring

WSNs are also increasingly used in the field of civil structure monitoring in structural health monitoring (SHM) systems to keep track of the condition of the respective civil structures. The main advantage of such systems is that they allow a paradigm shift from "fix-and-fail" to "predict-and-prevent" [30]. In the past, defects in civil structures such as buildings or bridges have been noticed when they have already manifested themselves on the surface. At this point, the damage has reached a scale where noticeable consequences can not be avoided anymore (like underground flooding in case of broken water pipes). More severe are those cases in which civil buildings collapsed due to structural defects that went unnoticed for longer times (e.g., the collapse of bridges [31]).

Generally speaking, there are two basic approaches to SHM [32]:

1. direct detection via visual inspection, x-rays, or similar
2. indirect detection by tracking changes in structural properties or system behaviors

While the former usually requires specially trained personnel, the latter is performed by technical systems in a more autonomous manner. For this purpose, indirect detection systems use modal parameters such as eigenfrequencies or eigenmodes for the analysis. Two discriminating factors for the analysis of these parameters are time-scale and the severity of the change, hence, how fast and to what extent the monitored parameters

change. As stated in [33], important characteristics of SHM systems are (i) adaptability, (ii) autonomy, and (iii) reliability. Especially reliability plays a crucial role in many SHM applications as no data has to be lost since these events happen rarely, cannot be duplicated, and may contain important information on the health state of the respective structure [32].

Today, SHM systems enable the permanent monitoring of different assets and the estimation of their structural health state, thus, proving measures to detect structural changes even before they significantly affect the performance of respective structures [32]. SHMs based on WSNs are applied to diverse kinds of civil structures, such as buildings [34], water pipes [35], railway tracks [36], highways [37] or tunnels [38,39]. The most prominent examples of WSN-based SHM systems are applied to bridges [40–42] as these structures have a higher risk of defects caused by vibrations (be it from strong wind or the vehicles moving across it) and the specific form of their architecture.

One of the first successfully applied sensor-based SHM was a setup on the Ben Franklin Bridge in Philadelphia which started around 2001 [43]. The installation was considered necessary as a considerable amount of high-speed trains are crossing this 2,918 m-long bridge regularly, causing significant vibrations whose effect on the bridge's structure could not be assessed before.

Another example is the retrofitting of the Tamar Bridge in southwest England with a SHM in 2001, which was further improved in 2006 and 2009, respectively. The bridge is a 335 m span suspension bridge whose original structure was opened in 1961. In the early 2000s, it was strengthened to meet European standards allowing heavy trucks with a wight of up to 40 tons to pass it. Due to the complexity of the resulting bridge structure, the SHM was considered relevant to survey the ambient vibration and the bridge's reaction to wind, temperature, and cable tension [44].

To date, one of the largest WSN-based SHM deployment was installed by a team of researchers of the University of California at Berkeley in cooperation with Crossbow Technology, Inc. in 2006 at the Golden Gate Bridge in San Francisco [32]. In this work, named the *Golden Gate Bridge Project*, a total of 64 sensor nodes were deployed on the 2,737 m long bridge in a 46-hop network.

As with environmental monitoring, also SHM applications usually have long battery lifetimes of several years and are often operated in relatively harsh and unpredictable environments. The data is transmitted in smaller packages in update intervals of usually minute granularity. As stated above, the loss of packets is in most cases critical as events signaling damage must not be missed.

2.2 Structure and Components

WSNs consist of spatially dispersed sensor nodes deployed in an area of interest to monitor their physical environment. The deployment of a WSN is a difficult and time-consuming task since several development aspects need to be considered, such as (i) the

communication technology including security concerns, (ii) the network topology and routing, (iii) the data storage and analysis, and (iv) the hardware to be used. Depending on the field of application and use-case specifics, a great variety of WSN architectures and deployment strategies exist.

However, they all consist of two main parts:

- the sensor nodes and
- the network architecture.

Both parts are equally important for the quality of service provided by the WSN. An overview of the network architecture is presented in Section 2.2.1. As the focus of this thesis is on senor node fault detection, a more detailed discussion of sensor nodes is given in Section 2.2.2 and an overview of recent node platforms is presented in Section 2.2.3.

2.2.1 Network Architecture

For the deployment of WSNs, two main network architecture structures exist, namely (i) a layered or (ii) a clustered structure. As explained below, both have different characteristics concerning their components and the underlying network topology.

Layered network architectures consist of a single powerful base station providing the access point to a large number of sensor nodes. The sensor nodes are commonly connected to the base station via multi-hop networks where nodes are grouped into layers according to the hop-count to the base station; thus, the name "layered architecture". Such network architectures have primarily been used with in-building sensor networks and military sensor-based infrastructure (cf. [45]).

The majority of WSNs use a *clustered network architecture*. In this architecture, sensor nodes located geographically close to each other are grouped into clusters. Each cluster usually has one dedicated node (often denoted as *cluster head* or *cluster leader*) responsible for collecting the information from its neighboring sensor nodes and collectively forwarding the data to one or more central services for further processing. These cluster heads are often equipped with higher resources than the sensor nodes. An architectural example of a clustered WSN is shown in Figure 2.2. In the figure, the three clusters depict the three most common network topologies used in clustered WSNs (i.e., tree, star, and mesh).

As can be seen in Figure 2.2, the term "layer" is also used in clustered WSNs for the particular stages in the data chain. From a global point of view, the WSN operates at the boundary of the data-processing chain (at the "edge") and, thus, the sensor nodes are sometimes denoted as *edge devices*. The cluster heads are called *edge gateways* that, together with the sensor nodes, form the *edge layer*. The data collected by the cluster heads are then either directly forwarded to a cloud system (i.e., central data endpoint) or pre-processed by intermediate systems before being uploaded to the cloud. In the latter case, the intermediate systems are commonly referred to as *fog devices* as they operate between the edge and the cloud. As shown in Figure 2.2, the number of devices per layer

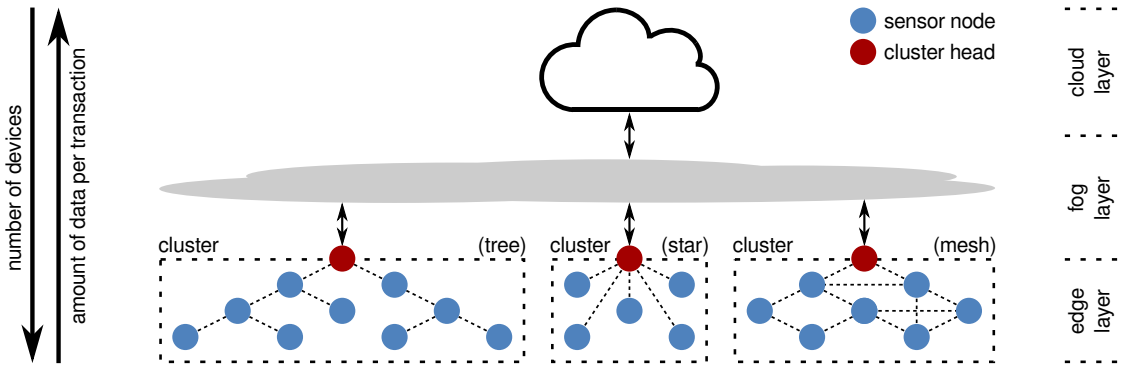


Figure 2.2: Architectural example of a clustered wireless sensor network

decreases from the edge to the cloud layer, while the amount of data transmitted per transaction significantly increases.

2.2.2 Sensor Nodes

On the lowest layer, WSNs consist of interlinked low-power embedded systems responsible for sensing specific physical quantities. These systems are commonly referred to as *sensor nodes* or sometimes also called *motes*[1]. The sensor nodes are key components of the WSN and have a significant impact on the network's performance and accuracy. They differ in the used hardware components, their size and weight, the supported power sources and battery lifetimes, and the sensors available or the analog/digital interfaces, respectively. Choosing suitable hardware components is crucial as they essentially determine the nodes' functionality and operational characteristics, including their energy efficiency and reliability. Regarding the latter, the components heavily influence the probability and nature of faults that can impair the nodes' proper function [5]. On the other hand, high energy efficiency is essential since most sensor nodes are battery-powered, and the available energy budget is strictly limited.

Typically, sensor nodes have to (i) acquire sensory information of certain physical quantities, (ii) perform some (pre-)processing of data, (iii) forward their data over wireless links, and (iv) operate in an energy-efficient manner to ensure long battery lifetimes. These basic requirements are reflected in the typical components of sensor nodes. Consequently, the hardware of most sensor nodes can be divided into four basic blocks as depicted in Figure 2.3, namely (cf. [47]):

(i) a set of sensors (see Section 2.2.2.1),

(ii) a processing unit (optionally with external memory; see Section 2.2.2.2),

(iii) a radio transceiver (see Section 2.2.2.3), and

(iv) a power unit with a power source (i.e., a battery; see Section 2.2.2.4).

[1] In this context, the term "mote" refers to sensor nodes of minimal size.

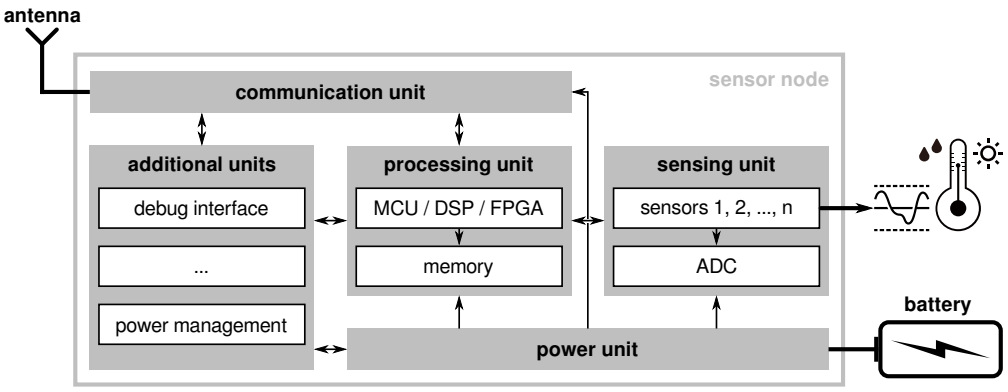


Figure 2.3: Basic components of a wireless sensor node (after [46, Fig. 1])

Additionally, the sensor nodes can be equipped with additional units such as a debugging interface to support the software development or with external power management capabilities on top of the power units.

2.2.2.1 Sensor Unit

The sensor unit is responsible for acquiring sensory information of specific physical quantities. Different sensors for various physical quantities are available depending on the application requirements. These sensors differ in the quality of the provided measurements (i.e., resolution, accuracy, conversion time) and in the way they provide the measurement to the subsequent processing unit. Thereby, two basic kinds of sensors can be distinguished: sensors available as integrated solutions and sensors (or sensory circuits) that output an analog signal proportional to the measured physical quantity. The former provide their measurements via a digital interface (e.g., universal synchronous/asynchronous receiver-transmitter (USART), serial peripheral interface (SPI), or inter-integrated circuit (I2C)). On the other hand, the latter requires an analog-to-digital converter (ADC) to make the measurements available to the processing unit that can be either an on-chip peripheral of the processing unit or a separate hardware component.

2.2.2.2 Processing Unit

The processing unit is the heart of the sensor node and gathers the measurements from the attached sensors, prepares these values for transmission (possibly including some pre-processing like normalization, conversion, or plausibility checks), and eventually forwards the data via the communication unit. While most sensor nodes use a microcontroller unit (MCU) as a processing unit sometimes extended with external flash memory, there are also solutions based on digital signal processors (DSPs), field-programmable gate arrays (FPGAs) or highly-integrated systems-on-a-chip (SoCs) with multicore architectures [48]. Aside from a shorter time-to-market, MCU-based nodes are beneficial due to their low prices and comparably low power consumption. The majority of MCU-based sensor

nodes currently either use an 8-bit AVR ATmega, a 16-bit TI MSP430, or a 32-bit ARM Cortex-M0/M3 MCU [47]. In the last years, a shift towards SoC-based nodes has been noticed, where the processing and communication unit are both integrated into a single chip [49].

2.2.2.3 Communication Unit

The choice of the communication unit depends on the transmission medium and the communication system to be used. Even though the majority of WSNs use radio frequency (RF)-based communication (e.g., using radio transceivers in the license-free industrial, scientific and medical (ISM) bands at 868/915 MHz and 2.4 GHz), some utilize other forms of communication such as ultrasonic-based systems used in submarine WSNs (cf. [50]). Except for multimedia WSNs, the majority of sensor networks use moderately low data rates of up to 250 kbit/s. Especially monitoring applications such as environmental monitoring usually require the transfer of comparably small network packages transmitted at lower data rates to keep the power consumption at a minimum. Depending on the range of the communication link, the radio transceivers can have an external, printed circuit board (PCB), or chip antenna attached. A summary of some commonly used RF modules can be found in [51], and an overview of wireless standards and technologies often used in WSNs is given in [47].

2.2.2.4 Power Unit

The power unit is responsible for providing the sensor node's components with power usually supplied by a battery. If the node uses energy harvesting (e.g., solar cells), the power unit must also control the battery's charging cycles. Especially for sensor nodes without energy harvesting, the choice of the power unit is often straightforward. Several nodes have the battery directly connected to the supply rail of the node's components. These nodes usually rely on the brown-out detection of components that disables them in case of a too low supply voltage to avoid unintended effects of a depleting battery (i.e., sinking battery voltage). Other sensor nodes cope with the effects of a possible undervolting on a higher level (cf. [52]). However, most sensor nodes use linear regulators to ensure a stable supply voltage, but at the cost of bad energy efficiency. Especially ultra-low-power sensor nodes tend to exploit the high efficiency of DC/DC converters where modern solutions only require a few additional (passive) components.

2.2.3 Node Platforms

Selecting proper hardware components for the sensor nodes is crucial to ensure a reliable operation (even under harsh environmental conditions) while providing high-quality data. The hardware selection, however, is not trivial as the sensor node design is challenged by the tradeoff between low-power operation and sufficient computational performance and using low-cost components while being small in size [48]. Nevertheless, the most

essential factor for sensor node design is the limited energy budget. Sensor nodes are usually battery-powered, and energy harvesting is not always possible or feasible.

Basically, there are three options for the sensor node development, namely:

(i) to build sensor nodes from scratch (*custom nodes*),

(ii) to utilize a generic embedded platform (*semi-custom nodes*), or

(iii) to use an available sensor node platform (*commercial* or *academic nodes*).

2.2.3.1 Custom Sensor Nodes

Custom sensor nodes are often designed for a particular application and, thus, offer the highest degree of specialization to the corresponding requirements. Their development requires considerable time and resources and a certain degree of expertise. Aside from selecting appropriate hardware components, considerations on the power supply, and developing the PCB, also the software consisting of an operating system (OS) or middleware, the sensor drivers, and the communication drivers as well as the respective toolchain need to be prepared.

2.2.3.2 Semi-Custom Sensor Nodes

Alternatively, the sensor nodes can be developed in a semi-custom fashion with a generic embedded platform (i.e., development and breakout boards) extended with application-specific hardware such as a radio transceiver and sensors. Such approaches usually require less development time than custom sensor nodes and often result in cheaper hardware costs as many embedded platforms are available at low prices. In addition, many of these embedded platforms are supported by a large community providing software drivers and example codes. The most-known generic embedded platforms include *Arduino*, *BeagleBoard*, *Raspberry Pi*, *Teensy*, *Espressif* (*ESP*), and *mbed*. However, these platforms usually target a wide range of applications. Hence, they are not explicitly designed for low-power sensor node operation. Such platforms often offer high computational power and particular onboard circuitry supporting the development process (e.g., light-emitting diodes (LEDs)) at the cost of a (significantly) higher overall power consumption (cf. [6]).

2.2.3.3 Commercial / Academic Sensor Nodes

Many wireless sensor nodes have been developed by universities, research institutions, or companies in the last two decades. Some of these nodes are (or have been) commercially available or were even made public as an open-source project. In the research community, probably the most-famous sensor nodes are the *Berkeley motes* including the well-known *Mica* or *Telos* motes [49, 53]. The advantage of using such ready-to-use nodes are the significantly shortened development times and the availability of the core software components in combination with hardware tailored for the (ultra) low-power requirements of wireless sensor nodes. On the downside, these nodes are often specific to particular

use-cases offering moderate flexibility, and several sensor nodes are not publicly available or not available anymore at all.

2.2.3.4 Node Platform Review

In [8], we conducted a literature review on recent sensor node platforms that extends the surveys presented in [47, 48, 51, 53]. It focuses on sensor nodes that either support high energy efficiency (i.e., ultra-low-power operation), include techniques for node-level fault tolerance (i.e., self-diagnostic measures), or both. An overview of sensor nodes found, their year of publication, and their characteristics are given in Table 2.1. The overview includes our sensor developed in the course of the present research. Details on this sensor node, named ASN(x), follow in Section 7.1. However, the overview depicted in the table focuses on the sensor nodes and excludes information on the radio transceivers and supported communications standards. For information on the network capabilities of the sensor nodes, we refer an interested reader to our full review in [8].

The table lists the core components of the sensor nodes, that are, the MCU including its specifications in terms of central processing unit (CPU) architecture, clock frequency (F_{CPU}), and available memory (i.e., flash, static random-access memory (SRAM), and electrically erasable programmable read-only memory (EEPROM)). Also, information on the nodes' supply voltage for the core components (V_{core}), the supported input voltage range (V_{bat}), and the typical power consumption[2] in the active and power-saving modes are listed. However, the radios' transmit and receive power consumption are neglected in the table as the values depend on the actual radio settings, such as the transmit power. Information on these values can be taken from the corresponding datasheets of the radio transceivers. Additionally, the table lists the *voltage regulation* technique where:

- ● refers to nodes using a DC/DC converter,
- ◑ denotes nodes using a linear regulator (e.g., low-dropout regulator (LDO)), and
- ○ highlights nodes that have the battery directly connected to the core supply rail.

Next, the columns *energy-efficiency* and *self-diagnostic* state to which extent energy-efficiency was considered in the node design and whether self-diagnostic measures are included, respectively. In the *open source* column, the extent to which the sources of the nodes' design and software are publicly available are highlighted where:

- ● means that all related information is publicly available,
- ◑ refers to nodes where only parts are available (mostly the software), and
- ○ shows that no information has been made publicly available.

Last, the availability status of the nodes (i.e., whether the nodes are (still) available on the market) and the price of one sensor node are listed. For commercial nodes, the

[2] As most authors present the current consumption of their sensor nodes rather than the power consumption, we calculated the corresponding power by multiplying the given current values with the nodes' core voltages to allow for better comparison.

Table 2.1: Overview of sensor node platforms

	Sensor node	Year	MCU/SoC	Arch. [bit]	F_{CPU} [MHz]	Flash [kB]	SRAM [kB]	EEPROM [kB]	V_{core} [V]	V_{bat} [V]	active mode [mW]	power-saving [μW]	Voltage regulation	Energy-efficiency	Self-diagnostic	Open source	Available	Price [$]
Commercial	UC Berkeley TelosB [54]	2005	MSP430F1611	16	8	1072	10	16	3.0	1.8–3.6	6.6	20.1	○	●	○	◑	○	99.00
	ETH Zürich Btnode [55]	2005	ATmega128L	8	8	128	64	4	3.3	0.5–4.4	39.6	9900	●	◑	○	◑	○	215.00
	UC Berkeley IRIS [56]	2007	ATmega1281	8	7.37	640	8	4	3.0	2.7–3.6	26.4	26.4	○	●	○	◑	○	115.00
	SHIMMER [57]	2010	MSP430F1611	16	8	48	10	16	3.3	1.8–3.6	5.9	16.8	◑	◑	○	○	○	269.00
	OpenMote CC2538 [58]	2015	CC2538SF53	32	32	512	32	–	3.3	2.0–3.6	42.9	×	●	●	○	◑	○	×
	Libelium Waspmote v15 [59]	2016	ATmega1281	8	14.75	128	8	4	3.0	3.3–4.2	56.1	99.0	◑	◑	○	◑	●	174.00
	Zolertia RE-Mote [60]	2016	CC2538SF53	32	32	512	32	–	3.3	3.3–16	66.0	4.3	◑	●	○	◑	●	112.00
	WiSense WSN1120L [61]	2019	MSP430G2955	16	16	56	4	128	3.0	1.8–3.6	56.1	56.1	○	●	○	◑	●	48.00
	OpenMote B [62]	2019	CC2538SF53	32	32	512	32	–	3.3	2.0–3.6	42.9	4.3	●	●	○	○	●	125.00
Academia	Kmote [63]	2007	MSP430F1611	16	8	8240	10	16	3.3	2.3–6.0	4.9	22.1	◑	◑	○	○	○	37.85
	Beasties [64]	2008	ATmega8L	8	4	8	33	0.5	5.0	7.0–20	77.5	40000	◑	○	○	○	○	139.00
	INGA [65]	2012	ATmega1284P	8	4	128	16	4	3.3	×	61.7	×	◑	◑	◑	◑	○	120.00
	Storm [66]	2014	ATSAM4LC8C	32	48	1536	64	–	3.3	1.8–3.6	4.5	7.6	●	◑	○	○	○	50.00
	Raju and Pratap [67]	2015	MSP430F5438	16	25	256	16	–	3.3	1.8–3.8	×	×	○	○	○	○	○	×
	Zeni *et al.* [68]	2015	ATmega328P	8	1	32	2	1	3.0	1.9–3.6	5.8	15	○	●	○	○	○	12.00
	panStamp NRG3 [69]	2016	CC430F5137	16	20	32	4	–	3.3	2.0–3.6	46.2	8.3	◑	◑	○	○	●	×
	EARNPIPE [70]	2016	AT91SAM3X8E	32	84	512	100	–	3.3	7.0–12	×	×	◑	◑	○	○	○	×
	uLoRa [71]	2017	STM32L051K8T6	32	32	64	8	2	3.3	×	34.7	1.2	×	●	○	◑	○	12.00
	Rusu and Dobra [72]	2017	STM32L443RC	32	80	256	64	–	3.3	×	×	×	×	◑	○	○	○	×
	Hamilton [49, 73]	2017	ATSAMR21	32	48	256	32	–	3.0	×	3.2	19.5	○	●	○	◑	●	25.00
	Hazelnut [74]	2019	ATtiny85	8	1	8	0.5	0.5	3.3	×	231	650	○	◑	○	○	○	×
	Raposo *et al.* [75]	2019	MSP430F5229	16	25	128	8	–	3.3	×	×	×	●	◑	◑	○	○	×
	Babusiak *et al.* [76]	2019	ATmega328P	8	1	32	2	1	3.0	1.5–3.6	10.5	22.2	○	●	○	○	○	11.00
	MEGAN [77]	2020	ATmega324PA	8	8	32	2	1	3.3	×	26.2	33.3	◑	◑	○	○	○	20.00
	ASN(x) [8]	2021	ATmega1284P	8	4	128	16	4	3.3	1.8–5.5	15.4	121.1	●	●	●	●	●	50.00

● supported ◑ partly supported ○ not supported – not available × no information available

price refers to the cost of one node on the market, while for nodes presented in academic papers, the cost estimation of the authors is stated. However, in both cases, the actual costs can vary depending on the distributor of the nodes or hardware components as well as the PCB manufacturer in the latter case. Also, some nodes come equipped with several sensors, while others provide the baseboard only. Therefore, the provided values shall be considered a reference value for coarse comparison.

In our review [8], we found that especially the energy characteristics stated by some authors have to be taken with care. In some cases, only the consumption of single components (sometimes just taken from the corresponding datasheets) is stated rather than the board's actual consumption, including peripherals and passive components. Also, the information provided in some of the surveys is incorrect, or at least questionable, especially if the source of information is missing.

However, the focus of this thesis lies on node-level fault-tolerant sensor nodes that also allow an energy-efficient operation. Therefore, sensor nodes focusing on energy efficiency and their power-saving approaches, as well as nodes enabling self-diagnostics to enhance the WSN's reliability, are discussed in the following.

Energy-efficient Sensor Nodes

The overview of sensor nodes in Table 2.1 reflects the importance of energy-efficiency in WSNs. Except for two designs, energy efficiency was at least partly considered in all nodes. Thereby, two main design criteria are important to ensure energy-efficient operation, namely:

(i) the duration of the active and the sleep phases (i.e., duty-cycling) and

(ii) the power consumption in both phases (i.e., energy-efficient hardware).

(i) Usually, the hardware components such as the MCU, the radio transceiver, and (where possible) also the sensors are kept in an active state for as short as possible. The rest of the time, the components are put to a power-saving or sleep mode to save energy (cf. [78]). In both states, the power consumption depends on the hardware used in combination with board assembly-related factors (i.e., passive components) and, in case used, OS-related characteristics. Consequently, the power consumption needs to be measured on an actual prototype as the sum of the datasheets' values is usually much lower than the reality.

Depending on the amount and type of sensors, the complexity of the data processing, and the communication standard, the active time is markedly smaller than the duration of the power-saving phase and is usually in the range of several milliseconds up to a few seconds. As a result, the hardware components also impact the duty-cycling as, for example, some sensors require a specific conversion time that can significantly prolong the active phase (e.g., the temperature measurement of the DS18B20 sensor takes up to 750 ms). On the other hand, the sleep time depends on the application requirements and is commonly in the range of several seconds or minutes (up to a few hours in rare cases). Thus, the energy spent in power-saving mode commonly dominates the overall

power consumption [49]. In this context, previous studies [79] found that one of the main contributors to active power consumption is wake-up energy. The hardware needs to be re-started and possibly re-configured for proper functioning during the wake-up. Additionally, the device may need to re-connect to the network. For example, the authors of [79] concluded that the length of the sleep phase in IEEE 802.11-based WSNs should be greater than 30 s with larger values resulting in better energy efficiency.

(ii) Aside from a suitable active/sleep schedule, the choice of hardware also affects the sensor node's energy efficiency. Most sensor nodes utilize low-power components (e.g., MCU or SoC) that have a comparably low power consumption in the active state, often in combination with the support of energy-saving (e.g., disabling of unneeded on-chip peripherals) and power-down possibilities (i.e., sleep modes). Regarding the latter, the power consumption of the digital circuits (mostly CMOS-based) mainly consists of a static fraction caused by a certain leakage current, a dynamic part resulting from changes in the charge of the (usually parasitic) capacitances of the circuitry, and a transient short circuit power dissipation during the switching. However, the dynamic part dominates the overall power dissipation. It has a linear dependency on the clock frequency and a quadratic dependency on the supply voltage. To lower the power demands of the circuitry, both the frequency and the supply voltage may be decreased (down to certain thresholds). The runtime adjustment of the frequency is commonly done via dynamic frequency scaling (DFS) schemes and the adaptation of the supply voltage by dynamic voltage scaling (DVS) techniques. While DFS is sometimes provided by the MCU/SoC, DVS usually requires additional onboard circuitry. In the past, DVS approaches applicable to sensor networks have been proposed that adjust the supply voltage level based on the system needs down to the minimum threshold specified by the manufacturer [80]. Besides, there is also so-called "active undervolting" where the supply voltage is even lowered below the specified minimum voltage level to decrease the power consumption further (cf. [52]).

A suitable hardware selection of the sensor nodes is also relevant for leveraging the sleep modes of the processing unit (i.e., MCU). Most recent MCUs provide different power-down modes with different levels of saving potentials by deactivating the clock source for specific on-chip components. In most power-saving modes, the clock of the CPU is deactivated. This, however, raises the need for external components able to wake the MCU up from its deep sleep. To do so, most often low-power real-time clocks (RTCs) are added to the nodes' design that can wake the MCU, for example, via an external interrupt. In such a case, the RTC can quickly become a single point of failure as a missing wake-up signal can cause the sensor node never to wake up again. Therefore, the chosen wake-up source and its consequences on the node's reliability must be carefully considered.

Coming back to the overview on energy-efficient sensor nodes, we found that the majority focuses on using (ultra) low-power components in their design [68, 71–73, 73, 76]. Some authors additionally took the passive hardware components required for proper functioning into their considerations [68, 73]. Surprisingly, the majority of nodes found use linear voltage regulators [57, 59, 60, 63–65, 69, 70, 77] or even no regulation at all [54, 56, 61, 67,

68, 73, 74, 76] instead of low-power DC/DC converters [55, 58, 62, 66, 75]. Especially if the battery is directly connected to the hardware components' supply rail, unintended effects (including soft faults) can happen at the end of the battery life due to a (passive) undervolting of the components (cf. [4, 52]). Also, the selection of the MCU's clock frequency has only been discussed in a few designs [64, 65, 68, 74, 76] by keeping the frequency as low as possible. Enhanced energy-saving approaches such as DVS (or active undervolting) were only considered in [52]. The impact of the software design on the nodes' overall energy efficiency was only presented in [68, 73] as the majority focused on the hardware design and left the software to the application developer.

Self-diagnostic Sensor Nodes

While energy efficiency was considered in almost all sensor node designs found in the literature review, self-diagnostic measures were only partially treated in two cases [65, 75]. All other designs relied on implementing reliability measures (e.g., fault detection and/or tolerance) on a network level rather than the node level. Other network participants have to mitigate the effects of faults occurring in sensor nodes before the affected data reach the subsequent data analysis. However, dealing with node faults on the network or even cloud level faces severe issues concerning the detection of soft faults such as silent data corruption. As a result, incorporating node-level information in the fault detection approaches is inevitable (cf. [3, 5]).

In the sensor nodes proposed in [75], a sensor node monitoring agent is deployed in the firmware of each sensor node that is able to collect specific runtime metrics provided by the OS or the firmware such as the CPU load (i.e., number of cycles executed by the MCU), the execution time, and the energy consumed (based on the approach proposed in [81]). Aside from that, no node-level self-diagnostics were applied.

In the approach proposed in [52] the primary focus is on energy saving by active undervolting. Thereby, the sensor nodes' supply voltage is decreased below the minimum voltage threshold suggested by the manufacturer down to the lowest value that allows a proper operation. However, this value depends on several factors, including manufacturing-related characteristics and the ambient temperature. As a result, the voltage level can drop below the lowest operational point, and the node stops working. Consequently, the used INGA nodes [65] are equipped with a secondary MCU supplied by a constant voltage in the safe region. This secondary MCU monitors the primary MCU using spot tests and resets the supply voltage in case an error is detected, hence, forming a type of node-level fault detection. The spot tests applied are matrix multiplications checking the proper functioning of the primary MCU's arithmetic logic unit (ALU). However, this approach has two main drawbacks that can endanger the WSN's reliability, namely:

1. the secondary MCU can be impaired by faults, too, and
2. checking the primary MCU's ALU alone is insufficient to ensure reliability.

Regarding the latter, in [4] we showed that such a spot-test needs to take all used on-chip peripherals into account since, for example, the minimum supply voltages for the ADC

or the USART interface are notably higher than those of the ALU. There are voltage levels where the ALU properly works, but the ADC and USART do not, and thus, the system is in a state susceptible to soft faults.

2.3 Characteristics and distinct Features

While WSNs offer a versatile opportunity to acquire qualitative data of diverse physical conditions, they usually impose strict limitations and particular features that distinguish them from traditional computing environments [82, 83]. These limitations and features have an impact on the sensor nodes, their behavior, and the network connecting them.

First of all, energy- and resource efficiency are of the highest importance. The nodes of WSNs are often expected to have battery lifetimes of ten years and more. Wired power connections or periodical battery charging/replacing are uneconomic or impossible in most applications. To ensure meeting this requirement, no highly complex algorithms, protocols, or the like can be used as increased complexity usually comes along with increased power/resource consumption.

Devices in the fog and cloud layers are usually operated in a controlled environment (e.g., data centers), have a wired power supply, and are commonly equipped with vast resources regarding their processing and memory capabilities. On the other hand, the devices of the edge layer most often operate outdoors with uncontrollable and mostly unpredictable conditions. While the cluster heads are sometimes powered by a wired power supply and have moderate resources, the sensor nodes are typically powered by batteries (with or without the possibility for energy harvesting) and have strictly limited resources. Additionally, several WSN applications densely deploy the sensor nodes to cover a wide area and/or have a fine spatial granularity resulting in large numbers of devices (ranging from tens up to thousands). This, however, usually requires the sensor nodes to mainly consist of low-cost components to keep the deployment costs as low as possible.

Directly affected by these limitations are the interconnections of the sensor nodes. As a result, energy-efficient communication tailored to the specific requirements is essential. For this reason, different communication standards and protocols are used to fit the given conditions best.

Near-field communication (e.g., Bluetooth, Zigbee) dominated the early years of WSNs. Today, several applications also demand long-distance communication [84]. Cellular networks are usually no option as their transceivers require too much energy. Thus, specialized long-distance networks with minimum power requirements known as LPWAN are often used (e.g., LoRaWAN or Sigfox).

As summarized in [85] and [86], besides the strict resource constraints, WSNs entail severe vulnerabilities associated with their wireless links over open and most often unreliable communication channels, their highly dynamic network structure, self-organization, and decentralized management scheme in combination with the harsh environment in which

they are deployed. Especially the harsh environment plays a crucial role for considerations on WSNs' architectures and designs. Not only do environmental conditions influence (or even temporarily prevent) the wireless links between the sensor nodes (like the temporary blocking of signals by obstacles [87] or the absorption of radio waves by rain or the surrounding vegetation [88]), they also pose a significant hazard to the sensor node's hardware and, thus, its proper operation. The sensor hardware has to operate in unpredictable and uncontrollable conditions influenced by environmental factors such as significant fluctuations in temperature, high levels of humidity, or strong vibrations that pose a severe risk of physical damage to the node. Fluctuations in air temperature and humidity impose the risk of water condensation inside the sensor node's housing, leading to short circuits and damaged components. The situation is even worse in maritime environments as the combination of sea fog and strong wind results in quick condensation of salty water and fast oxidation of metallic components [32].

Another critical aspect of the network is the general architecture and topology. While there are different, sometimes even hybrid approaches available, the majority of WSNs use either a star-, tree- or mesh network. In many WSN deployments, a significant number of sensor nodes is deployed in relatively high density to ensure fidelity through redundancy [89]. For such networks, a static configuration is inefficient, especially considering that some applications may need to support the mobility of sensor nodes. Therefore, such networks are usually self-organized and flexible, forming a dynamic network structure.

As stated in [90], for such large-scale networks, it is inefficient and costly (in terms of resources) to have all sensor nodes transmit their data directly to the final processing center (i.e., server/cloud). They suggest that it is more reasonable for energy-constrained sensor networks to form hierarchical structures where the sensor nodes form clusters and send their data to a local data aggregation node (i.e., cluster head). This central node is typically equipped with richer resources and can collect data, do some pre-processing and then forward relevant data for further processing. Especially in such hierarchical networks, efficient routing is mandatory to preserve the node's limited energy resources [91, 92].

In addition to the challenges stated above, sensor nodes are generally designed without measures for firmware upgrades or bug fixes after deployment or allow upgrades only under specific conditions. A significant number of sensor nodes run specialized OSs [93, 94], such as TinyOS or Contiki, that are not intended for software modifications at runtime. The reasons are mainly cost-considerations, power-consumption limitations, or inaccessibility of the nodes after deployment as well as safety and security aspects.

All of these limitations and unique characteristics of WSN make them a distinct area of communication systems that requires different treatments in terms of concepts, protocols, and algorithms applied (cf. [10]). The area of WSNs and its applications have drawn a significant amount of research interest over the last years as techniques well-established in other kinds of communication networks are mostly not applicable to WSNs.

CHAPTER 3

Anomaly Detection

The threat of faults to the dependability of WSNs and the resulting need for appropriate counter-measures are well-known and widely-researched topics. Especially the detection of sensor node faults is crucial for the WSN's reliability. However, most current node fault detection approaches rely on data anomaly detection and have significant limitations as discussed in this chapter.

With more expansive fields of applications and a correspondingly growing number of use cases for WSNs, appropriate measures to ensure proper functioning, as well as sensor data correctness, are becoming more relevant. Besides the standard soft- and hardware verification and testing activities – sometimes augmented with formal methods – especially runtime measures to detect erroneous behavior/data have become more significant. It is crucial to ensure high confidence in data correctness as all subsequent data analytics can only work correctly as long as the input data are accurate and correct. A wrong datum can influence and alter the outcome, ultimately leading to wrong assumptions, models, or predictions.

One key issue in dealing with faults is that ground-truth values (i.e., expert or domain knowledge) are needed to define proper fault models, or more precisely, their effect on the sensor nodes' behavior or data. Otherwise, we can only refer to deviations from expected behavior or model of the phenomenon, hence, the detection of anomalies [95]. As a consequence, anomaly detection is often used for fault detection approaches in WSNs (cf. [3]). Generally, an anomaly is "*[. . .] an observation which deviates so much from other observations as to arouse suspicion that it was generated from a different mechanism*" [96]. Or to put it into the context of sensor node behavior: "*an anomaly is considered as a state of a given system that is not consistent with the normal behavior of this system*" [97]. In the literature, different terms describing anomalous observations like *outliers*, *abnormality*, *aberrant*, *deviation*, *peculiarity*, or even *surprise* are used [98]. Although sometimes used interchangeably, not all of these terms refer to the same things as discussed in this chapter. However, most anomaly-detection approaches leverage

correlations of the sensors data (e.g., temporal, spatial, or functional) as a substitute for a missing ground-truth [99].

Anomaly detection schemes comprise techniques and algorithms to detect patterns that do "*not conform to an established, normal behavior*" [98]. The primary difficulty in anomaly detection lies in defining what has to be considered "normal" or in modeling the "normal" behavior, respectively. Especially the automatic constructions of models of "normality" from data is not a trivial task due to (i) huge data volumes, (ii) often a highly imbalanced distribution of available data, (iii) the difficulty in finding the boundaries between what is considered normal and abnormal, and (iv) the requirements for continuous adaptation in dynamic systems (cf. [98]). Also, anomaly detection approaches are known to often produce a high number of wrong alerts (sometimes referred to as "crying wolf" [100]). Although anomaly detection generally suffers from such issues that make efficient and thorough anomaly detection a complicated and challenging task, their benefits are significant enough for numerous researchers to invest effort in this topic. Consequently, we surveyed anomaly detection techniques with a focus on their applicability for sensor node fault detection in [3].

3.1 Challenges in Wireless Sensor Networks

To successfully apply anomaly detection to WSNs is by far no trivial task. Anomaly detection schemes are a complex topic as they have to be able to discover novel events in a tremendous amount of data. However, in the context of WSNs, several circumstances make efficient anomaly detection even harder, namely:

- the strictly limited resources of the sensor nodes,
- the restrictions entailed by the sensor nodes' interconnections,
- the communication patterns used in WSNs, and
- the unpredictable nature of their environment.

Node Restrictions

Sensor nodes are designed to be as energy-saving as possible to ensure the required lifespans without battery replacement. For this purpose, the computational capabilities, usable peripheral components, and available memory are strictly limited, making it impossible to use highly complex and powerful detection schemes as commonly used in other areas like computer networks. Additionally, WSNs have to deal with wildly varying numbers of partly heterogeneous sensor nodes ranging from several nodes up to several thousand nodes per network, thus, presenting a noticeable challenge for the scalability of the anomaly detection approaches.

For this reason, algorithms are favored that demand low computational power, low memory, and low energy [101]. Such algorithms are referred to as *lightweight* and are characterized by (i) a minimal overhead, (ii) an efficient distribution of work over the

single components, and (iii) the primary functions of the system are not adversely affected [102]. Examples of such lightweight anomaly detection systems in WSNs are the wrapper-based feature selection algorithm in [103], the lightweight decision tree classification and detection method in [104], and an approach for anomaly detection by sensing the nodes' energy consumption in [105].

Network Restriction

The network interlinking the sensor nodes has strictly limited communication capabilities affecting the available transfer rates and bandwidths for a maximal energy-saving operation. While several WSNs offer bi-directional communication, there are still scenarios where the sensor nodes are equipped with unidirectional communication capabilities, making the control of the nodes or the triggering of specific actions impossible. Additionally, a vast number of different network protocols are used in WSNs (cf. [89, 106–108]). As a result, it is difficult to develop a generic solution covering all possible protocols.

Communication Patterns

The communication patterns commonly used in WSNs significantly differ from most other network applications. While computer networks usually have more or less periodic traffic characteristics, probably with rare bursts in case of node updates/backups, notably aperiodic traffic is rather the exception and, therefore, often an indication of unusual network activity. In WSNs, on the other hand, the sensor nodes either send their data periodically in a continuous "*track & report*" manner or bursty in the case of certain events. For this reason, approaches originating from general-purpose computer networks are mostly unsuitable for WSNs even when leaving their often high power and memory requirements aside.

Environmental Influences

Sensor nodes are often operated under harsh environmental conditions affecting both the nodes' hardware and their wireless communication. The sensor nodes' hardware has to reliably operate in unpredictable and uncontrollable conditions influenced by environmental factors such as significant fluctuations in temperature, high levels of humidity, or strong vibrations that pose a severe risk of physical damage to the node. Fluctuations in air temperature and humidity include the risk of water condensation formation inside the sensor node's housing that may lead to (temporary) short circuits and/or damaged components. The situation is even worse in maritime environments as the combination of sea fog and strong wind results in quick condensation of salty water and fast oxidation of metallic components [32].

Besides the threat to proper node operation, the environmental factors also have a heavy impact on the quality of wireless communication like the temporary blocking of signals by obstacles [87] or the absorption of radio waves by rain and/or the surrounding

vegetation [88]. As discussed in [109], also the ambient temperature directly influences the link quality of the WSN's interconnections.

3.2 Anomaly Detection Metrics

The correctness and efficiency of anomaly detection algorithms and schemes depend on specific criteria. First of all, the technique used in an anomaly detection system significantly impacts its overall characteristics. Second, the field of application is a crucial factor in how good the detection works. With this, factors like how often anomalies occur and how much they differ from "normal" data are relevant. Also, the way how the technique is implemented affects the outcome as most of these methods have several parameters that need to be well adjusted. Nevertheless, the quality of the input data used to assess an approach also influences the resulting metrics.

Without a proper evaluation, one can make no case about an anomaly detection approach's correctness and efficiency, thus, its feasibility and suitability for a specific use-case. Although numerous papers on anomaly detection present their approach's correctness, often expressed in statistical metrics, we argued in [3] that especially for WSNs the evaluation of approaches also needs to take efficiency metrics (e.g., performance) into account. Analogously, three criteria for the assessment of detection approaches are defined in [110, ch. 7][1]: (i) the data quality, (ii) the correctness, and (iii) the efficiency. The corresponding metrics of these criteria with a focus on WSNs are discussed in the following sections.

3.2.1 Data Quality

The first criterion refers to the quality of the data used to evaluate a given approach. Selecting representative and qualitative data is essential for a meaningful evaluation. In this context, well-established benchmark datasets, application-specific practical testbeds, or real-world deployments can be used.

Over the last years, various benchmark datasets (cf. [111, ch. 9]) have been published to assess the correctness and effectiveness of detection schemes and to enable the direct comparison of competing approaches. Most of these datasets offer a multitude of attributes and are usually well-balanced, considering the contained classes/clusters and/or the presence of anomalies/outliers.

Practical testbeds and real-world deployment, on the other hand, deliver application-specific data that can contain data events and occurrences not included in public benchmark datasets, such as the effects of node faults that usually have a component-specific manifestation. As a result, they allow one to assess the detection approach's behavior in a specific application context but, on the downside, often impede the comparison with alternative approaches (except for the case when the acquired data are also assessed with the implementation of alternative schemes).

[1] These criteria refer to intrusion detection systems (IDSs), but are likewise valid for anomaly detection.

3.2.2 Correctness

Correctness metrics such as sensitivity and specificity allow one to make a statement about the suitability of an approach in detecting the events of interest. Several well-established metrics exist to evaluate the correctness of anomaly detection schemes (cf. [110, sec. 7.3]). Most of these common metrics are taken from classical statistics and express certain ratios of one or more of the following numerical values:

- true positives (TP) are instances correctly classified as anomalous
- true negatives (TN) are instances correctly classified as normal
- false positives (FP) are instances falsely classified as anomalous
- false negatives (FN) are instances falsely classified as normal

The following metrics based on these numerical values are commonly used to assess anomaly detection approaches. The true positive rate (TPR) (also called *detection rate* or *sensitivity*) expresses how many of the anomalous instances are correctly detected according to the relation:

$$TPR = \frac{TP}{TP + FN} \ . \tag{3.1}$$

The higher the value of the TPR the more anomalies are correctly detected.

Similar, the true negative rate (TNR) (or *specificity*) expresses the ratio of correctly classified normal instances and is defined as:

$$TNR = \frac{TN}{TN + FP} \ . \tag{3.2}$$

Again, the higher the value, the more normal instances have been labeled correctly.

Contrary, the false positive rate (FPR) (or *fall-out*) states how many normal instances were falsely recognized as anomalous. It is indirectly proportional to the TNR and can be derived with:

$$FPR = \frac{FP}{FP + TN} = 1 - TNR \ . \tag{3.3}$$

In the context of anomaly detection, the FPR is often called false alarm rate (FAR) as it relates to wrongly triggered alerts.

Analogously, the false negative rate (FNR) (or *miss rate*) shows how many anomalies were missed. It is indirectly proportional to the TPR and can be derived with:

$$FNR = \frac{FN}{FN + TP} = 1 - TPR \ . \tag{3.4}$$

For the FPR and FNR a low value is desirable.

To express how good the detection performs for a given task, often the positive predictive value (PPV) (or *precision*)

$$PPV = \frac{TP}{TP + FP} \tag{3.5}$$

and the *accuracy* (or short *ACC*)

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.6}$$

are used.

Since anomaly detection is often interpreted as a binary classification task (an event can be either "normal" or "anomalous"), also the *F1 score* known from statistical analysis is preferred over the simple accuracy metric. It is the harmonic mean of precision and sensitivity and is expressed as

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \ . \tag{3.7}$$

To express the detection characteristics of an anomaly detection scheme, often receiver operating characteristics (ROC) curves [112] are used. ROC curves are a graphical plot used in binary classifier systems to show the trade-off between the TPR and the FPR. In addition, also the area under the ROC curve (AUROC) metric (sometimes simply called area under curve (AUC)) aims to express an approach's correctness concerning its accuracy based on the ROC curve (cf. [113]).

3.2.3 Efficiency

While correctness metrics are often found in related works, most of these contributions lack considering efficiency measures like stability, timeliness, and performance. Such metrics are important as the detection of anomalies has to happen timely and properly [114, 115]. Especially for WSNs, a proper consideration of the performance and reliability is crucial, sometimes jointly considered as "performability" (a term coined by Meyer [116]).

In contrast to the correctness metrics mentioned above, efficiency metrics are often dependent on the implementation and partly the prevalent conditions during the evaluation of the detection system. As an example, the time between the occurrence of an anomaly and its detection is also an essential aspect of detection systems. For distributed approaches, information on the number of nodes that correctly identified the anomaly (true-positive detection) and the number of nodes that falsely classified a normal situation as anomalous (false-positive detection) also need to be considered when assessing the characteristics of the system.

However, most importantly, the sensor nodes in WSNs have strictly limited sources; hence, the efficiency of an approach especially needs to consider the overhead a detection approach imposes on the systems' resources. In particular, the memory footprint of the approach (regarding the consumption of flash, SRAM, and possibly EEPROM memory), the energy overhead, and an impact on the node's runtime behavior (e.g., prolongation of non-sleep phases) are metrics of interest.

3.3 Taxonomy for Anomaly Detection

Over the last years, a large number of approaches and schemes to detect various kinds of anomalies in different WSN applications have been developed. With a growing number of approaches, the need for a proper classification arose. Today, several classification schemes for anomaly detection approaches exist that often focus on certain aspects of (i) the anomalies considered, (ii) the locus of the anomalies, or (iii) the area of application where it is used.

Consequently, we proposed a taxonomy for anomaly detection in WSNs based on previous surveys and classification schemes augmented with own considerations and findings in [3]. As depicted in Figure 3.1, parts of this taxonomy have a more general nature. Unique attributes and characteristics of approaches applicable to WSNs are considered in the works we based our taxonomy on. Additionally, some categories are not entirely complimentary as some approaches combine features of different categories. In contrast to previous taxonomies, we explicitly included bio-inspired detection techniques (i.e., computational intelligence (CI)) as some of them offer suitable characteristics for anomaly detection approaches. Such techniques offer a reasonable compromise between resource requirements and detection efficiency for several WSNs applications.

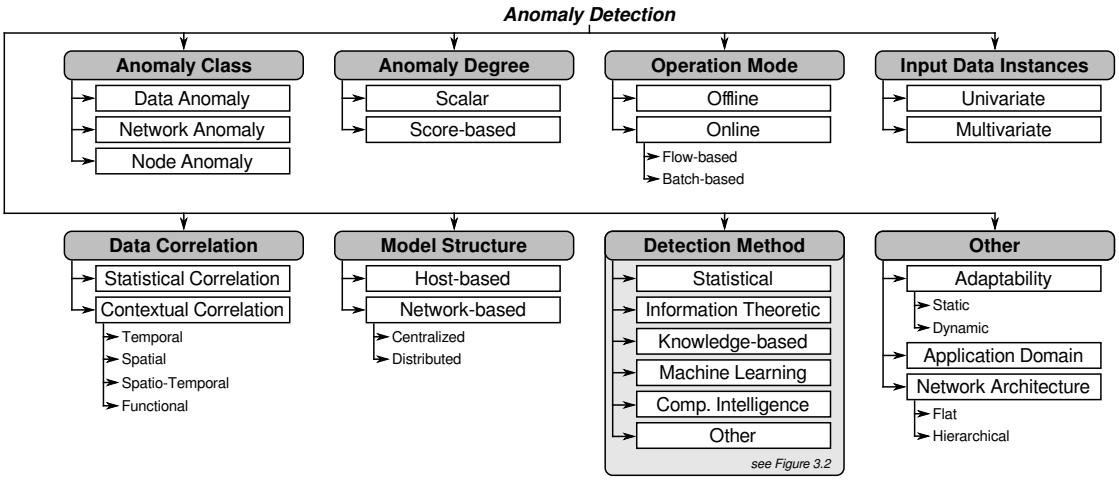


Figure 3.1: Taxonomy for anomaly detection in WSNs

3.3.1 Anomaly Classes

For WSNs, three basic classes of anomalies are defined, namely (i) data anomalies, (ii) network anomalies, and (iii) node anomalies (cf. [100]). A similar classification of anomaly classes was presented in [117] where the authors refer to data anomalies as "events" and to node anomalies as "noise & errors". Regarding network anomalies, they considered "malicious attacks" whereby, as presented below, a much broader range of anomalies can occur on the network.

Data Anomalies

Data anomalies are the oldest and most widely researched class of anomalies. Their study and analysis date back to the early 19th century [118]. Data anomalies refer to "*an observation (or a subset of observations) which appears to be inconsistent with the remainder of that set of data*" [119], thus, sensor readings that deviate from the expectations.

In WSNs, data anomalies can be caused by miscalibrated sensors, faulty hardware, design errors, or unusual phenomena in the monitored environment. It is important to note that data anomalies in sensor readings, be they spatially or temporally, can signify events of interest in the monitored phenomena.

Network Anomalies

Network anomalies are network activities that deviate from what is considered a "normal" network behavior/flow, such as (i) loss of connectivity, (ii) intermittent connectivity, (iii) highly irregular network traffic, (iv) routing loops, or (v) broadcast storms [100]. As with data anomalies, network anomalies can manifest themselves in a temporal, spatial, or spatio-temporal manner.

Network anomalies attracted much research interest in the past as the cause for such irregularities in network behavior often origins from malicious attacks on or intrusions in a network. Thereby two different kinds of attacks are considered in the literature:

1. internal attacks: Such attacks are performed by nodes that are already part of the target network (e.g., infected/altered nodes).
2. external attacks: These attacks are performed from outside to gain access to the target network (e.g., over the Internet).

In WSNs, network anomalies are often caused by physical conditions rather than by malicious attacks (although the latter happens as well). As mentioned above, the harsh environment in which some WSNs have to operate is not very beneficial for the wireless links due to absorbing or blocking effects caused by the surrounding nature. Also, in some scenarios, the connection to one or more sensor nodes may be ultimately lost due to unforeseen events. Examples of such events are (i) heavy rain or mudslides in environmental monitoring applications, (ii) overly heavy tidal forces in maritime applications, or (iii) the damaging or displacement of nodes in monitoring applications by uninformed humans or curious animals. Malfunctioning sensor nodes can also cause irregularities in the (sensor) network due to operational faults or simply because the node's battery is running down.

Node Anomalies

Node anomalies are particular hardware or software problems leading to abnormal behavior of the sensor node (cf. [99]). In this context, two main origins of node anomalies have to be distinguished, those stemming from (temporarily or permanently) changed

operational conditions (like the battery is running down) and those caused by actual defects of the sensor nodes, hence, faults.

While the former is most often an inevitable consequence of regular operation, it can even be accelerated by malicious attacks on the nodes (e.g., denial-of-sleep attacks). The latter, on the other hand, can arise from defective hardware, flawed software, and/or a bad integration of these [100]. As a result, the sensor node can end up with unexpected system performance (e.g., gradual degradation) that possibly influences the sensor node's operation and/or the readings forwarded over the network (i.e., soft faults).

Neighboring nodes can quickly detect faults resulting in node halts or crashes as an absence of network messages. Nevertheless, the effects of soft faults (e.g., silent data corruption) are often difficult or even close to impossible to be detected. Thus, they pose a severe threat to the quality of the service provided by WSNs.

3.3.2 Anomaly Degree

Anomaly detection is often considered as a binary classification problem using a *scalar scale* where data instances are either labeled as "normal" or "anomalous". Several approaches, however, do not label the data instances but provide a measure on the degree to which the instance is considered anomalous, thus, providing an *anomaly or outlier score* [117]. Usually, a threshold is defined (or learned), which defines the levels of anomaly scores that need to be treated. For more information on anomaly scoring techniques, we refer an interested reader to [120].

3.3.3 Operation Mode

Anomaly detection techniques can also be distinguished based on the data sources they process. In this context, two basic types of detection approaches exist: those using historical data and those processing real-time data. While historical data (e.g., data from files or databases) is available as a whole (or at least in blocks), real-time data (e.g., streaming data) becomes continuously available. Consequently, anomaly detection systems are categorized as (i) online or (ii) offline approaches [121].

(i) *Online* approaches analyze the data in real-time by keeping track of currently happening events (e.g., measurements). Such approaches are often denoted as streaming anomaly detection (SAD) as they process the incoming data in a streaming fashion. As discussed in [114], the processing can thereby be done in a continuous manner (i.e., *flow-based*) or based on packets of data (i.e., *packet- or batch-based*).

(ii) *Offline* techniques, on the other hand, process stored information to decide if an anomaly has happened. Such offline detection approaches often incorporate more powerful methods and larger amounts of data.

3.3.4 Input Data Instances

The anomaly detection approaches can be further categorized depending on the number of input data attributes that they consider. In WSNs, the sensor data is usually considered as a continuous stream of integer or real-valued data. A single sensor node can have one or more sensors equipped, thus, delivering one or more sensor measurements per time instance.

Consequently, anomaly detection approaches can be divided into techniques considering single attributes (i.e., *univariate data*) and techniques that take several attributes into account (i.e., *multivariate data*) [117, 121]. The advantage of the analysis of multivariate data is the possibility to identify anomalies only detectable by leveraging correlations between the attributes (see Section 3.3.5 and [122]).

3.3.5 Data Correlations

Anomaly detection approaches can also be categorized based on correlations and redundancies in the input data they exploit or dependencies among their attributes in the case of multivariate data [117]. This differentiation is essential as many of the proposed approaches base their detection on assumptions made on the nature (or granularity) of the data, or more precisely, on correlations between several instances [121].

The two main types of correlations leveraged in anomaly detection are [120, 122]: (i) statistical correlation and (ii) contextual correlation as discussed below.

Statistical Correlations

Statistical correlations refer to relations between the individual data instances of a data set. Such correlations can be used to identify:

- *point anomalies* (i.e., individual data instances are considered anomalous concerning the rest of the data set).
- *collective anomalies* (i.e., a group of data instances that is considered as anomalous concerning the rest of the data set).

Contextual Correlations

Contextual correlations make use of dependencies between the history of several sensor measurements and/or the measurements of neighboring nodes [117]. Four categories of contextual correlations can be distinguished (cf. [100]):

(i) Temporal correlations

(ii) Spatial correlations

(iii) Spatio-temporal correlations

(iv) Functional correlations

(i) *Temporal correlations* consider sensor readings that are temporally ordered and where a reading is compared to its preceding and subsequent sensor readings. Such correlations help to identify temporal anomalies possibly manifested as high variability in subsequent sensor readings, lack of change in sensor readings, gradual reading skews, or out-of-bound readings [100].

(ii) *Spatial correlations* allow to detect deviations of sensor readings of one node in comparison with the data acquired by its surrounding nodes (i.e., neighbors). As a result, it can be beneficial to take information on the location of the neighbors into account (e.g., floor plan for indoor applications and geographical attributes for outdoor deployments).

(iii) *Spatio-temporal correlations* use a combination of the two types above. The sensor readings are compared concerning time (temporal) and place (spatial) by considering the information stored on the sensor node itself and its neighbors. This form of correlation allows to detect more sophisticated anomalies possibly caused by miscalibrated sensors, faulty hardware, design errors, or unusual phenomena in the monitored environment.

(iv) *Functional correlations* exploit relationships and dependencies between several attributes of the sensor data to detect unreasonable or even impossible combinations. Therefore, additional information on the particular attributes and their relationships is needed to acquire the required "context". For example, a temperature reading is related to humidity and barometric pressure values due to physical laws [117].

3.3.6 Model Structure

Another classification criterion of anomaly detection techniques is the locus of detection. Here, two kinds of anomaly detection approaches are distinguished (cf. [121, 123]): (i) host-based and (ii) network-based detection systems.

(i) In *host-based* or local approaches, the analysis of events and the detection of anomalies are performed directly on the nodes (in a stand-alone manner).

(ii) *Network-based* approaches are further divided into *centralized* and *distributed* detection systems [100]. Centralized detection systems are placed on valuable points within the network where they can monitor the traffic to and from all relevant devices of the network, possibly by leveraging information from several communication layers [124]. Distributed approaches, on the other hand, usually rely on the cooperation of several nodes.

As stated in [125], in WSNs usually distributed approaches are preferred over centralized solutions since they often require lesser resources as the detection task is shared between several nodes. Additionally, such approaches usually scale much better with a growing number of network participants and often offer better robustness, reliability, and modularity than centralized solutions.

3.3.7 Detection Method

The most important criterion in choosing an anomaly detection approach is its underlying detection method or detection strategy in general. In this context, we also consider using specific metrics to detect anomalies as methods. This category offers the most remarkable diversity as ideas and concepts from several areas have been adapted and adopted for anomaly detection. For this reason, many reviews and surveys focus their classification scheme purely on the detection method applied.

The main classes of methods are depicted in Figure 3.2 which is part of the taxonomy shown in Figure 3.1. Examples and references for the particular methods can be found in the surveys and review articles reference in the respective parts as well as in our summary and comparison of these presented in Section 3.4.

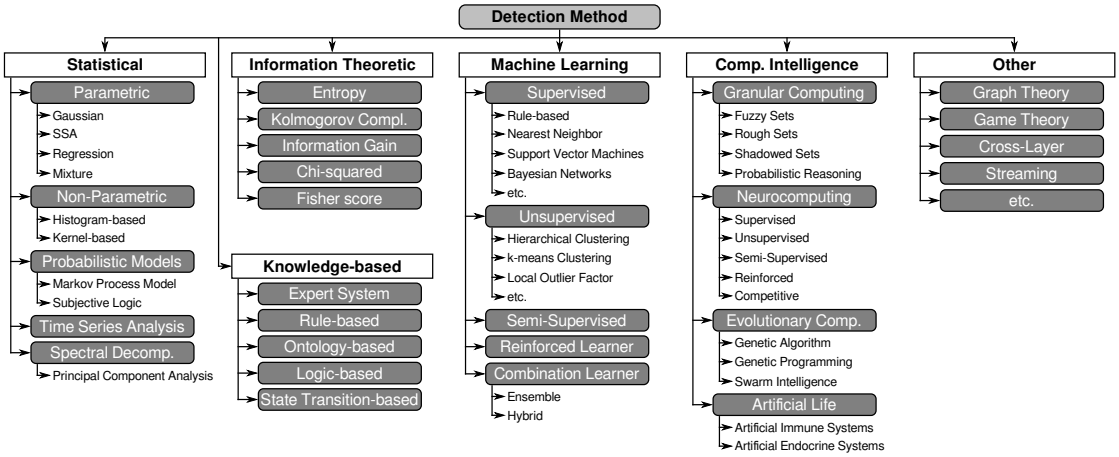


Figure 3.2: Taxonomy for anomaly detection methods

3.3.7.1 Statistical

Statistical anomaly detection methods and algorithms are based on statistical populations or models. It encompasses techniques such as (i) probability theory, (ii) real analysis, (iii) measure theory, (iv) asymptotic theory, (v) Markov chains, (vi) martingales, and (vii) ergotic theory [126]. There are two ways to build the (stochastic) model depending on which information is used [120]:

- based on *a priori* information (i.e., parametric modeling), or
- from the data itself (i.e., non-parametric modeling).

Parametric approaches rely on models based on *a priori* information of the data's distribution. The most common parametric model is Gaussian distribution [117]. Nevertheless, also non-Gaussian models such as segmented sequence analysis, regression, or mixture models have been successfully applied [121].

Non-parametric approaches do not use any prior knowledge about the data distribution and build their model based on the input data itself. The two most common non-parametric techniques are based on histograms or kernel functions [121].

Additionally, probabilistic models (e.g., Markov process models, subjective logic) and time series analysis to identify abnormal data instances have been proposed [127].

As presented in [114,128], methods based on spectral decomposition can identify deviations in the input data, too. For this purpose, several attributes of the input data instances are analyzed to find inherent dependencies. The most prominent of these methods is the principal component analysis (PCA) that, however, is more often used for dimensionality reduction of the input data [114] than for anomaly detection.

3.3.7.2 Information Theoretic

Anomaly detection methods based on information theory usually rely on metrics such as entropy since anomalies are assumed to distort the information content of the input data instances, thus, altering the entropy. Also, approaches based on Kolmogorov complexity, information gain, chi-square, or the Fisher score have been proposed [114, 129].

3.3.7.3 Knowledge-based

Especially for network anomaly detection, various knowledge-based methods have been proposed since a portion of the abnormal behavior is caused by malicious attacks on the network. The characteristics of these attacks are captured and modeled in the form of rules (possibly with associated knowledge, i.e., expert systems), expressive logic structures, by leveraging the knowledge of the domain (i.e., ontology-based) as well as by analyzing the state transitions of the system [110].

3.3.7.4 Machine Learning

Methods based on machine learning (ML) are increasingly used in anomaly detection since they are often capable of improving their detection accuracy while reducing the number of false alerts by adapting their model to the characteristics of the system and environment applied to [110]. Today, several light-weight and energy-efficient ML-based methods are available [130].

A major distinction in ML techniques is whether pre-knowledge is available and how feedback is incorporated into the model. ML schemes can be classified as:

1. supervised learner
2. unsupervised learner
3. semi-supervised learner
4. reinforced learner
5. combination learner

However, most learning techniques require (or implicitly assume) that normal instances occur far more frequently than anomalies [110].

Additionally, a review article on anomaly detection in WSNs [131] found that the majority of the approaches used in the past are purely statistical rather than learning-based methods. The reason for this is that learning approaches usually require more resources which are often not available in WSNs. Nevertheless, more and more approaches adapt and adopt ideas and methods from machine learning in resource-constrained environments.

Supervised Learning

Supervised learning techniques require a pre-labeled learning data set from which the algorithm infers a model based on the relation of input to output data. Thus, the models resulting from supervised learning are usually referred to as *predictive models*. The main learning problem associated with supervised learning is classification. Common supervised classification techniques suitable for anomaly detection encompass (i) Bayesian networks, (ii) support vector machines (SVMs), (iii) decision trees, or (iv) rule-based machine learning algorithms [114,117,120,121]. Although being treated separately in some surveys [117, 121, 128], also nearest neighbor algorithms are classification techniques, thus, supervised techniques. Nearest neighbor techniques identify anomalies by calculating the distance of the data instances to their respective neighbors. Anomalies are then identified either distance-based or density-based [114, 121].

Unsupervised Learning

Unsupervised learning techniques do not require pre-labeled data. They aim at finding patterns in data instances in a self-organized manner. Thereby, the data is usually grouped into clusters depending on the relation of the particular data instances with their neighboring instances. Standard clustering algorithms include hierarchical clustering, k-means clustering, density-based clustering, and local outlier factors [114, 117, 121]. As the resulting model tries to describe the relationships between the data instances, the term *descriptive model* is commonly used.

Semi-Supervised Learning

In semi-supervised learning, the learning process employs unlabeled data in addition to labeled data, whereby the amount of labeled data is usually much smaller than unlabeled data. One advantage of such techniques is reducing effort for properly labeling test data, a task that generally requires human interaction. Examples for approaches that apply semi-supervised techniques to anomaly detection are discussed in [132, 133].

Reinforced Learning

Like in unsupervised learning, also in reinforced learning, no labeled training data set is required. The model is trained in a goal-oriented manner. It takes decisions (or actions) to maximize its output or reach a particular objective. Thus, the algorithm learns to

react to its environment to reach a defined goal properly. Thereby, the decisions are reinforced by a certain kind of reward or penalty, depending on whether they help to improve the solution.

Combination Learners

Combination learners leverage the strengths of their parent techniques while minimizing their limitations and weaknesses [134]. Thereby, such combined techniques have shown to be often more effective and efficient than their respective parents. As presented in [123], there are two flavors of combination learners:

- tightly-coupled systems (or *hybrid systems*) where the techniques are combined inseparably
- loosely-coupled systems (or *ensemble systems*) where the techniques are connected, but distinguishable (like modules)

3.3.7.5 Computational Intelligence

Many learning schemes have been developed over the years that took inspiration from natural processes. Learning methods inspired by biological phenomena are referred to as computational intelligence (CI) [135] and belong to the broader class of complex adaptive systems (CAS). Such techniques usually offer (i) self-similarity, (ii) self-organization, (iii) emergent behavior, (iv) highly decentralized control, and (v) adaptability [136].

In an extensive survey in [123], Wu and Banzhaf argue that CI and artificial intelligence (AI) are often used synonymically, whereas there are distinct characteristics between these two fields. They state that AI and CI differ in their basic approach (top-down vs. bottom-up modeling), the problem representation (symbolic vs. numeric), and the cognitive functions applied (high-level vs. low-level). Also, CI is sometimes considered a synonym for soft computing. However, the former refers to nature-inspired computational techniques and the latter to the use of approximate calculations for complex computational problems. However, the boundaries between CI and ML (respectively AI) are not always apparent. In this work, we follow the classification presented in [135] which encompasses: (i) granular computing, (ii) neuro computing, (iii) evolutionary computing, and (iv) artificial life.

Granular Computing

Granular computing describes techniques for arranging entities (so-called granules) based on their characteristics (e.g., similarity or functional adjacency) [135, 137]. One common problem of anomaly detection approaches is crisp discrimination between “normal” and “anomalous”. As a result of this, the use of fuzzy rule-based approaches can help to introduce some “fuzziness” to the distinction [138]. Similarly, other granular concepts such as rough sets, shadowed sets, or probabilistic reasoning have been used [129, 139].

Neurocomputing

Neurocomputing (i.e., artificial neural networks (ANNs)) encompasses computational models inspired by the human neuron system [130]. Since ANNs can be used for supervised, semi-supervised as well as unsupervised learning tasks, they are often counted as ML techniques. Whether ANNs belong to ML or CI often depends on the characteristics of the ANN approach used. For supervised learning, two primary models depending on the type of feedback system are used (i.e., feed-forward and recurrent ANNs) [123]. The two most common unsupervised ANN models are self-organizing maps (SOMs) and the adaptive resonance theory. In the past, several approaches applying supervised or unsupervised ANNs for anomaly detection have been proposed [114, 138, 140] including deep learning methods (denoted as deep anomaly detection; cf. [141]).

Evolutionary Computing

Evolutionary computing is based on the principle of "survival of the fittest" inspired by the (natural) selection, crossover, and mutation found in natural systems [134]. Besides the most known genetic algorithms (GAs), there exist more approaches that exploit evolutionary processes, such as genetic programming (GP) and swarm intelligence (SI) [136]. In contrast to GAs and GP, the individuals within a swarm interact with each other rather than being modified by crossover or mutation operators. Although being usually used for optimization and problem-solving tasks, such evolutionary techniques have shown beneficial properties for discovering classification rules or clusters for adaptive anomaly detection [114, 123, 138, 140].

Artificial Life

Artificial life refers to techniques mimicking the behavioral characteristics of natural living systems. It encompasses techniques such as artificial endocrine systems and artificial immune systems (AISs).

Artificial endocrine systems are models of the human endocrine system that follows a homeostatic regulation approach based on chemical messengers (i.e., hormones) [142]. They are more suitable for fault-tolerant approaches than anomaly detection techniques [143].

AISs are based on models derived from the human immune system (HIS) [102]. The immune system is a widely distributed and inherently parallel network of a significant number of diverse entities. These entities are working simultaneously and in cooperation with each other to build up a highly complex self-organizing system with properties such as error-tolerance, adaptation, and self-monitoring [144]. Today, most AIS approaches are derived from one of four "classical AIS theories" (see Section 5.4):

- negative/positive selection
- clonal selection
- immune network theories
- danger theory

3.3.7.6 Other Detection Methods

Besides the methods mentioned above, several other techniques have been applied for the task of anomaly detection, such as graph theory, game theory, and cross-layer approaches [140] as well as streaming techniques [114] that exploit flow information to detect deviations.

3.3.8 Other Criteria

Additional characteristics and criteria of anomaly detection schemes are presented in the following. References to these criteria were only found in a minor number of related works. However, they still present valid criteria for the classification of anomaly detection approaches in WSNs and, therefore, have been included in our taxonomy.

3.3.8.1 Adaptability

In some WSN applications, the characteristics of the sensed environment as well as the operational parameters of the sensor nodes, the "normal behavior", can change over time. For this reason, the work in [121] differs between non-adaptive schemes (based on *static* models) and adaptive schemes (based on *dynamic* models). In dynamic systems, a corresponding updating of the system is crucial to maintain effective anomaly detection.

3.3.8.2 Application Domain

In [122], anomaly detection schemes are additionally classified based on the application domain for which they were developed. Such a distinction is feasible as different fields of application exhibit some inherent requirements that need to be considered when choosing a particular detection scheme (e.g., the criticality of false positives or time restrictions).

3.3.8.3 Network Architecture

A further criterion of anomaly detection schemes is based on the underlying network architecture. As presented in [115], centralized as well as distributed approaches for different architectures have been proposed in the past.

3.4 Related Work on Anomaly Detection in WSNs

Over the last years, a significant number of anomaly detection approaches tailored to the specific needs and requirements of WSNs have been proposed. Applying anomaly detection in WSNs helps to discover possible attacks or inconsistencies in the network, identify malicious or faulty sensor nodes or detect corrupted sensor readings. As a result, the overall energy consumption and, thus, the network's lifetime can be improved by mitigating possible attacks, removing impaired sensor nodes, and preventing the transmission of misleading data (i.e., outliers) [145]. In this thesis, however, we focus on applying anomaly detection schemes to identify sensor node faults.

Several classification schemes for anomaly detection have been proposed to cope with the vast amount of anomaly detection approaches over the years and their particular characteristics. In [3], we presented a comprehensive overview for anomaly detection focusing on WSN-specific techniques published in the years 2007–2019. Additionally, we provided a meta-survey on anomaly detection approaches as summarized in Table 3.1. In the table, the classification criteria used in the previous works in reference to our taxonomy [3] are highlighted.

Most of the works classify the anomaly detection approaches based on the underlying detection techniques, with some of them also taking the model structure into account [86, 93, 115, 122, 123, 125, 130, 147–150, 153]. Hereby, prevalently statistical and machine learning techniques are discussed [93, 117, 121, 122, 146, 148, 152–154, 156], where many account ANNs as ML rather than CI technique [86, 93, 100, 115, 120, 147–149, 151, 153]. As can be seen in Table 3.1, the most considered class of anomalies are network anomalies due to the broad use of anomaly detection in network security systems like IDSs [93, 115, 123, 125, 130, 148–151, 153]. Surprisingly, the number of works considering correlations in the input data is lower than expected, although the context is an essential factor in anomaly detection systems. More extensive classification schemes including a larger variety of criteria are presented in [100, 117, 120, 140, 146, 151, 152, 154]. To date, the most comprehensive schemes were presented in [121, 129, 139, 156]. However, none of these other taxonomies covers the complete set of criteria of our taxonomy presented in [3].

3.5 Limits of Anomaly Detection for Fault Diagnosis

Concerning the use of anomaly detection for fault diagnosis in WSNs, only a few anomaly detection approaches consider fault-induced node anomalies. Only a small number of approaches incorporated contextual information to detect faulty nodes. As presented in [117], using spatial similarity of the sensor data from neighboring nodes is one way to detect erroneous nodes as events in the monitored environment are assumed to be spatially correlated while faults are not. On the contrary, the work in [100] points out that detecting fault-induced anomalies or node errors by neighboring nodes may not work as such defects do not always exhibit any detectable symptoms by other network participants. Thus, we argue that node anomaly detection approaches need to consider node-level information. Only such information can help distinguish sensed data events from fault-induced deviations (i.e., errors).

Table 3.1: Classification criteria of anomaly detection taxonomies

			Class								Method						Other		
	Authors	Year	*Data*	*Network*	*Node*	Degree	Mode	Data	Corr.	Model	*Statistical*	*Inf. Theory*	*Knowledge*	*ML*	*CI*	*Other*	*Adaptability*	*Application*	*Architecture*
Taxonomy	Sebestyen *et al.* [122]	2018	●	○	○	○	○	●	●	○	●	○	○	●	○	○	○	●	○
	Vasilomanolakis *et al.* [125]#	2015	○	●	×	○	○	○	◑	●	○	○	○	○	○	○	●	×	×
	Wu & Banzhaf [123]#	2010	○	●	×	○	◑	○	●	●	○	○	○	○	●	○	◑	×	×
	Zhang *et al.* [146]	2007	●	○	●	●	◑	●	●	○	●	○	○	●	○	◑	○	●	○
Review / Survey	Kumar *et al.* [147]	2019	●	●	●	○	◑	○	◑	●	○	○	○	●	◑	○	○	○	●
	Kurniabudi *et al.* [129]	2019	●	●	●	●	○	●	●	●	●	●	●	●	●	◑	○	○	○
	Zamini & Hasheminejad [139]	2019	●	●	●	◑	●	●	●	●	●	○	○	●	●	○	●	◑	○
	Zhang & Xiao [148]	2019	○	●	○	○	○	○	○	●	●	○	○	●	◑	●	○	○	●
	Alaparthy *et al.* [149]	2018	○	●	○	●	○	○	○	●	○	○	○	○	◑	○	●	○	○
	Usman *et al.* [140]	2018	●	●	●	○	○	◑	●	●	●	○	○	●	●	●	◑	○	○
	Duhan & Padmavati [150]	2016	○	●	○	○	○	○	○	○	●	○	○	●	●	●	○	○	●
	Can & Sahingoz [86]	2015	●	●	●	○	○	○	○	●	○	○	○	◑	◑	◑	○	○	○
	Butun *et al.* [151]	2014	○	●	○	●	○	●	○	●	●	○	●	●	◑	○	◑	○	●
	O'Reilly [152]	2014	●	○	○	○	●	●	●	●	●	○	○	●	○	○	●	○	○
	Alrajeh & Lloret [130]	2013	○	●	○	○	○	○	○	◑	○	○	○	○	●	○	○	○	◑
	Ghosal & Halder [153]	2013	○	●	○	○	◑	○	○	●	●	○	○	●	◑	◑	◑	●	●
	Rassam *et al.* [121]	2013	●	●	●	○	●	●	●	●	●	○	○	●	○	◑	●	●	●
	Jurdak *et al.* [100]	2011	●	●	●	○	◑	○	●	●	●	○	◑	●	◑	◑	○	○	○
	Xie *et al.* [115]	2011	○	●	○	○	○	○	○	○	●	○	○	◑	◑	◑	○	○	●
	Lim [154]	2010	●	●	●	○	●	●	◑	●	●	○	○	●	○	○	◑	○	◑
	Zhang *et al.* [117]	2010	●	○	○	●	○	●	●	○	●	○	○	●	○	○	●	○	○
	Chandola *et al.* [120]	2009	●	○	○	●	○	●	◑	●	●	●	○	●	◑	○	○	●	○
	Farooqi & Khan [155]	2009	○	●	○	○	○	○	○	●	◑	○	○	◑	◑	○	○	○	○
	Rajasegarar *et al.* [156]	2009	●	●	●	○	●	●	●	●	●	○	○	●	○	○	●	○	◑

● considered ◑ partly considered ○ not considered × not applicable # not WSN-specific

CHAPTER 4

Sensor Node Fault Detection

As discussed in the previous chapter, one branch of fault detection schemes originates from anomaly detection as faults often cause abnormal behavior of sensor nodes (or their data). However, previous anomaly detection approaches for fault diagnosis have significant limitations, especially concerning the detection of sensor node faults. Consequently, we will focus on the peculiarities of sensor node faults and their detection in this chapter.

Wireless sensor nodes are prone to faults due to design issues (i.e., implementation flaws) or unforeseeable effects caused by the harsh environment. While one may argue that the former should be taken care of during the development phase, the latter contains highly dynamic effects caused by the interaction of the hardware components with the physical circumstances of the surrounding environment and can not be adequately captured with simulations or the like [99]. For this reason, runtime measures such as diagnostic or monitoring tools are required to ensure that failures of underlying components do not significantly influence the availability of the data acquisition service.

The difficulty in dealing with faults in WSNs is their unpredictable and diverse manifestation, often depending on the actual implementation and area of application. In [95], the authors report that they found faults to appear infrequently, but with no noticeable spatial or temporal correlation among them. On the other hand, several works concluded that faults occur regularly in WSNs [99,157]. Several studies confirmed significant changes in the wireless link quality caused by ambient temperature fluctuations showing diurnal (day/night) as well as seasonal (summer/winter) patterns [109].

The difference in experiences may stem from different views on the system and inconsistent use of terms. Several WSN applications use the inherent redundancy provided by densely deployed sensor nodes to enable some extent of fault tolerance. Such mechanisms help reduce failures of the data acquisition service by disrupting the propagation of faults throughout subsequent components. Faults can occur in different parts of the WSN where uncaught faults can have effects on the overall service. In this thesis, we focus

on the effects and the detection of sensor node faults. For a broader discussion on fault tolerance in WSNs including network and sink faults, we refer to [158, 159].

4.1 Danger posed by Node Faults

Sensor nodes are key components of WSNs, and they significantly influence the WSN's dependability, especially concerning the reliability of the data sources. They need to operate in a reliable and energy-efficient way to ensure accurate data acquisition while operating unattended for long times. These issues pose significant challenges to the sensor node design as the combination of (i) low-cost components, (ii) limited resources (especially energy), and (iii) the often harsh environmental conditions make the sensor nodes susceptible to impaired operation.

First, the hardware and software characteristics of sensor nodes are limiting factors. WSNs usually involve the deployment of a large number of nodes. This fact implies that the cost of the sensor nodes has to be as low as possible. The sensor nodes are usually equipped with low-cost components and strictly limited resources in terms of processing power and memory sizes to keep the costs low. Additionally, most sensor nodes are battery-powered and have to run for long times without the need for battery replacement or manual charging. As a result, the sensor nodes have to be as energy-efficient as possible, which in turn prohibits the use of resource-intense computations [158]. The strictly limited resources prevent the use of established reliability concepts such as redundancy by duplication or replication on the node level [160]. At the same time, the required functionality is becoming more complex [75].

Second, sensor nodes are usually deployed in an uncontrollable and unpredictable outdoor environment close to the monitored phenomena. The environment poses harsh conditions for proper operation, such as highly dynamic changes in temperature, high humidity, vibration or shocks, and the risk of physical impacts caused by animals or the surroundings (e.g., mudslides). Such effects can neither be entirely foreseen by developers nor completely covered by prior system tests or simulations. Additionally, environmental conditions such as high temperatures can accelerate the aging of the hardware components. A summary of studies on the effects of environmental conditions on sensor nodes can be found in [157].

On top of that, the limited resources of sensor nodes and the most often unprotected outdoor deployment make sensor nodes an easy target for attackers. In this thesis, we focus on node-fault detection and exclude security considerations. For security aspects, we refer an interested reader to the review on intrusion detection systems in [150].

Due to these reasons, sensor nodes are known to suffer from diverse faults that can affect (i) their hardware, (ii) their software, and/or (iii) their communication capabilities [158]. While software faults are primarily caused by development flaws or issues in the interaction with hardware components, the hardware part of the sensor node additionally has to deal with the imperfections of the natural world manifested as physical faults. Especially

temperature-dependent and aging effects have been shown to cause unpredictable behavior in hardware components regarding their severity as well as frequency and likelihood of occurrence. As a consequence, faults occurring on sensor nodes are often the norm rather than the exception, and, thus, the data reported by sensor nodes can become unreliable and inaccurate [161].

4.2 Terminology

In the context of fault detection in WSNs, several terms and definitions can be found that are not always used consistently in the literature. Additionally, the proper use of the corresponding terminology depends on the scope of the target system. For this reason, we will discuss the terminology used within this thesis in the following. Aside from an presentation of the basic dependability terminology (Section 4.2.1), we will especially highlight the relation of anomalies and the effects of faults (Section 4.2.2). In addition, we define our focus of consideration in Section 4.2.3 to show the boundaries of our target system and, thus, justify our use of the terms "faults", "errors", and "failures".

4.2.1 Chain of Dependability

Although the majority of works follow the dependability terminology proposed by Avizienis *et al.* [162] which also serves as the basis for the notion of dependability defined by the *IFIP Working Group 10.4 on Dependable Computing and Fault Tolerance*[1], the terms *faults*, *errors*, and *failures* are sometimes used inconsistently in the literature.

According to [162], a *fault* is a static defect in software or hardware components that can be either human-made (i.e., design fault), be related to the imperfections of the natural world that affect the hardware (i.e., physical faults), or can be caused by the interaction with external components (i.e., interaction faults). In case of design faults, the term *bug* is commonly used. A fault is *active* if it leads to an error, that is, an incorrect internal state such as a deviation from correctness or accuracy; otherwise, the fault is *dormant*. An error can propagate and ultimately lead to an observable deviation of the component's behavior from its specification, which is called a *failure*.

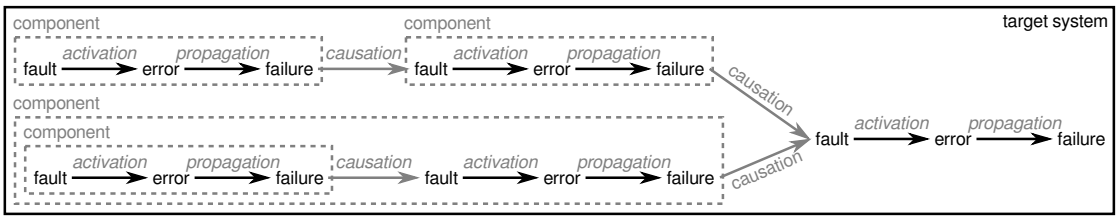


Figure 4.1: Fundamental chain of dependability (after [162, Fig. 10 and 11])

[1] IFIP Working Group 10.4, see https://www.dependability.org/wg10.4/

As depicted in Figure 4.1, a failure of one component can be the causation of a fault in a subsequent or superior component. It can eventually lead to the failure of the target system (i.e., system failure). This effect is covered by the *fundamental chain of dependability* and is a crucial issue for reliability considerations. Nevertheless, the classification of whether an undesired effect counts as fault or failure depends on the focus of considerations, that is, where the system or component boundaries are drawn.

The larger and more complicated a system is, the higher the probability of faults and, in turn, the higher the chance that a fault in an underlying component can lead to a system failure. In the case of a WSN, the situation is even worse as it usually consists of a large number of components (i.e., sensor nodes and cluster heads) that together form the system and contribute to the system's functionality (i.e., complex system). As shown in Figure 4.2, faults in the sensor nodes can propagate through the network and, in the absence of countermeasures, can cause the system to operate incorrectly or even crash completely. For this reason, it is essential to apply specific measures to prevent the propagation of component failures up to the system level and, thus, make the system fault-tolerant. Common practices include, for example, redundancy [163, 164], fault detection and mitigation [81, 165], or homeostatic approaches [166].

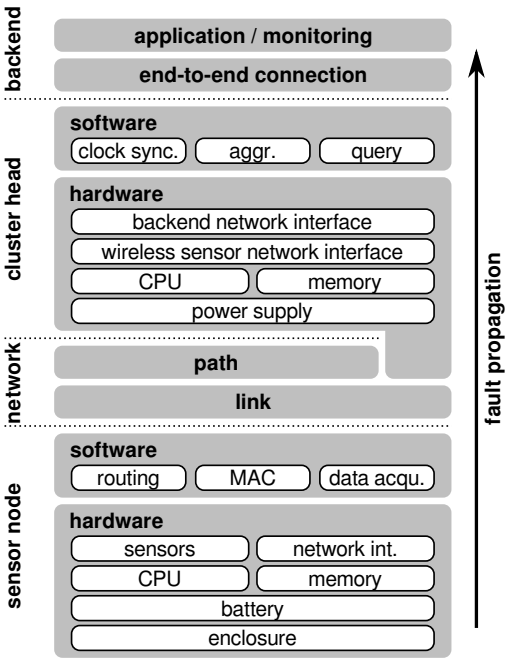


Figure 4.2: Fault propagation in wireless sensor networks (after [167, Fig. 1])

Measures to decrease the probability of having faults in a system are referred to as *fault avoidance*. Techniques to prevent active faults from causing erroneous systems states are denoted as *fault masking* and *fault tolerance* comprises actions to reduce the risk of errors leading to failures (see also Figure 4.3).

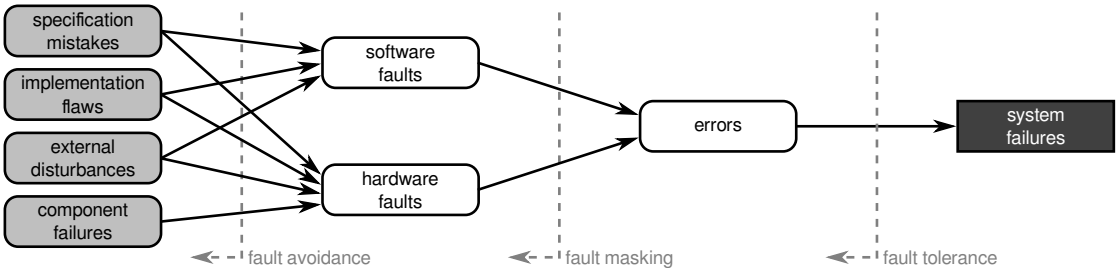


Figure 4.3: Cause-and-effect relationship of faults (after [168, Fig. 2])

Depending on the level where the fault-tolerant measures are applied, we can distinguish between:

- system-level fault tolerance,
- network-level fault tolerance, and
- node-level fault tolerance.

However, measures on all levels need to cooperatively work together to achieve a high degree of reliability. Nevertheless, depending on which level the measures are applied and where the focus of the system is laid (i.e., the boundaries), the terms *faults*, *errors*, and *failures* are sometimes confused and, thus, are used inconsistently in the literature.

4.2.2 Anomalies vs. Node Faults

The detection or diagnosis of faults generally suffers from one crucial problem as quoted from Ni *et al.* in [99]:

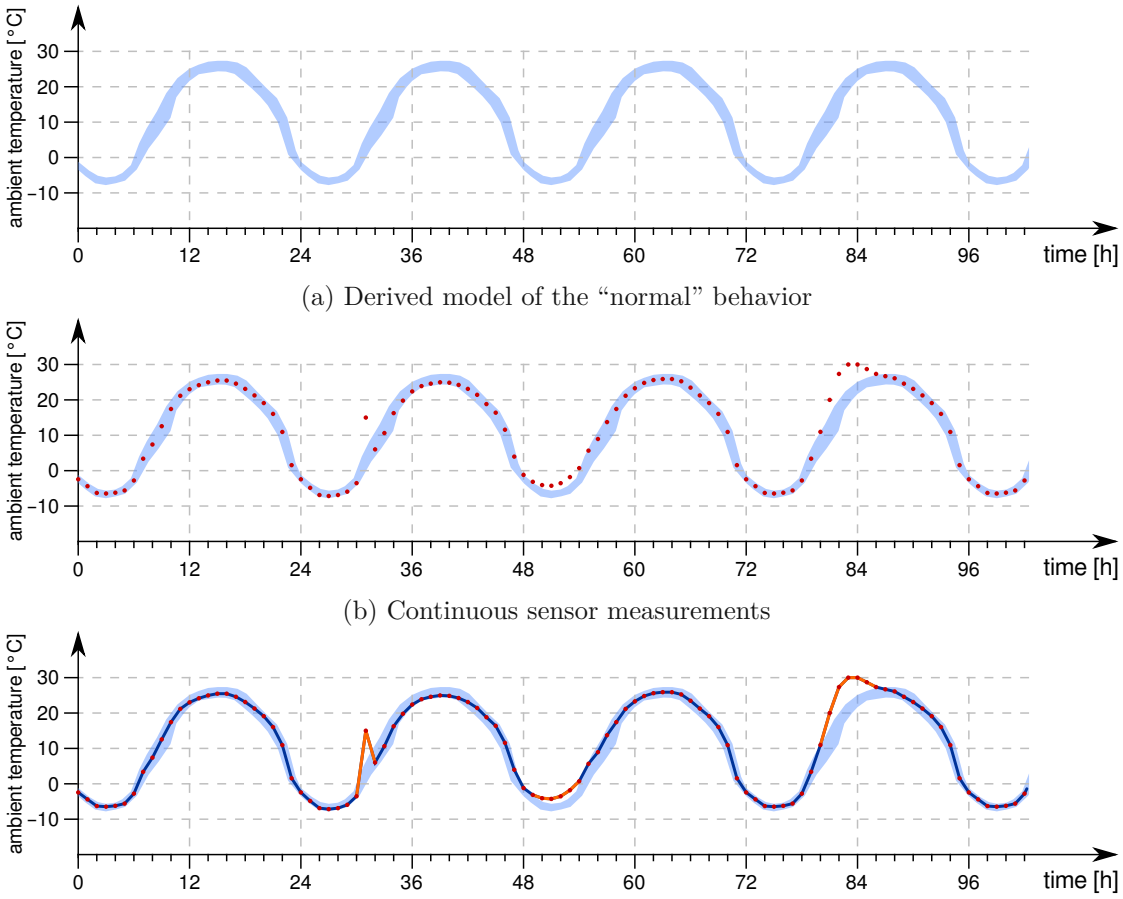
> "*Unless ground truth is known or given by something with high confidence, the term fault can only refer to a deviation from the expected model of the phenomenon.*"

Fault detection requires the effects of the considered faults to be known (i.e., their manifestation in the system behavior or data). Consequently, ground-truth values derived from expert or domain knowledge are required. If this information is not available, we can only detect deviations in the behavior (of the system or the data), that is, anomaly detection (see Chapter 3).

Therefore, the detection of faults is often considered an anomaly detection task and is purely based on the sensor data. This assumption, however, suffers from a crucial problem: anomalies do not need to be caused by faulty sensor nodes. They can also be the result of a rare but proper event in the sensed phenomena [5, 169]. Additionally, faulty sensor nodes can report incorrect sensor values that mimic non-faulty data [170].

As an example, consider a WSN used to continuously monitor and report the environmental conditions (i.e., temperature) of a particular area of interest. Based on prior

domain knowledge, available historical data, or after collecting sufficient data, a model of the expected (or normal) behavior of the temperature curve can be derived (see the blue-shaded area in Figure 4.4a). This process is usually done on a central point with sufficient resources, such as a cloud server. In outdoor applications, the temperature typically follows a diurnal pattern with day and night cycles. As the WSN continues to monitor the temperature, continuously new data instances become available depicted as red dots in Figure 4.4b. When analyzing the newly arriving data regarding the expected behavior (i.e., the "normal" model), certain deviations can be found in the reported data. These deviations can be manifested as drifts, offsets, or outliers as shown by the orange regions in Figure 4.4c.



(a) Derived model of the "normal" behavior

(b) Continuous sensor measurements

(c) Deviations in the data instances

Figure 4.4: Anomaly detection in an environmental monitoring example

The question is whether these anomalies in the sensor data stem from proper but rare events in the monitored phenomena or are deviations caused by faults in the sensor network. On the higher level of the data processing chain (e.g., the cloud), both effects are hard to distinguish or even impossible if no further information is available. For example,

a spike in the temperature curve may be a strong indicator of a fault, but can also be caused by direct sunlight that hits the area where the temperature is measured. So far, the distinction between data anomalies caused by actual events from those resulting from faults has only been sparsely addressed [171]. However, this distinction is crucial as anomalies do not need to be caused by faults and, on the other hand, faulty sensor data can mimic technically valid behavior. For this reason, the ability of a fault detection approach to distinguish data events from fault-induced variations is within the focus of this thesis.

4.2.3 Scope of Considerations

In this thesis, we follow the dependability terminology of Avizienis *et al.* [162]. Our target system is the entire WSN as it is cooperatively responsible for the data acquisition. Consequently, failures of the sensor nodes or their components are considered to be faults from a system-level perspective. To avoid confusion, within this thesis, we use the term "WSN" to refer to the entire system while the term "network" refers to the interconnects between the network participants.

4.3 Fault Taxonomy

The sources and manifestations of faults in WSNs are very diverse (cf. [162]). Faults can originate in different parts of the system, causing failure modes of different severities. A faulty component does not always cause the system to fail in the same way. In the following, we discuss the diverse kinds of faults based on the taxonomy of wireless sensor network faults, which we presented in [8]. A graphical representation of the taxonomy is depicted in Figure 4.5.

While parts of this taxonomy are generally applicable, it is tailored to the characteristics of WSNs especially concerning their hardware components, network structures, and fault types commonly appearing in sensor networks. Such a classification scheme helps develop appropriate countermeasures as it allows the identification of the relevant fault types, the components affected, and the level where the measures need to be applied.

Some categories (i.e., fault origin, severity, and persistence) are generally applicable to various kinds of systems. The categories fault type, level, and manifestations are system-specific and include unique attributes and characteristics of WSNs. However, some categories are not entirely complimentary as faults may combine features of different elements.

4.3.1 Fault Origin

Wireless sensor nodes are embedded systems consisting of tightly integrated software and hardware components. While the software is usually considered as a single component, the hardware part can usually be divided into the radio transceiver, the MCU, the sensors, and the power supply (i.e., battery). Both the software and hardware components can

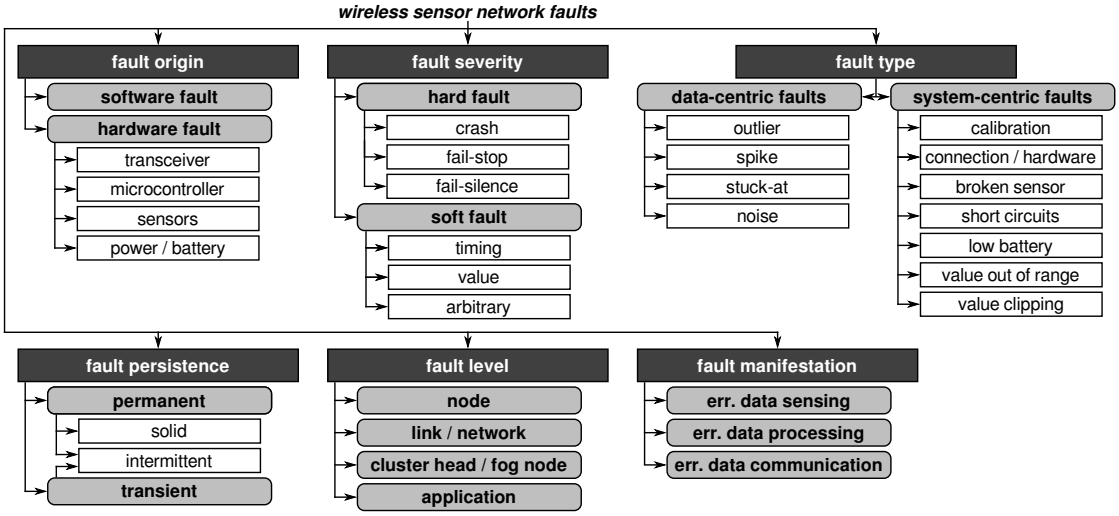


Figure 4.5: Wireless sensor network fault taxonomy

suffer from various faults where the manifestations depend on the actual origin of the fault. As shown in Figure 4.3, software mainly suffers from human-made faults such as specification or implementation mistakes (also called design flaws or simply bugs). On the other hand, hardware components also have to cope with component failures due to physical faults.

Aside from supply voltage-related effects, especially the ambient temperature has shown to cause unpredictable behavior or defects in hardware components [157]. For example, high ambient temperatures accelerate the aging of the components that bring forward effects such as hot carrier injection (HCI), time-dependent dielectric breakdown (TDDB), or negative-bias temperature instability (NBTI). High temperatures further facilitate hardware-stress-related effects such as increased electromigration or the forming of metal whiskers.

While design flaws can mostly be targeted with simulations or testing, physical faults caused by the imperfections of the natural world cannot be adequately captured before the WSN's deployment and, thus, runtime measures to enable fault-tolerance are needed.

4.3.2 Fault Severity

Faults do not always cause the system to fail in the same way, neither concerning their manifestations nor the related severity of their effects. While some faults may not even be noticeable, others can cause disruptions of the entire sensor network. In this context, two main groups of faults can be distinguished, namely *hard faults* and *soft faults*.

Hard faults include node crashes or the inability of a network participant to communicate with others, such as fail-stop or fail-silence states. Such faults usually require human intervention to resolve the situation. However, hard faulty network participants can

generally be easily detected by their neighbors indicated by an absence of messages over a certain period.

Soft faults, on the other hand, are a notably greater danger to the data quality of a WSN. While hard faults usually result in missing data, soft-faulty components continue to report data but with impaired quality. The effects of soft faults can range from deviations in the runtime behavior that can cause services to time out, over silent data corruption by incorrect data sensing or processing up to completely arbitrary effects [172]. In addition, soft faults pose a significant danger as such adverse effects are hard to detect by other network participants. Without additional information, such as expert- or domain-knowledge, it is close to impossible to decide whether an anomalous data instance was caused by a fault or reflects a rare but correct event in the observed physical phenomena [99]. Consequently, corrupted or even arbitrary sensor readings can be propagated to the subsequent data processing resulting in wrong decisions or (counter-)actions. Also, a missing data instance from a hard-faulty sensor node does not impair the subsequent data processing as much as a corrupted value from a soft-faulty sensor node. For this reason, especially soft faults are a severe risk to the reliability of WSNs and pose a crucial challenge for fault-tolerant networks.

4.3.3 Fault Type

Faults appearing in sensor networks can also be described according to their manifestation in the sensor data and/or the system behavior. As a consequence, there are two views on the types of fault models for fault detection approaches: data-centric and system-centric (cf. [99]). However, both views are not disjoint, and most of the faults from one view can be mapped to faults of the other one (see [99, Table IV]).

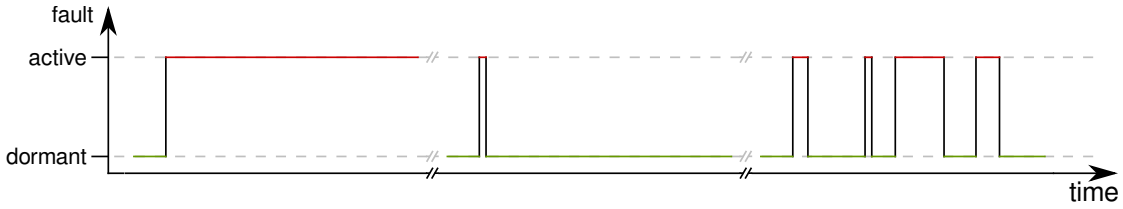
The *data-centric* view describes faults by the characteristics they cause in the data behavior (diagnostic approach). This approach can also describe faults where there is no clear explanation of its cause. Examples of data-centric faults are outliers, spikes or abrupt changes, stuck-at faults, or noise with a high variance.

The *system-centric* view defines faults based on the effect certain flaws occurring in the system cause in the data it produces. One of the most common sources for system-related data distortion are depleting batteries of the sensor nodes or calibration faults of the sensors used [95]. Also, hardware or connection failures (including short and open circuits) or environmental conditions such as a value out of sensor range (e.g., clipping) can cause inaccurate sensor data. However, in contrast to data-centric faults, system-centric faults depend on the actual system implementation, such as the used hardware components.

4.3.4 Fault Persistence

Another criterion to categorize faults is the persistence of faults. In this context, Avizienis *et al.* [162] defined two kinds of faults, namely *permanent faults* and *transient faults*. While the presence of permanent faults is assumed to be continuous in time (Figure 4.6a), the presence of transient faults is bounded in time (Figure 4.6b).

The persistence of faults can be further categorized based on their activation reproducibility. Faults with reproducible activation patterns are called "solid" (or hard), and those without systematically reproducible patterns are named "elusive" (or soft). Solid faults are the result of permanent faults. As discussed in [162], the manifestations of elusive (permanent) faults and transient faults are similar and, thus, are grouped together as *intermittent faults* (Figure 4.6c).



(a) permanent/solid (b) transient (c) intermittent

Figure 4.6: Fault categorization based on their persistence

Typical causes of permanent faults in sensor nodes are physical damage or design flaws. Transient faults can additionally be the result of external circumstances such as interference. While solid faults permanently affect the sensor nodes' operation, the effects of intermittent faults happen sporadically and with varying duration, hence, often causing an unstable device operation.

4.3.5 Fault Level

As depicted in Figure 4.2, faults happening on lower levels can propagate through the network affecting subsequent components in the data flow. Thus, faults can also be categorized based on the location where they originate (or the level, respectively). Fault tolerance measures can thereby focus on particular parts of the system or several components of the data (sub)flow.

However, based on the level where the measures are applied, the basic approaches may differ. While self-check techniques most efficiently detect certain node-level faults, faults in the links between network participants are better to be captured with group detection or centralized approaches. Additionally, the higher the network level, the more resources are usually available and, thus, the more sophisticated measures can be applied. For example, monitoring and fault diagnosis techniques applied on the cloud level can typically utilize a certain level of processing power, have significantly more memory available, and do not need to consider an energy-efficient operation. Devices in the edge layer, on the other hand, can only use lightweight techniques that do not contravene the resource limitation of those. As a result, the fault level considered is an essential criterion for selecting appropriate countermeasures.

4.3.6 Fault Manifestation

Faults can also be described based on the functionality they impair. The basic functionality of a sensor network comprises:

- the measurement of certain physical quantities (i.e., data sensing),
- the (pre)processing of the acquired data (i.e., data processing), and
- the forwarding of these data via the network (i.e., data communication).

A fault can affect one or more of these functionalities, possibly with different severity.

An erroneous data sensing can be caused, for example, by sensor hardware failures, electrical connection issues, or in case the sensed phenomena is outside of the sensor's measurement range (or in its saturated area; refer to [99, Fig. 2]). Erroneous data processing can be caused by design faults like software bugs or by physical faults affecting the processing units such as the sensor nodes' MCU. Errors in data communication can stem from a wide range of sources. Aside from flaws in the communication protocols, network attacks or environmental factors can also hinder the proper communication between participants. Thereby, primarily environmental factors such as the ambient temperature have been shown to influence (or even temporarily prevent) the communication within the network (cf. [109, 173]).

4.4 Related Fault Detection Schemes

Faults are a severe threat to the sensor network's reliability. They can significantly impair the quality of the data provided and the network's performance in terms of battery lifetimes. While design faults can be addressed during the development phase, it is nearly impossible to derive proper models for the effects of physical faults. Such effects are caused by the interaction of the hardware components with the physical environment and occur only in natural systems. For this reason, they can not be adequately captured with well-established pre-deployment activities such as testing and simulations. Hence, it is necessary to incorporate runtime measures to deal with the multilateral manifestation of faults in a WSN.

Fault tolerance is not a new topic and has been addressed in numerous areas for a long time already. Like WSNs, also systems used in automotive electronics or avionics mainly consist of interconnected embedded systems. Especially in such safety-critical applications where system failures can have catastrophic consequences, fault management schemes to mitigate the risks of faults are a must-have. Consequently, the automotive functional safety standard ISO 26262 provides methods and techniques to deal with the risks of systematic and random hardware failures. The most commonly applied concepts are hardware and software redundancy by duplication and/or replication [174]. Similarly, also cyber-physical systems used in, for example, industrial automation commonly use duplication/replication to enable a certain level of resilience [163, 164].

However, redundancy-based concepts often interfere with the requirements of WSNs as they require a significant overhead regarding the system sizes and costs, but especially concerning the energy consumption [160]. Therefore, such concepts are hardly suitable for WSNs. Fault management schemes suitable for sensor networks have to be energy efficient, provide a suitably high fault detection accuracy, need to be able to cope with the characteristics of the wireless network, and should not suffer from scalability issues [165].

In the following, an overview of the primary detection strategies of fault management schemes for WSNs published in the recent past is presented. The majority of approaches can be classified into three main categories based on their general detection strategy:

1. sensor data analysis (see Section 4.4.1),
2. group detection (see Section 4.4.2), and
3. local self-diagnosis (see Section 4.4.3).

Based on our survey of related sensor node fault detection schemes, we identified the research gap discussed in Section 4.5. However, for a detailed survey on fault detection and tolerance schemes applied to WSNs, we refer an interested reader to the literature reviews presented in [165, 175].

4.4.1 Sensor Data Analysis

One way to detect faults in a sensor network is to analyze the data reported by the sensor nodes. Faults often manifest as anomalies in the sensor data; hence, data anomaly detection approaches are commonly used [3]. Since faults can have different causes and result in effects of variable duration and impact, many data-oriented fault detection approaches leverage correlations available in the sensor data (e.g., temporal, spatial, or functional) to substitute for missing ground truth. However, to consider temporal correlations also, previous sensor data are required (i.e., the history). On the other hand, spatial correlations rely on the data from various sensor nodes within a specific neighborhood. As a result, many sensor data analysis approaches run centrally on systems with higher resources, such as the cluster head or even in the cloud layer.

Most of the data-oriented approaches can be categorized into:

(i) statistics-based,

(ii) rule-based,

(iii) time series analysis-based, or

(iv) learning-based methods.

To cover a broader spectrum of faults, to improve the detection rate, or to lower the false alarm rate, hybrids can be used that combine different methods. An overview of data-based fault detection approaches can be found in the outlier detection survey presented in [176] or the review on noise or error detection approaches given in [111].

(i) In statistics-based detection methods, mostly standard metrics such as the mean, the variance, or the gradient of the sensor data are considered for detection [177]. Additionally, there are more sophisticated approaches that, for example, apply the Mann-Whitney U statistical test or the Kolmogorov-Smirnov test to identify irregularities in the sensor data [178, 179] as well as 3σ-based techniques [180].

(ii) Rule-based methods derive heuristic rules and constraints for the sensor readings often by exploiting domain or expert knowledge. Such approaches can range from adaptive thresholds of the sensor data [181] over signature-based fault detection [182] up to applying distributed state filters on the sensor data [183].

(iii) Time series analysis-based methods leverage temporal correlations in timely ordered data of one or more sensor nodes collected over an interval of time to predict the expected values for future data (cf. [184]). An anomaly is then assumed to be the deviation of the measurements and the predicted values [185].

(iv) Another possibility to infer a model of the "normal" sensor data is the use of learning-based methods. Based on the derived model, deviations of the actual sensor readings from the expected values can be detected. Thereby, particularly neural networks [158, 186] and SVM-based detection approaches [187] have shown to be suitable in identifying anomalous sensor readings, especially when being augmented with statistical features as described in [188]. In addition, also approaches based on decision trees have been proposed for fault detection [189].

However, most data-centric detection approaches consider the sensor nodes as black boxes and neglect information available on a node level. Consequently, such approaches often suffer from difficulties in distinguishing anomalies caused by faults from actual events in the monitored phenomena. In addition, several approaches are not generally applicable because they require expert/domain knowledge that is often not available or base their detection technique on application-specific assumptions.

4.4.2 Group Detection

Group detection-based approaches leverage the detection of faults based on the spatial correlation of sensor data. Such approaches can either be run centrally on, for example, the cluster head or distributed on several (or even all) network participants. In some approaches, additional monitoring nodes with higher resources are added to the network to observe the behavior of their local neighbors. However, group detection approaches commonly rely on three major assumptions:

(i) the sensor nodes are deployed densely (i.e., the difference in the measurements of two error-free sensor nodes is negligibly small),

(ii) faults occur rarely and without systemic dependencies (i.e., the number of faulty nodes is much smaller than the number of non-faulty nodes), and

(iii) faults significantly alter the sensor data (i.e., a faulty sensor reading deviates from its local neighbors' proper readings notably).

Additionally, some approaches assume that faults occurring in the network are permanent (cf. [190]); hence, transient and intermittent faults are not considered. Aside from the approaches' architecture (i.e., centralized vs. distributed), the approaches differ in the way they decide on faulty readings (e.g., voting [191], aggregation [192]) and in the information used for their decision (e.g., sensor readings, battery level, link status). For example, the battery level in combination with the link status can be used to define the sensor nodes' state of health that is then shared with the node's neighbors (cf. [193]).

To detect faults, the approaches apply (spatial) anomaly detection methods [194], consider mutual statistical information of the neighbors [169], or use a (dynamic) Bayesian classifier [161]. The approach proposed in [195] extends a dynamic Bayesian network with a sequential dependency model separated in time slices where temporal correlations can be exploited in a single time slice. Spatial dependencies can be treated by exploiting time slices of different nodes. Another example of group fault detection is the algorithm presented in [196] that incorporates physical constraints of the monitored phenomena based on which the Kalman filter estimation value of adjacent nodes is calculated. Especially AIS-based approaches have properties beneficial to anomaly and fault detection in WSNs such as the distributed AIS-based fault diagnosis algorithm proposed in [197].

Although showing reasonable detection rates, group-based approaches suffer from significant drawbacks stemming from the assumptions mentioned above. For instance, to ensure that the distance between two neighboring nodes is always small enough to have negligibly small differences in their measurements would require a large number of nodes and, thus, would be expensive, and the network may suffer from scalability issues. Also, the assumptions on the faults cause difficulties as faults have shown to appear frequently in WSNs and their effects may be subtle such as silent data corruption. Group detection approaches often require a high communication overhead due to the message exchanges between the neighboring nodes. Consequently, the energy consumption of the nodes is significantly increased, resulting in shorter battery lifetimes.

4.4.3 Local Self-Diagnosis

The third main class of fault detection approaches is executed on the nodes locally. In contrast to the above-presented sensor data analysis and group detection concepts, the local self-diagnosis is applied close to the source of faults where node-level information can be used for better fault detection. Since these approaches exploit the nodes' internal information (i.e., node-level data), they can be seen as a form of glass-box (or white-box) runtime testing. In addition, such approaches do not suffer from scalability problems as the detection is run on the nodes locally.

One possibility for fault self-diagnosis is to run lightweight data-centric techniques on the nodes that detect statistical deviations in the node's measurements (i.e., mean and variance) or perform low-level anomaly detection similar to the methods described in Section 4.4.1. However, some researchers suggest including node-level information aside from the sensor readings to analyze the node status at runtime [3, 198]. Several works

have been presented in the last years that incorporate such node-level information in their approach. However, most of them use relatively simple checks based on the remaining battery charge (measured by the battery voltage level) or the nodes' link status (e.g., received signal strength indicator (RSSI) or signal-to-noise ratio (SNR); cf. [167,199,200]). In case the nodes are running an OS, also metrics such as the CPU load (i.e., number of cycles executed by the MCU), the memory consumption, or the execution time available from the OS have been included in the detection [75, 201]. Aside from information already available in software, it is also possible to extend the sensor nodes with specific hardware for fault diagnosis, for example, using a secondary MCU that supervises the main MCU [202] or a current monitor that allows detecting faults specific to certain sensors [170].

As with data-centric and group detection approaches, also self-diagnostic techniques are challenged by the limited resources of the sensor nodes. In the case of local self-diagnosis, the situation is even worse as the approach is applied on all nodes and, thus, has to be lightweight and energy-efficient. If additional hardware is required, also the cost factor has to be kept in mind. As a result, the majority of approaches so far rely on simplistic checks of, for example, the residual energy or OS-related metrics. Although mentioned before, incorporating additional node-level information into the fault detection has not been analyzed yet.

4.5 Research Gap

The research community has broadly targeted the topic of fault detection in WSNs over the last two decades. However, most related work on sensor node fault detection suffers from restrictive assumptions and significant limitations that hinder their effective and efficient use, as discussed in the following.

The majority of related approaches focus on fault models that cause data anomalies in the reported sensor readings, such as outliers, offsets, or drifts (see Section 4.4.1). Consequently, these "sensor data analysis" approaches consider fault detection as a data anomaly detection task performed on the sensor data only. Although several lightweight online anomaly detection schemes have been proposed in recent years, most of these approaches require a considerably high resource overhead that conflicts with the resource limitations of sensor nodes. Additionally, such approaches typically suffer from a disability to distinguish the effects of environmental events in the sensor data from data distortion caused by sensor node faults. A reliable distinction between both effects requires incorporating diagnostic data acquired on a node level.

Group detection approaches, as presented in Section 4.4.2, try to remedy this issue by considering spatial correlations in the measurements of several sensor nodes within a specific neighborhood. They mostly build on the assumption that (i) the sensor nodes are deployed densely, (ii) faults occur rarely and without systemic dependencies, and (iii) faults significantly alter the sensor data. The former assumption can often not be adhered to for economic reasons (i.e., deployment costs) and, depending on the size of

the area to be monitored, due to possible scalability issues of the underlying network. Similarly, the second assumption does not hold for all deployments, as reported by previous studies where node faults were found to occur frequently. And the latter assumption suffers from the same issue as with the sensor data analysis approaches (see above). In addition, most of the group detection approaches require an information exchange between the sensor nodes within a defined neighborhood that requires bidirectional communication and poses a significant communication overhead (i.e., energy overhead).

The local self-diagnostic approaches discussed in Section 4.4.3 aim at exploiting the information available on the node level to remedy the drawbacks of sensor data analysis and group detection schemes. The main difficulty in applying self-diagnostic approaches is the trade-off between the diagnostics' expressiveness and the required resource overhead. Therefore, most self-diagnostic approaches limit their consideration to simplistic checks of, for example, the residual energy or OS-related metrics. However, we found that such node-level diagnostics have to consider a broader spectrum of operational parameters and cover as many functional modules of the sensor node as possible (cf. [4]).

To sum up, an effective and efficient sensor node fault detection scheme needs to distinguish between data events and the effects of faults on the sensor data. It has to be generally applicable to not suffer from limitations caused by specific assumptions or restrictions on the application use cases. Moreover, it should not burden the WSN with high resource requirements, neither regarding the network (i.e., communication overhead) nor the sensor nodes (i.e., energy overhead).

CHAPTER 5

Artificial Immune Systems

One branch of anomaly detection schemes applicable to the detection of node faults are artificial immune systems (AISs); or immune-inspired approaches in general. Such approaches have shown promising capabilities and preliminary results for developing lightweight fault detection systems. This chapter discusses the underlying biological principles and the corresponding computational models developed from the human immune system (HIS).

In [100], the authors claim that "*... the process for characterizing a sensor network fault or anomaly is very similar to diagnosing an illness.*" Not only the authors of [100] draw the connection between the task of anomaly or fault detection in computing systems and the basic principles of immune systems. Similarly, the authors of [203] and [204] consider the normal behavior of a computer system to be free of anomalous occurrences, hence, "healthy". Detecting unhealthy circumstances is precisely what the HIS is doing. In other words, anomaly detection systems and the HIS share the same goal, which is to keep the system stable despite a continuously changing environment [205]. Thus, applying immune-inspired techniques to detect deviations from a "normal" operation seems reasonable. Regarding the use in WSNs, the authors of [89] go even further and claim that the basic architecture of sensor networks has a high structural similarity to the biological cell structure.

For this reason, a new discipline arose in the early 1990s aiming at deriving computational models inspired by concepts from immunology, namely the field of AISs [206] (also known as *computational immunology* or *immuno-computing*). AISs are bio-inspired schemes leveraging the immune system's characteristics for use in computational problem-solving [207, 208]. The field of AISs encompasses a collection of algorithms that are models or abstractions of mechanisms observed in the HIS [209–212]. New insights in the functioning of the HIS and the processes involved to detect infections and regulate the responses serve as inspiration for novel computational models.

A brief history and the prevalent immunological theories are presented in Section 5.1. The unique properties of the HIS are discussed in Section 5.2. One of the latest immunological models, the so-called "Danger Theory", is presented in Section 5.3. Section 5.4 gives an overview of the four "classical AIS theories".. The Danger Theory-based dendritic cell algorithm (DCA) formed the basis of our immune-inspired fault detection approach. Details on the DCA and its use for fault detection are presented in Section 5.5.

5.1 History and Immunological Theories

Several works [213–215] define the beginning of immunology with the discovery of the basic principle of immunization and phagocytosis by Pasteur and Metchnikoff in 1870. In 1890, von Behring discovered the presence of antibodies in the body of mammals, followed by the detection of cell receptors by Ehrlich around 1900. Based on the work of von Behring and Ehrlich, Bordet and Landsteiner found in the 1930s that the antibodies have a particular specificity. Thus, they only react to certain other types of cells. From there, it took another 20 years until La Verne and Burnet started to work on a theory on clonal selection of specific lymphocytes (B and T cells in particular) in the 1950s, on whose basis Burnet developed the clonal selection theory in 1957 (cf. [216]).

These fundamental discoveries led to the development of the so-called self/non-self (SNS) model (or "one-signal model") in 1959 [217]. The name "self/non-self" refers to the basic process of B cells that is to distinguish between entities that originate from the own system ("self") and those that are foreign to the host ("non-self"). Similarly, the name "one-signal" model originates from the primary hypothesis that the immune reaction is triggered by recognizing non-self entities. As a result, only one signaling factor is required to trigger an immune response (see Figure 5.1a).

Soon, the SNS model got challenged by Oudin et al. [218] with questions that the original model could not answer. As a consequence, Bretscher and Cohn proposed in 1969 their associative recognition theory, sometimes referred to as "two-signal model" [219]. In their model, antigen recognition alone is insufficient to trigger an immune response. It requires a second "signal" which they named *help signal* as shown in Figure 5.1b. This help signal is necessary to trigger the B cells. If only signal 1 (antigen recognition) is present without the secondary help signal, the B cell simply dies.

Meanwhile, Jerne was working on another aspect of the HIS which he published in 1973 as his idiotypic network theory [220]. Jerne focused on the interaction of the particular parts of the immune system and suggested that the immune system consists of complementary idiotypes and paratopes that coexist and form some kind of a formal network. The idiotypes and paratopes act as stimulatory or suppressive factors in this network. Usually, these factors are balanced. In case the stimulatory parts become rife, an immune response is triggered. For a long time, the idiotypic network theory was seen as a competitive model to Cohn and Bretscher's associative recognition theory. However, today Jerne's propositions on the regulation of the HIS by such an idiotypic network is considered complementary to the prevalent models of immune response activation [221].

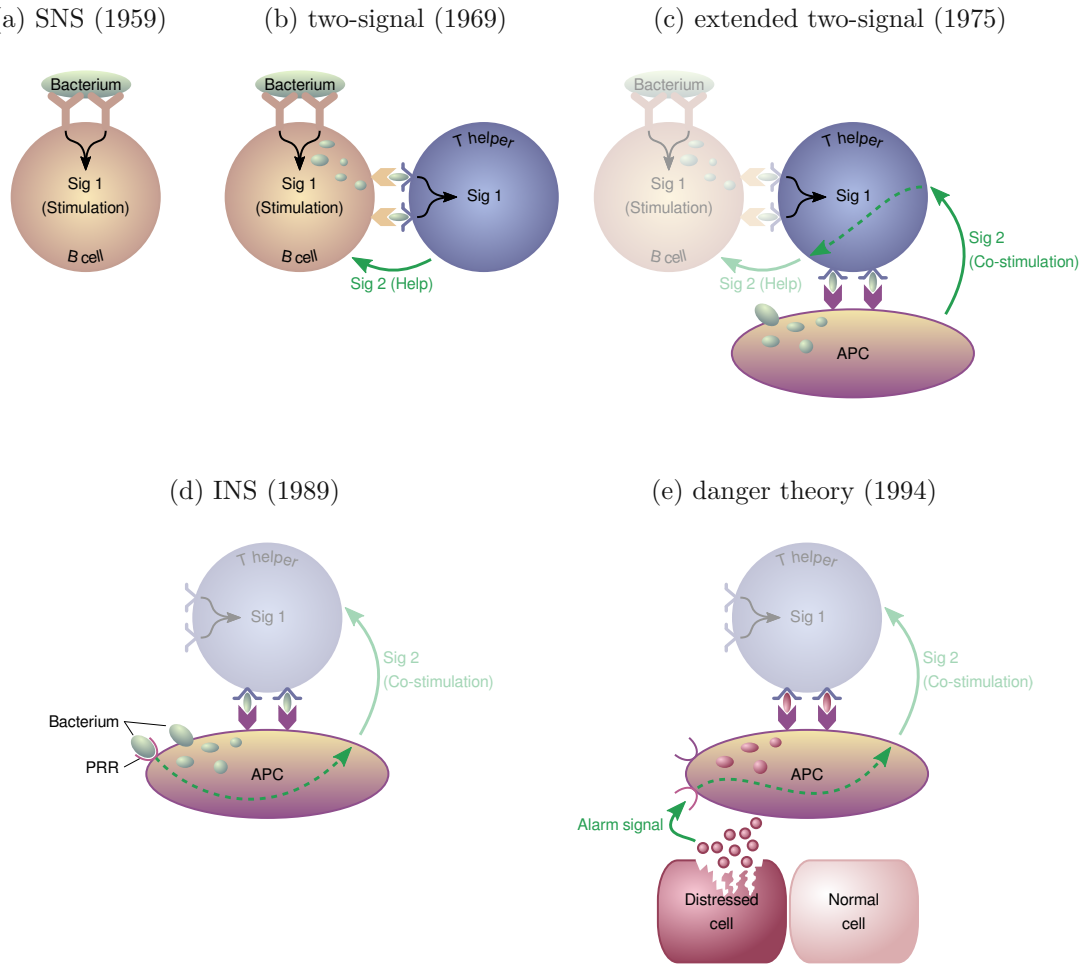


Figure 5.1: A history of immunological models (after [214, Fig. 1])

Lafferty and Cunningham further refined and extended the two-signal model in 1975 [222]. As depicted in Figure 5.1c, they claimed that the T helper cells themselves need to be co-stimulated by antigen-presenting cells (APCs) (e.g., dendritic cells (DCs)) to provide the help signal to the B cells. If the T helper cell sees signal 1 (antigen recognition), but gets no co-stimulation from an APC, it dies. Consequently, it does not relay the co-stimulation as a help signal to the B cell, causing this to die, too. Thus, the presence of two signaling factors in conjunction is needed to trigger an immune response by activating the B cells: (i) antigen recognition (i.e., the affinity between T cell receptors and certain antigens) as well as (ii) the co-stimulation by T helper cells.

The extended two-signal model by Lafferty and Cunningham served as a sound basis for the functioning of the HIS and stayed untouched for quite some time. Later, new observations on how vaccines worked led to questions not answerable by the model. In particular, it was found that adjuvants were needed in combination with vaccines to

stimulate immune responses. It was Janeway in 1989 who presented a new, refined model of the immune system, the infectious non-self (INS) model (cf. [223]) as shown in Figure 5.1d. This model suggests that the APCs themselves need to be activated before being able to provide the co-stimulation signal. For this reason, the APCs have their own form of SNS discrimination that is based on the detection of conserved pathogen associated molecular patterns (PAMP) (essentially exogenous signals) through pattern recognition receptors (PRRs).

For a long time, it was believed that the critical element in activating immune responses is antigen recognition, that is, the discrimination of entities that originated from the own system ("self") from those that are foreign ("non-self"). This "self/non-self" view became increasingly challenged by observations that the model could not explain, for example, transplants (no attack against "non-self") as well as tumors or autoimmunity (both attacks of "self"). Another prominent example is the absence of immune responses to foreign bacteria in the gut or the food we eat [215]. As a consequence, the model of the HIS was continuously refined to be able to explain new findings. However, the core mechanisms remained the discrimination between self and non-self.

This view was significantly changed when Polly Matzinger presented her "danger theory" in 1994 (cf. [214, 224]). According to this theory, the immune system reacts to entities causing damage rather than those considered foreign. Unlike the INS model, the danger theory builds upon the suggestion that DCs are natural information fusion entities able to combine signals from both endogenous and exogenous sources [225]. These signals either stem from exogenous sources (e.g., foreign bacteria) or from endogenous cellular signals [214]. The cellular signals are further distinguished based on their origin. As highlighted in Figure 5.1e, there are signals from distressed or injured cells (necrotic signals) that imply danger. In contrast, cellular signals from cells that died naturally (apoptotic signals) present a somewhat safe situation [205].

However, even today, immunologists are not fully sure how the immune system works in its entirety and which entities and processes are actually involved. So far, the INS model and the danger theory are two of the most hotly debated theories, and their basic principles are accepted by the majority of immunologists [226]. Still, the danger theory implies some problems similar to those of previous immune models. Similar to the question of how to discriminate self from non-self of the SNS model [211], the danger theory faces the difficulty of how to distinguish between danger and non-danger [215].

5.2 Unique Properties of the Immune System

The human body is unquestionably one of the most complex systems known to humanity. There are three main regulation systems in the human body:

- the nervous system,
- the endocrine system, and
- the immune system.

These three systems are integrated into one ultimate information communication network within the human body [227]. However, each regulation system has its specific roles and unique properties. Understanding these unique properties is necessary for building effective and efficient computational models based on mechanisms and processes observed in natural systems.

In the following, we will first provide a brief overview of these three regulation systems in Section 5.2.1. Then, the multi-layer defense mechanism of the HIS is presented in Section 5.2.2. Finally, the role of leukocytes and, in particular, the lymphocytes is discussed in Section 5.2.3.

5.2.1 Nervous, Endocrine-, and Immune System

The nervous system is a highly ramified network with hierarchical order controlled by a central controller (the brain). Information is transported via electrical impulses that can be amplified or blocked by messengers. The nervous system and particularly the brain have been used as inspiration for computer scientists for a long time (e.g., in ANNs).

On the other hand, the endocrine system is a regulation system purely based on chemical messengers (i.e., hormones; cf [228]). These chemical messengers are secreted by different source organs (called glands) in the human body. The regulation itself happens with specific feedback loops of the hormones as almost every hormone has a complementary hormone [229]. The endocrine system tries to establish homeostasis (or feedback inhibition) between the chemical messengers by regulating the secretion of the respective complementary hormones. The endocrine system has some interesting properties [143] such as (i) self-organization, (ii) synchronization and (iii) cascading effects that offer inspiration for certain computational problems.

In [143], Sinha and Chaczko compared the basic structure and working principle of the endocrine system with large-scale Internet of Things (IoT) infrastructures. Based on this view, they argue that models derived from the endocrine system offer great potential to solve problems prevalent in such large-scale networks. For this reason, several computational models based on the endocrine system have been proposed in the past, such as the autonomous decentralized system [230–232], the digital hormone system used for self-organized robot swarms [233–235], the computational model of hormones as first proposed in [236] and extended in [237], the regulation model of hormones [238, 239] as well as the artificial hormone system [240, 241].

The third regulation system, the immune system, is a widely distributed and inherently parallel network of a significant number of diverse entities. These entities are working simultaneously and in cooperation with each other to reach the overall goal, to keep the body healthy [242,243]. It is a decentralized system without a central controlling instance (such as the brain for the nervous system). One of the most significant advantages of the HIS is its vast amount of resources. The immune system of an adult consists of around 10^{12} lymphocytes, 10^{20} soluble antibody molecules with about 5 million different antibody types, and a daily turn-over of these components of approximately 2 % (cf. [213]).

Also, the HIS operates on different levels using various components, such as physical barriers (e.g., skin), chemical barriers (e.g., antimicrobial substances like sweat and saliva), cellular proteins (e.g., cytokines), and a large number of different cells (e.g., macrophages and DCs). All these components and their interaction build up a highly complex self-organizing system with beneficial properties such as error-tolerance, adaptation, and self-monitoring [144]. Certain parts of the immune system even have learning, memory, and associative capabilities (cf. [244]).

5.2.2 Innate and Adaptive Immunity

The immune system has an ingenious multi-layer defense mechanism consisting of two distinct yet interrelated immune mechanisms [245]:

- innate (non-specific) immunity
- adaptive (specific) immunity

The combination and interaction of both forms versatile and efficient protection for the human body. Both parts of the immune systems use many different cells of diverse specialization to protect the host efficiently.

5.2.2.1 Innate Immunity

The innate immune system [246] provides non-specific protection and defense mechanisms as well as general immune responses. There are four types of defense barriers in innate immunity, namely (i) anatomic, (ii) physiologic, (iii) endocytic & phagocytic, and (iv) inflammatory [245]. The anatomic barriers (e.g., skin or mucous) prevent the penetration of foreign entities and, thus, build the first level of defense. Also, innate immunity consists of a large number of different cells providing a defense against the general properties of pathogens [247]. Hereby, the APCs (a kind of leukocytes, or more specifically monocytes) play in important role, especially the DCs (see Section 5.3.3). The innate immune system is an essential first line of defense against invading pathogens using generic responses [248]. The innate immune system does not develop memory and, thus, does not offer specific responses [249]. A review of innate immunity and its biological principles and properties can be found in [250].

5.2.2.2 Adaptive Immunity

The adaptive immune system [251] provides more specific and compelling response mechanisms as well as the capability to learn from previous occurrences of pathogens (i.e., immune memory [252]). It is sometimes called *acquired immunity* as the specific responses are developed over the lifetime of the host [144]. The main components of the adaptive immune system are lymphocytes, in particular B and T cells. In contrast to the leukocytes constituting to the innate immunity, these cells can evolve over the lifetime of the host by specializing their receptors [253]. Based on these cells and their contribution

to adaptive immunity, two primary adaptive immune responses can be distinguished, the *humoral response* and the *cellular response* [249, 254, 255].

The *humoral response*, or humoral immunity, refers to the interaction of B cells with antigens by producing specific antibodies that detect and eliminate foreign entities. B cells are produced by the bone marrow, where they have to survive a negative selection process before being released into the bloodstream. This negative selection process is part of the SNS theory and makes sure that the B cells surviving are self tolerant; thus, they do not attack native (self) cells. If a B cell matches a particular antigen, they respond by multiplying themselves by clonal expansion. In this process, B cells divide into several clones with slightly mutated antibodies to cover a broader spectrum of antigens and increase the chance of an even better antigen matching [256]. B cells with a high affinity can evolve to memory B cells capable of identifying the same pathogen much faster in the future (as the activation and stimulation process is shorter for memory B cells [257]). Such an immune response from memory B cells is called immune memory (also referred to as secondary immune response or strong immunity [252]) and provides an essential characteristic of the adaptive immune system, namely the ability to learn through the interaction with the environment. Approximately 90 % of the B cells die after their responses or lifespans, and the rest remain as memory cells [258].

The second adaptive immune response is the *cellular response*. It refers to the behavior of T cells that have two main tasks: (i) the detection of intrusions by T helper cells (T_h) and (ii) the attraction of cytotoxic T cells (T_c) for the disposal of infected cells [255]. To be more precise, the T_c becomes activated on the recognition of infected cells and starts producing molecules that destroy the infected cell.

Besides T_h and T_c, there exists a third type of T cells, the regulatory T cells. These regulatory T cells exist in two different stages: naive or active [258]. After being produced in the bone marrow, these regulatory T cells migrate to the thymus where they undergo a negative/positive selection process similar to B cells (but in the thymus instead of the bone marrow). Regulatory T cells that survived the selection process and that have not experienced an antigen yet are called *naive T cells*. Naive T cells can become *activated T cells* if they successfully bind to an antigen in combination with co-stimulation from an APC (or DC to be precise; see immune models in Section 5.1). Thereby, the degree of activation depends on the degree of signaling from the DC. In the case of excessive levels of co-stimulation, the T cells die to prevent overly excessive immune responses, a process called activation induced cell death [259].

5.2.3 White Blood Cells

Although many cells are involved in immunity, white blood cells build the core of the immune system. These cells are primarily produced and matured in lymphoid organs (e.g., thymus or bone marrow) and are categorized in general white blood cells, the *leukocytes*, and specific subtypes of white blood cells, the *lymphocytes* [206]. While leukocytes form the bases of innate immunity (i.e., monocytes such as macrophages and

APC), the adaptive immune responses are primarily performed by lymphocytes (i.e., T and B cells as well as natural killer cells) [214]. The three most important white blood cells for immunity are:

- **dendritic cells (DCs)** are a particular class of APCs that move in blood and process information about antigens and dead cells found on their way.
- **T cells** are produced by the bone marrow and are responsible for destroying infectious cells.
- **B cells** are also produced by the bone marrow and stimulate the production of antibodies.

Due to their way of detecting foreign antigens, the antibodies are often called detectors, especially in the context of AIS.

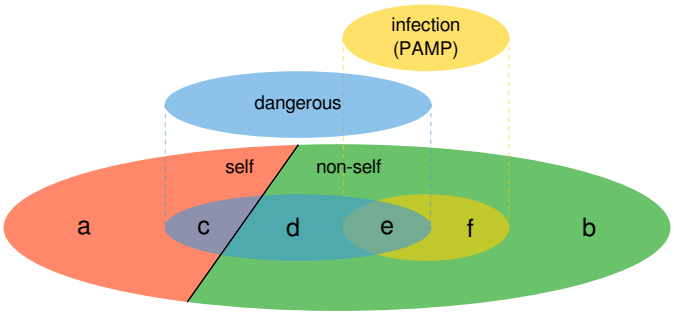
Besides the white blood cells, a large number of other cells and molecules are essential for the functioning of the immune system. Thereby the ligands (or keys) play an important role as they are responsible for activating the cells' receptors. As with the endocrine system, also the immune system contains regulating molecules called cytokines. Additionally, chemokines are specialized molecules that stimulate cell movement [206].

Altogether the immune system shows characteristics also found in other bio-inspired systems (as presented in Section 3.3.7.5). As the cells and their interaction share similar properties with swarm-like systems, the immune system is often considered a swarm system, too [260]. Also, to detect foreign entities, the immune system uses affinity measures that are, in their fundamental principle, similar to the fitness function in GA [254]. A detailed overview of the (natural) immune system can be found in [245, 261].

5.3 The Danger Theory

The danger theory states that the immune system does not primarily react to foreignness but to circumstances that pose a danger to the host. Therefore, it changes the discrimination of "self from non-self" of the SNS model to a discrimination of "some self from some non-self" depending on the presence of danger to the system (see Section 5.3.1). This difference in the antigen discrimination is shown in Figure 5.2, where SNS refers to the self/non-self model, INS to the infectious non-self model, and DT to the danger theory. In the figure, a "+" states that the theory reacts to this kind of antigens while a "–" means that the theory ignores antigens of that kind.

In the danger theory, the danger is represented by the presence of so-called danger signals in the absence of down-regulating safe signals within a specific area (refer to Section 5.3.2). These necrotic (danger) signals and apoptotic (safe) signals in combination with PAMP are integrated by the dendritic cells to instruct the immune system to respond appropriately. Thus, the dendritic cells are one of the major control mechanisms in immune systems (cf. Section 5.3.3).



	a	b	c	d	e	f
SNS	−	+	−	+	+	+
INS	−	−	−	−	+	+
DT	−	−	+	+	+	−

Figure 5.2: Antigen responses of different immune theories (after [214, Fig. 2])

5.3.1 Basic Concept

While previous immune models often focused on the role of adaptive immunity, Matzinger also stressed the importance of innate immunity [214, 224]. From a biological point of view, the innate immune system has three main roles [244]:

- defending the host in early stages of infection
- initiation of adaptive immune responses
- determination of the actual type of adaptive response through APCs (i.e., DCs)

In the danger theory, signal two is provided by "professional" APCs, the DCs, which provide a vital link between innate and adaptive immunity [226]. Due to their way of collecting and evaluating the information on the current condition of the host, these DCs are sometimes denoted as the crime-scene investigators of the HIS [262]. Therefore, the danger theory suggests that there are two key elements responsible for immunity: (i) the tissue with the signals contained and (ii) the alignment of innate and adaptive immunity by DCs. The signals are discussed in more detail in Section 5.3.2.

As a result, the danger theory further implies a notable change regarding the control of immune responses. It highlights the role of the tissue for the immune system as it suggests that it is the tissue that controls immune responses and the evolution of the immune system [226, 263].

The danger theory was initially hotly discussed within the immunology community and by far not accepted by all members [264]. However, Matzinger and other advocates of the danger theory found more and more evidence for their claims as well as observations in nature that can not be explained by the previously prevalent theories. The danger theory states that the "foreignness" of a pathogen alone is not enough to trigger an immune response and that, on the other hand, "selfness" is no ultimate guarantee of tolerance [214]. As shown by Matzinger in [224, 265], changes do happen in the human body over the lifetime, be it of natural cause (e.g., pregnancy) or due to external intervention (e.g., surgeries); thus, the self changes as well. More detailed information on the danger theory from an immunologist's view can be found in [214].

5.3.2 Immunological Signals

The danger theory states that the affinity between an antigen and an antibody ("signal one") is not enough to trigger an immune response [245, 266]. In addition, there needs to be a co-stimulation by APCs such as the DCs ("signal two"; see Figure 5.1). DCs reside in the tissue and collect antigenic material and contextual information (commonly termed signals). According to the danger theory, it is the correlation of the contextual information (i.e., the signals) that triggers immune responses. Matzinger [224] groups these signals into three main categories (see also [253]):

- **apoptosis**: natural death of cells (the "safe signals")
- **necrosis**: unnatural death of cells (the "danger signals")
- **PAMP**: biological signatures of potential intrusions (e.g., foreign bacteria)

The danger signals can be further divided into (i) endogenous (generated by the body such as heat shock proteins, nucleotides, neuromediators, and cytokines) and (ii) exogenous (caused by invading organisms) [267]. These necrotic (danger) signals and apoptotic (safe) signals in combination with PAMP signals are integrated by the DCs to instruct the immune system to respond appropriately [253]. For more information on necrosis, apoptosis, and their processes and characteristics, we refer to [268].

However, Matzinger admits that the exact nature of the danger signals is unclear, resulting in the above-mentioned difficulty of how to discriminate danger from non-danger [215]. Since the advent of the danger theory in 1994, many signals affecting the DCs have been empirically revealed [267]. As argued by Aickelin and Cayzer [215], a connection to the classical SNS theory is to consider the presence of non-self as a kind of danger signal.

5.3.3 The Role of Dendritic Cells

The danger theory focuses on the DCs since they can stimulate naive T cells and thus initiate primary immune responses [267]. DCs are monocytes (i.e., white blood cells) that were initially identified by Steinman and Cohn [269] and are native to the innate immune system [225]. Due to their function, they can be seen as the body's own intrusion detection agents [270]. DCs provide a vital link between the innate and the adaptive immune system as they link the initial detection (innate) to the actual effector response (adaptive) [226]. Additionally, DCs are one of the major control mechanisms in immune systems as they coordinate the T cell responses by producing certain pro- or anti-inflammatory cytokines (chemical messengers). Pro-inflammatory cytokines have an activating effect on immune responses, while anti-inflammatory cytokines have a suppressing effect.

DCs are produced by the bone marrow and exist in three states of maturity with different functions respectively [250, 271]. After being produced, the DCs are in an *immature* state (denoted as `iDC`). The `iDCs` reside in the tissue and have the primary task of collecting cellular debris via ingestion [226, 272]. Thereby they collect antigens and receive the signals mentioned above (i.e., danger, safe, and PAMP). After being exposed to a certain

quantity of signals, the iDC becomes activated. Exposure to PAMPs accelerates the process of maturation.

The activated iDC then migrates from the tissue to the lymph nodes where they either become *semi-mature* (smDC) or *mature* (mDC). In case the iDC experienced a higher concentration of danger-related signals (i.e., a greater quantity of either PAMP or danger signals), it maturates into an mDC. Otherwise, it becomes an smDC.

In the lymphoid tissues, the smDC and mDC interact with naive T and B cells to either initiate (in case of mDC) or suppress (in case of smDC) an adaptive immune response. The naive T cells respond by differentiating further into activated T cells (see Section 5.2.2.2 as well as [258]). This is achieved by the production of small quantities of anti-inflammatory cytokines by the smDC and the production of pro-inflammatory cytokines by the mDC respectively. Also, the mDC produces co-stimulatory molecules that have an amplifying effect on both the PAMPs and danger signals in the surrounding area [273].

However, also iDC have a suppressing effect as the encounter of iDC with T cells results in the deactivation of the T cell due to a lack of co-stimulatory molecules or inflammatory cytokines [272]. The DCs do not perform their function in isolation as there are numerous of these DCs in the tissue. Thus, they are forming a population-based system offering high error-tolerance and robustness through diversity as well as a low FAR [270]. Further information on the DCs functioning with a focus on AIS is available in [274].

5.4 Classical AIS Theories

Since the first AIS emerged in the 1990s, much research has been done in the field. With growing research interest, the field of AIS became more comprehensive and the areas of applications more numerous. Generally, research on AISs can be grouped into three main areas: (i) immune modeling, (ii) theoretical AISs, and (iii) applied AISs (cf. [275]). Immune modeling is concerned with the biological processes of the HIS and is predominantly covered by immunologists or biologists. Theoretical AISs take inspiration from immune models to develop computational models capable of solving defined problems on a theoretical model. In this context, especially the mapping from immunological to computational entities remains a problematic task [276]. However, especially applied AISs have gained popularity over the last years as an increasing number of use cases and real-world scenarios arose where AIS can be efficiently applied to solve computational problems [277].

However, over the last two decades, the AIS models have notably evolved. While in the beginning most AISs mimicked adaptive immune response mechanisms only, today more models incorporate processes of both the innate and the adaptive immunity. For this reason, usually models including only adaptive immunity are referred to as *first generation AIS* and those that include both are denoted as *second generation AIS* [206]. One primary reason for this paradigm shift was the findings on the data fusion capabilities of DCs. Including DCs into an AISs allows the system to correlate data from multiple

noisy sensors that help to improve the overall stability of the AIS, especially in the presence of unknown time delays of the signals [206].

AIS have characteristics that make them suitable for optimization or anomaly detection tasks, especially their ability of self-adapting, self-learning, self-organizing, highly parallel processing, and their distributed coordination [205]. Their efficiency can be further improved by auxiliary antigen libraries or concepts from the idiotypic network theory (see Section 5.4.3). AISs also provide mechanisms for self-regulation by adjusting the lifetime of the cells used and their probability of reproduction [278]. By fine-tuning these parameters, the performance of AIS can be significantly improved [276]. Additionally, these regulatory mechanisms allow the system to adapt to dynamic environments, which in turn is vital as the human body undergoes specific changes over its lifetime [215], which can also be the case for WSNs.

AISs have shown to perform comparably well on certain benchmark data sets when compared to existing statistical and machine learning techniques [206]. In some cases, they even presented a more efficient solution than prevalent techniques. Nevertheless, many AIS models have some significant drawbacks that limit their applicability. The most severe ones are their usually high resource consumption (especially for memory) and their ordinarily bad scaling properties [279]. As an example, the authors of [106] compared an AIS-based misbehavior detection with a second instance based on an ANN. They showed that the AIS offers comparable results, in some cases even better than the ANN, but at the cost of resources, especially memory. In their experiments, the AIS-based approach required nearly six times more memory than the ANN approach.

Today, most AIS approaches are derived from one of the following four theories, sometimes called “classical AIS theories” [242]:

- **negative/positive selection** (mainly based on T cells; see Section 5.4.1)
- **clonal selection** (mainly based on B cells; see Section 5.4.2)
- **immune network theories** (i.e., idiotypic network theory; see Section 5.4.3)
- **danger theory** (i.e., dendritic cell-based algorithms; see Section 5.4.4)

Aside from these common techniques, several other immunology-inspired algorithms and computational tools have been developed, such as humoral immune response systems [249] and the pattern recognition receptor model [280]. Review work for general AIS approaches is given in [242, 242, 281–284] as well as focused on anomaly detection and IDS in [102, 210, 285].

AIS can also be combined with other (learning) techniques to build more efficient ensemble/hybrid systems. One common goal is to decrease the FAR, which is usually high in self-organized (unsupervised) approaches. A typical example are immune genetic algorithms [286–288] for optimization problems as well as to lower the FAR of an immunity-based anomaly detection system [289]. A more sophisticated approach was proposed in [290]. This model consists of three evolutionary stages to optimize the overall performance. These stages are (i) gene library evolution [252], (ii) negative selection [291],

and (iii) clonal selection [292]. For more examples on ensemble/hybrid AIS, we refer to the survey on AIS hybrids during the years 2008–2011 presented in [293].

5.4.1 Negative and Positive Selection

In the HIS, negative selection is a process taking place in the bone marrow (for B cells) or the thymus (for T cells). It uses self/non-self discrimination based on a naive model of central tolerance developed in the 1950s [258] and, together with clonal selection, forms the core concepts of the SNS model. The SNS model assumes that the self is defined in early life, and anything that comes later is considered as non-self [214]. The selection process aims at eliminating antibodies (i.e., lymphocytes) that are reactive to entities of the self space. For this purpose, it checks their affinity based on the degree of binding between, for example, T cell receptors and specific antigens. The antibodies failing the selection process are removed from the population.

There are two basic selection processes, namely, positive and negative selection. In positive selection, the antibodies are selected to cover the self space. Thus, only those who match the self are kept while the others are removed. On the other hand, in negative selection, the antibodies are selected to match the non-self space. Nevertheless, positive selection has not been found in the selection of T cells [294]. As a result, the majority of immune-inspired approaches use negative selection. However, which of these two selection processes better suits a given task depends on the size of self and non-self, or their ratio, respectively.

Methods based on negative/positive selection are typically used for classification and pattern recognition problems (e.g., anomaly detection [295]). In anomaly-based IDS, the pathogens represent the potential attacks, and the antibodies are a way to identify that attacks [296].

Inspired by the HIS' negative selection processes, the first negative selection algorithm (NSA) model was proposed in 1994 in [291]. The crucial part of the NSA is to find a suitable mapping from the biological entities (e.g., antigens, antibodies, pathogens) to the computational problem. In the area of AISs, the antibodies are usually called detectors as their job is to detect certain circumstances (i.e., the presence of non-self). The detectors are often represented as feature vectors representing antigenic patterns able to detect changes in behavior [33]. Often the problem space is represented by an n-dimensional space, and the detectors are hyperspheres that use a matching rule based on an individual membership or distance function (e.g., Euclidean distance). In some NSA-based approaches, immune memory is introduced by promoting detectors that produce many alarms to memory cells with a lower activation threshold [210].

Basically, the NSA has two important components: (i) the detectors and (ii) the matching rule. The problem of how to generate detectors to minimize their number while maximizing the covering of the non-self space is one of the major fields of research for NSA [297, 298]. Usually, the number of detectors required to cover a certain self-space grows exponentially

with its size [208]. Also, the shape of the self space and the detectors has shown to have a significant impact on the number of detectors needed [106].

Related work on the improvement of the detectors focuses on (i) their representation (e.g., binary or real-valued; see [299]), (ii) their shape (e.g., hyperspheres or hyperellipsoids; see [300]), (iii) the parameters involved in their creation [301], (iv) the influence of variable radius [302] as well as (v) the effects of growing or shrinking the detectors surface [303]. An extensive analysis of the effects of different detectors used in NSA as well as the development of improved detector generation algorithms is summarized in [97]. Another way to efficiently cover the entire non-self space is to combine detectors of different types (with their respective matching rules) to reduce the number of holes [208].

Directly intertwined with detectors are the affinity measures (or matching rules) applied. As presented in [242], the metric to measure the affinity (similarity) depends on the choice of vectors attributes as it determines the detectors' shape space type. In [304], different detectors shape spaces and suitable affinity metrics are analyzed, such as (i) real-valued shape spaces (with Euclidean distance or Manhattan distance), (ii) Hamming shape spaces (with Hamming distance or r-continuous bit rule) and (iii) symbolic shape spaces. Also, alternative representations have been proposed such as (iv) feature-feature relations [305], or (v) dictionary-based basis decomposition methods [306]. However, choosing an expressive metric is a non-trivial task in most cases.

As stated in [307], most works so far used an antigen representation based on binary feature vectors and applied binary matching rules (e.g., r-contiguous matching [291], r-chunk matching [299], landscape-affinity matching [308], or Hamming distance matching rules [308,309] and its variations such as Rogers and Tanimoto (R&T) matching rule [308]). Especially the r-contiguous matching rule has found application in many NSA-based approaches [208, 291, 299]. The r-contiguous rule matches two strings if they have an identical sequence of r bits.

Although approaches based on negative selection had a promising start, they have been found to have severe problems regarding scalability and coverage [102, 211]. As pointed out in [310], the required amount of detectors to sufficiently cover the non-self space becomes unmanageable for most problems. The authors of [97] counter this claim and argue that the problem is not with the algorithm itself, but with unsuitable (binary) representations of the problem space (see also [311]).

In addition, there are two common problems with the traditional SNS model applied to AIS, that are a high FPR when using negative selection (leading to missed anomalies) and a high FNR when applying positive selection (resulting in a high FAR; cf. [215]). Directly connected with these issues is the problem of a dynamic or changing self as the SNS model assumes a static self that does not change over the lifetime. One way to cope with changing selves is the balance the life-cycle of immune cells, enabling an adaptive coverage of the non-self space [252, 312].

Possible solutions to these problems are hybrid approaches. One way to overcome the difficulties with detector coverage is to apply evolutionary algorithms to continuously

evolve the detectors, such as GAs [313] or clonal optimization [314]. A prominent example is the evolutionary negative selection algorithm, a hybrid evolutionary immune algorithm that was extended with a niching technique to prevent the algorithm from ending up in a local optima [290]. Also the usage of gene libraries to avoid random detectors at the initialization is a promising way [210]. These gene libraries lead the generation process of antibodies and can improve the overall efficiency [315].

Another approach dealing with the problem of crisp transitions between the self and non-self space is the combination of negative selection with fuzzy rules [316, 317]. Such fuzzy-based NSA have shown favorable characteristics when applied to immunity-based IDS [316]. For a network IDS, also the efficiency of a hybrid AIS combining positive and negative selection has been analyzed [318, 319]. Fuzzy rules in combination with Q-learning were used in the cooperative fuzzy artificial immune system proposed in [108] that showed superior properties in comparison with other learning techniques (i.e., C4.5 decision tree, artificial immune recognition system (AIRS), clonal selection algorithm (CLONALG), fuzzy logic controller, and fuzzy Q-learning).

Two of the more complex hybrid approaches are Bayesian artificial immune systems [320, 321] and the complex artificial immune system [322]. The former are based on Bayesian networks and are intended for solving hard optimization problems. On the other hand, the complex artificial immune system is a layered model that takes antigens as inputs and proposes antibodies as output. It is best suited for pattern detection problems as it can deal with several transformations such as scaling or rotation of patterns.

However, NSAs have been applied to many problems so far, including anomaly detection [323], fault detection [324] or function optimization [325]. An approach to apply negative selection to an active defense IDS is presented in [326]. Similarly, an immunity-based IDS with a multi-agent architecture is shown in [327]. A survey on NSA applications from 2011 can be found in [328].

5.4.2 Clonal Selection

Clonal selection theory [217, 292] is based on the functions of lymphocytes in immune systems, especially the maturation phase of B cells. The foundation of this theory was introduced by Burnet in 1957 as an explanation for the observed diversity of antibodies during an immune response [216]. The clonal selection theory suggests that lymphocytes activated by antigen-binding trigger a clonal expansion to evolve antibodies with a better affinity to the present antigens. During this clonal expansion, the lymphocytes undergo an affinity maturation where they are subject to somatic hypermutation (a mutation of the cell's antigen-binding coding sequences) and a subsequent selection mechanism [275]. In hypermutation, the degree of mutation depends on the affinity measure, where a lousy affinity value results in a higher degree of mutation. As a consequence, the generality and coverage of the detection are increased through the process of hypermutation [210].

The clonal selection and the algorithms derived from it, like the CLONALG [329], are commonly applied to optimization problems and clustering problems (such as pattern

recognition) [281]. Additionally, it is often used in conjunction with NSA or an affinity calculator [252]. As the task of affinity evaluation can be partitioned, a parallel version of CLONALG was proposed in [330].

As summarized in [149], the original CLONALG has a relatively high FAR and is not able to cope with dynamic environments. It is impracticable for dense environments making it not suitable for WSN applications. But it can be deployed in a highly distributed manner and offers an efficient detection rate. It allows the development of memory detectors that help to reduce the response time, especially when combined with negative selection. For this reason, an improvement of the original CLONALG was introduced in [331]. Another algorithm based on the CLONALG with influences from artificial immune network (AIN) [332] (see Section 5.4.3) is the artificial immune recognition system (AIRS) [333, 334], one of the first AIS-based supervised learning algorithms. In [330], a version of AIRS is presented in which the affinity evaluation is parallelized.

Although clonal selection approaches rather deal with optimization problems, several attempts of applying it to anomaly or intrusion detection have been proposed (cf. [335]).

5.4.3 Artificial Immune Networks

Artificial immune networks (AINs) are a class of immune-inspired algorithms that are based on the idiotypic network theory proposed by Jerne [220]. They can be seen as an extension of the clonal selection with the interaction between the antibodies and antigens, or B cells respectively [206]. The AIN model was first proposed in [309] followed by the first AIN algorithm in [336] and an improved version in [337]. Today, one of the most common AIN-based algorithms is aiNet [338] and its variations [339].

Similar to the clonal selection, AIN-based concepts are usually used for optimization and clustering problems as well as data visualization and control where they share properties with ANN [340].

5.4.4 Danger Theory-based Approaches

The unique role of APCs and especially the DCs for (innate) immunity is well known since Lafferty and Cunningham's extended two-signal model from 1975 [222]. DCs are one of the most important immune response regulation mechanisms. Their importance for the immune system became even more evident with the advent of the danger theory [224]. Since then, several computational approaches based on the danger theory, or the DCs' functionality have been proposed.

A first in-depth discussion on the potential of the danger theory for AISs was presented in [144]. The authors stressed on the natural anomaly detection capabilities of DCs and their possible applications in computing systems. Thereby, especially a low FPR in combination with a high TPR are desirable properties for anomaly detection techniques [253]. The anomaly detection is performed by the DCs by correlating the collected antigens with the fused contextual signals. It is necessary to consider the signals in combination

as the analysis of particular signals in isolation is insufficient to indicate anomalies [341] or to produce classification [209]. Additionally, the danger theory provides a way of grounding the response by linking it directly to the source for abnormality [144].

Danger theory-based approaches have shown good anomaly detection capabilities while using minimal resources [144]. In contrast to other immune-inspired techniques, the danger theory bases its detection on the presence of danger to the host, represented by so-called danger signals, in combination with an absence of down-regulating safe signals [270]. Thus, danger theory-based approaches use pre-defined signals to derive the system's context and react to "dangerous" states rather than all kinds of deviations. These signals are collected over time and in different places allowing the system to leverage spatio-temporal correlation.

Over the years, the danger theory has inspired the development of several AISs. Especially the unique role of the DCs has paved the way for several novel algorithms such as the toll-like receptors (TLR) algorithm [342] and the conserved self pattern recognition algorithm (CSPRA) [280].

The TLR algorithm [342] models the interaction of DC and T cell populations. It uses binary signals (i.e., present and not-present) to stimulate immune responses in a way similar to PAMP signals. For more information on the TLR algorithm and the detailed steps involved, see [343].

Another AIS model influenced by the danger theory is the CSPRA [280]. It allows detecting anomalies by replicating the negative selection of T cells in combination with the self pattern recognition of APCs. It adds the APCs part of the function as the negative selection is naturally involved from the PRR model.

Nevertheless, the main danger theory-inspired algorithm is the dendritic cell algorithm (DCA) [209, 226, 253, 262, 270, 274, 296, 341, 344, 345] originally proposed in [226] as part of the so-called "Danger Project" [144]. The DCA is suitable for use in resource-constrained systems and can perform context-aware anomaly detection. Both are properties desirable for fault detection approaches in WSNs. For this reason, an introduction to the DCA, its working principle, its variants and further developments as well as related work for fault detection are presented in Section 5.5.

5.4.5 AIS Applications in WSNs

As revealed by several reviews [210, 275, 312, 346], the majority of AIS research was focused either on negative selection or the danger theory. In the field of WSNs, there is a noticeable trend towards danger theory-based approaches. The reason is mainly scaling problems of the NSA that is even worse when being applied to real network traffic [310]. Secondly, the danger theory's distributed and simple concept is suitable for most WSN applications [149]. Therefore, we will give a brief overview of the application of immune-inspired techniques for computational problems in general and specific to the use in WSNs.

Based on the aim of the HIS to keep the host healthy by eliminating threats to proper functioning, several researchers claimed that the HIS could be seen as a natural anomaly detection system [102, 210, 281]. It can detect pathogens without prior knowledge of their structure [102] and offers a very low FPR as well as FNR [210]. Thus, making it a perfect example of a (distributed) anomaly detection system.

In general, AIS-inspired anomaly detection found application in a great number of different fields [281], such as virus detection [347], intelligent spam mail filter [348], credit card fraud detection [349] or different other computer security related topics [285, 308].

Especially the application of AIS for network anomaly detection, as part of an IDS, has drawn much attention from the research community [242, 291, 312, 350–352]. In this context, the expected behavior is usually considered as the self space, and any deviation from it counts as non-self [136]. To increase efficiency while reducing the FAR, hybrid approaches can be beneficial (cf. [353]).

For applying AISs to WSNs, the mapping of entities of immunity to those of the WSN is a crucial task. For network-based approaches, often the antigen is derived from information extracted from network packets and stored in feature vectors [346]. On the other hand, host-based systems often use OS-related information such as system calls to derive the antigens [253]. Examples of immune-inspired IDS applied to WSNs are given in [354, 355].

A combination of negative and clonal selection for network anomaly detection in WSNs is proposed in [276] and an extension of it in [212]. The authors define antigens as random low-level bit patterns and, as in the HIS, let the immunity-inspired mechanisms take care of their evolution.

Concerning the use of AISs for fault detection, immune-inspired approaches can also be used to detect internal deviations rather than focusing on attacks from the outside (similar to the HIS). In this context, several fault diagnosis systems inspired by the HIS have been proposed [356]. As with IDSs, also such systems often assume a fault-free system behavior at the early stages [357]. However, many of these approaches suffer from a high FPR [358]. Based on immune models, a maintenance architecture able to detect faulty behavior has been developed [359]. Another network fault diagnosis approach based on AIS is presented in [360]. Specialized systems to detect hardware faults are introduced in [360] as well as systems leveraging co-stimulation in [203, 361]. The fault detection technique in [362] is tailored for WSNs and consists of a linear vector quantization-based training phase and a subsequent AIS-based diagnosis mode. Also for SHM with WSN, some immune-inspired approaches have been proposed [33, 363].

As argued in [279], an efficient fault detection system would combine AIS with an artificial endocrine system. The AIS is suitable for detecting low-level faults that can be corrected locally, and the artificial endocrine system is better suited to recognize chronic faults.

5.5 The Dendritic Cell Algorithm

The dendritic cell algorithm (DCA) was one of the first algorithms that used the functioning of dendritic cells as suggested by the danger theory for solving computational problems. Its initial version (also called "classical DCA") was introduced by Julie Greensmith in 2005 [226]. The DCA is based on the DCs's ability to combine multiple signals to assess the current context of their environment. In contrast to other AISs, it relies on the correlation of information from the population of DCs rather than pattern matching based on similarity metrics [253]. Further differences to other AIS algorithms are the combination of multiple signals from diverse sources as well as the correlation of signals with antigens in a temporal and distributed manner to form a context-aware anomaly detection system [209].

To confirm the algorithm's basic working principle, it was initially used to classify data provided by the UCI Wisconsin breast cancer dataset with signals derived from the data attributes. The original intention for the development of the DCA was its use in an immune-inspired IDS where it has then been applied for the detection of port scans and the detection of botnets in computer networks (cf. [262]) as well as for attack detection in an Open Platform Communications Unified Architecture (OPC UA) framework [364].

5.5.1 Working Principle

The DCA describes an abstract model of the functioning of dendritic cells based on Matzinger's danger theory [214]. For this purpose, it uses a population of abstracted dendritic cells, each with (i) a collection of antigens the cell encountered during its life, (ii) a finite lifetime with a pre-defined threshold, and (iii) a contextual value depending on the concentration of the input signals as described below. As depicted in Figure 5.3, the original DCA consists of three main stages [209]:

1. initialization (setting of various parameters),
2. cell update (event-driven update of variables), and
3. data aggregation.

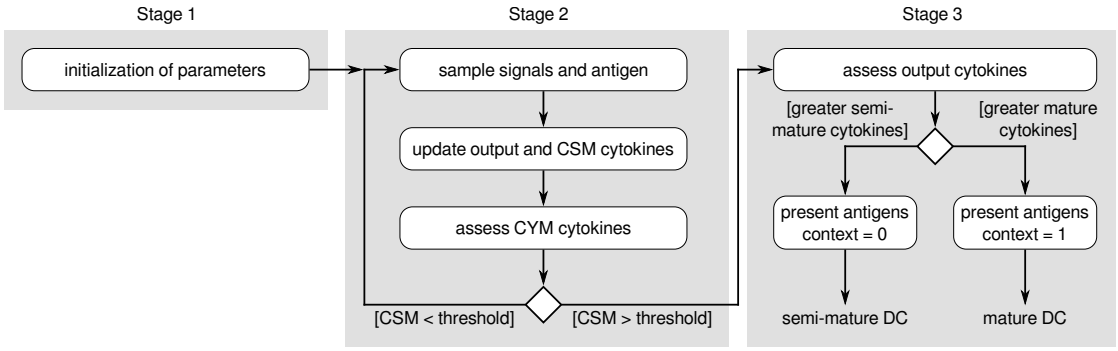


Figure 5.3: Key features of DC biology used in the DCA (after [270, Fig. 5.3])

5.5.1.1 Cell Update

Until the lifetime of a cell is exceeded (i.e., update stage), each cell iteratively performs three functions: 1.) the sampling of antigens, 2.) the update of the input signals, and 3.) the calculation of the cell's interim output signals. The core mechanism of the cell update stage is the collection of antigens and signals over the DCs' lifetime. Four types of input signals are combined to acquire contextual information on the status of the target system (cf. [226]). They are analog to the natural signals observed in the HIS [224]:

- PAMP (P) – signals that are known to be pathogenic.
- safe (S) – signals that are known to be normal.
- danger (D) – signals that indicate changes in behavior.
- inflammatory (I) – signals that amplify the other signals.

Based on these input signals, the DCA calculates three intermediate output values:

- co-stimulatory molecule (CSM): expresses the cell's maturation status
- semi-mature value: response to a safe environment
- mature value: response to a dangerous environment

The correlation of input to output signals is shown in Figure 5.4. In this illustration, the thickness of the lines expresses the transforming weights.

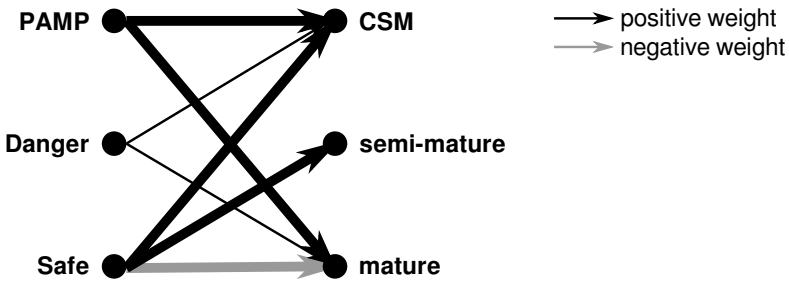


Figure 5.4: Abstract model of the DCA signal processing (after [270, Fig. 5.4])

In biology, PAMP are occurrences known to be not produced by the host, hence, a clear sign of danger [270]. In the DCA, they lead to an increase in CSM and mature output signals resulting in an earlier maturation with an anomalous context (i.e., `mDC`). The CSM expresses the maturation status of the cell, that is, whether the cell is ready for antigen presentation [270]. Danger signals are indicators of possible anomalies and influence the CSM and mature output signals, but, as can be seen in Figure 5.4, much lower than the PAMP signals [270]. On the other hand, safe signals suppress the production of the mature output signal (negative weight) and cause an increase in the semi-mature output value. Still, they contribute to the DC's maturation (i.e., increase of the CSM value).

The intermediate output signals are derived from the input according to the equations presented in [226]:

$$C_{csm} = \left(\underline{2} \sum_{i=0}^{I} P_i + \underline{1} \sum_{i=0}^{I} D_i + \underline{2} \sum_{i=0}^{I} S_i \right) \cdot (1 + IC) \tag{5.1}$$

$$C_{semi-mature} = \left(\underline{0} \sum_{i=0}^{I} P_i + \underline{0} \sum_{i=0}^{I} D_i + \underline{3} \sum_{i=0}^{I} S_i \right) \cdot (1 + IC) \tag{5.2}$$

$$C_{mature} = \left(\underline{2} \sum_{i=0}^{I} P_i + \underline{1} \sum_{i=0}^{I} D_i + (\underline{-3}) \sum_{i=0}^{I} S_i \right) \cdot (1 + IC) \tag{5.3}$$

where C_{csm}, $C_{semi-mature}$, and C_{mature} are the intermediate output signals respectively, P_i represent the PAMP signals, D_i represent the danger signals, S_i represent the safe signals, and IC are inflammatory cytokines. The respective weights of the single terms (underlined numbers) are based on the suggestions in [341].

One effect present in this equation, but not shown in Figure 5.4, are inflammatory cytokines (IC) expressing an already ongoing infection. These signals have an amplifying effect on the other three input signals (i.e., PAMP, danger, and safe).

5.5.1.2 Data Aggregation

As shown in Figure 5.3, when a dendritic cell reaches the end of its life (i.e., its CSM value exceeds a defined threshold), its interim output signal concentrations are assessed to define its contextual status (i.e., semi-mature or mature). Based on this information, the accumulated antigens are classified based on whether more dendritic cells experienced this antigen in a normal or an anomalous context (i.e., binary classification). In the case of a dominating semi-mature signal, the group of antigens is assigned a "normal" context; otherwise, it is assigned an "anomalous" context. As opposed to most other immune-inspired algorithms, the DCA uses the collected antigens merely for labeling and tracking of data rather than for detection purposes.

5.5.1.3 Algorithmic Properties

The DCA was initially designed as an offline anomaly detection algorithm to be applied to network intrusion detection. Due to the replication of the DCs' functioning, it shows similarities with certain filtering techniques. In addition, the DCA has lower computational complexity (in comparison with other ML techniques) and it does not require extensive training periods [262]. For this reasons, it has also shown preliminary success in resource-constrained applications such as sensor networks and mobile robotics [270].

Since the lifespan of the individual DC instances is limited and influenced by the environment, the DCA forms a filter-based correlation algorithm that includes a time window effect that reduces false positive errors [274]. Experiments on the DCA have shown a high accuracy [253], but also a comparably high FAR (cf. [108]). Additionally, the

initial DCA does not involve any learning mechanisms regarding the selection, mapping, and weighting of the signals used, making manual tuning and preparation necessary [345]. Therefore, there is great potential for future improvements regarding the signal sources and their respective mapping.

5.5.2 Variants and further Developments

The classical DCA gained promising results but contained stochastic elements and required the fine-tuning of more than ten parameters that made it more challenging to apply. Consequently, its foundation was theoretically analyzed, and some simplifications were introduced based on which the deterministic dendritic cell algorithm (dDCA) was proposed in [262]. The main changes concerned (i) the lifetime of the dendritic cells, (ii) the way antigens are sampled and stored, and (iii) the processing of the input signals. Regarding the latter, the calculation of the interim output signals was significantly reduced to one signal expressing the maturation (lifetime) status of the cell (i.e., co-stimulatory signal) and a second one keeping track of the experienced system context (i.e., context value). In the dDCA, the co-stimulatory signal is calculated with

$$csm = S + D \tag{5.4}$$

and the context value is expressed as

$$k = D - 2S \tag{5.5}$$

where D refers to the sum of danger signals and S to the sum of safe signals, respectively. The theoretical analysis for the reduction to these two interim signals is provided in [365]. However, for both only the danger and safe signals are used. Thus, the special roles of the PAMP and inflammatory processes were neglected. As a consequence, the parameters of the dDCA were reduced to (i) the input signals (danger and safe), (ii) the dendritic cell population size, and (iii) the lifetime of the single cells. While the input signals determine the detection capabilities of the dDCA, the population size and lifetime of the dendritic cells influence the smoothing and noise reduction properties of the algorithm, both responsible for decreasing the false positives rate (cf. [345]).

The classical DCA and the dDCA were used for a (binary) classification of offline data. Therefore, all data must be already available when the algorithm is applied. However, many anomaly detection systems require runtime (or even real-time) detection capabilities. A first approach to transform the DCA into a runtime detection algorithm by utilizing segmentation techniques is discussed in [366]. To avoid the need for segmentation, the authors of [367] proposed the minimized dDCA (min-dDCA). Their min-dDCA replaced the usual population sampling strategy with a one-to-one correlation between signals and antigens. Most importantly, they reduced the population size to one single dendritic cell with a lifetime of one iteration. Thus, the dendritic cell assigns a context to the present antigen in each iteration. But the runtime processing comes at the cost of missing result smoothing and noise reduction.

5.5.3 Related Work on DCA-based Fault Detection

So far, the majority of DCA-related works utilized the algorithm for IDS, predominantly in computer networks. The input signals are mainly derived from the network interface (e.g., number of messages received during a specific period) and the host operation system (e.g., process-related meta-information). The inclusion of node-level diagnostic information is only sparsely addressed in related work.

In the following, an overview of DCA-based fault detection approaches is presented. We focused on works that base their detection on the concepts inspired by the danger theory (i.e., dendritic cell behavior) and extend the review of DCA-based methods presented in [368]. However, an overview of detection approaches based on negative selection, clonal selection, and immune networks is available in [369, 370]. For a general overview of fault detection strategies and approaches, we refer an interested reader to the survey on fault detection in WSNs given in [3, 371].

One of the first works that used the DCA for fault detection was presented in [372]. The authors applied the principles of the DCA on a fault diagnosis system for rotating machinery in industrial facilities. Their input signals focused on the vibration pattern acquired from vibration sensors. Five signals derived from the vibration data, such as the kurtosis, were considered. The authors claimed that their approach achieved an overall diagnostic accuracy of over 93 %. However, they gave no details on their implementation and signal combination.

In [373], a DCA-based fault detection system for sensor faults in wind turbines is proposed. The approach used redundant sensor measurements to acquire the input signals for the DCA-based fault detection. In addition, the authors compared their approach with a NSA-based implementation. The results show that both immune-inspired techniques offer a similarly good fault detection rate, but the NSA suffered from a higher false alarm rate.

So far, the only work that incorporates node-level information in an immune-inspired fault detection approach is presented in [374] that was applied to a robotic system. The authors defined a set of so-called health indicators that are used as input for the DCA. These health indicators are derived from operational characteristics on the node level, such as energy consumption, battery level, component temperature readings, and task completion status. All proposed health indicators are calculated as the difference between two consecutive measurements. The authors present an extensive analysis of their approach that resulted in an overall fault detection rate of 98 % with only 0.128 % false alarm rate.

Similar to the health indicators proposed in [374], we presented our so-called fault indicators in [5] (see Section 6.2) that we implemented on a wireless sensor node [8] as described in Section 7.1.2. These fault indicators are also derived from node-level diagnostics that express possibly abnormal system conditions. In [5], we showed the ability of our fault indicators to mark potentially faulty circumstances, but did not incorporate them in an fault detection approach, yet.

5.5.4 Limitation of current Approaches

As stated above, most AISs and immune-inspired approaches are derived from one of the four "classical AIS theories". Concerning anomaly or fault detection, primary approaches based on negative selection (self/non-self discrimination) or techniques based on the functioning of the dendritic cells (contextual information fusion) have been proposed. While negative selection techniques dominated the early stages of AIS-based detection systems, an increasing number of dendritic cell-based algorithms have been proposed over the years. The reason for this is the usually high memory consumption and comparably high false positives rate of most negative selection approaches. Both disqualify negative selection approaches, especially from a meaningful use in resource-constrained systems like WSNs [149].

In most DCA-based approaches, the definition and pre-classification of the input signals (e.g., danger and safe) is a manual process that requires a certain level of knowledge and expertise of the target system. Similarly, the mapping and weighting of the input signals require manual intervention. In addition, the basic working principle of the DCA has no learning mechanisms. To cope with these limitations, several works suggested replacing the classification stage of the DCA with machine learning capabilities (cf. [368]). Especially the use of fuzzy inference systems has gained promising results [375]. Such approaches, however, entail a significant overhead on the memory and processing that prevent them from being used in resource-constrained systems like WSNs.

In addition, most of the proposed approaches, especially those for fault detection, are based on assumptions that significantly limit their applicability. For example, some require the WSN to consist of homogeneous sensor nodes with static positions and static network topology. These limitations are impracticable as real deployments often use heterogeneous nodes whose interconnects may change over time.

Also, the majority of the DCA-based fault detection systems derive the input signals purely from the sensor data where the considered fault models assume that faults significantly alter the sensed data. However, such data analytical detection approaches suffer from a disability in distinguishing rare but proper events from data anomalies caused by soft faults (cf. [8, Section 2.4]).

In contrast, our fault detection approach considers sensor data as well as node-level diagnostic information. For this reason, it can distinguish between the effects of events and faults in the measured data. Additionally, our approach is generally applicable as it:

- removes the need for domain or expert knowledge,
- does not need manual intervention and analysis,
- can also be used with heterogeneous sensor nodes, and
- is suitable for static and dynamic networks.

Therefore, it does not suffer from the limitations entailed by the assumption of the related works described before.

CHAPTER 6

Immune-inspired Node Fault Detection Approach

In the previous chapters, we elaborated on the importance of sensor node fault detection as an inevitable measure for dependable WSNs. Also, we discussed the shortcomings and limitations of current node fault detection approaches. This chapter presents our immune-inspired node fault detection approach that remedies several limitations of previous fault detection schemes.

Our fault detection approach took inspiration from the danger theory and its computational counterparts, in particular, the deterministic dendritic cell algorithm (dDCA). One of the main challenges in applying immune-inspired principles to fault detection systems is to define a suitable mapping from the entities of the HIS to the computational elements. Concerning the dDCA, especially the definition and mapping of the input signals is crucial for its proper functioning. In the following, we present our approach for a dDCA-based fault detection approach applicable to the detection of node faults in WSNs. We refer to the inputs of the dDCA as indicators rather than signals to avoid confusion as the term signal usually has another meaning in the areas of computer science and electrical engineering than it has in immunology. An overview of our approach is depicted in Figure 6.1.

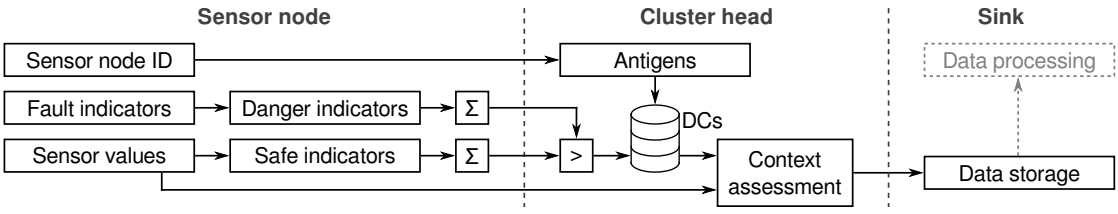


Figure 6.1: Overview of our immune-inspired fault detection approach

In this overview, a clustered WSN architecture is assumed where a dedicated cluster head forwards the data received from the sensor nodes to a central sink. However, our approach is not limited to this architecture as the dDCA-based detection can also be performed on any other system in the data chain able to perform the assessment of received sensor data (e.g., intermediate node or even the sink).

In the following, we first discuss the fault models considered in our work in Section 6.1. Our detection approach assumes that a fault is indicated by the presence of danger in combination with the absence of counteracting safeness; thus, the total value of the danger indicators is greater than the sum of safe indicators. In this context, our danger indicators are derived from our node-level fault indicators that we initially proposed in [5] and that we have implemented on a self-developed sensor node platform [8] (see Section 6.2). Their aggregation as well as the safe indicators derived from the sensor data are discussed in Section 6.3. Our modified dDCA with runtime detection capabilities is presented in Section 6.4. With Section 6.5, we complete the description of our approach with some reflections on its benefits and limitations.

6.1 Considered Fault Models

Faults can occur on different levels of the WSN due to various reasons and, as a consequence, can manifest themselves in diverse and often unpredictable ways. It is not possible to analyze or even model all potential faults that can occur on WSN nodes. But we can look for qualitative attributes that allow us to reason about possible faults. Therefore, we concentrate our analysis on the fault models that have been reported to have commonly happened in different WSN deployments (i.a., [75, 160, 173]), namely:

- ambient temperature faults (e.g., extreme values or fluctuations; see Section 6.1.1)
- supply voltage faults (e.g., undervolting; see Section 6.1.2)

In addition, we discuss the effect of high humidity and strong vibrations on the sensor nodes' operation in Section 6.1.3. An essential first step to identify available indications of faults on the node level is to analyze the effects and propagation of named faults in sensor nodes. Therefore, the selected fault models and their causes and effects are presented in the following.

6.1.1 Ambient Temperature Faults

Most WSNs are deployed outdoors where the ambient temperature is neither stable nor precisely predictable. Depending on the geographical location and the place of deployment, sensor nodes have to deal with extreme temperatures and significant temperature fluctuations. The former may be even worse when the sensor nodes' enclosure is exposed to direct sunlight as the infrared radiation can cause high temperatures inside the housing. Regarding the temperature, fluctuations as high as 34.9 °C within one hour have been reported [109]. These temperatures have a substantial impact on the wireless link quality [109, 173], but also have a considerable effect on the nodes' functionality [75, 160].

Besides the temperature changes caused by the environment, some effects can considerably impact the temperature experienced by a sensor node, such as the heating-up of the enclosure by direct sunlight. Also, components can suffer from (partial) short circuits or high loads that can additionally warm up the sensor node. As stated in [376], the ambient temperature can have a direct influence on the power consumption of electronic components resulting in an increase of the current of up to 15 times the average consumption under extreme temperatures, which can result in additional heating of the sensor node.

As a consequence, temperature-dependent hardware effects or material deformations can surface that may result in short or open circuits, timing variations, or varying CMOS threshold voltages. High temperatures also accelerate the aging of hardware components facilitating effects such as HCI, TDDB, or NBTI. Furthermore, extreme temperatures can stress the hardware components resulting in increased electromigration or the forming of metal whiskers. Hence, an analysis of the effects of the temperature on the performance of wireless sensor nodes is inevitable.

6.1.2 Supply Voltage Faults

Also, the supply voltage significantly impacts the proper functioning of wireless sensor nodes. This is especially true in WSNs as the sensor nodes are usually battery-powered with often no or limited power regulation or voltage management capabilities. As a result, the supply voltage levels can vary, causing different effects in the sensor node.

The main reason for differences in the supply voltage is a depleting battery. Thereby, the speed of depletion can be influenced by environmental conditions (see above) or by various kinds of faults occurring on the sensor node, such as (partial) short circuits caused by humidity in the enclosure. Also, bad connections between the sensor nodes' components can result in differences in their respective supply voltages.

The main effect caused by varying, or more specifically, sinking supply voltages is an undervolting of the sensor node. Undervolting refers to the operation of components with a supply voltage below their nominal supply voltage. Different components react differently to undervolting. For example, components with brownout detection tend to shut down before unintended effects can happen. Similarly, components with a voltage regulator (internal or external) tend to stop working at specific voltage levels, preventing them from operating at dangerously low voltage levels. If such protection is not available, components are possibly operated at voltage levels where they can produce wrong results. Additionally, the components used in a sensor node often have different minimum supply voltages; thus, there are voltage levels on which some components continue to operate while others have already stopped working or even show incorrect behavior (cf. [4, 75]). As a result, the sensor node can report wrong sensor readings.

Rapid changes in the supply voltage can also cause another effect known as power-glitch (also called VCC-glitch). Since CMOS gates are vulnerable to negative supply voltage spikes, an abrupt change in the supply voltage level can cause the MCU to skip certain

instructions resulting in an altered execution flow. This phenomenon is sometimes exploited in security attacks. Still, it can also happen unintentionally in sensor nodes due to rapid voltage changes, such as temporary short circuits caused by humidity in the enclosure.

6.1.3 Humidity and Vibration Faults

Although previous studies suggest that the main threat for a reliable sensor node operation stems from unpredictable changes in the node's ambient temperature and fluctuations in its supply voltage, other environmental factors also pose a risk to impaired node functionality. Depending on the deployment area, the sensor nodes may experience high levels of humidity and/or strong vibrations that pose a severe risk of physical damage to the node.

In combination with fluctuations in the ambient temperature, high levels of humidity can lead to the condensation of water inside the nodes' housing. The soaking of humidity into the housing can often not be prevented entirely. Specific sensors require direct contact with the sensed physical quantity (i.e., temperature, relative humidity, or gas sensors). If this condensation happens on the node's PCB, there is a significant danger of (partial) short circuits that can further lead to a damaging of the electrical components. Additionally, humidity in the sensor area can lead to abnormal sensor readings [377]. As a result, they can temporarily or even permanently impair the node's operation. However, in our outdoor experiments, we have found that even partial short circuits can have severe long-term effects as they lead to an increased corroding of electrical contacts.

Some WSN applications require the sensor nodes to be deployed in places where strong vibration can occur. For example, sensor nodes used in SHM or industrial process automation are susceptible to significant vibration, temporarily as well as permanent. The vibration causes physical stress on the components and, most of all, on their solder connections. As a result, the vibration can, either over time or in the case of intense vibration, damage the connections and/or components. The situation is even worse in case of manufacturing flaws (i.e., cold solder or dry joint). Consequently, the sensor node may experience (sporadic) open circuits on its PCB or inside components.

6.2 Node-level Diagnostics

An effective fault detection approach needs to consider node-level diagnostic data that allow inferring information on the sensor node's state of operation. As mentioned above, it is impossible to analyze all faults that can occur on sensor nodes. However, we can look for qualitative attributes that allow us to reason about possible faults. These attributes can be seen as node-level symptoms caused by faults (i.e., their manifestation).

An example for such node-level diagnostics are the fault indicators tailored for resource-constrained sensor nodes that we have initially presented in [8]. These indicators are metrics derived from self-checks and functional diagnostics performed on the sensor nodes.

The indicators are numerical metrics expressing the probability that the sensor node is currently affected by a soft fault where a higher number means a higher likelihood. Consequently, we define *fault indicators* as the subset of node-level data that is capable of indicating possibly faulty behavior. They aim at supporting the detection of permanent, transient, or intermittent soft faults to mitigate the risk of silent data corruption. While a single indicator can already be a good hint for faulty behavior, the real benefit comes from the fusion of several indicators.

The question remains which data is available on the sensor nodes that can indicate possibly faulty behavior. So far, most self-diagnostic approaches rely on the monitoring of the battery voltage as a measure of the remaining energy as well as specific link-related metrics such as the RSSI. In this section, we show that several more indicators are available that can be used to augment the detection of soft faults. Thus, they help to reduce the risk of transmitting corrupted sensor data. We based our work on the fault indicators partly on the node-centric metrics presented in [378] and the sensor network features proposed in [99].

The availability and quality of some indicators depend on the hardware used and the actual application. For this reason, we start by analyzing the possible sources of indicators for generic and specific data. As depicted in Figure 6.2, we broadly differ between two categories of fault indicators, namely those which are *inherently available* and those that can be *artificially added*. In the following, we will elaborate on both categories of indicators and present examples for them.

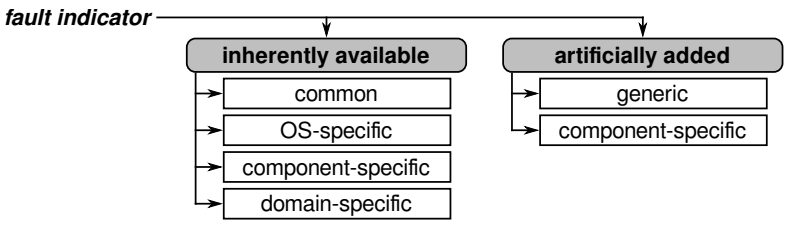


Figure 6.2: Fault indicator classification

Depending on application specifics such as the actual components used, more or even better indicators may be available, especially if specialized hardware or software is deployed. However, our work provides general considerations and shall serve as a starting point as we cannot capture all aspects of all possible combinations of applications and components.

6.2.1 Inherently-available Indicators

The first category of fault indicators is based on data that is inherently available on the sensor node. For these indicators, no additional hardware is required. Hence, these indicators are based on data available in the software.

As shown in Figure 6.2, the inherently available indicators can be further classified based on the source of the underlying data. We define four classes of inherent indicators which are 1) common, 2) OS-specific, 3) component-specific, and 4) domain-specific inherent indicators.

6.2.1.1 Common Inherent Indicators

These indicators rely on software additions that do not require specific hardware components. Such indicators are metrics derived from, for example, the control flow of the software. For example, we defined an *incident counter* which is increased by one every time a function failed (up to a predefined threshold) and decreased if a function returned with success (the lower boundary is 0). Alternatively, different increment/decrement values can be assigned depending on the function that failed/succeeded (i.e., its importance to a proper operation). If the threshold is exceeded, the sensor node is reset. The decreasing of the counter is necessary not to reset the node in case of transient faults. This indicator requires the sensor node's software to have functions that return information whether the function's execution was successful or not (e.g., a timeout occurred).

6.2.1.2 OS-specific inherent indicators

In several WSN deployments, the sensor nodes run an operating system (OS) such as Contiki, TinyOS, RiotOS, or FreeRTOS. Most OSs offer metrics such as CPU utilization (e.g., task execution and idle times), memory usage (e.g., stack and heap consumption), number of interrupts, and total operation time. Adding complexity to the node's software in the form of an OS or similar offers more potential for fault indicators. However, higher complexity increases the risk of faults and often requires an energy overhead in relation to the size of the addition.

6.2.1.3 Component-specific Inherent Indicator

The components of the sensor node (e.g., MCU, radio, sensors) can offer status information that can be leveraged to derive fault indicators. For example, the IEEE 802.15.4 radio transceiver XBee 3 from Digi provides diagnostic information such as the module temperature or the supply voltage level. Similarly, most MCUs offer ways to retrieve their core temperature and/or supply voltage. If at least two different sources for supply voltage information are available, the difference of both can be used as an indicator, too, especially if both devices are connected to the same voltage path. Also, the status registers of the CPUs can be used to infer information on the state of the current operation to detect, for example, the occurrence of over- or underflows in variable assignments. Some MCUs provide further helpful information such as the MCU status register (MCUSR) available in most AVR controllers that gives information on the source of the latest reset (i.e., watchdog, brownout, external, or power-on).

6.2.1.4 Domain-specific Inherent Indicators

Also domain-specific information as utilized in sensor data analysis (cf. Section 4.4.1) can be used to derive fault indicators [99]. For example, environmental features derived from the sensor location and observed physical certainties can be exploited. In this context, considerably large time gradients can be evidence of abnormal behavior as the rate of change of certain sensor measurements is limited by the natural laws of the observed phenomena and the characteristics of the sensors used. Abrupt changes in sensor values can indicate a short circuit in the internal sensor wiring. In contrast to the sensor data analysis approaches, the domain-specific inherent indicators can be combined with other indicators to distinguish correct events from fault-induced variations.

6.2.2 Artificially-added Indicators

The second category of fault indicators requires additional hardware and, therefore, are artificially added to the sensor node. We limit our considerations to artificial indicators requiring a small amount of additional hardware and feasibly small energy overhead. Complex circuitry and specialized hardware could offer indicators of high quality. However, such additions may consume too much energy or significantly increase the cost of the sensor nodes, which conflicts with their basic requirements.

The artificial indicators can be further divided into two classes: 1) generic and 2) component-specific artificial indicators.

6.2.2.1 Generic Artificial Indicators

This class of indicators encompasses possible additions for almost all sensor nodes. Examples are external power monitors to directly analyze the node's power consumption or additional sensors such as temperature sensors to measure the sensor node's surface temperature. In both cases, only a communication interface is required that is available on most sensor nodes. Power monitors such as the *INA219* can be used to gain additional information on the supply voltage and the current consumption, hence, the power used by the sensor node. Also, adding the measurement of the node's surface temperature (e.g., with a thermistor) offers potential for fault indicators, especially in combination with the core or ambient temperature readings (e.g., their differences).

6.2.2.2 Component-specific Artificial Indicators

Besides the generic additions, there are also hardware additions to the sensor node that are specific to its components and may therefore not be usable on all sensor nodes. Examples are external diagnostic hardware for the node's MCU such as debuggers or power tracers that usually require specific interfaces to work. Such additions provide data on the operational state of the node and, thus, can serve as a reasonable basis for fault indicators.

6.2.3 Remarks on Fault Indicators

As stated above, these fault indicators are meant to be additional inputs collected on each sensor node to support an existing fault detection approach. The actual detection can happen either locally with self-diagnosis techniques, distributed in group detection approaches, or a hybrid of both to leverage temporal and spatial correlations. In the case of self-diagnosis, the fault indicators are evaluated locally and do not cause any communication overhead. Also, it is unnecessary to define indicators for all connected sensors; thus, scalability is not impaired. In [5], we showed that adding such fault indicators to the detection approach enhances its detection rate and allows the distinction between soft faults and events.

Nevertheless, the fault indicators can be corrupted by faults, too. In such a case, there are two possible outcomes:

- faulty data are labeled correct
- correct data are labeled faulty

For the former, the reliability of the WSN is as good as without the use of indicators in the worst case. In the latter case, faulty indicators can lead to a rejection of data (i.e., dropping of messages) that reduces the WSN's availability.

However, in both cases, no corrupted data is caused by the use of the fault indicators. Also, the indicators are meant as an addition to an existing detection approach. The benefit of an improved detection rate combined with the ability to differentiate between faults and events outweighs the risk of discarding data due to faulty indicators.

6.3 Danger and Safe Indicators

The danger and safe indicators are contextual information used by the dDCA to assess the system's state of health, that is, fault detection. In our approach, we utilize node-level diagnostics based on the fault indicators presented in Section 6.2. For the use in our approach, these fault indicators need to be normalized in the range [0, 1]. Detailed information on the implemented fault indicators and their normalization is given in Section 7.1.2. However, our approach is not limited to the use of precisely these fault indicators. They can be adapted and extended if more suitable indicators are available on a target platform.

We derive the cumulative danger indicator D with:

$$D = \min\left(\sum \chi'_i, 1\right) \tag{6.1}$$

where χ'_i are the normalized fault indicators. The value of D is directly proportional to the probability of faulty circumstances on the sensor node ($D \in \mathbb{R} \mid 0 \leq D \leq 1$).

While the danger indicators rely on node-level diagnostics, the counter-regulatory safe indicators are derived from the reported sensor values. Following other DCA-based

approaches [372–374], we derive our safe indicators from the difference between successive sensor measurements where smaller differences are considered safer. Therefore, we base the safe indicators on the standard deviation σ_j of N consecutive measurements of sensor j. However, we express safeness as the absence of significant changes. Therefore, the maximum deviation of all sensors is decisive for the safe indicator. To avoid an overreaction to small changes that often happen in sensors due to natural noise, the maximum deviation is multiplied by a sensitivity factor ($q_{\text{sen}} \in \mathbb{R} \mid 0 < q_{\text{sen}} \leq 1$). Finally, the resulting numerical value is used as the negative exponent of an exponential function to have a decreasing value in case of higher deviations and limit the resulting numerical value in the range $[0, 1]$. Consequently, the aggregated safe indicator for m sensors is calculated with:

$$S = e^{-\max(\sigma_1,\ldots,\sigma_m)\cdot q_{\text{sen}}} \tag{6.2}$$

with ($S \in \mathbb{R} \mid 0 \leq S \leq 1$). Our safe indicators assume continuous sensor data. For discontinuous sensor data, the safe indicators can be derived from other metrics that express the safeness of operation, such as the distribution and/or amount of events over time.

6.4 Modified Runtime DCA

The entire process of our modified dDCA works as follows. Every time new data is received from the sensor nodes, four main tasks are performed:

1. acquire the present antigen
2. update the danger and safe indicators
3. update the population of dendritic cells
4. classify the new sensor values

The single tasks are described in the following, and considerations on the modifications of the dDCA are discussed.

6.4.1 Antigen Definition

The antigens are used as unique labels for data from one instance to be assessed (i.e., sensor node). They have no direct influence on the classification process. In our case, we used antigens that are based on the lower 32-bit of the media access control (MAC) address of the used Zigbee radio transceivers as they are (i) unique and (ii) inherently available on the Xbee modules. Our approach is not limited to Zigbee networks, as the antigens could also be taken from another source or even be hard-coded; they simply serve as data labels.

6.4.2 Indicator Update

The update of the indicators is based on the fault indicators and the sensor data received from the sensor nodes. It follows the procedure discussed in Section 6.2 and 6.3.

6.4.3 DC Population Update

The dDCA uses a population of abstract dendritic cells, each storing the antigens they experienced during their life and the cumulative context defined by the indicators, or signals, respectively. However, the dDCA and its predecessor, the DCA, are both designed for static data analysis. The runtime min-dDCA, on the other hand, lost the smoothing and noise reductions capabilities by using only a single dendritic cell.

To have both runtime processing and the beneficial properties of the population-based assessment, we modified the min-dDCA by re-introducing a population of dendritic cells with specific properties. In our approach, a dendritic cell is linked to precisely one antigen where a maximum of M dendritic cells exists for each distinct antigen. Thus, every dendritic cell has two properties: (i) the antigen it is linked to and (ii) the danger/safe context associated with this specific antigen.

We adapted the process of how the population of cells is built so that in every iteration (i.e., update), a new dendritic cell is created with the currently observed antigen assigned. If the number of cells for this antigen exceeds the population size M, the oldest cell linked to this antigen is deleted. Hence, after experiencing the same antigen for the M-th time, a constant population of M dendritic cells is kept. This procedure is summarized in Algorithm 6.1 where DC is the population of dendritic cells represented as a list of tuples (Ag, k), Ag is short for antigen, k is the cell's context value, and M is the maximum population size per antigen type. In the algorithm, the *filter* method returns the subset of list elements that satisfy the given predicate.

Algorithm 6.1: Update of the dendritic cell population

Input: list of dendritic cells $DC : [(Ag, k)]$,
current antigen Ag'
Output: updated list of dendritic cells $DC' : [(Ag, k)]$

$DC' = DC$
Add $cell = (Ag', 0)$ to DC'
if $|DC'.filter(Ag = Ag')| > M$ **then**
  Remove oldest cell with $Ag = Ag'$ from DC'
return DC'

Next, the dendritic cells' context values are updated. These values express the context a specific antigen was experienced in, that is, whether the antigen was found in a context with predominantly danger or safe indication. Consequently, the update of the context values depends on the previously updated values of the indicators. The context value is updated by adding the difference between the danger and the safe indicator (i.e., $k = k + (D - S)$). Algorithm 6.2 depicts the procedure of the cells' context update.

By applying this form of dendritic cell population management scheme, a sliding window of temporally ordered context values is utilized for the subsequent fault classification.

Algorithm 6.2: Update of the cells' context value

Input: list of dendritic cells $DC : [(Ag, k)]$,
current antigen Ag',
danger and safe indicators D, S
Output: updated list of dendritic cells $DC' : [(Ag, k)]$

$DC' = DC$
for $cell \in DC'$ **do**
  if $cell.Ag = Ag'$ **then**
    $cell.k \leftarrow cell.k + D - S$
return DC'

The size of the window can be controlled via the maximum population size M (similar to the original dDCA as analyzed in [345]). Regarding the fault detection accuracy, an optimal value of M depends on the sensor update interval and the maximum change of the sensor value to be expected between two successive measurements, both with a directly proportional relation.

6.4.4 Sensor Value Classification

Whether the present antigen (i.e., sensor node) is considered faulty depends on the context values of the dendritic cells associated with said antigen. It involves two steps:

- Assess the context value of each associated cell
- Classify the antigen based on the cells' context labels

Regarding the former, a single cell considers the respective antigen as faulty if the cumulative context value k is greater than or equal to zero. That implies that the cell experienced higher values of danger indicators than safe indicators. Otherwise, the antigen is considered normal as no evidence of faulty behavior has been found. Thus, the context label Cx of dendritic cell m is determined as:

$$Cx_m = \begin{cases} 1, & \text{if } k_m \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$

where 1 signals a faulty and 0 a normal context, respectively.

After that, a majority voting of the related context labels Cx is performed to classify the sensor data as either normal or faulty. To do so, the mean value of Cx of all dendritic cells linked to the current antigen Ag' is calculated with:

$$\overline{Cx_{Ag'}} = \frac{1}{M_{Ag'}} \sum_{m=1}^{M_{Ag'}} Cx_m \tag{6.4}$$

where $M_{Ag'}$ is the total number of cells with antigen Ag' ($M_{Ag'} \in \mathbb{N} \mid 0 < M_{Ag'} \leq M$). If $\overline{Cx_{Ag'}} \geq 0.5$, the current antigen and, thus, the corresponding sensor data are labeled as *faulty*, otherwise they are considered *normal*.

Alternatively, the value of $\overline{\mathrm{Cx}_{Ag'}}$ can be directly used as a fault coefficient, for example, to weight the corresponding sensor values in the further data processing. Thereby, sensor values with a lower fault coefficient have a more significant impact on the subsequent data processing than values considered faulty (i.e., sensor values with a higher fault coefficient).

6.5 Considerations on the Detection Approach

By applying our modified dDCA, the received data is assessed during runtime and the sensor data can immediately be classified as either normal or faulty. Thus, we combine the strengths of the dDCA with those of the min-dDCA, which is a lightweight and real-time assessment of the diagnostic and sensor data to detect faulty circumstances.

However, our proposed approach includes parameters that need to be adjusted appropriately (e.g., maximum population size M, sensitivity factor q_{sen}). Unlike in other approaches, our modified dDCA requires fewer parameters that are not application-specific and, thus, can be used in almost all systems once optimal values have been found. If adaptions of the parameters are necessary to improve the correctness, they can be used for an entire domain and do not need to be adjusted for every application.

We currently use a unified weighting of the single fault indicators for our danger indicator. Further experiments may suggest that specific fault indicators are more sensitive to node faults than others and, thus, a different weighting scheme could score better results.

Additionally, the DCA and most of its variants do not include learning capabilities. A theoretical analysis of the algorithmic basis revealed that the DCA is, in principle, a collection of linear classifiers (cf. [368]). As presented in Section 5.5.4, several works proposed concepts to incorporate machine learning into the DCA for two purposes: (i) an automated mapping and weighting of the input parameter (i.e., signals or indicators) and/or (ii) an introduction of immune-memory. The latter refers to the capability of the system to learn from previous fault encounters to achieve a faster reaction in case the same situation is experienced again. Currently, neither the automated mapping/weighting nor the immune memory are included in our approach. However, they are promising future research directions to improve and extend our concept.

CHAPTER 7

Concept Evaluation

In the following, we present the evaluation of our fault detection approach. We show that it enhances the reliability of the nodes by (i) improving the accuracy of sensor node fault detection while (ii) posing only a reasonably small resource overhead. To evaluate our immune-inspired fault detection approach, we used simulations in combination with an implementation of the entire concept in a practical WSN. The latter consisted of two parts, namely (i) lab experiments in a controlled environment and (ii) a WSN testbed (indoor and outdoor). This tripartite experiment setup is necessary as the analysis of (soft) fault detection in WSNs is a challenging and non-trivial task. The considered faults depend on many factors influencing the sensor nodes' operation that can, either alone or in combination, lead to soft faults such as silent data corruption (cf. Section 6.1).

Each part of this tripartite setup targets a different aspect of the approach evaluation. The simulation is primarily used to analyze the detection characteristics of our immune-inspired approach and to compare its detection characteristics with alternative methods. An investigation of the approach's reaction to specific fault models is the focus of the lab experiments where the control over the environment allows us to force the sensor nodes into particular conditions (i.e., perform physical fault injection). Long-term characteristics and energy efficiency are targeted in the WSN testbed. Moreover, the testbed consists of an indoor and an outdoor deployment to evaluate our approach in a real-world setting where also faults occur that are not covered by our previously defined fault models. Therefore, our tripartite evaluation considers many factors relevant for correct and efficient sensor node fault detection.

First, we will present our self-developed sensor node platform, the ASN(x), in Section 7.1. The design of the ASN(x) includes several diagnostic measures (i.e., fault indicators) to support the detection of faults with qualitative node-level information. We have implemented the entire fault detection approach as discussed in Section 7.2. In Section 7.3, we present the simulation environment used for the correctness analysis and a benchmark

with alternative fault detection methods. The lab experiments performed are shown in Section 7.4 followed by details on our indoor and outdoor WSN testbed in Section 7.5.

7.1 The ASN(x) Platform

Applying fault detection on a node level is crucial to improve the WSN's overall reliability. Nevertheless, most related sensor nodes in our literature review enable low-power operation but do not offer node-level fault detection capabilities. That, however, has significant drawbacks as presented in Section 4.4. To improve the detection of node-level faults and allow one to distinguish between faults and data events, we argue that it is inevitable to incorporate self-diagnostic measures into the sensor nodes' design.

The development of a sensor node from scratch is a complicated task that takes much time and entails many pitfalls. For this reason, researchers often utilize ready-to-use sensor node platforms like the Arduino [6]). As presented in Section 2.2.3.4, various commercial products from different manufacturers have been introduced over the years. While these node platforms usually are highly optimized and are proven in practical use, they are often limited in their usage (i.e., tailored for specific applications), require specific toolchains or programming languages, are difficult to acquire (limited availability on the market) or are too expensive (refer to Table 2.1). Additionally, several custom sensor nodes have been used by researchers in the literature. However, most of them are not publicly available, use obsolete components, or lack support (especially after the corresponding research projects ended). As a result, finding a suitable platform for WSN deployments is often not easy but is necessary for practical research.

Based on the concept of fault indicators that we initially proposed in [5] and our findings on efficient sensor node design in [6], we developed a new sensor node platform named *AVR-based Sensor Node with Xbee radio*, or short *ASN(x)*, that is completely open-source[1] and free to use[2]. As the name implies, the ASN(x) is based on an Atmel AVR MCU that commonly employ a modified 8-bit Harvard reduced instruction set computer (RISC) architecture. It enables node-level fault detection by using specialized self-tests and diagnostic data, a concept we named *active node-level reliability*. The detection is based on the use of fault indicators that allow us to infer information on the state of health of the sensor node's operation and, in turn, can indicate possibly erroneous operational conditions. This additional information can be used to augment existing WSN-specific fault detection approaches (centralized as well as distributed) to improve the fault detection rate and, most importantly, to enable the distinction between data anomalies caused by rare events and fault-induced data corruption. Thereby, the fault indicators require only a justifiable resource overhead to keep the hardware costs as well as the energy consumption at a minimum while significantly improving the WSN's reliability. Security on the device and communication level was not in the focus of our

[1]The ASN(x) is available at https://github.com/DoWiD-wsn/avr-based_sensor_node.
[2]Published under the MIT license, see https://choosealicense.com/licenses/mit/.

work. However, security and dependability are integrated concepts (cf. [162]); hence, increased reliability typically also significantly influences security positively.

7.1.1 Design and Components

The ASN(x)[3] is a wireless sensor node platform that offers several beneficial features for sensor nodes as it:

- enables active node-level reliability by incorporating self-check capabilities,
- offers an energy-efficient operation especially suitable for monitoring applications,
- is versatile regarding its usage (i.e., modular expandability),
- is based on current components that are highly available on the market,
- is comparably cheap (less than $50 per node including the radio), and
- is completely open-source published on Github under the MIT license.

As depicted in Figure 7.1, the ASN(x) incorporates the basic components of a wireless sensor node presented in Section 2.2.2 (see also Figure 2.3), that are:

- a processing unit (see Section 7.1.1.1),
- a sensing unit (see Section 7.1.1.2),
- a power unit (see Section 7.1.1.3), and
- a transceiver unit (see Section 7.1.1.4).

Aside from an onboard temperature sensor (TMP275), the ASN(x) can be freely extended with application-specific sensors via extension headers with several general-purpose input/outputs (GPIOs), two one-wire interface (OWI) connectors with separate data lines, and two two-wire interface (TWI) connectors (i.e., I2C). The ASN(x) can be easily programmed via a 6-pin AVR in-system programming (ISP) connector. Additionally, for timestamping purposes or to have an external wake-up source for the MCU, a PCR85263A low-power RTC is available on the sensor node. Nevertheless, one of the essential features of the ASN(x) are the self-check measures added to the design to allow node-level fault diagnosis as presented in Section 7.1.2. The particular units, their components, and their characteristics are described in the following.

7.1.1.1 Processing Unit

The core of the ASN(x) forms the ATmega1284P, a high-performance 8-bit AVR RISC-based MCU with rich on-chip peripherals. It has 128 kB ISP flash memory, 16 kB SRAM, and 4 kB EEPROM. The ATmega1284P can be clocked either by the internal 8 MHz RC oscillator or an external clock source with up to 20 MHz. In the ASN(x), the MCU is clocked via an external 4 MHz crystal oscillator. As the MCU can execute most instructions in a single clock cycle, the processor runs almost 4 million instructions per

[3]The information in this thesis refers to the hardware revision v1.5 (2022-02) of the ASN(x).

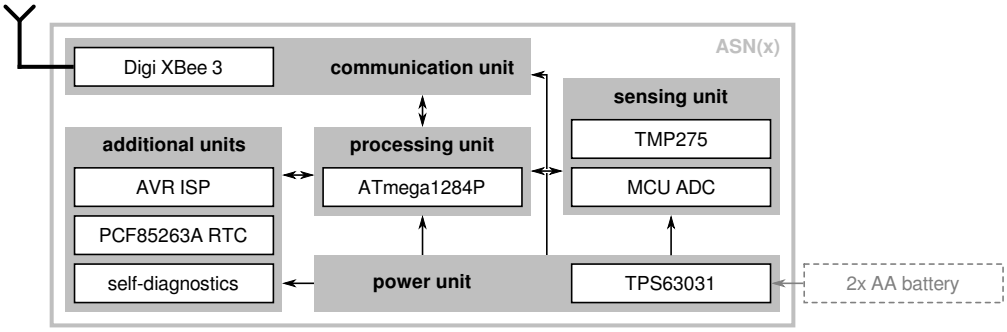


Figure 7.1: Basic components of the ASN(x) sensor node platform

second (MIPS). The ATmega1284P's operational temperature range is -40 up to +85 °C, which makes it usable for indoor and (most) outdoor applications. Additionally, it offers many on-chip peripherals such as a 10-bit ADC with eight input channels, 8- and 16-bit timers, and several communication interfaces (i.e., two USART, one SPI, and one TWI) while requiring only a minimal amount of external (passive) components. We found that having two USART interfaces is beneficial for sensor nodes as one is often used to communicate with the radio transceiver, and a second one is helpful when developing and debugging the node's software. Also, the ASN(x) has two user LEDs that can be physically disconnected if not needed to save energy.

To upload the node software onto the MCU, the 6-pin AVR ISP connector can be used. It is connected to the MCU's SPI and allows writing data to the ISP flash memory. However, a programmer to connect the ASN(x) with the host computer is needed, which is available for around $20. Alternatively, a bootloader could be used that allows uploading new programs via USART like it is done with most Arduino boards. Such a bootloader would occupy a certain amount of flash memory (e.g., 500 bytes in case of an *optiboot*-based bootloader[4]) but would allow to easily update the sensor node's software with just a serial connection (e.g., via an FTDI universal serial bus (USB)-to-serial adapter).

One of the essential characteristics of an MCU to be used in wireless sensor nodes is its power consumption and the availability of suitable power saving modes (i.e., sleep modes). The ATmega1284P provides six different software-selectable power saving modes with varying domains of clock remaining active and other wake-up sources for the MCU. In the most power-saving mode, the power-down mode, the external oscillator is stopped; only the watchdog timer (WDT) (if enabled) continues to operate. Since almost the entire MCU core is disabled, only an external event such as an external interrupt, an TWI address match, or a reset (either external, brown-out, or initiated by the WDT) can wake the MCU up from this mode. The power consumption of the MCU can be further decreased by deactivating the WDT and the brown-out detector. Additionally, the power consumption is affected by the external (passive) wiring.

[4]For more information on *optiboot*, we refer to https://github.com/Optiboot/optiboot.

As sensor nodes in environmental monitoring applications are usually active for a short time and spend the rest of the time in a sleeping state, a reliable wake-up source allowing for intervals in the granularity of minutes up to a few hours is needed. For this purpose, an external RTC is commonly utilized that generates an external interrupt for the MCU after a defined period. For the ASN(x), we included a PCF85263A low-power RTC that can be operated either as a calendar-optimized clock or as a stopwatch (i.e., an elapsed time counter). The stopwatch mode is most suitable to generate a periodic wake-up signal (i.e., external interrupt) where the desired interval can be easily configured. The PCF85263A is clocked by an external 32.768 kHz quartz crystal. However, it is of utmost importance to ensure that the interrupt generated by the RTC reliably wakes up the MCU (i.e., proper RTC and MCU configuration). Otherwise, the node may end up in a state where it never wakes up from the power-down mode again.

7.1.1.2 Sensing Unit

The ASN(x) has an onboard TMP275 low-power temperature sensor connected via TWI/I2C. It enables temperature measurements for ambient temperatures between -40 and +125 °C with an accuracy of ±1 °C over the full range and ±0.5 °C for temperatures between -20 and +100 °C, respectively. The conversion resolution can be configured in software between 9-bit (0.5 °C granularity with 27.5 ms typical conversion time) and 12-bit (0.0625 °C granularity with 220 ms typical conversion time). Additionally, it can be configured for a one-shot temperature measurement mode where the sensor performs one conversion on demand and remains in a low-power state for the rest of the time.

Since the ASN(x) is meant to be a generic platform for monitoring applications, however, the sensor node provides interfaces for various types of sensors rather than having several sensors mounted on the PCB. Thereby, the costs are kept to a minimum as no unused sensors are included, and similarly, the power consumption is not burdened by mounted but unneeded sensors. Depending on the application, the sensors required can be connected to the available pin headers offering GPIOs (9x), ADC inputs (6x) as well as digital interfaces such as USART (1x), SPI (1x), OWI (2x), and TWI (2x). To connect the sensors, either cables connected to the pin headers can be used or a sensor add-on[5] can be developed. The latter is beneficial if numerous nodes with the same set of sensors have to be deployed.

Also, some of the self-diagnostic measures (i.e., fault indicators) are sensorial. However, since their primary purpose is node-level fault detection rather than actual sensor value monitoring/reporting, we will specifically discuss them in Section 7.1.2.

7.1.1.3 Power Unit

As shown in Table 2.1, most of the available sensor nodes are directly powered by (two AA) batteries or use linear regulators. Directly supplying the sensor node does not need any additional hardware for voltage regulation which saves costs and does not add

[5] An ASN(x) add-on template is provided at https://github.com/DoWiD-wsn/asnx_addon_template.

any extra power dissipation. However, this option entails the hazard of undervolting of the components by a depleting battery that can result in severe soft faults, as we have analyzed in [4]. This threat is removed by using a linear regulator to ensure a stable supply voltage at the cost of bad energy efficiency. Those regulators convert the voltage surplus to heat. Additionally, linear regulators only work as long as the input (battery) voltage exceeds the desired supply voltage. If the sensor node is supplied with two AA batteries resulting in a nominal voltage of 3 V, a supply voltage of 3.3 V can not be realized with a linear regulator.

For these reasons, we employed a single inductor buck-boost DC/DC converter with a fixed output voltage of 3.3 V in our ASN(x), more specifically, the TPS63031. Due to its buck-boost capability, a wide input voltage range of 1.8 to 5.5 V is supported with an input-to-output efficiency of above 65 %. Thus, the input voltage range perfectly fits the voltage range of two AA batteries/accumulators but also offers the possibility to attach other forms of energy sources as long as they do not exceed 5.5 V.

The energy efficiency of the TPS63031 mainly depends on two factors, namely the input voltage and the output current. Regarding the former, the DC/DC converter operates more efficiently in buck mode, that is, in cases where the input voltage is higher than the output voltage. When supplied with two AA batteries, the TPS63031 is operated in boost mode that offers a slightly worse efficiency that is still above 65 % and, thus, much better than a linear regulator-based solution.

Concerning the output current, the TPS63031 can provide up to 500 mA in boost mode and even up to 800 mA in buck mode when operated in normal operation mode. Such high values are rarely needed on sensor nodes. For output currents below 100 mA, the DC/DC converter offers a power-save mode. In this mode, the converter is operated asynchronously and stops whenever the output voltage is at or above its nominal value. Only if the output voltage drops below its nominal value, the converter is started and ramps up the output voltage for one or several pulses. For an analysis and discussion of the efficiency of the converter over the full battery-powered supply range as well as the effects of the power-save mode on the node's operation, we refer to [8].

7.1.1.4 Transceiver Unit

The ASN(x) has a 20-pin socket with a pin assignment commonly used in XBee through-hole technology (THT) modules, whereby not all signals are connected to the MCU. In the current version of the ASN(x), the USART and the SPI signals are connected as well as the two pins responsible for controlling the pin-sleep functionality, that is, the sleep request and the sleep indication pins.

Initially, the ASN(x) was designed for using a Digi XBee 3 RF module, hence, the "(x)" in its name. The module supports different networking protocols (i.e., Zigbee, IEEE 802.15.4, and DigiMesh), where Digi provides a corresponding firmware for each protocol. For the ASN(x), we currently use the Zigbee firmware to establish a Zigbee 3.0 network.

The Xbee 3 module additionally has a BLE interface that can be activated for debugging or configuration purposes.

However, the XBee modules' 20-pin footprint and pin layout have become a standard design. Today, numerous modules featuring different RF technologies are available in the "XBee" (or sometimes simply "Bee") layout, such as the Core2530 module[6] with a TI CC2530F256 Zigbee radio. For this reason, the "(x)" in ASN(x) may also be seen as an abbreviation for "eXtensible" as the sensor node can be easily extended with any radio transceiver available in the XBee layout. The same concept is also used, for example, in the Libelium Waspmote to support a wide range of different applications and use cases with a single hardware platform (cf. [59]).

7.1.2 Node-level Indicators

To the best of our knowledge, the ASN(x) is the first sensor node that enables active node-level reliability by including node-level fault indicators. Fault indicators are enhanced self-diagnostic measures added to the sensor node design, both in hardware and in software as presented in Section 6.2. The diagnostic information obtained from the fault indicators allows us to infer qualitative data on the sensor node's operation state. In other words, they indicate circumstances that facilitate erroneous behavior. These simple self-checks offer an excellent way to enhance the node's reliability while only requiring minimal resource overhead. Sensor data reported in times of active fault indicators are more likely to be corrupted (or even arbitrary) than data acquired in phases of dormant fault indicators.

We denote our fault indicators with the Greek letter χ (small chi) due to the name similarity of our fault indicators with indicator functions used in mathematics that are commonly represented with the letter χ. In the current version of the ASN(x), eight different fault indicators are implemented as listed in Table 7.1. Each indicator targets a different component or operational aspect of the sensor node, as described in the following. These diagnostic data can either be checked on the sensor node and/or be sent to other network participants to establish a distributed fault-detection scheme. As presented in Section 6.2, the fault indicators can be categorized based on whether they are inherently available on the node (i.e., software metrics) or can be artificially added (i.e., additional hardware). However, the possibilities of fault indicators are not limited to the set currently implemented on the ASN(x). There are further sources of node-level data that could be leveraged for improved fault detection such as those mentioned in Section 6.2. In [379], also composed indicators are utilized that allow to link couples of indicators by a given relation. Such sophisticated indicators offer a promising future research direction.

[6] For more information on the Core2530 module, see https://www.waveshare.com/wiki/Core2530_(B).

Table 7.1: Overview of the available ASN(x) fault indicators

Indicator		Category	Section
χ_{NT}	Node temperature monitor	Artificial generic	7.1.2.1
χ_{VS}	Supply voltage monitor	Inherent component-specific	7.1.2.2
χ_{BAT}	Battery voltage monitor	Artificial generic	7.1.2.3
χ_{ART}	Active runtime monitor	Inherent component-specific	7.1.2.4
χ_{RST}	Reset monitor	Inherent component-specific	7.1.2.5
χ_{IC}	Software incident counter	Inherent common	7.1.2.6
χ_{ADC}	ADC self-check	Artificial generic	7.1.2.7
χ_{USART}	USART self-check	Artificial component-specific	7.1.2.8

7.1.2.1 Node Temperature Monitor

The first fault indicator is derived from temperature measurements of the ASN(x)' components, in particular, the MCU's surface temperature, the board temperature, and the radio transceiver core temperature. It is denoted as χ_{NT} where "NT" stands for "node temperature".

The MCU's surface temperature (T_{MCU}) is measured via a 103JT-025 thin-film negative temperature coefficient (NTC) thermistor affixed to the MCU's surface. To get the respective temperature, the thermistor is used in a voltage divider in combination with a 10 kΩ balance resistor placed on the high side (i.e., connected to the supply voltage), and the thermistor is located on the low side (i.e., connected to ground). The voltage divider midpoint is connected to the MCU's ADC. In order not to waste energy, the voltage divider can be enabled and disabled via an N-channel metal-oxide-semiconductor field-effect transistor (MOSFET) controlled by a GPIO. We use the Steinhart-Hart equation to calculate an approximation of the thermistor's temperature based on the ADC's conversion result and the thermistor's characteristics (i.e., beta value).

The board temperature (T_{BRD}) is provided by the onboard TMP275 temperature sensor connected to the MCU via TWI. In our setup, we configured the sensor to provide us with temperature measurements with a 10-bit resolution (0.25 °C granularity) that takes approximately 55 ms for single conversions. As a result, it offers a good balance between measurement accuracy and the required conversion time.

The radio transceiver core temperature (T_{TRX}) of the XBee 3 module is available as part of the diagnostic information provided by the module. It can be read from the module using AT commands (i.e., Hayes commands) issued via the USART interface.

All three temperature measurements are taken from places on the sensor node inside the node's housing. Consequently, all three measurements should be pretty similar. While the absolute value may differ (i.e., have an offset of a few degrees Celsius), the trends of the temperature measurements should have a negligibly small difference as we expect the measurements to react to external influences equally. Therefore, the node temperature

monitor fault indicator χ_{NT} is defined as the standard deviation of the changes of the three temperature measurements considering two consecutive measurements denoted as ΔT_{MCU}, ΔT_{BRD}, and ΔT_{TRX}, respectively. χ_{NT} is calculated with:

$$\chi_{NT} = \sqrt{\frac{(\Delta T_{MCU} - \mu_{NT})^2 + (\Delta T_{BRD} - \mu_{NT})^2 + (\Delta T_{TRX} - \mu_{NT})^2}{3}} \tag{7.1}$$

where μ_{NT} is the mean value of the temperature readings calculated as:

$$\mu_{NT} = \frac{\Delta T_{MCU} + \Delta T_{BRD} + \Delta T_{TRX}}{3} \ . \tag{7.2}$$

Thereby, a higher value of χ_{NT} indicates a higher probability of experiencing a faulty system condition. So far, we use a uniform weighting of the single temperature measurements. However, a future analysis may suggest using different weights.

Regarding the fault indicator categorization, the node temperature monitor implemented on the ASN(x) requires additional hardware (i.e., TMP275 temperature sensor, thermistor circuit) that can be, however, added to almost every sensor node. For this reason, this fault indicator belongs to the artificial generic indicators.

Some MCUs such as the ATmega328P have a core temperature sensor implemented into their ADC peripheral and, thus, would allow acquiring the MCU core temperature without the need of additional hardware. In such a case, a simpler χ_{NT} could be implemented as an inherent component-specific indicator by considering the MCU and radio core temperatures only.

7.1.2.2 Supply Voltage Monitor

Our previous analysis [5] has shown that aside from the ambient temperature, especially the supply voltage of the sensor node has a significant influence on its proper operation. Consequently, our second fault indicator χ_{VS} considers the supply voltage on the sensor node measured by the MCU and the XBee independently.

The supply voltage level of the MCU can be acquired without the need for additional hardware or circuitry using the on-chip ADC as described in Microchip's application note AN2447 [380]. Similar to the temperature, the supply voltage level of the XBee is provided as a diagnostic value retrievable via an AT command. For this reason, χ_{VS} is an inherent component-specific indicator as the measurements are inherently available but are specific to the hardware components used.

The onboard DC/DC converter regulates the supply voltage and, in a fault-free operation, should constantly be 3.3 V (with minor fluctuations). We derive χ_{VS} as the absolute difference between the measured MCU supply voltage (V_{MCU}) and the radio transceiver supply voltage (V_{TRX}) with:

$$\chi_{VS} = |V_{MCU} - V_{TRX}| \tag{7.3}$$

where the probability of a faulty condition is directly proportional to the value of χ_{VS}.

7.1.2.3 Battery Voltage Monitor

Aside from the supply voltage, also the battery voltage offers vital information on the node's state of operation. Thereby, especially the deviation between several consecutive measurements and the rate of change are essential characteristics. We added a voltage divider consisting of two 10 kΩ resistors between the battery input voltage (before the DC/DC converter) and ground-level to measure the battery voltage. The midpoint of the voltage divider is connected to the MCU's ADC. As two equal resistor values are used, the highest voltage level of the midpoint equals

$$V_{ADC,max} = V_{BAT,max} \cdot \frac{R_2}{R_1 + R_2} = \frac{V_{BAT,max}}{2} = 2.75\,V \tag{7.4}$$

and, thus, stays below the maximum ADC input voltage of 3.3 V as long as the battery voltage does not exceed the maximum of 5.5 V. Due to the voltage divider ratio, the voltage level applied to the ADC is half the level of the battery voltage. Therefore, the corresponding battery voltage can be calculated with:

$$V_{BAT} = V_{ADC} \cdot \frac{2 \cdot V_{VS}}{ADC_{max}} \tag{7.5}$$

where V_{VS} is the supply voltage level (i.e., 3.3 V) and ADC_{max} is the maximum conversion result depending on the ADC's resolution (1023 in case of a 10-bit resolution). The voltage divider can be also be enabled/disabled via an N-channel MOSFET.

We defined the battery voltage monitor fault indicator χ_{BAT} to be the standard deviation of N consecutive measurements[7] of the battery voltage as:

$$\chi_{BAT} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(V_{BAT,i} - \mu_{BAT})^2} \tag{7.6}$$

where μ_{BAT} is the mean value of the measurements calculated as:

$$\mu_{BAT} = \frac{1}{N}\sum_{i=1}^{N} V_{BAT,i}\ . \tag{7.7}$$

A higher value of χ_{BAT} represents high deviations between consecutive measurements and, therefore, indicates possibly erroneous circumstances.

An additional voltage divider to measure the battery voltage is used for the battery voltage monitor that can, however, be added to almost every sensor node. Therefore, this indicator counts as an artificial generic indicator.

[7] For the implementation of indicators involving the calculation of the standard deviation of consecutive values, we suggest using Welford's online algorithm [381] as presented in [382, p. 232].

7.1.2.4 Active Runtime Monitor

The active runtime fault indicator monitors the length of the period the sensor node is active. The active phase follows pre-defined sequential processing of specific tasks and should be of constant length in every iteration. Significant deviations in the length of the active phase can indicate possibly erroneous circumstances.

In the current version of the ASN(x), the active runtime monitor indicator χ_{ART} is realized using the 16-bit timer1 peripheral of the MCU. The timer is started as soon as the node wakes up and stopped shortly before entering power-down mode. The counter value after stopping the timer is directly proportional to the length of the active phase. In our implementation, we configured the timer module to run with a prescaler of 1024, resulting in a tick length of 256 μs for a clock frequency of 4 MHz. The time spent in the active phase equals the counter value multiplied by the length of a tick. Therefore, the measurable time interval of the 16-bit timer is [256 μs, 16.78 s]. If other time intervals (e.g., shorter or longer) are needed, the timer's prescaler needs to be adjusted.

As we expect the period of the active phase to be of more or less constant length, we define χ_{ART} as the standard deviation of N consecutive measurements (measured in milliseconds). Thereby, we consider the magnitude of the difference rather than the absolute values, hence, we calculate χ_{ART} as the common logarithm of the standard deviation with:

$$\chi_{ART} = \log_{10}\left(\sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_{active,i} - \mu_{ART})^2}\right) \tag{7.8}$$

where $t_{active,i}$ is the length of the i-th measurement and μ_{ART} is the mean value of the measurements calculated as:

$$\mu_{ART} = \frac{1}{N}\sum_{i=1}^{N} t_{active,i} \ . \tag{7.9}$$

To avoid negative values of χ_{ART}, the logarithm is only calculated in case the standard deviation is higher than one. In case the standard deviation is smaller or equal to one, μ_{ART} is defined to be zero as the difference is negligibly small. Again, a higher value refers to a higher probability of abnormal circumstances possibly caused by faults. In our implementation, we used five consecutive values ($N = 5$) for the evaluation of χ_{AT}.

As only on-chip resources of the MCU are used, χ_{ART} refers to an inherent component-specific indicator. It could be argued that it is an inherent common indicator as almost all MCUs have timer modules; however, it still depends on the MCU and, thus, is component-specific.

7.1.2.5 Reset Monitor

A node reset is usually triggered by the hardware or software when a proper operation can not be continued anymore (such as a watchdog reset). Therefore, a node reset is a

clear sign of an unsafe operational condition often originating from faults. While the node may continue its proper operation after a reset, the probability of faulty circumstances is higher after a reset, especially if several resets happen during a short period. Additionally, the reason for the reset is relevant in deciding how probable faulty conditions are.

As a consequence, we implemented a reset monitor indicator χ_{RST} that is based on the number of resets happening in a certain timespan and the sources of the resets (e.g., the MCU module causing the reset). Thereby we leverage the 8-bit MCUSR available on most AVR MCUs. It provides information on which source caused the latest reset. The available sources indicated by corresponding flags in the MCUSR are:

- bit 0: power-on reset,
- bit 1: external reset (via the reset pin),
- bit 2: brown-out reset (in case the brown-out detection is enabled), and
- bit 3: watchdog reset.

We defined that the probability of faults is higher after a watchdog reset than after a power-on reset. Correspondingly, we use the bit position of the flags to weight the reset sources, where a higher weight refers to a higher probability of impaired operation. The ATmega1284P also has a flag for resets caused by the Joint Test Action Group (JTAG) interface (bit 4), but as we do not use JTAG we ignored it. Bits 5 to 7 are not used and always read as zero. However, the MCUSR needs to be cleared manually to detect whether new resets have happened since it was last read.

Aside from the reset source, the amount of resets during a specific period is also considered. For this reason, we implemented χ_{RST} as a function based on its previous value, the current value of the MCUSR, and a time-dependent exponential decay with a pre-defined rate. In each iteration (active phase), the value of χ_{RST} is updated using the following equation:

$$\chi_{RST,n} = \chi_{RST,n-1} \cdot \lambda_{RST} + \text{MCUSR} \tag{7.10}$$

where λ_{RST} is the inverse decay rate and the initial value $\chi_{RST,0} = 0$. In our implementation, we defined λ_{RST} as 0.92, resulting in a decay of 8 % of the previous value per iteration. However, to keep the value of χ_{RST} during a reset, its value is written to the EEPROM. Since the EEPROM has a limited number of write cycles, the value is only updated in the EEPROM if the current value differs from the previous one.

The χ_{RST} is categorized as an inherent component-specific indicator as it requires the availability of EEPROM and reset-related information (such as the MCUSR in case of AVR MCUs), but does not need additional hardware or circuitry.

7.1.2.6 Software Incident Counter

To monitor the stability of the software execution, we defined an incident counter indicator χ_{IC} that collects information on how many software function calls failed during a specific period. This counter, however, relies on the availability of corresponding function return

values expressing whether the function call was successful or failed. For this reason, we implemented the majority of our software modules in a way that they provide such information; that is, if the processing was successful, a *pass* is returned. In case of problems (e.g., function timeouts, incorrect responses, failed value checks) the functions return *fail*.

Based on these return values, we defined χ_{IC} as

$$\chi_{IC,n} = \begin{cases} \chi_{IC,n-1} + 1, & \text{if a function returned with } \textit{fail} \\ \chi_{IC,n-1} - 1, & \text{if } \textit{pass} \text{ AND } \chi_{IC,n-1} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{7.11}$$

where the initial value is $\chi_{IC,0} = 0$. Thereby, a high counter value indicates a rather unsafe state of operation in which faults are more likely to occur. However, if χ_{IC} exceeds a predefined threshold, the sensor node is reset.

As the incident counter indicator χ_{IC} does not require any additional hardware and has no dependency on the components used, it belongs to the inherent common indicators that can be added to any sensor node.

7.1.2.7 ADC Self-Check

Our previous research [4] revealed that especially the MCU's ADC and USART modules are susceptible to faults caused by fluctuations in the supply voltage and/or temperature. These modules, however, are essential for the correct operation of the sensor node, and thus, checking their proper function during runtime is crucial.

To check the ADC module's operation, we added a voltage divider with a fixed ratio to the ASN(x) design. The voltage divider is placed between the supply rail and the ground potential. Its midpoint is connected to the MCU's ADC. As the ratio of the voltage divider is fixed, the ADC's conversion result of the respective channel should be stable (aside from minimal conversion-related fluctuations).

In the current version of the ASN(x), two 10 kΩ resistors are used. Again, the voltage divider can be enabled/disabled via an N-channel MOSFET to save energy. The voltage level at the ADC input depends on the resistor values ($R_1 = R_2 = 10\,k\Omega$) and equals:

$$V_{ADC,CH0} = V_{VS} \cdot \frac{R_2}{R_1 + R_2} = \frac{V_{VS}}{2} \ . \tag{7.12}$$

The expected conversion result of the ADC (10-bit resolution and the analog reference voltage V_{AREF} connected to V_{VS}) equals:

$$ADC_{expected} = \frac{ADC_{max}}{V_{AREF}} \cdot V_{ADC} = \frac{2^{10} - 1}{V_{VS}} \cdot \frac{V_{VS}}{2} \approx 511 \ . \tag{7.13}$$

The ADC self-check indicator χ_{ADC} is now defined as the deviation of the actual conversion result from the expected value and is calculated with:

$$\chi_{ADC} = |ADC - ADC_{expected}| \tag{7.14}$$

where larger values indicate faulty circumstances. Since this indicator requires an additional voltage divider that can, however, be added to any sensor node with an ADC, it is an artificial generic indicator.

7.1.2.8 USART Self-Check

Similarly to the ADC self-check, we also implemented a way to evaluate the USART's operation for correctness. Thereby, we leverage the availability of two USART modules in the ATmega1284P, named USART0 and USART1. USART0 is used to communicate with the XBee radio module, and USART1 can be used for debugging during the development phase. After deployment, the USART1 can be used to monitor the USART0 interface by employing a loopback test. To do so, the ASN(x) has two open solder jumpers that can be bridged to form a loop. After that, every time data is sent via USART0, the same data should arrive at USART1. Consequently, the USART self-check indicator χ_{USART} is defined as:

$$\chi_{USART} = \sum_{i=1}^{||D_{TX}||} \Delta_{TXRX,i} \tag{7.15}$$

with

$$\Delta_{TXRX,i} = \begin{cases} 1, & \text{if } D_{RX,i} \neq D_{TX,i} \\ 0, & \text{otherwise} \end{cases} \tag{7.16}$$

where D_{RX} refers to the array of bytes received by USART1 and D_{TX} refers to the array of bytes transmitted by USART0, both regarding the data of one message transmission. Thus, it expresses the number of bytes that have not been correctly received by the loopback interface (i.e., USART1).

The implementation of χ_{USART} requires the availability of two USART interfaces (component-specific) and an external connection between both (loopback; additionally added). Therefore, χ_{USART} counts as an artificial component-specific indicator.

7.1.2.9 Indicator Normalization

The fault indicators presented above have different value ranges that hamper an automated analysis. For this reason, we applied feature scaling to the fault indicators to use them in our dDCA-based fault detection system as described in Section 6.3. In addition, the resulting real numbers are limited to a maximum value of one to prevent an overreaction of the indicators. Consequently, the scaled fault indicators are calculated as:

$$\chi_i' = \min\left(\frac{\chi_i}{\chi_{i,max}}, 1\right) \tag{7.17}$$

where χ_i are the original fault indicators presented above, $\chi_{i,max}$ are the divisors of the scale factors, and χ_i' are the resulting normalized fault indicators ($\chi_i' \in \mathbb{R} \mid 0 \leq \chi_i' \leq 1$). Table 7.2 provides a list of the fault indicators and scale factors used in our approach. These scale factors are derived from prior empirical analyses using experiments and

simulations. In this context, we combined the insights from observations of our practical testbeds with the results of directed lab experiments to find meaningful thresholds for the value normalization.

Table 7.2: Scale factor divisors for normalized fault indicators

Indicator	Scale factor divisor
Node temperature monitor (χ_{NT})	$\chi_{NT,max} = 5\,°\mathrm{C}$
Supply voltage monitor (χ_{VS})	$\chi_{VS,max} = 1\,\mathrm{V}$
Battery voltage monitor (χ_{BAT})	$\chi_{BAT,max} = 1\,\mathrm{V}$
Active runtime monitor (χ_{ART})	$\chi_{ART,max} = 5$
Reset monitor (χ_{RST})	$\chi_{RST,max} = 25$
Software incident counter (χ_{IC})	$\chi_{IC,max} = 10$
ADC self-check (χ_{ADC})	$\chi_{ADC,max} = 25$
USART self-check (χ_{USART})	$\chi_{UART,max} = 5$

As an example, we performed several experiments where we controlled the ambient temperature (maximum values and rate of change) and monitored the temperature readings used for χ_{NT}. We found that the particular sensors reacted differently fast to the temperature changes. However, these differences mostly stayed within a specific range. Consequently, a temperature threshold of 5 °C for the normalization of χ_{NT} was found to (i) not cause an overreaction in case of normal operation and (ii) provides a sufficiently large indication in case of abnormal conditions (i.e., operation). Analogously, also the voltage and time thresholds for χ_{VS}, χ_{BAT}, and χ_{ART} have been derived.

The scale factor divisors of χ_{RST}, χ_{IC}, and χ_{USART} are based on abstract simulations of the expected indicator behavior and their reaction to specific situations in combination with a usual update interval in minute granularity. When defining the divisors for normalization, it has to be minded that the reset monitor (χ_{RST}) is significantly influenced by the application's update interval, and the software incident counter should be chosen in relation to the complexity of the node software (i.e., number of function calls).

For χ_{ADC}, the value of the scale factor divisor depends on the quality of the used ADC. The more stable the ADC performs, the lower the threshold can be set. Thereby, the accuracy is commonly expressed as the number of least significant bit (LSB) affected by internal noise and inaccuracies. In our case, the ADC of the ATmega1284P has an absolute accuracy of ±2 LSB [383, ch. 21]. Therefore, the ADC's conversion result can have a fluctuation of up to 1.5% in regular operation. We added a safety margin on top and defined the threshold as 5% of the ADC's maximum value. This threshold was then evaluated and confirmed by experiments in our lab setup.

7.2 Prototype Implementation

We have implemented our entire fault detection approach[8] according to the procedure depicted in Figure 6.1. Concerning the architecture of the WSN, we assume a clustered structure where every cluster consists of precisely one cluster head (CH) and a variable number of sensor nodes (SNs). Additionally, there is one dedicated sink node (SK) responsible for storing the reported sensor data and providing some basic data visualizations. The particular implementation of the approach on each of these participants is presented in the following in a top-down manner (i.e., SK, CH, and SNs).

7.2.1 Use Case

The use case for our implementation was a WSN used in a smart garden system. Such a system is used to monitor specific physical conditions related to plant growth; thus, it presents an environmental monitoring system (cf. Section 2.1.1). It aims to increase the plants' yield by optimizing the conditions that impact plant growth. Nevertheless, our work focuses on the sensory part of the smart garden application. Actuators such as irrigation and shadowing systems are not in the scope of our work.

Several physical quantities directly influence plant growth. The most important ones are (i) air temperature and humidity, (ii) soil temperature and moisture, (iii) light intensity, and (iv) nutrient content of the soil. In our implementation, we implemented sensory information on the first two points, namely air temperature and relative humidity as well as soil temperature and moisture level. Other quantities can easily be included by adding corresponding sensors to the sensor nodes.

However, the smart garden only acts as a use case for our fault detection approach. Therefore, we are interested in the system behavior rather than the information acquired. The sensor data are collected, but their analysis and knowledge gained from these are not considered in our work.

7.2.2 Sink Node

The sink node (SK) provides the central point for data storage in our network. Its tasks included hosting the database (DB), providing data visualization, and making the data accessible for possible further processing (e.g., data analysis). In our setup[9], we used a Raspberry Pi 3 Model B+.

The SK additionally provided Python 3 scripts to analyze the modified dDCA processing characteristics by running it on pre-recorded data either provided by the DB (see Section 7.3) or by the lab experiments (refer to Section 7.4).

[8] The resources are publicly available as summarized at https://dowid-wsn.github.io/ftdca_landing/
[9] Detailed information on the setup is available at https://github.com/DoWiD-wsn/RPi_sink_node.

7.2.3 Cluster Head

The cluster head (CH) acted as the gateway between the sensor nodes in its cluster and the rest of the WSN. As illustrated in Figure 7.2, its main tasks are receiving the messages from the SNs and storing the data in a DB. Additionally, the CH performed the modified dDCA to classify the received data as described in Section 6.4 (depicted as dark boxes in Figure 7.2). Consequently, it received the data from the sensor nodes via Zigbee, assessed the data, and forwarded the use case-related data in combination with the class label to the SK via WiFi. Our CH was realized with a Raspberry Pi 3 Model B+ extended with a Digi XBee 3 radio.

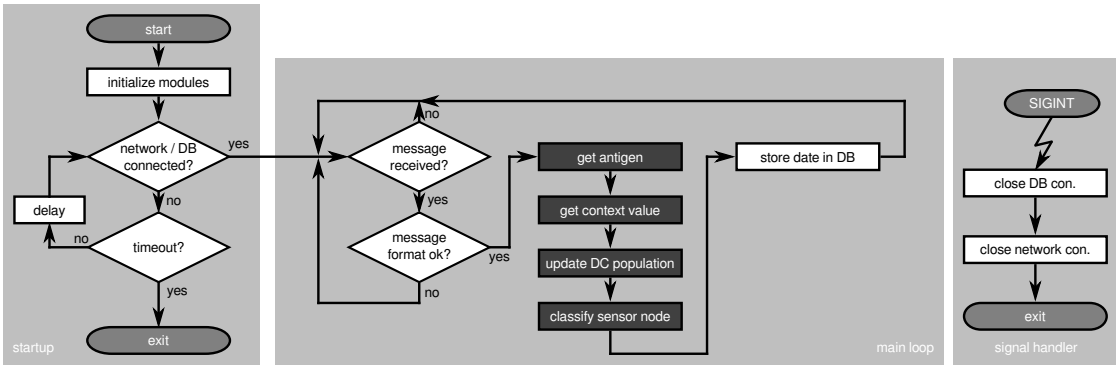


Figure 7.2: Simplified cluster head software flowchart

We implemented the CH's functionality as a Python 3 script[10] which the cluster head continuously executed. The script is executed until it is terminated by a process signal (i.e., keyboard interrupt). Concerning our modified dDCA's parameters, we found that $M = 3$, $N = 10$ and $q_{\text{sen}} = 0.1$ attained the best results in previous experiments. The dDCA-based assessment is run centrally on the cluster head for each cluster, where only the indicators of each particular SN are considered. In a future version, the safe values of the SN's neighbors could be incorporated to improve the sensor node fault detection.

7.2.4 Sensor Nodes

Our SNs are based on the ASN(x) platform which we initially presented in [8] (see also Section 7.1). All sensor nodes were programmed in C-language on the bare metal (without an OS) following the procedure shown in Figure 7.3. Again, the dark boxes show the fault detection-specific parts of the software. Each sensor node measured the environmental conditions (use case data) and performed the node-level diagnostics (see Section 6.2) based on the fault indicators presented in Section 7.1.2.

If the sensor node software runs into an erroneous situation not resolvable by the software, the sensor node is reset. Since the AVR MCU does not offer a software reset, we exploited the watchdog timer (WDT) for this purpose. In case the node shall be reset, the WDT

[10] For more information, we refer to https://github.com/DoWiD-wsn/RPi_cluster_head/tree/dca.

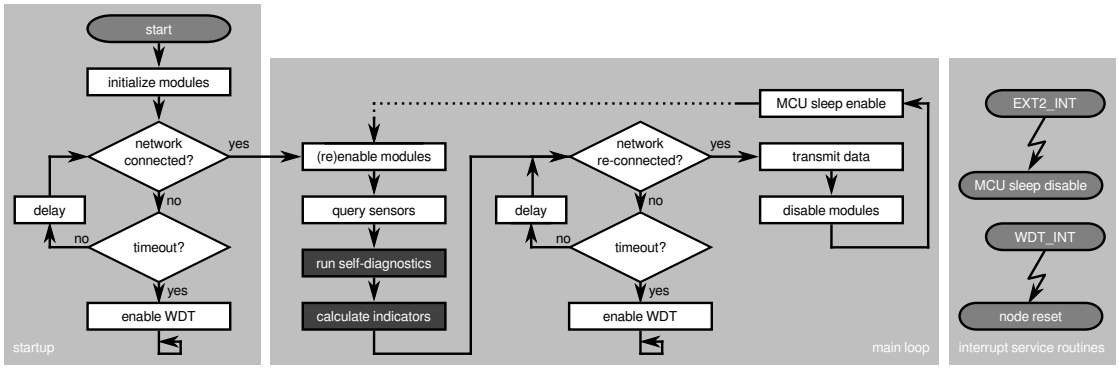


Figure 7.3: Simplified sensor node software flowchart

is started and the software waits in an endless loop for the WDT to trigger and reset the MCU (depicted as "WDT_INT" in Figure 7.3).

Every SN forwarded its data in a defined update interval to the CH via Zigbee and remained power-saving the rest of the time. For the sleep schedule, the onboard RTC is used which triggers the MCU's external interrupt periodically with a defined interval (see "EXT2_INT" in Figure 7.3). All values were converted and stored as fixed-point numbers with six fractional bits that provide sufficient granularity and resolution for the respective value ranges to minimize the memory overhead. The use case-related data were stored as 16-bit and the indicators as 8-bit variables, thus, having ten and two integer bits, respectively. These data and a local timestamp (incremental message number) were then transmitted to the cluster head via Zigbee. The local timestamp is used as a simple means to detect a loss of messages on the CH. Figure 7.4 shows the resulting message format, including the size of the respective data fields in bits. In the figure, T_{air} is the air temperature, H_{air} is the relative air humidity, T_{soil} is the soil temperature, and H_{soil} is the soil moisture level, respectively.

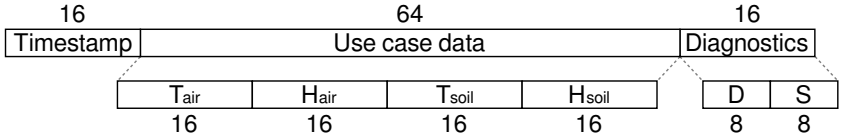


Figure 7.4: Sensor node message format

7.3 Simulations

We ran a set of simulations to assess the detectability of faults when using our fault detection approach. These simulations used pre-recorded, fault-free datasets (in the following referred to as "base datasets") in which faults were randomly injected. We used three different base datasets, each containing data for a period of seven days (see Section 7.3.1). In addition, we utilized fault signatures identified in our works in [5, 8] as

presented in Section 7.3.2. As described in Section 7.3.3, we used a script to inject these fault signatures into the base datasets randomly. The resulting "faulty" data were then fed to our approach and the results were assessed.

7.3.1 Base Datasets

The base datasets form a basis for our simulations by providing data from fault-free operation (i.e., baseline data). We used three base datasets with different characteristics regarding the behavior of the sensed phenomena to have a broad basis for assessing our approach's fault detection capabilities. In the following visualizations of our datasets, the measured use-case data combined with the monitored node-level diagnostics and the resulting danger and safe indicator values are shown. The latter were calculated according to the scheme discussed in Section 6.3. Concerning the use case date, only the third dataset contains all sensor data discussed in Section 7.2.4; the first two sets neglect the soil-specific measurements (i.e., the respective data fields contain only zeros).

The first base dataset was recorded from a sensor node in a living space (i.e., indoor deployment) between 2021-07-04 06:00 and 2021-07-11 06:00. It contains data from 7 days with an update interval of 10 minutes. As visualized in Figure 7.5, a normal diurnal pattern with minor magnitude is visible in the sensor data. However, no unexpected or otherwise noticeable occurrences happened during this time, thus, providing a dataset of a stable environment.

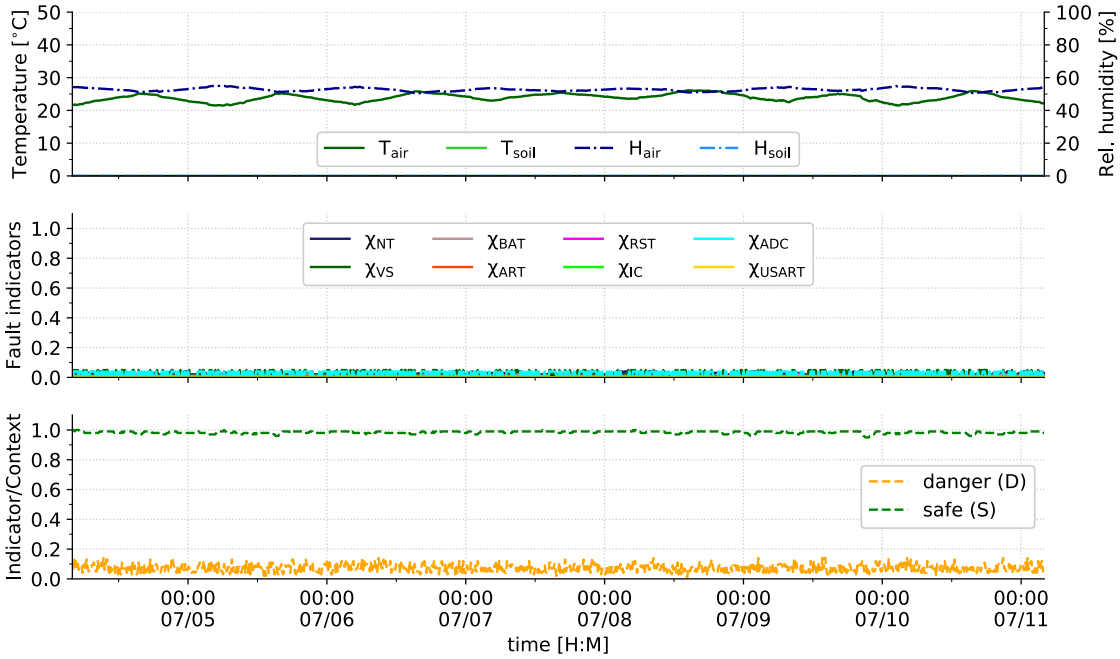


Figure 7.5: Base dataset with a stable indoor environment

Similarly, the second base dataset depicted in Figure 7.6 was gathered from a sensor

node in an indoor deployment, but this time in an office between 2021-11-22 00:00 and 2021-11-29 00:00. Unlike the first base dataset, this dataset contains sensor data with an update interval of one minute (instead of the usual 10-minute interval) to also analyze the effect of the update interval on the detection process. During this week, the sensor node was operating in a mostly stable environment but with several events in the recorded data caused by an irregular opening of nearby windows (i.e., airing of the room). Although these events are small in their magnitude, they provide deviations from normal behavior that data-centric fault detection approaches could misclassify. This base dataset provides a good example of the benefit of combining all sensor readings into our safe indicator. As can be seen in the plot, the safe indicator reacts to significant deviations in both the temperature and the relative humidity as well as to smaller changes in single data streams. Consequently, the minor fluctuations in the relative air humidity measurements that happened on Nov. 24, 25, and 26 also caused a temporary decrease in the safe indicator value.

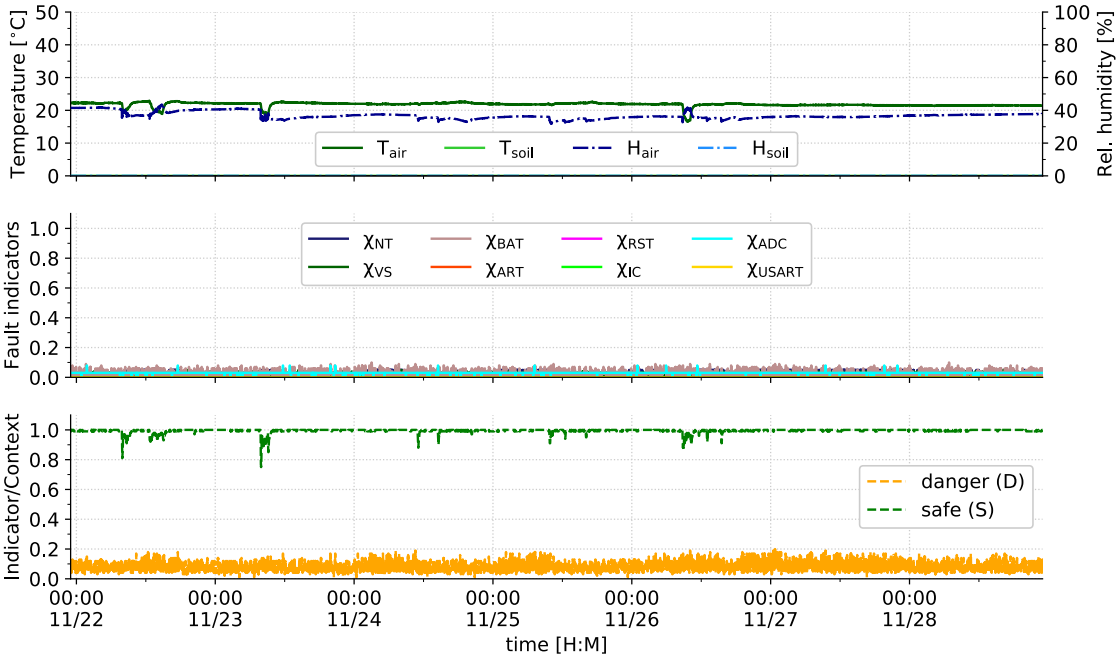


Figure 7.6: Base dataset with several events in an indoor environment

In contrast to the first two base datasets, the third one was recorded from a sensor node in an outdoor deployment during the late summer of 2021. In particular, it shows the measurements of a sensor node deployed in a raised bed located in Lower Austria. The data was recorded between 2021-08-12 04:00 and 2021-08-19 04:00 with a 10-minute update interval. On 2021-08-16 morning, heavy rain started to fall that lasted for several days. Consequently, the persistent rain caused substantial deviations in the usually diurnal pattern in the second half of the dataset, as visible in Figure 7.7.

Such substantial deviations are evident anomalies in the sensor data and lead to false

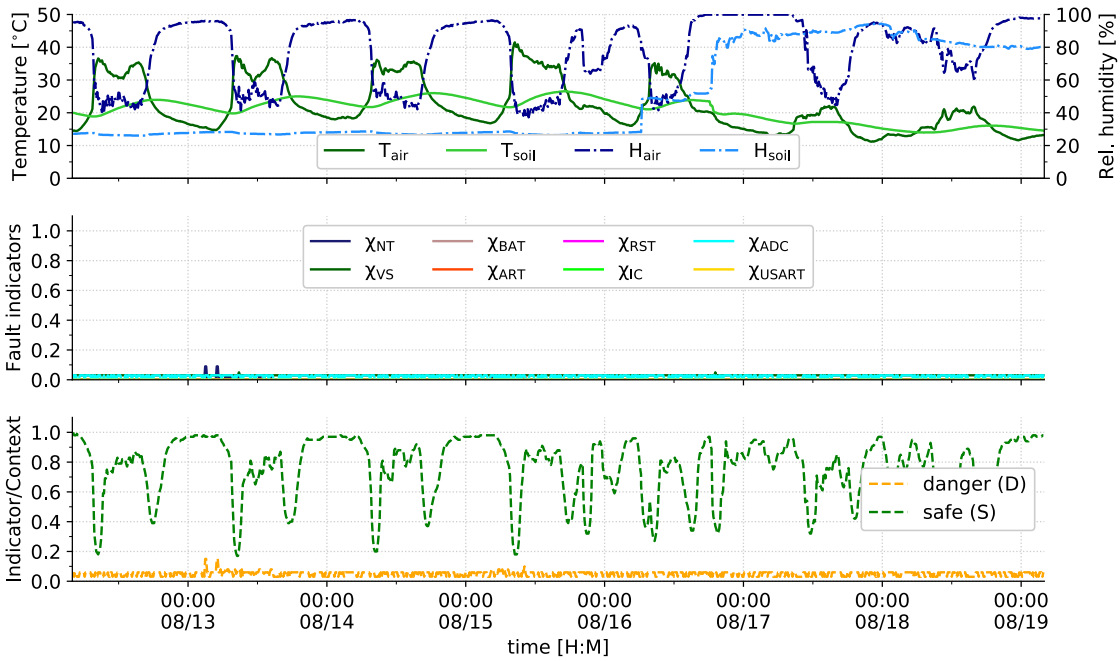


Figure 7.7: Base dataset with strong deviations in an outdoor environment

alarms of most related fault detection schemes, especially those focusing on the sensor data. A detailed discussion of the results regarding the distinction of data events from node faults provided by our novel fault detection approach can be found in Section 8.1.

7.3.2 Fault Signatures

Aside from the base datasets, our simulation exploits fault signatures identified in our works on the detectability of sensor node faults (i.e., fault indicators) in [5] and the use of this information for fault diagnosis presented in [8]. These fault signatures were extracted from actual fault actions observed on the sensor nodes during our experiments. The fault signatures contain two kinds of information: (i) the relative change on the sensor data and (ii) the reaction of the implemented fault indicators caused by the respective fault model. We identified five distinct fault signatures originating from different components of the sensor nodes. The visualizations of the fault signatures provided below show the relative impact of the fault model on the sensor measurements and the respective values of the eight implemented fault indicators.

The first fault signature illustrated in Figure 7.8 was caused by a bad electrical connection of the air temperature sensor and the sensor node (i.e., OWI-based sensor). It caused transmission problems in the serial communication that resulted in the failing of several functions in the sensor node software (indicated by χ_{IC}) and differences in the onboard temperature measurements (indicated by χ_{NT}) probably caused by partial short circuits in the sensor's supply rail. The manifestation of this fault in the sensor data was a

negative offset of the air temperature measurement.

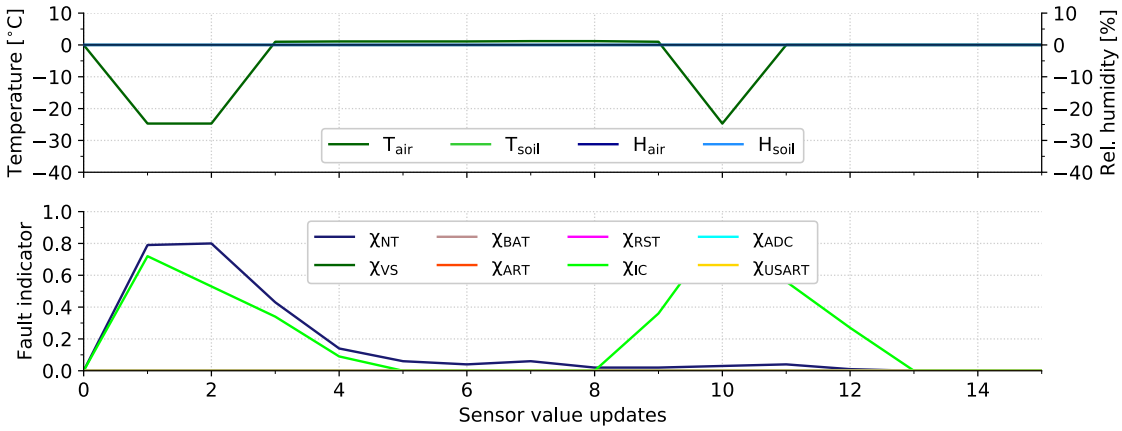


Figure 7.8: Fault signature of a node fault caused by bad sensor connection

As visible in Figure 7.9, the second fault signature influenced both the air temperature and relative humidity measurements. This fault was caused by water in the sensor node's housing that caused partial short circuits in the sensors' connection. The battery voltage monitor (χ_{BAT}) reacted to this node fault as the short circuits impaired the underlying voltage measurements.

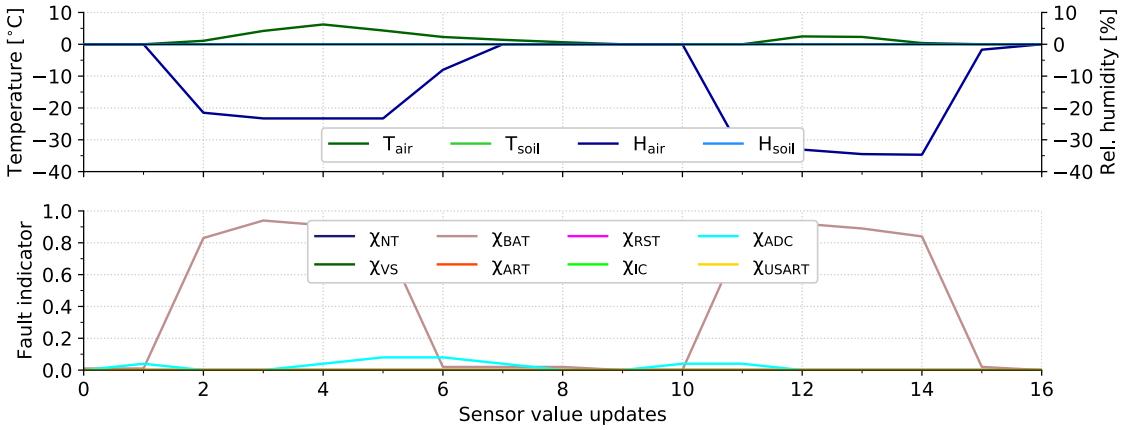


Figure 7.9: Fault signature of a sensor fault caused by humidity in sensor housing

Similarly, humidity caused partial and temporary short circuits in the supply of the air temperature sensor (see Figure 7.10). The short circuits occurred in the connection between the sensor and the sensor node. Consequently, it caused an impact on the sensor node observable by the fault indicators. In particular, the node temperature monitor (χ_{NT}) and the supply voltage monitor (χ_{VS}) indicated the presence of this fault.

Like in the third fault signature, also in the fourth fault signature, humidity causes partial short circuits, but this time in the supply of the soil-related sensors (i.e., T_{soil}

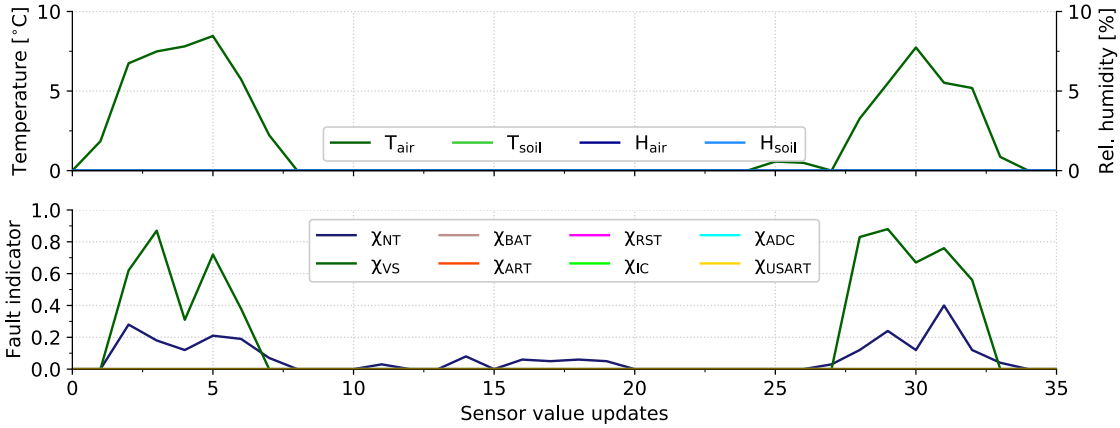


Figure 7.10: Fault signature of an air sensor short circuit caused by humidity

and H_{soil}). As shown in Figure 7.11, the fault caused deviations in the sensor readings and a corresponding reaction of several fault indicators.

Surprisingly, although most of the fault signatures captured are related to humidity, all of them have a distinct impact on the sensor node's operation. Consequently, different fault indicators (and combinations of these) reacted to each fault signature.

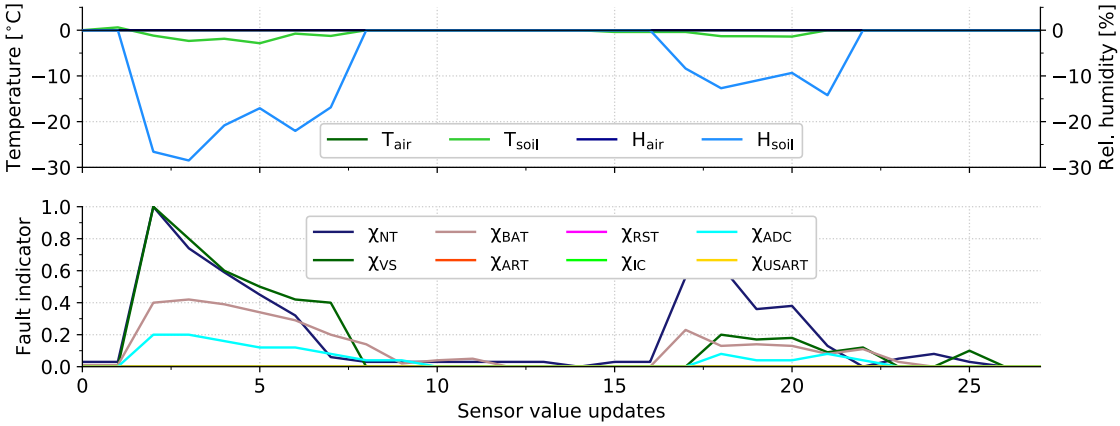


Figure 7.11: Fault signature of a soil sensor short circuit caused by humidity

In contrast to the other faults, the fifth fault signature presents a type of fault that does not cause a reaction of any of the fault indicators implemented. The fault shown in Figure 7.12 originated from humidity inside the soil temperature sensor that did not cause any observable symptoms outside the sensor. Consequently, also the fault indicators of the sensor node did not react. Nevertheless, the reported sensor measurements showed substantial deviations of more than 60 °C between two consecutive measurements.

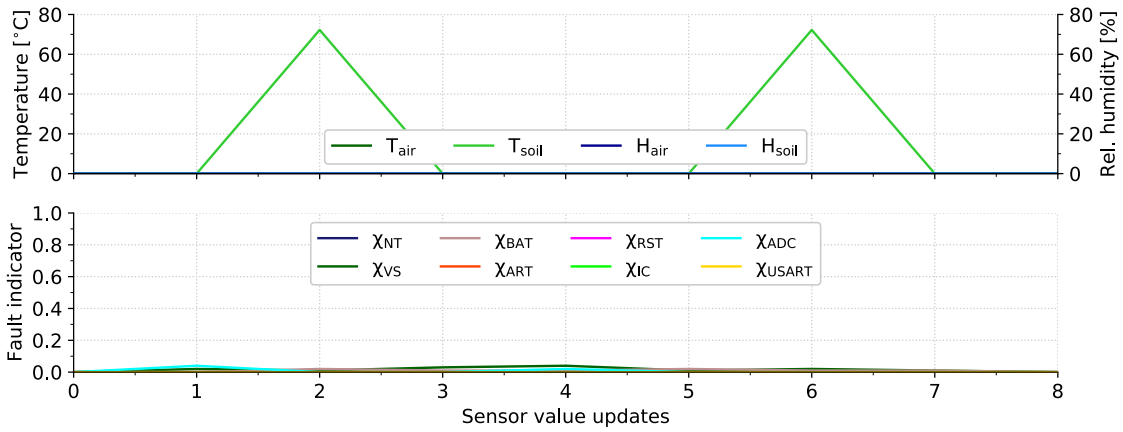


Figure 7.12: Fault signature of a sensor communication fault caused by humidity

7.3.3 Fault Injection

Using a Python 3 script[11], we randomly injected fault signatures into the base datasets. For this purpose, the script injected a random number of faults at random positions in the base dataset. Additionally, the injected faults are labeled accordingly. By running the fault injection script 20 times per base dataset, we acquired 60 datasets[12] with unique characteristics concerning the amount and type of faults injected.

We then executed our modified dDCA on these "faulty" datasets and stored the fault context derived by our approach (based on $\overline{\mathrm{Cx}_{Ag'}}$; see Section 6.4.4) along with the input data. Next, we compared these resulting context values with the fault labels stored during the fault injection to determine the number of TP, TN, FP, and FN. Based on these numbers, the approach's sensitivity (i.e., TPR), specificity (i.e., TNR), and accuracy (i.e., F-score) were calculated. The simulation results are discussed in Section 8.1.

7.3.4 Benchmark

Based on the simulation data described before (i.e., base datasets and fault models), we additionally ran benchmarks to compare the detection correctness of our proposed approach with alternative methods. We applied three state-of-the-art streaming anomaly detection (SAD) methods on the 60 "faulty" datasets described in Section 7.3.3. In particular, we used (i) the Exact-Storm method [384], (ii) the isolation forest algorithm for streaming data (IForestASD) [385], and (iii) the robust random cut forest (RRCF) method [386]. that are provided by the *PySAD* package (cf. [387]).

All of these methods support the detection of anomalies in streams of multivariate data. To do so, they successively build and update their internal model with newly arriving data instances. Then, an anomaly score is calculated for the given data instance. If this

[11] For details, see https://github.com/DoWiD-wsn/asnx_analyses/blob/master/dca/README.md.
[12] Located at https://github.com/DoWiD-wsn/asnx_analyses/tree/dca-central-analysis/dca/results/.

score exceeds a defined threshold, the data instance is labeled as anomalous; otherwise, it is considered correct. We used a threshold of 90%, stating that the probability of the data instance being normal is less than 10%.

Additionally, we implemented an outlier detection based on the running standard deviation of the sensor values within a pre-defined window. In this approach, a data instance is considered anomalous (or faulty) if the absolute value of the resulting standard deviation is greater than three (i.e., 3σ-based detection).

By performing fault detection using these alternative methods on the same datasets used for the correctness evaluation of our approach, we provide an objective base for the effectiveness assessment of our approach. Moreover, we claim that effective sensor node fault detection needs to include node-level information, considering only the sensor data is insufficient. To support our claim, we ran a second set of simulations of the SAD methods where the node-level diagnostics (i.e., danger and safe indicators values) were added to the multivariate input data instances. The results of our benchmark with alternative approaches are discussed in Section 8.1.3.

Nevertheless, the provided benchmark serves as a basis for comparison but is not sufficient to generically demonstrate the superiority of the proposed approach. To support such a claim, a comparison with more sophisticated anomaly detection solutions and more fault models would be necessary.

7.4 Lab Experiments

We used a lab experiment setup to further investigate the effects of the supply voltage and ambient temperature (separate and in combination) on the sensor node's operation. Therefore, our lab experiments focused on the two main fault models described in Section 6.1 (i.e., ambient temperature and supply voltage faults). For this purpose, we created a lab setup based on our embedded testbench (ETB) (cf. [8]) that provides a controllable environment for the sensor nodes (see Section 7.4.1). It offers an adjustable ambient temperature between 25 and 70 °C. Concerning the supply voltage, several supply channels have been used to either supply the sensor node via the battery voltage input with a voltage between 0 and 5.5 V, or to directly supply the node bypassing the voltage regulator (i.e., DC/DC converter) where voltages between 0 and 3.3 V were applied. We used Python scripts to control the environmental conditions and automate the test procedures, thus, making our experiments repeatable and reproducible (see Section 7.4.2).

With the lab experiments, we analyzed the behavior of an actual sensor node and, respectively, our approach to external effects (i.e., environmental events) and internal disturbances such as (partial) short circuits that could be caused by humidity inside the node's housing. Therefore, we performed a kind of physical fault injection by forcing the sensor node into conditions that endanger its proper functioning. We then manually analyzed the results to determine how the approach reacts to proper environmental events in contrast to situations resulting in node faults (i.e., fault-induced deviations).

With our directed lab experiments, we extended the correctness analysis of our simulations by evaluating specific corner cases on an actual sensor node. Additionally, we analyzed the resource requirements to assess the approach's efficiency. The acquired results and findings are discussed in Chapter 8.

7.4.1 Embedded Testbench

The embedded testbench (ETB)[13] is a platform to enable the testing, analyzing, and profiling of embedded systems focused on low-power devices such as sensor nodes. We originally developed the ETB to facilitate the experiments presented in [4]. It was then continuously extended in our succeeding works in [5–8]. It encompasses a Raspberry Pi hardware add-on and Python libraries facilitating the experiment creation and execution. The primary lab experiment setup is shown in Figure 7.13. It consists of one dedicated sensor node (based on the ASN(x)) and the ETB acting as an experiment controller. In this setup, the sensor node is equipped with a DS18B20 temperature sensor (used for soil temperature measurements) as well as an AM2302 and an SHTC3 for air temperature and relative humidity measurements. The measurement of the soil moisture level was neglected in this setup. As visible in Figure 7.13, all sensors are duplicated with one set connected to the sensor node under test and the second connected to the ETB for reference measurements. Using the reference measurements, we can identify corrupted sensor data due to node-level effects.

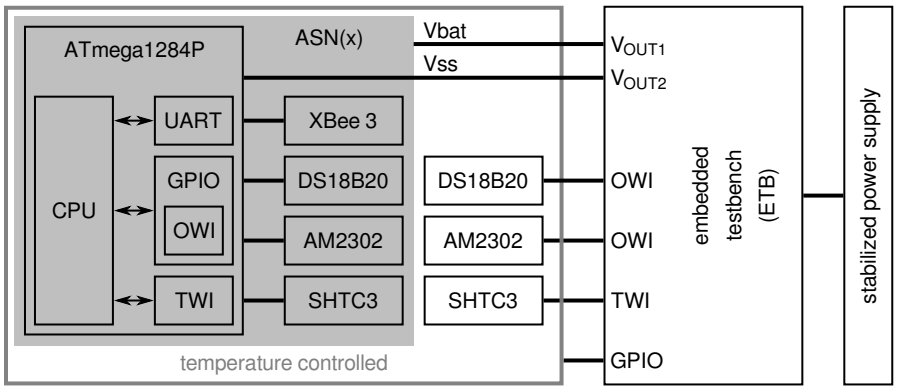


Figure 7.13: Lab experiment setup utilizing the embedded testbench

As shown in Figure 7.14, it offers a voltage scaling module (VSM) consisting of four independent power outputs, each equipped with a wattmeter. Each power output consists of a MIC24045 buck converter with a programmable output voltage between 0.64 V and 5.25 V. Using the VSM, we can precisely adjust the ASN(x)'s supply voltage to mimic the effects of a depleting battery or other effects such as temporary voltage fluctuations (e.g., caused by short circuits). Additionally, the ETB is equipped with two auxiliary wattmeters, a four-channel 16-bit ADC, and connectors for various communication

[13] Information on the ETB are available at https://github.com/DoWiD-wsn/embedded_testbench.

interfaces (i.e., USART, SPI, OWI, and I2C). Four dedicated test control signals are available for low-level experiment control and data exchange with the sensor node. These signals and the USART interface have MOSFET-based bi-directional level shifters [388] to prevent effects caused by different voltages of the logic levels.

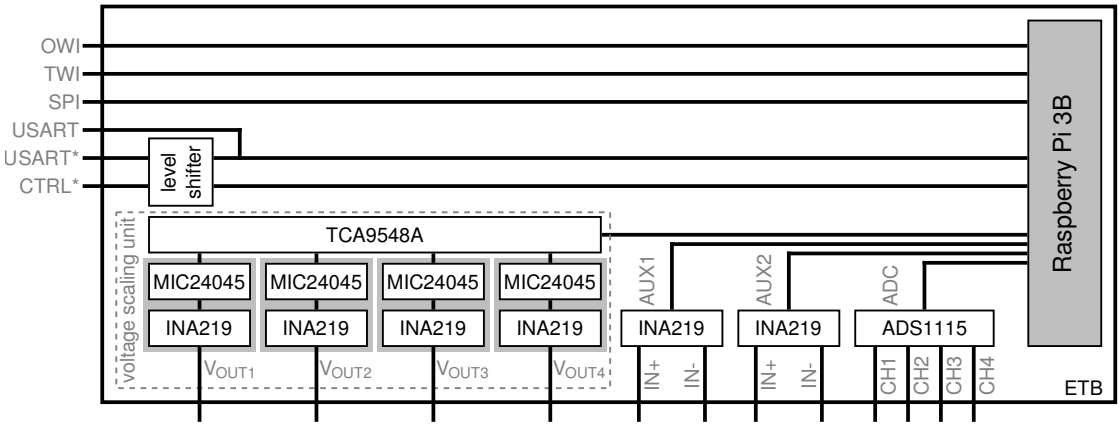


Figure 7.14: Basic components of the embedded testbench

In our lab experiment setup, we can also vary the ambient temperature using a 100 W infrared lamp in combination with a modified hair dryer (both controlled via a relay card connected to GPIOs of the ETB). Due to the adjustable environmental parameters, the lab experiment setup allowed us to analyze the ASN(x)' behavior during an impaired operation in a controlled environment. However, as the ETB controls the sensor node supply voltage and ambient temperature, our experiments can be automated using Python scripts and, therefore, the experiments are reproducible to better distinguish between sporadic and recurring effects.

The sensor node was configured to send updates every minute to have data of fine granularity. Additionally, the ETB kept track of the node's supply voltage and current consumption as well as its reference measurements. Concerning the current measurements, we found that the measurement resolution provided by the ETB of 0.1 mA was too coarse. The current drawn by the ASN(x) is comparably small, especially when the node is in a power-saving state (i.e., sleeping). For this reason, we augmented the power consumption measurements with a Joulescope[14] connected between the ETB and the ASN(x). The Joulescope was configured to monitor the power consumption and the respective electrical charge consumed during a defined period with a current resolution of 1.5 nA and a voltage resolution of 0.4 mV, both measured with 2 MS/s and a reduction frequency of 2 Hz.

7.4.2 Test Automation

The ETB was primarily used to analyze specific corner cases conditioned by the combination of environmental factors (i.e., ambient temperature and supply voltage). The

[14]For information on the Joulescope, see https://www.joulescope.com/ (last accessed on 2022-08-23).

experiments were repeated several times in different settings to distinguish between random effects and sporadic external influences. Consequently, developing an experiment setup that facilitated automated testing was necessary.

For this purpose, we created the ETB that not only provides the hardware platform and interfaces for embedded system testing but also offers a rich and easily accessible set of functionalities to support the development and execution of various experiments. While the hardware modules could have been controlled by any supported programming or scripting language, we decided to use Python (specifically Python 3) due to its widespread use and the availability of supportive libraries.

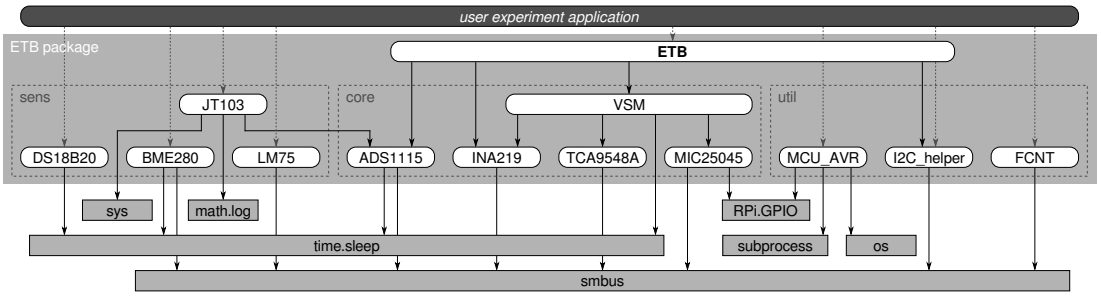


Figure 7.15: Software library structure of the embedded testbench

As summarized in Figure 7.15, the ETB provides functionality for the onboard modules such as the VSM, the power monitors, and the ADC. Several sensors accessible via different interfaces are also supported by corresponding libraries contained in the ETB package. Additionally, auxiliary functionality is included, for example, to support the flashing of firmware on the MCU (provided by the *MCU_AVR* library), the detection of available I2C devices (via the *I2C_helper* library), or the use of our AVR-based frequency counter[15] (included in the frequency counter library, or short *FCNT*).

7.5 WSN Testbed

Concerning a practical evaluation of our approach, the lab setup allows us to specifically analyze the effects of the voltage and temperature on the node's operation. However, the sources of faults occurring on sensor nodes are of a much more comprehensive range and include temporal effects as well as network-related influences. For this reason, we additionally deployed a WSN testbed to give our experiments a broader scope. This testbed consisted of ten SNs deployed both indoor and outdoor. The WSN testbed additionally evaluated the system behavior of our approach, including several nodes and the CH. As presented in Section 7.2.1, our practical experiments were performed in a smart garden setting where four environmental parameters related to plant growth were monitored, namely ambient air temperature and relative humidity as well as soil

[15]The frequency counter module is available at https://github.com/DoWiD-wsn/frequency_counter.

temperature and moisture level. In the following, the setup of the WSN testbed is presented. For a discussion of the results and findings, see Chapter 8.

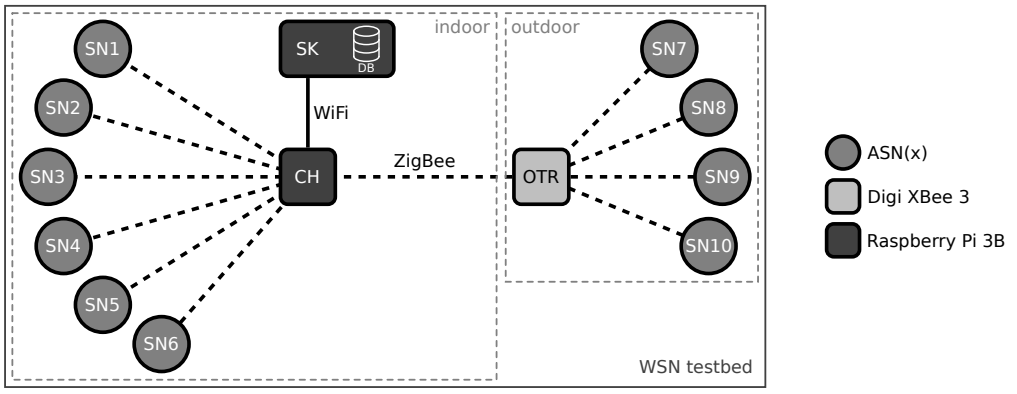


Figure 7.16: Architecture of the indoor and outdoor WSN testbed

The architecture of the WSN is illustrated in Figure 7.16. It shows the indoor deployment consisting of six SNs and the outdoor deployment consisting of four SNs. In both settings, the SNs were equipped with an XBee 3 radio configured to transmit at the lowest power level (i.e., at -5 dBm) to decrease the overall power consumption of the nodes. To ensure a reliable Zigbee network connection of the SNs placed outdoors, we additionally deployed an outdoor relay node (OTR) for increased network connectivity. This OTR consisted of an XBee 3 module operated standalone in a network router configuration and was supplied by a wired power supply. The CH and SK deployed were implemented as described in Section 7.2. In contrast to the SNs, the XBee radios of the OTR and CH use the highest power level available, which is +8 dBm.

7.5.1 Indoor

Our indoor deployment consisted of six nodes (denoted as *SN1* to *SN6* in Figure 7.16) that were placed on top of plant pots. The sensor nodes were equipped with DS18B20 temperature sensors to measure the soil temperature and sensors to measure the soil's moisture level (i.e., Adafruit STEMMA capacitive soil sensor). Additionally, half of the nodes were equipped with AM2302 and the other half with SHTC3 digital temperature and relative humidity sensors. The sensor nodes were first deployed in the living area of a residential building, where they ran for six months with an update interval of 10 minutes. Then the indoor deployment was moved to an office room and reconfigured to send updates every 5 minutes (running for ten months).

With the indoor deployment, we analyzed the behavior of the ASN(x) including their fault indicators during a regular operation in a mostly controlled environment. No extreme environmental disturbances such as high temperatures or heavy rain compromised the nodes' operation in this environment. However, there were some irregularities in the temperature and the humidity readings caused by the opening of nearby windows.

Such events are a deviation from an expected (modeled) behavior of the environmental conditions and could, depending on their scale, be misinterpreted as faults by data-centric detection approaches. Still, the data acquired from the indoor deployment provide reference measurements; thus, how the sensor nodes behave in a stable environment.

7.5.2 Outdoor

Especially the harsh conditions posed by the environment of outdoor deployments have been shown to significantly impact the behavior of sensor nodes and, therefore, the probability of node faults. For this reason, we deployed four sensor nodes (named *SN7* to *SN10* in Figure 7.16) in different locations of raised beds planted with different crops. The raised bed was placed on a south-facing balcony located in Lower Austria. In contrast to the sensor nodes deployed indoors, where some had a SHTC3 sensor connected, all four outdoor nodes were equipped with AM2302 sensors. The outdoor testbed was active during the summer of 2021, where several weather extremes such as sudden heavy rain, strong winds, and significant temperature fluctuations occurred, which posed perfect conditions for our evaluation.

In contrast to the indoor deployment, the outdoor installation provided us with data from sensor nodes in regular operation but in an uncontrolled and harsh environment. Thereby, especially direct sun radiation and heavy rainfalls posed challenging conditions for our ASN(x) where the latter also caused the leaking of water into the housing of some nodes resulting in partial short circuits. By comparing the data from the outdoor with the indoor deployment, we were able to identify differences in the node/sensor behavior caused by the environmental influences.

CHAPTER 8

Result Discussion

This chapter presents the results of our tripartite evaluation setup described before. We discuss the main findings of our approach's fault detection and operational characteristics. Additionally, we show that our approach indeed remedies the drawbacks of previous node fault detection schemes indicated by increased detection accuracy.

Two characteristics are of paramount importance for sensor nodes, namely: (i) reliability and (ii) energy efficiency. Only if both are satisfied the sensor nodes can provide high-quality data over a long time. Consequently, we evaluated the correctness and efficiency of our approach in various operational and environmental conditions.

The correctness refers to the approach's capability to detect faults properly. Standard correctness metrics include (i) the sensitivity, (ii) the specificity, and (iii) the accuracy of the approach. High correctness is crucial to ensure the quality of the data provided and, thus, the WSN's reliability.

On the other hand, efficiency primarily focuses on the resource overhead of the approach. Especially in resource-constrained sensor nodes, high efficiency is essential for an approach to be meaningfully applicable. Moreover, high efficiency is necessary to enable the sensor nodes' long lifetimes. In WSN applications, standard efficiency metrics include network traffic overhead, memory consumption, computation time, energy overhead, and processing delay (i.e., time until a result is available).

For this purpose, we employed our tripartite experiments described in Section 7. We have grouped the findings of these experiments in two categories based on whether they account to:

- the correctness of our detection approach (Section 8.1)
- the efficiency of our implementation (Section 8.2)

Concerning the former, we additionally provide a comparison of our approach's correctness with alternatives (i.e., data-centric outlier and anomaly detection) in Section 8.1.3.

However, we could not benchmark our approach based on common benchmark datasets as those datasets do not include the required node-level information. In Section 8.3, we conclude the discussion of our results with a summary of the main findings and an exposition of the lessons learned in the course of the present research.

8.1 Correctness Evaluation

Concerning our fault detection's correctness, we analyzed its ability to detect anomalous sensor data caused by node faults and distinguish them from abnormal data related to environmental events. The former is commonly expressed via sensitivity, and the specificity metric indicates the latter. Accuracy metrics such as the F-score are commonly used to state the detection's overall correctness.

In this context, we manually analyzed the results of the lab experiments and the data obtained from the WSN testbed (i.e., indoor and outdoor deployment) to identify faulty conditions and the reaction of our approach to those. We found that all data accountable for node faults were accordingly labeled by our detection approach. Despite extensive data analysis, no false alarms were found in the gathered data.

In addition, we used numerous simulations based on fault-free datasets in which faults were randomly injected as described in Section 7.3. The simulations allowed us to automate the process of fault injection, performing our modified dDCA on the datasets, and finally assess the results of our approach. In this way, numerous simulations were performed with reproducible results, thus removing the possibility of human error in a manual inspection.

8.1.1 Sensitivity, Specificity, and Accuracy

In total, we ran 60 different simulations using three distinct base datasets and five distinct fault signatures (see Section 7.3.3). We used standard metrics for anomaly and fault detection techniques to assess the correctness objectively. In particular, we assessed the sensitivity, specificity, and accuracy of our approach. Regarding the latter, we express the accuracy with the F-score commonly used in the statistical analysis of fault or anomaly detection approaches (cf. Section 3.2).

To assess the simulation results, the acquired fault context of each data instance was compared to the available fault labels. Each data instance was accounted to one of the following four classes:

- true positives (TP): correctly detected faulty data
- true negatives (TN): correctly identified normal data
- false positives (FP): falsely labeled faulty data
- false negatives (FN): falsely accounted normal data

Based on these numbers, the sensitivity, specificity, and accuracy of each simulation run were calculated according to the equations presented in Section 3.2.2. Finally, the results of the single simulations were aggregated to express the overall correctness of our fault detection approach, as summarized in Table 8.1. The table shows that the number of sensor measurements assessed during the simulations varied between 997 sensor readings for the smallest datasets and 10,078 measurements for the largest datasets.

Table 8.1: Aggregated simulation results

	Mean	Min	Max
Sensor measurements	4,025.67	997	10,078
True positives (TP)	37.78	3	77
True negatives (TN)	3,977.35	905	10,072
False positives (FP)	3.80	0	15
False negatives (FN)	6.73	0	20
Sensitivity (TPR)	0.86	0.70	1.00
Specificity (TNR)	1.00	0.99	1.00
Accuracy (F-score)	0.87	0.67	0.95

The simulations revealed that our fault detection approach favors long-lasting faults due to the majority voting-based classification process (cf. Section 6.4.4). Depending on the number of dendritic cells considered for the voting, our approach requires the fault to stay active for at least $\left\lceil \frac{M_{Ag'}}{2} \right\rceil$ update cycles. As a result, faults affecting only a few measurement cycles can be missed due to the voting characteristic of our approach.

The results of the simulations in combination with the data obtained from the lab experiments and the WSN testbed show that our approach reliably detects the majority of faults. Most of all, our approach has shown that it can reliably distinguish between fault-induced data anomalies and correct but rare events in the monitored physical phenomenon. This ability is mandatory for a low false alarm rate.

8.1.2 Fault and Event Detection Examples

As described in Section 6, our approach considers sensor data as faulty if they were acquired in a context dominated by the danger indicator in combination with a low safe indicator value. An example for such a situation is depicted in Figure 8.1.

In the example shown, the sensor node experienced a node fault caused by condensed humidity in the node's housing. During the active fault, incorrect sensor data were reported as visible as a temporary offset in the temperature curve (T_{air}). Our approach correctly identified this node fault as indicated by the resulting fault context *Cx*. However, the identification of the fault happened with a short delay due to the voting of the involved dendritic cells (see Section 6.4.4).

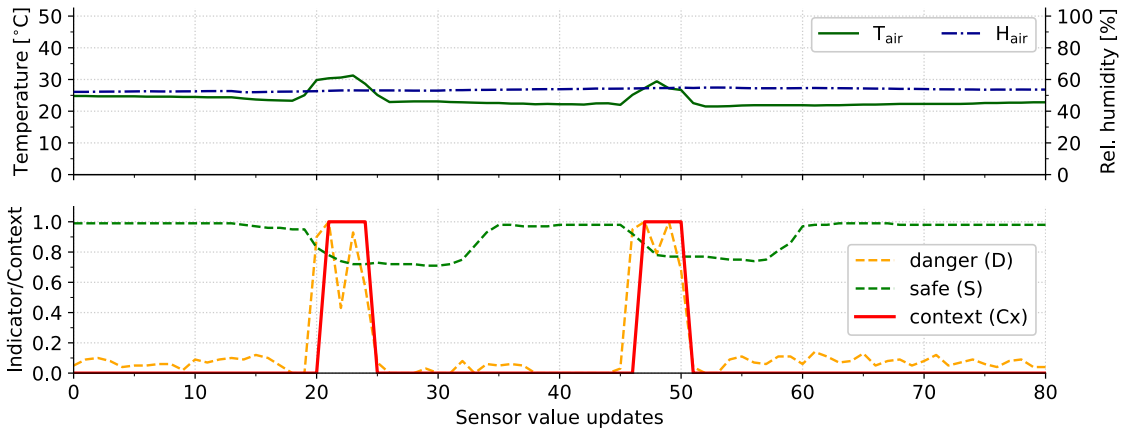


Figure 8.1: Example of a detected sensor node fault

Figure 8.2 shows another example of a successfully detected fault that was caused by a partial short circuit in the board supply rail. The data was obtained in our lab experiments while directly supplying the node with a constantly decreasing supply voltage (V_{BRD}) while bypassing the voltage regulator. Additionally, the microcontroller's brown-out detection was disabled, which is commonly used to decrease the node's energy consumption further. In this example, the ADC self-check (χ_{ADC}) dominantly indicated the presence of faulty conditions.

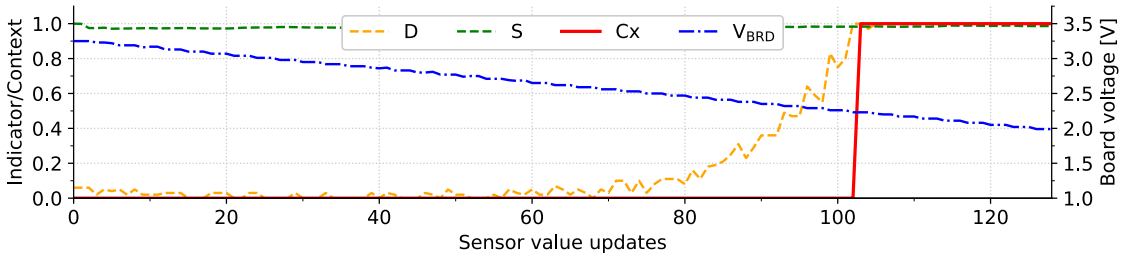


Figure 8.2: Emulation of partial short circuit on board

Due to the combination of the node-level diagnostics with metrics derived from the sensor data, our approach can also identify faults that manifest as a significantly abnormal sensor reading even if the danger indicator does not react. As a result, our approach was able to identify sensor node faults that the fault indicators alone could not (see Figure 8.3). In the figure, two abnormal spikes in the soil temperature measurement are visible that, however, did not cause any fault indicator to react. Consequently, also the aggregated danger indicators stayed at a low value. The fault was still detected thanks to the sensor data-based safe indicator that dramatically dropped during the fault's active time resulting in the successful detection of the fault as indicated by the fault context *Cx*.

Aside from an effective detection of node faults, our approach does not cause false alarms in case of rare but proper events, even if they significantly impact the reported sensor

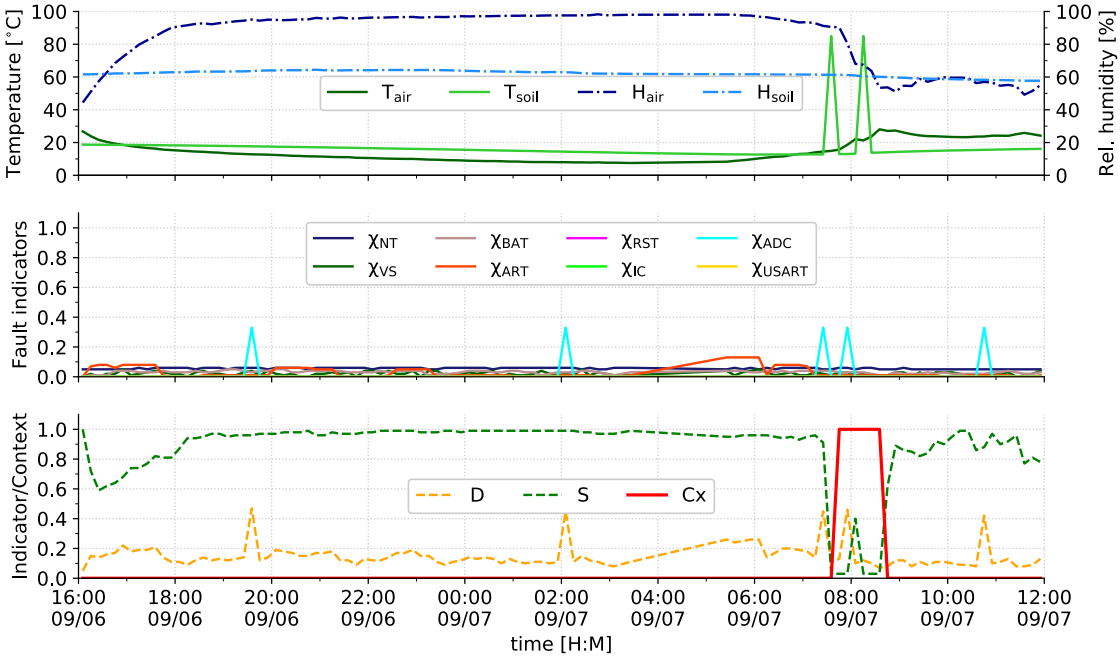


Figure 8.3: Fault detected by our approach but not the fault indicators alone

data. Figure 8.4 depicts an example where the first half of the seven-day record showed a typical diurnal pattern. In contrast, the second half was remarkably influenced by sustained heavy rain. There was no reason to consider these data faulty as the danger indicator did not react.

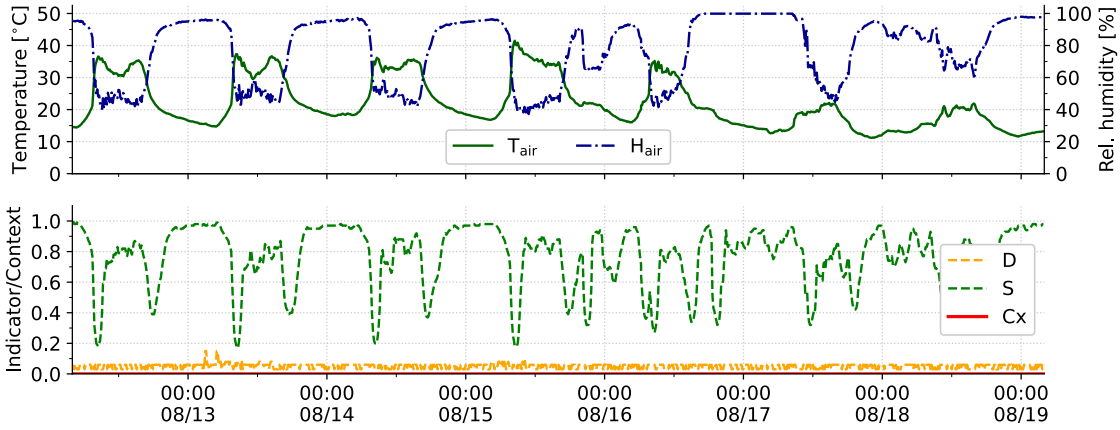


Figure 8.4: Example of intense events not causing false alarms

As can be seen in Figure 8.4, the value of S shows fluctuations for both, the first half with "stable" weather conditions and the second half where heavy rain occurred. This is due to the way the safe indicator value is derived based on changes in the sensor values.

Also during the "stable" half of the dataset, notable diurnal differences between the air temperature and the related relative air humidity happened that caused the safe indicator value to temporally decrease. However, these fluctuations did not cause any false alarm for two reasons: (i) an absence of a danger indicator reaction and (ii) the smoothing characteristic of the context classification provided by the cell population-based assessment. Regarding the latter, the smoothing can be adjusted via the maximum population size M (see Section 6.4.4).

8.1.3 Comparison with alternative Approaches

To objectively assess the detection performance of our approach, we ran a benchmark with state-of-the-art streaming anomaly detection (SAD) methods and a 3σ-based outlier detection as described in Section 7.3.4. The benchmarks utilized the same datasets as used in our correctness analysis presented above; thus, the benchmark results allow an objective comparison with the alternative approaches. Table 8.2 summarizes the results of these benchmarks. It contains the mean sensitivity, specificity, and F-score gained by running the alternative methods on the 60 simulation datasets.

Table 8.2: Summary of comparison results

	Excl. indicators			Incl. indicators		
Method	TPR	TNR	F-score	TPR	TNR	F-score
ExactStorm [384]	0.69	0.90	0.25	0.73	0.90	0.26
IForestASD [385]	0.53	0.86	0.18	0.73	0.88	0.24
RRCF [386]	0.66	0.90	0.22	0.71	0.90	0.22
Standard deviation (3σ)	0.79	0.80	0.29	0.94	0.78	0.32
Our detection approach	×	×	×	0.86	1.00	0.87

The results depicted in Table 8.2 confirm that fault detection based on anomaly or outlier detection in the sensor data is not sufficient. Although the alternative methods utilized notably more resources in the detection process, our approach scores a significantly better accuracy than these alternative methods. These results show that by incorporating node-level diagnostics sensor node faults can be detected more reliably.

Additionally, the results of the SAD method simulations that included the node-level indicators in form of the aggregated danger indicator[1] are notably better than those without them. Only in the case of the 3σ-based approach the specificity (i.e., TNR) was slightly worse due to the inclusion of the indicator value in the running standard deviation that also affected the assessment of the succeeding data instances. Consequently, these results support our claim of the inevitability of including node-level information in the fault detection process. By including such node-level data, the detection of node faults

[1]The safe indicator was neglected as the sensor values are analyzed by the methods themselves.

becomes more accurate. Moreover, it allows the approaches to distinguish between the effects of environmental events from those caused by node faults.

Surprisingly, the 3σ-based approach scored better results than the SAD methods considered. In this context, we found that the SAD methods suffer from three problems that negatively impacted the acquired results. First, these methods build their internal model successively with the incoming data. As a result, the data instances are mostly considered anomalous in the first few iterations as the internal model is not yet sufficiently evolved. Second, even when the model has been properly trained, unexpected events (such as rain) distort the model and cause data instances to be classified incorrectly. Third, the SAD methods require faults to have a significant influence on the reported sensor values to be able to detect deviations. However, sensor node faults do not always cause significant changes in the reported values. The five used fault signatures cover different manifestations of node faults, including cases where the sensor data are altered insignificantly. Those faults are almost always missed by the alternative methods used. As a result, the overall correctness metrics of the alternative methods are mediocre.

8.2 Efficiency Analysis

Aside from assessing the correctness, we also evaluated our approach's efficiency. For this purpose, we analyzed the resource footprint of our proposed fault detection including:

- the overhead on the network traffic (Section 8.2.1)
- the memory requirements (Section 8.2.2)
- the impact on the runtime behavior (i.e., computation time; Section 8.2.3)
- the resulting energy overhead (Section 8.2.4)
- the corresponding processing delay of the fault detection (Section 8.2.5)

In the efficiency evaluation presented in the following, a sensor node equipped with an AM2302 sensor was used. For the measurements of the active processing time and the power consumption of the sensor node, we used a stabilized power supply and a Joulescope as described in Section 7.4.1. We measured the duration the nodes spent in an active state and the corresponding energy consumed. The Joulescope's software kept track of the power consumed during the measurements and, thus, measured the consumed energy (i.e., electrical charge). For the measurements, the sensor node was configured to send a measurement update every 10 minutes. We calculated the mean average of 20 measurements for each version (i.e., with and without fault detection enabled).

However, we focused our efficiency evaluation on the sensor nodes as their resources are strictly limited, especially their energy budget. In our implementation, the time complexity of the cluster head's part is $\mathcal{O}(1)$, and the space complexity grows linear with the number of sensor nodes (i.e., $\mathcal{O}(n)$). However, the cluster head's efficiency was not further analyzed as it had sufficient resources and was powered by a wired power supply.

8.2.1 Network Traffic Overhead

For a networked system, considering the impact of an approach on the network traffic is an integral part of the efficiency analysis. This is especially crucial for WSNs for two reasons: (i) the network capabilities of the sensor nodes are often limited and (ii) the transmission of data over the wireless links requires a significant amount of energy. Concerning the latter, depending on the network interface and its settings, the transmission of a single bit can cost as much energy as the execution of 800–1000 instructions in the MCU (cf. [389]). For this reason, it is crucial to keep the overhead on the network messages as small as reasonably possible. Consequently, we analyzed the network traffic overhead of our approach.

As shown in the overview depicted in Figure 6.1, our approach requires diagnostic data from the sensor nodes to be transmitted to the system performing the fault assessment, such as the cluster head. This diagnostic data consists of the aggregated danger and safe indicators, which are, as described in Section 7.2.4, transmitted as fixed-point numbers with six fractional and two integer bits. As a result, two bytes (i.e., 16 bits) have to be added to each message a sensor node transmits to the cluster head. However, this overhead is fixed and does not depend on the application domain or the use case.

In our evaluation use-case (smart garden system), each sensor node message contains two bytes with the local timestamp, eight bytes of use-case data (two bytes per measurement), and the said two bytes with diagnostic data (see also Figure 7.4). Therefore, our approach requires a network traffic overhead of 20 % for the presented use case.

8.2.2 Memory Consumption

To evaluate the memory requirements imposed by our approach, we prepared two versions of the sensor node software, one without and the second with the fault detection included. Both were compiled with *avr-gcc* with optimization for size (*-Os*) and link-time optimization (*-flto*). We then obtained the memory requirements of both versions using the *avr-size* tool. Table 8.3 summarizes the results of the memory consumption comparison. It shows the absolute consumption of flash memory (i.e., program space), SRAM (i.e., data space), and EEPROM as well as the relative consumption (in percent) concerning the resources available on the used ATmega1284P microcontroller. The difference expresses the memory overhead imposed by our self-diagnostics.

Table 8.3: Our approach's sensor node memory consumption

	Flash memory		SRAM		EEPROM	
	[Byte]	[%]	[Byte]	[%]	[Byte]	[%]
Detection excluded	6,250	4.8	256	1.6	0	0.0
Detection included	12,578	9.6	331	2.0	4	0.1
Difference	6,328	4.8	75	0.4	4	0.1

While the consumption of SRAM and EEPROM is negligibly small, the implementation of our fault detection approach on the sensor nodes requires a considerable amount of flash memory. This is mainly caused by the multitude of intermediate diagnostic data calculated in each iteration. However, an approximated overhead of 6 kB flash memory is acceptable as most modern sensor nodes offer significantly higher resources as presented in the sensor node platform overview in Table 2.1.

8.2.3 Computation Time

Our detection approach extends the sensor node software with the node-level diagnostics described in Section 7.1.2. Consequently, it alters the runtime behavior of the nodes. In this context, it prolongs the time the sensor node is active to perform the diagnostics (cf. Figure 7.3). During this time, the node consumes more energy than if it was in sleep mode; thus, it also affects the energy consumption and the battery life. Therefore, we measured the time the sensor node is active and the energy consumed of (i) a sensor node without the fault detection added and (ii) a sensor node with the fault detection activated. The effect of our approach on the nodes' active phase is discussed in the following. In Section 8.2.4, the resulting impact on the energy consumption is presented. A summary of both results is given in Table 8.4.

Table 8.4: Our approach's sensor node runtime impact

	Active time		Charge per 1h	Est. battery life	
	[ms]	[μAh]	[μAh]	[h]	[years]
Detection excluded	198	0.93	48.8	53,279	6.08
Detection included	204	0.95	51.0	50,980	5.82
Difference	6	0.02	2.2	-2,299	-0.26

The sensor node spent an average of 198 ms in an active state without fault detection and remained the rest of the time sleeping. With a 10-minute update interval, the sensor node consumed an average of 48.8 μAh per hour.

On the other hand, the sensor node stayed active for an average of 204 ms with the fault detection enabled. Therefore, the diagnostics prolonged the active time by 6 ms (equals 3 %). The node with an enabled detection consumed an average of 51.0 μAh per hour, which equals a plus of 2.2 μAh (or 4.5 %).

At first glance, this overhead appears to be negligibly small. However, when analyzing the node's active phase in more detail, we found that the network connectivity of the used Xbee 3 transceiver in Zigbee configuration requires a minimum active time of the module. Consequently, the node's active time also had to exceed a certain threshold. When analyzing the runtime behavior of the node in more detail using the Joulescope and specific GPIOs to indicate the current phase of the processing, we found that the diagnostics actually take an average of about 77 ms. Although the majority of the node-level diagnostics require simple calculations only, some of them depend on the acquisition

of sensory values (e.g., from the ADC peripheral) that take a certain time. Therefore, the overhead can become more apparent in other use cases or with other network interfaces.

8.2.4 Energy Overhead

Although the impact of the fault detection on the sensor node is justifiably small, it still causes a shortened battery life. In the following, we show the impact based on estimating the battery life when powering the sensor node with two Alkaline LR6 AA batteries. These batteries have a nominal capacity of about 2600 mAh.

By applying a linear approximation, the expected battery lifetime of the node without our fault detection can be estimated as:

$$t_{\mathrm{w/o}} = \frac{2600\,\mathrm{mAh}}{48.8\,\mu\mathrm{Ah}} \cdot 1\,\mathrm{h} \approx 53,279\ \mathrm{h} \equiv 6.08\ \mathrm{years} \tag{8.1}$$

With the detection enabled, the sensor node would last for:

$$t_{\mathrm{w/}} = \frac{2600\,\mathrm{mAh}}{51.0\,\mu\mathrm{Ah}} \cdot 1\,\mathrm{h} \approx 50,980\ \mathrm{h} \equiv 5.82\ \mathrm{years} \tag{8.2}$$

As a consequence, the battery life is decreased by approximately 3 months (0.26 years; see Table 8.4). Nevertheless, the sensor node with the fault detection enabled can operate for a considerably long time, especially considering that node faults can be detected during this time.

8.2.5 Processing Delay

On the system where the dDCA-based fault detection is implemented, the assessment of incoming data consists of a deterministic sequence of computational steps. It neither requires any lead time nor significant delays in its sequential processing, hence ensuring a real-time assessment. However, that is not the case in most related DCA-based detection approaches, where the detection is usually performed offline on an entire dataset.

8.3 Lessons learned

Our tripartite experiments showed that our approach offers promising results as it requires a justifiable resource overhead and has a high detection accuracy. It confirms that immune-inspired schemes such as those based on the functioning of dendritic cells (i.e., DCA-based approaches) have indeed properties beneficial for the fault detection in resource-constrained systems such as WSNs. However, aside from the contributions and results previously published in our papers [3–8], two vital lessons were learned in the course of the present research that will be presented in the following.

After our initial literature review revealed the research gap treated in this thesis, that is, the detection of sensor node faults by exploiting node-level information, the research

plan assumed that the most critical part would be the detection algorithm based on a suitable adoption and adaptation of the DCA (or one of its variants). In this initial plan, the input data (i.e., node-level fault indicators) fed to the algorithm were considered a means to an end but were not particularly considered. However, during our research, we found that it is the other way round: the input data are the essential part, and the detection algorithm is mostly a vehicle for the automated assessment (cf. Section 8.1.3). In other words, the best and most efficient algorithm still relies on the quality of the input data. Therefore, the majority of the work presented in this thesis deals with the node-level diagnostics (i.e., fault indicators) that made effective and efficient node-fault detection possible in the first place. Still, the algorithm has a significant impact on the characteristics of the detection approach and its final results. For example, the drawback of missing short-term faults by our majority voting-based assessment may be remedied with a better assessment scheme. This, however, is part of future work and not considered within this thesis.

The second main lesson refers to utilizing immune mechanisms (or bio-inspired processes in general) for solving computational problems. As pointed out in related works, the basic structure of WSNs shows a certain similarity with the biological entities involved in the HIS. Consequently, it was commonly acknowledged that the key to successfully applying immune-inspired schemes lies in a suitable mapping of biological entities to computational counterparts. In this context, we also found that the general characteristics of the considered biological systems need to be minded. The HIS involves an enormous amount of various cells where quantity has more effect on the system's performance than quality. In computing systems, it is usually the other way round. Although there are WSN that incorporate thousands of sensor nodes, these numbers are no comparison to the number of cells cooperatively proving immunity (cf. Section 5.2.1). Consequently, researchers usually have to develop suitable abstractions of the underlying processes or need to come up with creative solutions to overcome the limitations of computing systems.

CHAPTER 9

Summary

In this thesis, we have presented our immune-inspired sensor node fault detection approach for wireless sensor networks (WSNs). Most related approaches base their detection merely on the sensor data gathered by the sensor nodes. In contrast, our approach additionally incorporates node-level diagnostics in the decision process. For this reason, our approach offers higher accuracy than alternative detection approaches, especially concerning its specificity. Regarding the latter, our approach can distinguish data events from fault-induced data deviations. This is not possible in most related approaches.

A brief introduction to the importance of fault detection in wireless sensor networks (WSNs) and an organizational overview of the present thesis were given in Chapter 1. We described the targeted research questions and the applied methodology to answer them, resulting in the presented scientific contributions.

Chapter 2 provided a more detailed introduction to WSNs to highlight the particular characteristics and distinctive features entailed. The provided considerations are essential to understand better the importance of sensor node fault detection and the challenges imposed. In addition, we presented a survey on recent sensor node platforms and a summary of their characteristics.

The detection of node faults is often considered an anomaly detection task and is purely based on the sensor data, as discussed in Chapter 3. Aside from standard metrics used in anomaly detection, we presented our taxonomy for anomaly detection in WSNs with a focus on node-level techniques. As found in our presented meta-survey, most related approaches suffer from a disability to distinguish the effects of environmental events in the sensor data from data distortion caused by sensor node faults.

Consequently, we discussed the peculiarities of sensor node faults in Chapter 4 and presented our taxonomy for the classification of faults on wireless sensor nodes. In addition, we elaborated on related fault detection approaches for WSNs that can be

broadly categorized as either (i) sensor data analysis, (ii) group detection, or (iii) local self-diagnosis schemes. However, approaches of all three categories entail specific assumptions or suffer from certain limitations that hinder their fault detection efficacy.

In the context of resource-efficient fault detection, several approaches were inspired by the promising properties of the natural anomaly detection performed by the human immune system (HIS). For this reason, Chapter 5 first presents an overview of immunological models followed by a discussion of immune-inspired fault detection approaches, in particular the dendritic cell algorithm (DCA) which is based on the findings of the so-called danger theory. Additionally, the limitation of current approaches when applied to sensor node fault detection are treated.

Chapter 6 presented our immune-inspired fault detection approach applicable to the detection of node faults in WSNs. Our approach remedies several limitations of previous fault detection schemes, especially concerning the distinction of data events from the manifestation of sensor node faults. We developed node-level diagnostic metrics denoted as fault indicators. These fault indicators and statistical metrics derived from the sensor data are fed to a modified DCA to classify the acquired sensor data as either normal or faulty in real-time.

The evaluation of our approach, including the concept implementation and our tripartite experiment setup, is described in Chapter 7. In this chapter, a prototype implementation of the entire approach is presented that is also publicly available on Github. To appropriately evaluate our approach, we leveraged a combination of simulations, directed experiments in a lab setup, and a practical WSN including both an indoor and outdoor deployment.

The results of our concept evaluation and the corresponding findings are discussed in Chapter 8. As confirmed by our evaluation, our approach offers a higher detection accuracy than alternative approaches while it requires a reasonably small resource overhead only. In addition, we presented lessons learned in the course of the present research.

To complete this thesis, we give an overview of the research findings and our dissemination in Section 9.1. A summary of related work for sensor node fault detection is shown in Section 9.2 followed by our conclusion in Section 9.3. Finally, an outlook for future work is presented in Section 9.4.

9.1 Research Findings and Dissemination

As mentioned in Section 1.3, this thesis mainly presents a summary of work already published in corresponding conference proceedings and journal articles. In the following, we provide an overview of the particular publications and their specific contribution to answering the research questions (RQs) presented in Section 1.1.

In [3], we present a comprehensive taxonomy for anomaly detection in WSNs that mainly answers RQ#1. It results from extensive initial literature research covering a broad

spectrum of anomaly detection applications, including the detection of faults. The contributions of [3] include:

- a refined taxonomy extending previous WSN anomaly detection taxonomies
- a meta-survey of reviews on anomaly detection schemes for WSNs
- an elaboration on new insights into (node-level) anomaly detection in WSNs
- a discussion of research challenges concerning node-level anomaly detection

The main reasons, the probability, and the severity of node-faults in WSNs, as well as the impact of the node's hardware choices on those, are treated in [4], thus, focusing on RQ#2 and RQ#4. We found that specific environmental influences are the main drivers for node faults, aside from design flaws. In particular, the sensor node's ambient temperature and supply voltage have significantly impacted their operation's stability. Thereby, we explicitly analyzed the effects of so-called undervolting. This effect can either happen on purpose as part of an energy-saving technique or unintentionally if the battery runs low. Consequently, the contributions of [4] include:

- an analysis of undervolting on the component- and system-level including on-chip and onboard component interaction
- an investigation of the stability of the node's operation at different supply voltage levels under various ambient temperatures and different clock sources
- an identification of critical voltage regions leading to faults resulting in silent data corruption that a fault-detection approach can not trivially detect

The previously identified importance of node-level information for efficient and correct fault detection and possible sources of such information are elaborated on in [5]. It targets RQ#2, RQ#4, and RQ#5 by introducing a diagnostic concept we named "node-level indicators". These indicators form the core of the subsequent fault detection approach. We showed that suitable fault indicators exist and help distinguish data anomalies between correct data caused by rare events and erroneous data caused by soft faults. For these reasons, the contributions of [5] include:

- an analysis of the manifestation and risk of soft faults in wireless sensor nodes
- an inspection of node-level information sources (the fault indicators)
- an evaluation of the benefits of using such fault indicators for soft-fault detection

Although being focused on the Arduino platform, our considerations and findings presented in [6] resulted in generally applicable sensor node design insights (RQ#4) that further led to a self-developed sensor node platform. The restrictions and possible improvements for the use of Arduino in WSNs presented in the paper highlighted critical aspects of an efficient sensor node design, especially regarding its overall power consumption and total energy efficiency. The contributions of [6] include:

- a comparison of recent Arduino boards with common sensor nodes
- an analysis of the power consumption of recent low-power Arduino boards

- an evaluation of Arduino core software components (e.g., the bootloader)

A comprehensive discussion of node faults in WSNs including a fault taxonomy and relevant fault models has been published in the journal article in [8]. In addition, the article presents the aforementioned self-developed sensor node platform that was then used for the majority of practical experiments. As a result, the article provides answers to RQ#3 and RQ#5. In addition, our article in [8] provides a summary and extension of topics previously published in [3–6]. The contributions of [8] include:

- an extensive literature review on recent sensor node platforms
- a comprehensive taxonomy for faults in WSNs
- a practical evaluation of the fault indicator concept proposed in [5]
- a detailed introduction of the ASN(x)
- the introduction of our ETB, a Raspberry Pi hardware add-on for the analysis and profiling of embedded systems like sensor nodes

Finally, an answer to RQ#6 is presented in [7]. The paper discusses the immunological background and the theoretical basis for developing our immune-inspired node fault detection approach that forms the core of this doctoral research. Our key contribution in [7] is to integrate node-level diagnostics with the characteristics of the sensor data. Especially the choice and representation of the algorithms' input data (i.e., node-level diagnostics) have shown to be crucial for proper functioning. Expressive input data significantly influences fault detection's effectiveness, much more than the aggregation by the algorithm does. Additionally, the paper provides:

- an introduction of our immune-inspired runtime fault detection approach
- a presentation of our implementation of the detection approach
- an extensive quantitative evaluation of our concept

To sum up, the RQs are answered in our publications as follows:

- RQ#1: node fault detection → research gap [3]
- RQ#2: fault analysis → fault models [4, 5]
- RQ#3: fault classification → fault classes [8]
- RQ#4: fault detectability → design guide [5, 6]
- RQ#5: fault diagnosis → node diagnostics [5, 8]
- RQ#6: DCA fault detection → detection concept [7]

9.2 Related Work

In our literature reviews (cf. [3, 8]), the detection of sensor node faults has been found to be most often considered a data anomaly detection task and is purely performed on the sensor data. Consequently, related fault detection approaches presented in the

Sections 3.4, 4.4, and 5.5.3 mostly use data anomaly detection schemes to identify abnormal sensor readings. Utilizing data anomaly detection for fault diagnosis suffers from a crucial problem: anomalies do not need to be caused by faulty sensor nodes. Similarly, not all node faults cause distinct irregularities in the reported sensor data. Therefore, such approaches suffer from a disability in distinguishing between data events and fault-induced deviations, as discussed in Section 4.2.2.

Also, other related approaches such as those based on group detection or local self-diagnosis impose limitations and restrictions that reduce their detection accuracy and general applicability. As summarized in Section 4.5, especially their resource and communication overhead disqualify them from a meaningful use in most WSN applications.

In contrast, our fault detection approach does not suffer from the limitations entailed by the assumptions and drawbacks of the related works. It requires a reasonably small resource overhead and enables the distinction between data events and sensor data distortion caused by node faults. Additionally, our approach is generally applicable as it: (i) removes the need for domain or expert knowledge, (ii) does not need manual intervention and analysis, (iii) can also be used with heterogeneous sensor nodes, and (iv) is suitable for static and dynamic networks. However, our approach offers a first line of defense against faults happening in the WSN. The WSN still has to include measures to deal with faults introduced later in the subsequent data chain as well as to analyze the provided data and acquire meaningful insights.

9.3 Conclusion

Our approach combines statistical metrics derived from the sensor data with node-level diagnostics acquired on a node level. Aside from an increased detectability of node faults, this combination allows our approach to distinguish between actual data events and fault-related data distortion. That, however, is not possible in the majority of related fault detection approaches. We have found that the availability of suitable input data (i.e., node-level diagnostics) is crucial and influences the fault detection effectiveness more than the actual detection algorithm (i.e., the modified dDCA) does. Our approach requires a reasonable resource overhead regarding its memory consumption, processing time, and overall energy requirements. It can be applied to most WSNs as it does not suffer from the limiting assumptions imposed by several related works. Additionally, the entire approach has been implemented and is publicly available.

As discussed in Section 6.5, our implementation includes several parameters that have an impact on the correctness of the fault classification. In contrast to related approaches, our approach only requires a small number of parameters whose optimal values are not application-specific. They are generally applicable, or at least for an entire domain of applications. We performed several experiments and simulations to derive suitable parameter choices. However, further analyses may propose even better parameter values.

9.4 Future Work

Like the original DCA, our approach does currently not incorporate learning mechanisms. Several works propose to replace the linear classifiers of the DCA with machine learning algorithms to enable an automated parameter adjustment. Also, learning techniques could be used to implement an immune memory allowing for faster detection of fault patterns that have been experienced in the past.

Our fault detection approach focuses on the temporal correlation between the sensor measurements and the diagnostic data. The dendritic cells in the HIS, however, also perform a spatial correlation of the information provided. Similarly, the assessment of the abstract dendritic cells in our approach performed on the cluster head could consider the measurements of several sensor nodes. To do so, a suitable definition of the antigens is necessary. It has to facilitate the classification of the sensor values instead of the sensor nodes, for example, by deriving the antigens from a combination of the sensor values.

So far, we have evaluated our approach based on one type of sensor node with a set of eight fault indicators. Further experiments with heterogeneous sensor nodes are necessary to evaluate the approach's generality. Although the core detection approach (i.e., the modified dDCA) is generally applicable, the underlying node-level diagnostics used for the calculation of the danger indicator depend on the used hardware. In this context, the analysis of further sources of fault indicators such as system features provided by operating systems or specific middleware are a promising future research direction.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ADC	analog-to-digital converter
AI	artificial intelligence
AIN	artificial immune network
AIRS	artificial immune recognition system
AIS	artificial immune system
ALU	arithmetic logic unit
ANN	artificial neural network
APC	antigen-presenting cell
ASN(x)	AVR-based Sensor Node with Xbee radio
AUC	area under curve
AUROC	area under the ROC curve
AVR	Alf and Vegard's RISC
BLE	Bluetooth low energy
CAS	complex adaptive systems
CH	cluster head
CI	computational intelligence
CLONALG	clonal selection algorithm
CMOS	complementary metal oxide semiconductor
CPU	central processing unit
CSM	co-stimulatory molecule
CSPRA	conserved self pattern recognition algorithm
DB	database
DC	dendritic cell
DCA	dendritic cell algorithm
dDCA	deterministic dendritic cell algorithm
DFS	dynamic frequency scaling
DSP	digital signal processor
DVS	dynamic voltage scaling
EEPROM	electrically erasable programmable read-only memory
ETB	embedded testbench

FAR	false alarm rate
FN	false negatives
FNR	false negative rate
FP	false positives
FPGA	field-programmable gate array
FPR	false positive rate
GA	genetic algorithm
GP	genetic programming
GPIO	general-purpose input/output
HCI	hot carrier injection
HIS	human immune system
I2C	inter-integrated circuit
IDS	intrusion detection system
IEEE	Institute of Electrical and Electronics Engineers
IForestASD	isolation forest algorithm for streaming data
INS	infectious non-self
IoT	Internet of Things
ISM	industrial, scientific and medical
ISP	in-system programming
JTAG	Joint Test Action Group
LDO	low-dropout regulator
LED	light-emitting diode
LoRaWAN	long-range wide-area network
LPWAN	low-power wide-area network
LSB	least significant bit
MAC	media access control
MCU	microcontroller unit
MCUSR	MCU status register
min-dDCA	minimized dDCA
MIPS	million instructions per second
ML	machine learning
MOSFET	metal-oxide-semiconductor field-effect transistor
NBTI	negative-bias temperature instability
NSA	negative selection algorithm
NTC	negative temperature coefficient
OPC UA	Open Platform Communications Unified Architecture
OS	operating system
OTR	outdoor relay node
OWI	one-wire interface

PAMP	pathogen associated molecular patterns
PCA	principal component analysis
PCB	printed circuit board
PPV	positive predictive value
PRR	pattern recognition receptor
RF	radio frequency
RISC	reduced instruction set computer
ROC	receiver operating characteristics
RQ	research question
RRCF	robust random cut forest
RSSI	received signal strength indicator
RTC	real-time clock
SAD	streaming anomaly detection
SHM	structural health monitoring
SI	swarm intelligence
SK	sink node
SLR	systematic literature review
SN	sensor node
SNR	signal-to-noise ratio
SNS	self/non-self
SoC	system-on-a-chip
SOM	self-organizing map
SPI	serial peripheral interface
SRAM	static random-access memory
SVM	support vector machine
TDDB	time-dependent dielectric breakdown
THT	through-hole technology
TLR	toll-like receptors
TN	true negatives
TNR	true negative rate
TP	true positives
TPR	true positive rate
TWI	two-wire interface
USART	universal synchronous/asynchronous receiver-transmitter
USB	universal serial bus
VSM	voltage scaling module
WDT	watchdog timer
WSN	wireless sensor network

Bibliography

[1] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manage. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, dec 2007.

[2] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7 – 15, 2009, special Section - Most Cited Articles in 2002 and Regular Research Papers.

[3] D. Widhalm, K. M. Goeschka, and W. Kastner, "SoK: A taxonomy for anomaly detection in wireless sensor networks focused on node-level techniques," in *The 15th International Conference on Availability, Reliability and Security (ARES '20)*, 2020.

[4] ——, "Undervolting on wireless sensor nodes: a critical perspective," in *The 23rd International Conference on Distributed Computing and Networking (ICDCN '22)*, 2022.

[5] ——, "Node-level indicators of soft faults in wireless sensor networks," in *The 40th International Symposium on Reliable Distributed Systems (SRDS '21)*, 2021, pp. 13–22.

[6] ——, "Is arduino a suitable platform for sensor nodes?" in *The 47th Annual Conference of the IEEE Industrial Electronics Society (IECON '21)*, 2021, pp. 1–6.

[7] ——, "Sensor node fault detection in wireless sensor networks utilizing node-level diagnostics," in *The 20th ACM Conference on Embedded Networked Sensor Systems (SenSys'22)*, 2022, (in review).

[8] ——, "An open-source wireless sensor node platform with active node-level reliability for monitoring applications," *Sensors*, vol. 21, no. 22, 2021.

[9] H. Mahmoud and A. Fahmy, "WSN applications," in *Concepts, Applications, Experimentation and Analysis of Wireless Sensor Networks*. Springer International Publishing, Nov. 2020.

[10] D. Culler, D. Estrin, and M. Srivastava, "Guest editors' introduction: Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, Aug 2004.

[11] A. Ali, Y. Ming, S. Chakraborty, and S. Iram, "A comprehensive survey on real-time applications of wsn," *Future Internet*, vol. 9, no. 4, 2017.

[12] I. Akyildiz and M. Vuran, *Wireless Sensor Networks*, ser. Advanced Texts in Communications and Networking. Wiley, 2010.

[13] S. Kharb and A. Singhrova, "Review of industrial standards for wireless sensor networks," in *Next-Generation Networks*, D. K. Lobiyal, V. Mansotra, and U. Singh, Eds. Singapore: Springer Singapore, 2018, pp. 77–87.

[14] S. Barillaro, S. Rhee, G. Escudero, R. Kacker, L. Badger, and D. R. Kuhn, "Low-power wide area networks (lpwan) for communications of mobile sensor data," in *Proceedings of the 2nd ACM/EIGSCC Symposium on Smart Cities and Communities*, ser. SCC '19. New York, NY, USA: Association for Computing Machinery, 2019.

[15] J. Zenisek, J. Wolfartsberger, C. Sievi, and M. Affenzeller, "Modeling sensor networks for predictive maintenance," in *On the Move to Meaningful Internet Systems: OTM 2018 Workshops*, C. Debruyne, H. Panetto, W. Guédria, P. Bollen, I. Ciuciu, and R. Meersman, Eds. Cham: Springer International Publishing, 2019, pp. 184–188.

[16] T. Mujawar and L. Deshmukh, "Smart environment monitoring system using wired and wireless network: A comparative study," in *Atmospheric Air Pollution Monitoring [Working Title]*. IntechOpen, Aug. 2019.

[17] M. F. Othman and K. Shazali, "Wireless sensor network applications: A study in environment monitoring system," *Procedia Engineering*, vol. 41, pp. 1204 – 1210, 2012, international Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

[18] A. Rajput, V. B. Kumaravelu, and A. Murugadass, "Smart monitoring of farmland using fuzzy-based distributed wireless sensor networks," in *Lecture Notes on Multidisciplinary Industrial Engineering*. Springer Singapore, Jun. 2019, pp. 53–75.

[19] A. Abid, A. Kachouri, and A. Mahfoudhi, "Anomaly detection through outlier and neighborhood data in wireless sensor networks," in *2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, March 2016, pp. 26–30.

[20] S. Ullo, M. Gallo, G. Palmieri, P. Amenta, M. Russo, G. Romano, M. Ferrucci, A. Ferrara, and M. De Angelis, "Application of wireless sensor networks to environmental monitoring for sustainable mobility," in *2018 IEEE International Conference on Environmental Engineering (EE)*, March 2018, pp. 1–7.

[21] K. Grover, D. Kahali, S. Verma, and B. Subramanian, "WSN-based system for forest fire detection and mitigation," in *Lecture Notes on Multidisciplinary Industrial Engineering*. Springer Singapore, Jun. 2019, pp. 249–260.

[22] A. Molina-Pico, D. Cuesta-Frau, A. Araujo, J. Alejandre, and A. Rozas, "Forest monitoring and wildland early fire detection by a hierarchical wireless sensor network," *Journal of Sensors*, vol. 2016, pp. 1–8, 2016.

[23] K. Bouabdellah, H. Noureddine, and S. Larbi, "Using wireless sensor networks for reliable forest fires detection," *Procedia Computer Science*, vol. 19, pp. 794 – 801, 2013, the 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013).

[24] L. Muduli, D. P. Mishra, and P. K. Jana, "Application of wireless sensor network for environmental monitoring in underground coal mines: A systematic review," *Journal of Network and Computer Applications*, vol. 106, pp. 48 – 67, 2018.

[25] J. Valverde, V. Rosello, G. Mujica, J. Portilla, A. Uriarte, and T. Riesgo, "Wireless sensor network for environmental monitoring: Application in a coffee factory," *International Journal of Distributed Sensor Networks*, vol. 8, no. 1, p. 638067, Jan. 2012.

[26] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, ser. WSNA '02. New York, NY, USA: ACM, 2002, pp. 88–97.

[27] P. Juang, H. Oki, Y. Wang, Y. Wang, Y. Wang, Y. Wang, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," *SIGPLAN Not.*, vol. 37, no. 10, pp. 96–107, Oct. 2002.

[28] A. L. Barriuso, G. Villarrubia González, J. F. De Paz, Á. Lozano, and J. Bajo, "Combination of multi-agent systems and wireless sensor networks for the monitoring of cattle," *Sensors (Basel, Switzerland)*, vol. 18, no. 1, p. 108, Jan 2018, 29301310[pmid].

[29] K. H. Kwong, T. T. Wu, H. G. Goh, B. Stephen, M. Gilroy, C. Michie, and I. Andonovic, "Wireless sensor networks in agriculture: Cattle monitoring for farming industries," *PIERS Online*, vol. 5, no. 1, pp. 31–35, 2009.

[30] J. Lee, M. Ghaffari, and S. Elmeligy, "Self-maintenance and engineering immune systems: Towards smarter machines and manufacturing systems," *Annual Reviews in Control*, vol. 35, no. 1, pp. 111 – 122, 2011.

[31] J. Scheer and L. Wilharm, *Failed Bridges: Case Studies, Causes and Consequences*. Wiley, 2011.

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

[33] B. Chen, "Agent-based artificial immune system approach for adaptive damage detection in monitoring networks," *Journal of Network and Computer Applications*, vol. 33, no. 6, pp. 633 – 645, 2010, advances on Agent-based Network Management.

[34] M. Giammarini, D. Isidori, E. Concettoni, C. Cristalli, M. Fioravanti, and M. Pieralisi, "Design of wireless sensor network for real-time structural health monitoring," in *2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits Systems*, April 2015, pp. 107–110.

[35] Z. Liu and Y. Kleiner, "State-of-the-art review of technologies for pipe structural health monitoring," *IEEE Sensors Journal*, vol. 12, no. 6, pp. 1987–1992, June 2012.

[36] M. Rajendra Dhage and S. Vemuru, "Structural health monitoring of railway tracks using wsn," in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, Aug 2017, pp. 1–5.

[37] F. X. Li, A. A. Islam, A. S. Jaroo, H. Hamid, J. Jalali, and M. Sammartino, "Urban highway bridge structure health assessments using wireless sensor network," in *2015 IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNet)*, Jan 2015, pp. 75–77.

[38] A. Zrelli and T. Ezzedine, "Localization of damage using wireless sensor networks for tunnel health monitoring," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, June 2017, pp. 1161–1165.

[39] J. Joshi, A. Bagga, A. Reddy, D. Akhil, H. Munnangi, B. Nikhil, and J. Reddy, "Structural health monitoring of earth air tunnel (eat) using wireless sensor network," in *2016 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS)*, June 2016, pp. 1–5.

[40] P. K. Patil and S. R. Patil, "Review on structural health monitoring system using wsn for bridges," in *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, vol. 1, April 2017, pp. 628–631.

[41] I. Khemapech, W. Sansrimahachai, and M. Toahchoodee, "A real-time health monitoring and warning system for bridge structures," in *2016 IEEE Region 10 Conference (TENCON)*, Nov 2016, pp. 3010–3013.

[42] P. R. Kamble and R. A. Vatti, "Structural health monitoring of river bridges using wireless sensor networks," in *2015 International Conference on Pervasive Computing (ICPC)*, Jan 2015, pp. 1–5.

[43] S. Arms, J. Galbreath, A. Newhard, and C. Townsend, "Remotely reprogrammable sensors for structural health monitoring," *Proc. NDE/NDT for Highways and Bridges, Structural Materials Technology VI*, 01 2004.

[44] K. Y. Koo, J. M. W. Brownjohn, D. I. List, and R. Cole, "Structural health monitoring of the tamar suspension bridge," *Structural Control and Health Monitoring*, vol. 20, no. 4, pp. 609–625, Mar. 2012.

[45] J. Ding, K. Sivalingam, B. Li, and Y. Hu, "Design and analysis of an integrated mac and routing protocol framework for wireless sensor networks." *Ad Hoc & Sensor Wireless Networks*, vol. 2, January 2006.

[46] V. Ramasamy, "Mobile wireless sensor networks: An overview," in *Wireless Sensor Networks - Insights and Innovations*. InTech, October 2017.

[47] P. Kumar and S. R. N. Reddy, "Wireless sensor networks: a review of motes, wireless technologies, routing algorithms and static deployment strategies for agriculture applications," *CSI Transactions on ICT*, vol. 8, no. 3, May 2020.

[48] F. Karray, M. W. Jmal, A. Garcia-Ortiz, M. Abid, and A. M. Obeid, "A comprehensive survey on wireless sensor node hardware platforms," *Computer Networks*, vol. 144, 2018.

[49] H.-S. Kim, M. P. Andersen, K. Chen, S. Kumar, W. J. Zhao, K. Ma, and D. E. Culler, "System architecture directions for post-soc/32-bit networked sensors," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys '18)*. New York, NY, USA: Association for Computing Machinery, 2018.

[50] S. Fattah, A. Gani, I. Ahmedy, M. Y. I. Idris, and I. A. Targio Hashem, "A survey on underwater wireless sensor networks: Requirements, taxonomy, recent advances, and open research challenges," *Sensors*, vol. 20, no. 18, 2020.

[51] R. P. Narayanan, T. V. Sarath, and V. V. Vineeth, "Survey on motes used in wireless sensor networks: Performance & parametric analysis," *Wireless Sensor Network*, vol. 8, 04 2016.

[52] U. Kulau, F. Büsching, and L. Wolf, "Idealvolting: Reliable undervolting on wireless sensor nodes," *ACM Trans. Sen. Netw.*, vol. 12, no. 2, 2016.

[53] S. Gajjar, N. Choksi, M. Sarkar, and K. Dasgupta, "Comparative analysis of wireless sensor network motes," in *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, 2014.

[54] *TelosB Mote Platform*, Crossbow, document Part Number: 6020-0094-01 Rev B. [Online]. Available: https://www.willow.co.uk/TelosB_Datasheet.pdf

[55] E. Z. BTnode Project, "Btnodes - a distributed environment for prototyping ad hoc networks," Online, 2007, accessed on 2021-07-28. [Online]. Available: http://www.btnode.ethz.ch/

[56] *MPR-MIB Users Manual*, Crossbow Technology, Inc., June 2007, revision A, PN: 7430-0021-08. [Online]. Available: http://cpn.unl.edu/?q=system/files/devices/moteManual.pdf

[57] A. Burns, B. R. Greene, M. J. McGrath, T. J. O'Shea, B. Kuris, S. M. Ayer, F. Stroiescu, and V. Cionca, "Shimmer – a wireless sensor platform for noninvasive biomedical research," *IEEE Sensors Journal*, vol. 10, no. 9, pp. 1527–1534, 2010.

[58] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, "Openmote: Open-source prototyping platform for the industrial iot," in *Ad Hoc Networks*, N. Mitton, M. E. Kantarci, A. Gallais, and S. Papavassiliou, Eds. Cham: Springer International Publishing, 2015, pp. 211–222.

[59] *Waspmote v15 Datasheet*, Libelium Comunicaciones Distribuidas S.L., February 2019, v8.2. [Online]. Available: http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf

[60] A. Lignan, "Zolertia re-mote platform," Online, November 2016, accessed on 2021-07-28. [Online]. Available: https://github.com/Zolertia/Resources/wiki/RE-Mote

[61] WiSense Technologies, "Wisense wsn1120l datasheet," December 2019. [Online]. Available: https://wisense.in/wp-content/uploads/2019/12/WSN1120L_Datasheet.pdf

[62] "Open mote b user guide," Industrial Shields, August 2019, sKU: IS.OMB-001, Rev. 0: 22-08-2019. [Online]. Available: https://www.industrialshields.com/web/content?model=ir.attachment&field=datas&id=208800

[63] N. Madabhushi, "Kmote - design and implementation of a low cost, low power hardware platform for wireless sensor networks," mathesis, Indian Institute of Technology, Kanpur, 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.135.6395&rep=rep1&type=pdf

[64] A. Hoskins and J. McCann, "Beasties: Simple wireless sensor nodes," in *2008 33rd IEEE Conference on Local Computer Networks (LCN)*, 2008, pp. 707–714.

[65] F. Büsching, U. Kulau, and L. Wolf, "Architecture and evaluation of inga an inexpensive node for general applications," in *SENSORS, 2012 IEEE*, 2012, pp. 1–4.

[66] M. P. Andersen, G. Fierro, and D. E. Culler, "System design trade-offs in a next-generation embedded wireless platform," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-162, 2014.

[67] K. S. Raju and N. V. P. R. Pratap, "A reliable hardware platform for wireless sensor networks," in *2015 International Symposium on Advanced Computing and Communication (ISACC)*, 2015, pp. 310–314.

[68] M. Zeni, E. Ondula, R. Mbitiru, A. Nyambura, L. Samuel, K. Fleming, and K. Weldemariam, "Low-power low-cost wireless sensors for real-time plant stress detection," in *Proceedings of the 2015 Annual Symposium on Computing for Development (DEV '15)*. New York, NY, USA: Association for Computing Machinery, 2015.

[69] D. Berenguer, "panstamp wiki," Online, July 2018, accessed on 2021-07-28. [Online]. Available: https://github.com/panStamp/panstamp/wiki

[70] F. Karray, A. Garcia-Ortiz, M. W. Jmal, A. M. Obeid, and M. Abid, "Earnpipe: A testbed for smart water pipeline monitoring using wireless sensor network," *Procedia Computer Science*, vol. 96, pp. 285–294, 2016, knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 20th International Conference KES-2016.

[71] H. Sallouha, B. Van den Bergh, Q. Wang, and S. Pollin, "Ulora: Ultra low-power, low-cost and open platform for the lora networks," in *Proceedings of the 4th ACM Workshop on Hot Topics in Wireless*, ser. HotWireless '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 43–47.

[72] A. Rusu and P. Dobra, "The implementation of an arm-based low-power wireless process control system," in *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*, 2017, pp. 666–670.

[73] M. P. Andersen, H.-S. Kim, and D. E. Culler, "Hamilton: A cost-effective, low power networked sensor for indoor environment monitoring," in *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, ser. BuildSys '17. New York, NY, USA: Association for Computing Machinery, 2017.

[74] B. Pervan, E. Guberovic, and F. Turcinovic, "Hazelnut - an energy efficient base iot module for wide variety of sensing applications," in *Proceedings of the 6th Conference on the Engineering of Computer Based Systems*, ser. ECBS '19. New York, NY, USA: Association for Computing Machinery, 2019.

[75] D. Raposo, A. Rodrigues, S. Sinche, J. S. Silva, and F. Boavida, "Security and fault detection in in-node components of IIoT constrained devices," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, 2019.

[76] B. Babusiak, M. Smondrk, and S. Borik, "Design of ultra-low-energy temperature and humidity sensor based on nrf24 wireless technology," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019, pp. 397–401.

[77] S. Misra, S. K. Roy, A. Roy, M. S. Obaidat, and A. Jha, "Megan: Multipurpose energy-efficient, adaptable, and low-cost wireless sensor node for the internet of things," *IEEE Systems Journal*, vol. 14, no. 1, pp. 144–151, 2020.

[78] Z. Zhang, L. Shu, C. Zhu, and M. Mukherjee, "A short review on sleep scheduling mechanism in wireless sensor networks," in *Quality, Reliability, Security and Robustness in Heterogeneous Systems*, L. Wang, T. Qiu, and W. Zhao, Eds. Cham: Springer International Publishing, 2018, pp. 66–70.

[79] S. Tozlu and M. Senel, "Battery lifetime performance of Wi-Fi enabled sensors," in *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, 2012, pp. 429–433.

[80] A. Pughat and V. Sharma, "Optimal power and performance trade-offs for dynamic voltage scaling in power management based wireless sensor node," *Perspectives in Science*, vol. 8, pp. 536–539, 2016, recent Trends in Engineering and Material Sciences.

[81] D. Raposo, A. Rodrigues, S. Sinche, J. S. Silva, and F. Boavida, "Industrial IoT monitoring: Technologies and architecture proposal," *Sensors*, vol. 18, no. 10, October 2018.

[82] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug 2002.

[83] D. Estrin, D. Culler, K. Pister, and G. Sukhatme, "Connecting the physical world with pervasive networks," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 59–69, 2002.

[84] D. Ismail, M. Rahman, and A. Saifullah, "Low-power Wide-area Networks: Opportunities, Challenges, and Directions," in *Proceedings of the Workshop Program of the 19th International Conference on Distributed Computing and Networking*, ser. Workshops ICDCN '18. New York, NY, USA: ACM, 2018, pp. 8:1–8:6.

[85] H. M. Salmon, C. M. de Farias, P. Loureiro, L. Pirmez, S. Rossetto, P. H. de A. Rodrigues, R. Pirmez, F. C. Delicato, and L. F. R. da Costa Carmo, "Intrusion detection system for wireless sensor networks using danger theory immune-inspired techniques," *International Journal of Wireless Information Networks*, vol. 20, no. 1, pp. 39–66, Mar 2013.

[86] O. Can and O. K. Sahingoz, "A survey of intrusion detection systems in wireless sensor networks," in *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, May 2015, pp. 1–6.

[87] L. Paradis and Q. Han, "A survey of fault management in wireless sensor networks," *Journal of Network and Systems Management*, vol. 15, no. 2, pp. 171–190, Jun 2007.

[88] N. Savage, D. Ndzi, A. Seville, E. Vilar, and J. Austin, "Radio wave propagation through vegetation: Factors influencing signal attenuation," *Radio Science*, vol. 38, no. 5, 2003.

[89] J. Kim, P. Bentley, C. Wallenta, M. Ahmed, and S. Hailes, "Danger is ubiquitous: Detecting malicious activities in sensor networks using the dendritic cell algorithm," in *Artificial Immune Systems*, H. Bersini and J. Carneiro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 390–403.

[90] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 8, no. 4, pp. 48–63, Fourth 2006.

[91] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325 – 349, 2005.

[92] A. Ahmadi, M. Shojafar, S. F. Hajeforosh, M. Dehghan, and M. Singhal, "An efficient routing algorithm to preserve $$k$$k-coverage in wireless sensor networks," *J. Supercomput.*, vol. 68, no. 2, pp. 599–623, May 2014.

[93] M. O. Farooq and T. Kunz, "Operating systems for wireless sensor networks: A survey," *Sensors*, vol. 11, no. 6, pp. 5900–5930, 2011. [Online]. Available: https://www.mdpi.com/1424-8220/11/6/5900

[94] T. Vu Chien, H. Nguyen Chan, and T. Nguyen Huu, "A comparative study on operating system for wireless sensor networks," in *2011 International Conference on Advanced Computer Science and Information Systems*, Dec 2011, pp. 73–78.

[95] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Trans. on Sensor Networks*, vol. 6, no. 3, Jun. 2010.

[96] D. Hawkins, *Identification of Outliers*, ser. Monographs on applied probability and statistics. Chapman and Hall, 1980.

[97] F. González, "A Study of Artificial Immune Systems Applied to Anomaly Detection," Ph.D. dissertation, The University of Memphis, May 2003. [Online]. Available: http://dis.unal.edu.co/~fgonza/papers/gonzalez03study.pdf

[98] M. L. Shahreza, D. Moazzami, B. Moshiri, and M. Delavar, "Anomaly detection using a self-organizing map and particle swarm optimization," *Scientia Iranica*, vol. 18, no. 6, pp. 1460 – 1468, 2011.

[99] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, "Sensor network data fault types," *ACM Trans. Sen. Netw.*, vol. 5, no. 3, Jun. 2009.

[100] R. Jurdak, X. R. Wang, O. Obst, and P. Valencia, *Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 309–325.

[101] R. Roman, Jianying Zhou, and J. Lopez, "Applying intrusion detection systems to wireless sensor networks," in *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006.*, vol. 1, Jan 2006, pp. 640–644.

[102] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross, "Immune system approaches to intrusion detection – a review," *Natural Computing*, vol. 6, no. 4, pp. 413–466, Dec 2007.

[103] S. S. S. Sindhu, S. Geetha, and A. Kannan, "Decision tree based light weight intrusion detection using a wrapper approach," *Expert Systems with Applications*, vol. 39, no. 1, pp. 129 – 141, 2012.

[104] A. Garofalo, C. D. Sarno, and V. Formicola, "Enhancing intrusion detection in wireless sensor networks through decision trees," in *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, 2013, pp. 1–15.

[105] C. D. Sarno and A. Garofalo, "Energy-based detection of multi-layer flooding attacks on wireless sensor network," in *Lecture Notes in Computer Science.* Springer International Publishing, 2014, pp. 339–349.

[106] M. Becker, M. Drozda, S. Jaschke, and S. Schaust, "Comparing performance of misbehavior detection based on Neural Networks and AIS," in *2008 IEEE International Conference on Systems, Man and Cybernetics*, Oct 2008, pp. 757–762.

[107] H. Qu, Z. Qiu, X. Tang, M. Xiang, and P. Wang, "Incorporating unsupervised learning into intrusion detection for wireless sensor networks with structural co-evolvability," *Applied Soft Computing*, vol. 71, pp. 939 – 951, 2018.

[108] S. Shamshirband, N. B. Anuar, M. L. M. Kiah, V. A. Rohani, D. Petković, S. Misra, and A. N. Khan, "Co-FAIS: Cooperative fuzzy artificial immune system for detecting intrusion in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 42, pp. 102 – 117, 2014.

[109] C. Boano, H. Wennerström, M. Zuniga, J. Brown, C. Keppitiyagama, F. Oppermann, U. Roedig, L.-A. Norden, T. Voigt, and K. Römer, "Hot packets: A systematic evaluation of the effect of temperature on low power wireless transceivers," in *Extreme Conference on Communication.* United States: ACM, 2013.

[110] M. Bhuyan, D. Bhattacharyya, and J. Kalita, *Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools*, ser. Computer Communications and Networks. Springer International Publishing, 2017.

[111] B. Chander and G. Kumaravelan, "Outlier detection strategies for wsns: A survey," *Journal of King Saud University - Computer and Information Sciences*, 2021.

[112] F. J. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 445–453.

[113] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[114] S. Baddar, A. Merlo, and M. Migliardi, "Anomaly detection in computer networks: A state-of-the-art review," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 5, pp. 29–64, 12 2014.

[115] M. Xie, S. Han, B. Tian, and S. Parvin, "Anomaly detection in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1302 – 1325, 2011.

[116] J. F. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Transactions on Computers*, vol. C-29, no. 8, pp. 720–731, Aug 1980.

[117] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 12, no. 2, pp. 159–170, Second 2010.

[118] F. Y. Edgeworth, "On observations relating to several quantities," *Hermathena*, vol. 6, no. 13, pp. 279–285, 1887.

[119] V. Barnett, P. Barnett, and T. Lewis, *Outliers in Statistical Data*, ser. Wiley Series in Probability and Statistics. Wiley, 1994.

[120] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[121] M. Rassam, A. Zainal, and M. Maarof, "Advancements of data anomaly detection research in wireless sensor networks: A survey and open issues," *Sensors*, vol. 13, no. 8, pp. 10 087–10 122, Aug. 2013.

[122] G. Sebestyen, A. Hangan, Z. Czako, and G. Kovacs, "A taxonomy and platform for anomaly detection," in *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. IEEE, May 2018.

[123] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied Soft Computing*, vol. 10, no. 1, pp. 1–35, 2010.

[124] A. Amouri, S. Morgera, M. Bencherif, and R. Manthena, "A cross-layer, anomaly-based IDS for WSN and MANET," *Sensors*, vol. 18, no. 2, p. 651, Feb. 2018.

[125] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Computing Surveys*, vol. 47, no. 4, pp. 1–33, May 2015.

[126] J. H. Friedman, "Data mining and statistics: What's the connection," in *Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics*, 1997.

[127] Z. Ferdousi and A. Maeda, "Unsupervised outlier detection in time series data," in *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, April 2006, pp. x121–x121.

[128] N. Peng, W. Zhang, H. Ling, Y. Zhang, and L. Zheng, "Fault-tolerant anomaly detection method in wireless sensor networks," *Information*, vol. 9, no. 9, 2018.

[129] K. Kurniabudi, B. Purnama, S. Sharipuddin, D. Darmawijoyo, D. Stiawan, S. Samsuryadi, A. Heryanto, and R. Budiarto, "Network anomaly detection research: a survey," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 7, no. 1, Mar. 2019.

[130] N. A. Alrajeh and J. Lloret, "Intrusion Detection Systems Based on Artificial Intelligence Techniques in Wireless Sensor Networks," *International Journal of Distributed Sensor Networks*, vol. 9, no. 10, p. 351047, 2013.

[131] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Anomaly detection in wireless sensor networks," *IEEE Wireless Communications*, vol. 15, no. 4, pp. 34–40, Aug 2008.

[132] K. Burbeck and S. Nadjm-Tehrani, "Adaptive real-time anomaly detection with incremental clustering," *Information Security Technical Report*, vol. 12, no. 1, pp. 56 – 67, 2007.

[133] J. Zhang, C. Chen, Y. Xiang, and W. Zhou, "Semi-supervised and compound classification of network traffic," in *2012 32nd International Conference on Distributed Computing Systems Workshops*, June 2012, pp. 617–621.

[134] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016.

[135] A. Abraham, R. Falcon, and M. Koeppen, *Computational Intelligence in Wireless Sensor Networks: Recent Advances and Future Challenges*, ser. Studies in Computational Intelligence. Springer International Publishing, 2017.

[136] K. Shafi and H. A. Abbass, "Biologically-inspired Complex Adaptive Systems approaches to Network Intrusion Detection," *Information Security Technical Report*, vol. 12, no. 4, pp. 209 – 217, 2007.

[137] R. Kozik, M. Pawlicki, M. Choraś, and W. Pedrycz, "Practical employment of granular computing to complex application layer cyberattack detection," *Complexity*, vol. 2019, pp. 1–9, Jan. 2019.

[138] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," *Computer Networks*, vol. 136, pp. 37 – 50, 2018.

[139] M. Zamini and S. M. H. Hasheminejad, "A comprehensive survey of anomaly detection in banking, wireless sensor networks, social networks, and healthcare," *Intelligent Decision Technologies*, vol. 13, no. 2, pp. 229–270, May 2019.

[140] M. Usman, V. Muthukkumarasamy, X. Wu, and S. Khanum, *Mobile Agent-Based Anomaly Detection and Verification System for Smart Home Sensor Networks*. Springer Singapore, 2018.

[141] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys*, vol. 54, no. 2, mar 2021.

[142] C. Wei, S. Qiang, and X. Z. Gao, "Artificial endocrine system and applications," in *2006 Chinese Control Conference*, Aug 2006, pp. 1433–1437.

[143] S. Sinha and Z. Chaczko, "Concepts and observations in artificial endocrine systems for iot infrastructure," in *2017 25th International Conference on Systems Engineering (ICSEng)*, Aug 2017, pp. 427–430.

[144] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod, "Danger theory: The link between ais and ids?" in *Artificial Immune Systems*, J. Timmis, P. J. Bentley, and E. Hart, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 147–155.

[145] M. A. Alsheikh, S. Lin, D. Niyato, and H. Tan, "Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1996–2018, Fourthquarter 2014.

[146] Y. Zhang, N. Meratnia, and P. Havinga, *A taxonomy framework for unsupervised outlier detection techniques for multi-type data sets*, ser. CTIT Technical Report Series. Netherlands: Centre for Telematics and Information Technology (CTIT), 11 2007, no. Paper P-NS/TR-CTIT-07-79.

[147] D. P. Kumar, T. Amgoth, and C. S. R. Annavarapu, "Machine learning algorithms for wireless sensor networks: A survey," *Information Fusion*, vol. 49, pp. 1 – 25, 2019.

[148] R. Zhang and X. Xiao, "Intrusion detection in wireless sensor networks with an improved NSA based on space division," *Journal of Sensors*, vol. 2019, pp. 1–20, Apr. 2019.

[149] V. T. Alaparthy, A. Amouri, and S. D. Morgera, "A study on the adaptability of immune models for wireless sensor network security," *Procedia Computer Science*, vol. 145, 2018.

[150] S. Duhan and P. Khandnor, "Intrusion detection system in wireless sensor networks: A comprehensive review," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, March 2016.

[151] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 266–282, First 2014.

[152] C. O'Reilly, A. Gluhak, M. A. Imran, and S. Rajasegarar, "Anomaly detection in wireless sensor networks in a non-stationary environment," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1413–1432, Third 2014.

[153] A. Ghosal and S. Halder, "Intrusion detection in wireless sensor networks: Issues, challenges and approaches," in *Signals and Communication Technology*. Springer Berlin Heidelberg, 2013, pp. 329–367.

[154] T. H. Lim, "Detecting anomalies in wireless sensor networks," Ph.D. dissertation, University of York, Aug. 2010.

[155] A. H. Farooqi and F. A. Khan, "Intrusion detection systems for wireless sensor networks: A survey," in *Communication and Networking*. Springer Berlin Heidelberg, 2009, pp. 234–241.

[156] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Detecting data anomalies in wireless sensor networks," in *Security in Ad Hoc and Sensor Networks*. WORLD SCIENTIFIC, Sep. 2009, pp. 231–259.

[157] A. H. Dehwah, M. Mousa, and C. G. Claudel, "Lessons learned on solar powered wireless sensor network deployments in urban, desert environments," *Ad Hoc Netw.*, vol. 28, no. C, May 2015.

[158] G. Krivulya, I. Skarga-Bandurova, Z. Tatarchenko, O. Seredina, M. Shcherbakova, and E. Shcherbakov, "An intelligent functional diagnostics of wireless sensor network," in *2019 7th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, 2019.

[159] Z. Alansari, A. Prasanth, and M. R. Belgaum, "A comparison analysis of fault detection algorithms in wireless sensor networks," in *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2018.

[160] U. Kulau, D. Szafranski, and L. Wolf, "Effective but lightweight online selftest for energy-constrained WSNs," in *2018 IEEE 43rd Conference on Local Computer Networks Workshops (LCN Workshops)*, 2018, pp. 23–29.

[161] C. Titouna, M. Aliouat, and M. Gueroui, "FDS: Fault detection scheme for wireless sensor networks," *Wireless Personal Communications*, vol. 86, no. 2, Aug. 2015.

[162] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

[163] Y. Xu, I. Koren, and C. M. Krishna, "AdaFT: A framework for adaptive fault tolerance for cyber-physical systems," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 3, March 2017.

[164] T. Tomiyama and F. Moyen, "Resilient architecture for cyber-physical production systems," *CIRP Annals*, vol. 67, no. 1, pp. 161–164, 2018.

[165] E. Moridi, M. Haghparast, M. Hosseinzadeh, and S. J. Jassbi, "Fault management frameworks in wireless sensor networks: A survey," *Computer Communications*, vol. 155, 2020.

[166] I. Gerostathopoulos, D. Skoda, F. Plasil, T. Bures, and A. Knauss, "Architectural homeostasis in self-adaptive software-intensive cyber-physical systems," in *Software Architecture*, B. Tekinerdogan, U. Zdun, and A. Babar, Eds. Cham: Springer International Publishing, 2016, pp. 113–128.

[167] D. Hamdan, O. Aktouf, I. Parissis, A. Hijazi, M. Sarkis, and B. El Hassan, "Smart service for fault diagnosis in wireless sensor networks," in *2012 Sixth International Conference on Next Generation Mobile Applications, Services and Technologies*, 2012, pp. 211–216.

[168] B. W. Johnson, "Fault-tolerant microprocessor-based systems," *IEEE Micro*, vol. 4, no. 6, pp. 6–21, 1984.

[169] M. Z. A. Bhuiyan, G. Wang, J. Wu, J. Cao, X. Liu, and T. Wang, "Dependable structural health monitoring using wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 4, pp. 363–376, 2017.

[170] S. Marathe, A. Nambi, M. Swaminathan, and R. Sutaria, "CurrentSense: A novel approach for fault and drift detection in environmental IoT sensors," in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, ser. IoTDI '21. New York, NY, USA: ACM, 2021, pp. 93–105.

[171] J. Marzat, H. Piet-Lahanier, and S. Bertrand, "Cooperative fault detection and isolation in a surveillance sensor network: a case study," *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 790–797, 2018, 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018.

[172] G. Kakamanshadi, S. Gupta, and S. Singh, "A survey on fault tolerance techniques in wireless sensor networks," in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 2015.

[173] H. Wennerström, F. Hermans, O. Rensfelt, C. Rohner, and L. Nordén, "A long-term study of correlations between meteorological conditions and 802.15.4 link performance," in *2013 IEEE International Conference on Sensing, Communications and Networking (SECON)*, 2013.

[174] G. Xie, G. Zeng, J. An, R. Li, and K. Li, "Resource-cost-aware fault-tolerant design methodology for end-to-end functional safety computation on automotive cyber-physical systems," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 1, September 2018.

[175] T. Muhammed and R. A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 78, 2017.

[176] A. Ayadi, O. Ghorbel, A. M. Obeid, and M. Abid, "Outlier detection approaches for wireless sensor networks: A survey," *Computer Networks*, vol. 129, pp. 319–333, 2017.

[177] W. Li, L. Galluccio, F. Bassi, and M. Kieffer, "Distributed faulty node detection in delay tolerant networks: Design and analysis," *IEEE Transactions on Mobile Computing*, vol. 17, no. 4, pp. 831–844, 2018.

[178] B. R. Senapati, P. M. Khilar, and R. R. Swain, "Composite fault diagnosis methodology for urban vehicular ad hoc network," *Vehicular Communications*, vol. 29, 2021.

[179] R. R. Swain, T. Dash, and P. M. Khilar, "A complete diagnosis of faulty sensor modules in a wireless sensor network," *Ad Hoc Networks*, vol. 93, 2019.

[180] P. Yu, S. Jia, and P. Xi-yuan, "A self detection technique in fault management in wsn," in *2011 IEEE International Instrumentation and Measurement Technology Conference*, 2011, pp. 1–4.

[181] N. A. M. Alduais, J. Abdullah, A. Jamil, L. Audah, and R. Alias, "Sensor node data validation techniques for realtime iot/wsn application," in *2017 14th International Multi-Conference on Systems, Signals Devices (SSD)*, 2017, pp. 760–765.

[182] Q. yan Sun, Y. mei Sun, X. jiao Liu, Y. xin Xie, and X. guang Chen, "Study on fault diagnosis algorithm in WSN nodes based on RPCA model and SVDD for multi-class classification," *Cluster Computing*, vol. 22, no. S3, Jan. 2018.

[183] Y. Zhao, X. He, and D. Zhou, "Distributed fault source detection and topology accommodation design of wireless sensor networks," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017, pp. 5529–5534.

[184] Y. Gao, F. Xiao, J. Liu, and R. Wang, "Distributed soft fault detection for interval type-2 fuzzy-model-based stochastic systems with wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 334–347, 2019.

[185] B. Mali and S. H. Laskar, "Incipient fault detection of sensors used in wastewater treatment plants based on deep dropout neural network," *SN Applied Sciences*, vol. 2, no. 12, December 2020.

[186] R. Arunthavanathan, F. Khan, S. Ahmed, S. Imtiaz, and R. Rusli, "Fault detection and diagnosis in process system using artificial intelligence-based cognitive technique," *Computers & Chemical Engineering*, vol. 134, 2020.

[187] A. Theissler, "Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection," *Knowledge-Based Systems*, vol. 123, pp. 163–173, 2017.

[188] V. Joshi, O. Desai, and A. Kowli, "High accuracy sensor fault detection for energy management applications," in *2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, 2017, pp. 1–6.

[189] U. Saeed, S. U. Jan, Y.-D. Lee, and I. Koo, "Fault diagnosis based on extremely randomized trees in wireless sensor networks," *Reliability Engineering & System Safety*, vol. 205, 2021.

[190] S. Chessa and P. Santi, "Comparison-based system-level fault diagnosis in ad hoc networks," in *Proceedings 20th IEEE Symposium on Reliable Distributed Systems*, October 2001, pp. 257–266.

[191] K. Alshammari and A. E. S. Ahmed, "An efficient approach for detecting nodes failures in wireless sensor network based on clustering," in *2017 International Symposium on Networks, Computers and Communications (ISNCC)*, 2017, pp. 1–6.

[192] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 2, March 2005, pp. 902–913 vol. 2.

[193] L. Wang, X. Zhang, Y.-C. Tseng, and C.-K. Lin, "Parallel and local diagnostic algorithm for wireless sensor networks," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2017, pp. 334–337.

[194] H. H. Bosman, G. Iacca, A. Tejada, H. J. Wörtche, and A. Liotta, "Spatial anomaly detection in sensor networks using neighborhood information," *Information Fusion*, vol. 33, pp. 41–56, 2017.

[195] H. Zhang, Y. Jiang, X. Song, W. N. N. Hung, M. Gu, and J. Sun, "Sequential dependency and reliability analysis of embedded systems," in *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2013, pp. 423–428.

[196] Y. Ji, "Application of fault detection using distributed sensors in smart cities," *Physical Communication*, vol. 46, 2021.

[197] S. Mohapatra and P. M. Khilar, "Artificial immune system based fault diagnosis in large wireless sensor network topology," in *TENCON 2017 - 2017 IEEE Region 10 Conference*, 2017, pp. 2687–2692.

[198] P. Chanak and I. Banerjee, "Fuzzy rule-based faulty node classification and management scheme for large scale wireless sensor networks," *Expert Systems with Applications*, vol. 45, pp. 307 – 321, 2016.

[199] W. Elsayed, M. Elhoseny, A. M. Riad, and A. E. Hassanien, "Autonomic self-healing approach to eliminate hardware faults in wireless sensor networks," in *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017*. Springer International Publishing, August 2017, pp. 151–160.

[200] S. Das, P. Kar, and D. K. Jana, "SDH: Self detection and healing mechanism for dumb nodes in wireless sensor network," in *2016 IEEE Region 10 Conference (TENCON)*, 2016, pp. 2792–2795.

[201] C. Ioannou, V. Vassiliou, and C. Sergiou, "RMT: A wireless sensor network monitoring tool," in *Proceedings of the 13th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, ser. PE-WASUN '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 45–49.

[202] X. Liu, H. Zhou, S. Xiong, K. M. Hou, C. De Vaulx, and H. Shi, "Development of a resource-efficient and fault-tolerant wireless sensor network system," in *2015 2nd International Symposium on Dependable Computing and Internet of Things (DCIT)*, 2015, pp. 122–127.

[203] M. Burgess, "Computer immunology," in *Proceedings of the 12th USENIX Conference on System Administration*, ser. LISA '98. Berkeley, CA, USA: USENIX Association, 1998, pp. 283–298.

[204] A. Somayaji, S. Hofmeyr, and S. Forrest, "Principles of a computer immune system," in *Proceedings of the 1997 Workshop on New Security Paradigms*, ser. NSPW '97. New York, NY, USA: ACM, 1997, pp. 75–82.

[205] Lu Hong and Jing Yang, "Danger theory of immune systems and intrusion detection systems," in *2009 International Conference on Industrial Mechatronics and Automation*, May 2009, pp. 208–211.

[206] J. Twycross and U. Aickelin, "Information fusion in the immune system," *Information Fusion*, vol. 11, no. 1, pp. 35 – 44, 2010.

[207] M. Burgess, H. Haugerud, S. Straumsnes, and T. Reitan, "Measuring system normality," *ACM Trans. Comput. Syst.*, vol. 20, no. 2, pp. 125–160, May 2002.

[208] P. D'haeseleer, S. Forrest, and P. Helman, "An immunological approach to change detection: algorithms, analysis and implications," in *Proceedings 1996 IEEE Symposium on Security and Privacy*, May 1996, pp. 110–119.

[209] J. Greensmith, U. Aickelin, and J. Twycross, "Articulation and clarification of the dendritic cell algorithm," in *Artificial Immune Systems*, H. Bersini and J. Carneiro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[210] U. Aickelin, J. Greensmith, and J. Twycross, "Immune system approaches to intrusion detection – a review," in *Artificial Immune Systems*, G. Nicosia, V. Cutello, P. J. Bentley, and J. Timmis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 316–329.

[211] S. A. Hofmeyr and S. Forrest, "Immunity by design: An artificial immune system," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2*, ser. GECCO'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 1289–1296.

[212] S. Sarafijanovic and J. . Le Boudec, "An artificial immune system approach with secondary response for misbehavior detection in mobile ad hoc networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1076–1087, Sep. 2005.

[213] K. Eichmann, Ed., *The idiotypic network theory.* Basel: Birkhäuser Basel, 2008, pp. 82–94.

[214] P. Matzinger, "The danger model: A renewed sense of self," *Science*, vol. 296, 2002.

[215] U. Aickelin and S. Cayzer, "The danger theory and its application to artificial immune systems," *CoRR*, vol. abs/0801.3549, 2002.

[216] F. M. Burnet, "A modification of jerne's theory of antibody production using the concept of clonal selection," *CA: A Cancer Journal for Clinicians*, vol. 26, no. 2, pp. 119–121, Mar. 1976.

[217] F. R. Fekety, "The clonal selection theory of acquired immunity," *The Yale Journal of Biology and Medicine*, vol. 32, no. 6, pp. 480–480, Jun 1960, pMC2604340[pmcid].

[218] J. Oudin and P. A. Cazenave, "Similar idiotypic specificities in immunoglobulin fractions with different antibody functions or even without detectable antibody function," *Proceedings of the National Academy of Sciences*, vol. 68, no. 10, pp. 2616–2620, Oct. 1971.

[219] P. Bretscher and M. Cohn, "A theory of self-nonself discrimination: Paralysis and induction involve the recognition of one and two determinants on an antigen, respectively," *Science*, vol. 169, no. 3950, pp. 1042–1049, Sep. 1970.

[220] N. Jerne, "Towards a network theory of the immune system," *Annales d'immunologie*, vol. 125C, no. 1-2, p. 373—389, January 1974.

[221] R. Langman and M. Cohn, "The 'complete' idiotype network is an absurd immune system," *Immunology Today*, vol. 7, no. 4, pp. 100–101, Apr. 1986.

[222] K. Lafferty and A. Cunningham, "A new analysis of allogeneic interactions," *Australian Journal of Experimental Biology and Medical Science*, vol. 53, no. 1, pp. 27–42, Feb. 1975.

[223] C. Janeway, "Approaching the asymptote? evolution and revolution in immunology," *Cold Spring Harbor Symposia on Quantitative Biology*, vol. 54, no. 0, pp. 1–13, Jan. 1989.

[224] P. Matzinger, "Tolerance, danger, and the extended family," *Annu. Rev. Immunol.*, vol. 12, pp. 991–1045, 1994.

[225] T. R. Mosmann and A. M. Livingstone, "Dendritic cells: the immune information management experts," *Nature Immunology*, vol. 5, no. 6, pp. 564–566, Jun. 2004.

[226] J. Greensmith, U. Aickelin, and S. Cayzer, "Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection," in *Artificial Immune Systems*, C. Jacob, M. L. Pilat, P. J. Bentley, and J. I. Timmis, Eds. Springer Berlin Heidelberg, 2005.

[227] Q.-z. Xu and L. Wang, "Recent advances in the artificial endocrine system," *Journal of Zhejiang University SCIENCE C*, vol. 12, no. 3, pp. 171–183, Mar 2011.

[228] L. Sherwood, *Human Physiology: From Cells to Systems*. Cengage Learning, 2015.

[229] J. Neal, *How the Endocrine System Works*, ser. The How it Works Series. Wiley, 2016.

[230] Ihara and Mori, "Autonomous decentralized computer control systems," *Computer*, vol. 17, no. 8, pp. 57–66, Aug 1984.

[231] S. Miyamoto, K. Mori, H. Ihara, H. Matsumaru, and H. Ohshima, "Autonomous decentralized control and its application to the rapid transit system," *Computers in Industry*, vol. 5, no. 2, pp. 115 – 124, 1984, special Issue: Computers in Japanese Industry.

[232] K. Mori, "Autonomous decentralized systems technologies and their application to a train transport operation system," in *The Kluwer International Series in Engineering and Computer Science*. Springer US, 2001, pp. 89–111.

[233] W.-M. Shen and C.-M. Chuong, "The digital hormone model for self-organization," in *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior on From Animals to Animats*, ser. ICSAB. Cambridge, MA, USA: MIT Press, 2002, pp. 242–243.

[234] Wei-Min Shen, Cheng-Ming Chuong, and P. Will, "Simulating self-organization for multi-robot systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, Sep. 2002, pp. 2776–2781 vol.3.

[235] F. Heylighen, C. Gershenson, S. Staab, G. W. Flake, D. M. Pennock, D. C. Fain, D. De Roure, K. Aberer, Wei-Min Shen, O. Dousse, and P. Thiran, "Neurons, viscose fluids, freshwater polyp hydra-and self-organizing information systems," *IEEE Intelligent Systems*, vol. 18, no. 4, pp. 72–86, July 2003.

[236] E. Kravitz, "Hormonal control of behavior: amines and the biasing of behavioral output in lobsters," *Science*, vol. 241, no. 4874, pp. 1775–1781, Sep. 1988.

[237] R. A. Brooks, "Integrated systems based on behaviors," *SIGART Bull.*, vol. 2, no. 4, pp. 46–50, Jul. 1991.

[238] O. Avila-Garcia and L. Canamero, "Using hormonal feedback to modulate action selection in a competitive scenario," in *In From Animals to Animats: Proceedings of the 8th International Conference of Adaptive Behavior (SAB'04*. MIT Press, 2004, pp. 243–252.

[239] ——, "Hormonal modulation of perception in motivation-based action selection architectures," *Procs of the Symposium on Agents that Want and Like*, 01 2005.

[240] U. Brinkschulte, M. Pacher, and A. von Renteln, "An artificial hormone system for self-organizing real-time task allocation in organic middleware," in *Organic Computing*. Springer Berlin Heidelberg, 2009, pp. 261–283.

[241] A. von Renteln, U. Brinkschulte, and M. Pacher, "The artificial hormone system—an organic middleware for self-organising real-time task allocation," in *Organic Computing — A Paradigm Shift for Complex Systems*. Springer Basel, 2011, pp. 369–384.

[242] L. N. de Castro and J. I. Timmis, "Artificial immune systems as a novel soft computing paradigm," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 7, no. 8, pp. 526–544, Aug. 2003.

[243] D. Dasgupta, *Artificial Immune Systems and Their Applications*. Springer Berlin Heidelberg, 2012.

[244] Ki-Won Yeom and Ji-Hyung Park, "An artificial immune system model for multi agents based resource discovery in distributed environments," in *First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06)*, vol. 1, Aug 2006, pp. 234–239.

[245] R. A. Goldsby and R. A. K. i. Goldsby, *Immunology*, 5th ed. W.H. Freeman, 2003.

[246] C. A. Janeway, "How the immune system recognizes invaders," *Scientific American*, vol. 269, no. 3, pp. 72–79, Sep. 1993.

[247] C. A. Janeway and R. Medzhitov, "Innate immune recognition," *Annual Review of Immunology*, vol. 20, no. 1, pp. 197–216, Apr. 2002.

[248] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*, 4th ed. Garland Science, 2002.

[249] D. Dasgupta, S. Yu, and N. S. Majumdar, "Mila – multilevel immune learning algorithm and its application to anomaly detection," *Soft Computing*, vol. 9, no. 3, pp. 172–184, Mar 2005.

[250] J. Twycross and U. Aickelin, "Towards a conceptual framework for innate immunity," in *Artificial Immune Systems*, C. Jacob, M. L. Pilat, P. J. Bentley, and J. I. Timmis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 112–125.

[251] E. Vivier and B. Malissen, "Innate and adaptive immunity: specificities and signaling hierarchies revisited," *Nature Immunology*, vol. 6, no. 1, pp. 17–21, Jan. 2005.

[252] J. Kim and P. Bentley, "Immune memory and gene library evolution in the dynamic clonal selection algorithm," *Genetic Programming and Evolvable Machines*, vol. 5, no. 4, pp. 361–391, Dec. 2004.

[253] J. Greensmith, U. Aickelin, and G. Tedesco, "Information fusion for anomaly detection with the dendritic cell algorithm," *Information Fusion*, vol. 11, no. 1, pp. 21 – 34, 2010, special Issue on Biologically-Inspired Information Fusion.

[254] U. Aickelin and D. Dasgupta, *Artificial Immune Systems*. Boston, MA: Springer US, 2005, pp. 375–399.

[255] J. M. Vidal, A. L. S. Orozco, and L. J. G. Villalba, "Adaptive artificial immune networks for mitigating DoS flooding attacks," *Swarm and Evolutionary Computation*, vol. 38, pp. 94 – 108, 2018.

[256] P. J. Delves, S. J. Martin, D. R. Burton, and I. M. Roitt, *Roitt's Essential Immunology*, 13th ed., ser. Essentials. Wiley-Blackwell, 2017.

[257] S. Venkatesan, R. Baskaran, C. Chellappan, A. Vaish, and P. Dhavachelvan, "Artificial immune system based mobile agent platform protection," *Computer Standards & Interfaces*, vol. 35, no. 4, pp. 365–373, 2013.

[258] R. Coico and G. Sunshine, *Immunology: A Short Course*, 7th ed., ser. Coico, Immunology. Wiley-Blackwell, 2015.

[259] D. R. Green, N. Droin, and M. Pinkoski, "Activation-induced cell death in t cells," *Immunological Reviews*, vol. 193, no. 1, pp. 70–81, Jun. 2003.

[260] C. Jacob, S. Steil, and K. Bergmann, "The swarming body: Simulating the decentralized defenses of immunity," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 52–65.

[261] J. Punt, *Kuby Immunology*. W. H. Freeman, may 2018.

[262] J. Greensmith and U. Aickelin, "The deterministic dendritic cell algorithm," in *Proceedings of the 7th International Conference on Artificial Immune Systems*, ser. ICARIS '08. Springer Berlin Heidelberg, 2008.

[263] P. J. Bentley, J. Greensmith, and S. Ujjin, "Two ways to grow tissue for artificial immune systems," in *Artificial Immune Systems*, C. Jacob, M. L. Pilat, P. J. Bentley, and J. I. Timmis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 139–152.

[264] T. Pradeu and E. L. Cooper, "The danger theory: 20 years later," *Frontiers in Immunology*, vol. 3, 2012.

[265] P. Matzinger, "An innate sense of danger," *Seminars in Immunology*, vol. 10, no. 5, pp. 399 – 415, 1998.

[266] L. M. Sompayrac, *How the Immune System Works*. Wiley-Blackwell, 2019.

[267] S. Gallucci and P. Matzinger, "Danger signals: Sos to the immune system," *Current Opinion in Immunology*, vol. 13, no. 1, pp. 114 – 119, 2001.

[268] J. F. R. Kerr, C. M. Winterford, and B. V. Harmon, "Apoptosis. its significance in cancer and cancer therapy," *Cancer*, vol. 73, no. 8, pp. 2013–2026, Apr. 1994.

[269] R. M. Steinman, "Identification of a novel cell type in peripheral lymphoid organs of mice: I. morphology, quantitation, tissue distribution," *Journal of Experimental Medicine*, vol. 137, no. 5, pp. 1142–1162, May 1973.

[270] J. Greensmith, U. Aickelin, and S. Cayzer, *Detecting Danger: The Dendritic Cell Algorithm*. London: Springer London, 2008, pp. 89–112.

[271] M. L. Kapsenberg, "Dendritic-cell control of pathogen-driven t-cell polarization," *Nature Reviews Immunology*, vol. 3, no. 12, pp. 984–993, Dec. 2003.

[272] J. Kim, J. Greensmith, J. Twycross, and U. Aickelin, "Malicious code execution detection and response immune system inspired by the danger theory," *CoRR*, vol. abs/1003.4142, 2010.

[273] R. Medzhitov, "Decoding the patterns of self and nonself by the innate immune system," *Science*, vol. 296, no. 5566, pp. 298–300, Apr. 2002.

[274] J. Greensmith, "The dendritic cell algorithm," Ph.D. dissertation, University of Nottingham, Oct. 2007.

[275] D. Dasgupta, S. Yu, and F. Nino, "Recent Advances in Artificial Immune Systems: Models and Applications," *Applied Soft Computing*, vol. 11, no. 2, pp. 1574 – 1587, 2011.

[276] J.-Y. Le Boudec and S. Sarafijanović, "An artificial immune system approach to misbehavior detection in mobile ad hoc networks," in *Biologically Inspired Approaches to Advanced Information Technology*, A. J. Ijspeert, M. Murata, and N. Wakamiya, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 396–411.

[277] J. Timmis, A. Hone, T. Stibor, and E. Clark, "Theoretical advances in artificial immune systems," *Theoretical Computer Science*, vol. 403, no. 1, pp. 11 – 32, 2008.

[278] T. W. Mak, "Order from disorder sprung: recognition and regulation in the immune system," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 361, no. 1807, pp. 1235–1250, May 2003.

[279] M. Read, P. S. Andrews, and J. Timmis, *An Introduction to Artificial Immune Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1575–1597.

[280] S. Yu and D. Dasgupta, "Conserved self pattern recognition algorithm," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 279–290.

[281] E. Hart and J. Timmis, "Application areas of ais: The past, the present and the future," *Applied Soft Computing*, vol. 8, no. 1, pp. 191 – 201, 2008.

[282] J. Timmis, P. Andrews, N. Owens, and E. Clark, "An interdisciplinary perspective on artificial immune systems," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 5–26, Jan. 2008.

[283] S. Forrest and C. Beauchemin, "Computer immunology," *Immunological Reviews*, vol. 216, no. 1, pp. 176–197, Mar. 2007.

[284] D. Dasgupta, "Advances in artificial immune systems," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 40–49, Nov 2006.

[285] K. P. Anchor, P. D. Williams, G. H. Gunsch, and G. B. Lamont, "The computer defense immune system: current and future research in intrusion detection," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 2, May 2002, pp. 1027–1032 vol.2.

[286] P.-C. Chang, W.-H. Huang, and C.-J. Ting, "A hybrid genetic-immune algorithm with improved lifespan and elite antigen for flow-shop scheduling problems," *International Journal of Production Research*, vol. 49, no. 17, pp. 5207–5230, Sep. 2011.

[287] C. A. C. Coello and N. C. Cortes, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 163–190, Jun. 2005.

[288] X. Luo and W. Wei, "A new immune genetic algorithm and its application in redundant manipulator path planning," *Journal of Robotic Systems*, vol. 21, no. 3, pp. 141–151, 2004.

[289] A. Graaff and A. Engelbrecht, "Optimised coverage of non-self with evolved lymphocytes in an artificial immune system," *International Journal of Computational Intelligence Research Research India Publications*, vol. 2, pp. 973–1873, 01 2006.

[290] J. Kim and P. Bentley, "Negative selection and niching by an artificial immune system for network intrusion detection," in *In Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, 1999, pp. 149–158.

[291] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1994, pp. 202–212.

[292] M. Cohn, N. A. Mitchison, W. E. Paul, A. M. Silverstein, D. W. Talmage, and M. Weigert, "Reflections on the clonal-selection theory," *Nature Reviews Immunology*, vol. 7, no. 10, pp. 823–830, Oct. 2007.

[293] G. Costa Silva and D. Dasgupta, *Handbook on Computational Intelligence*. World Scientific, 03 2016, vol. 2, ch. A Survey of Recent Works in Artificial Immune Systems, pp. 547–586.

[294] D. Dasgupta and F. Gonzalez, "An immunity-based technique to characterize intrusions in computer networks," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 281–291, June 2002.

[295] P. Mostardinha, B. F. Faria, A. Zúquete, and F. V. de Abreu, "A negative selection approach to intrusion detection," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 178–190.

[296] J. Greensmith, J. Feyereisl, and U. Aickelin, "The dca: Some comparison: A comparative study between two biologically-inspired algorithms," *CoRR*, vol. abs/1006.1518, 2008.

[297] M. Ayara, J. Timmis, R. Lemos, L. De Castro, and R. Duncan, "Negative selection: How to generate detectors," *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, 01 2002.

[298] Lu Hong, "Artificial Immune System for Anomaly Detection," in *2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop*, Dec 2008, pp. 340–343.

[299] J. Balthrop, F. Esponda, S. Forrest, and M. Glickman, "Coverage and generalization in an artificial immune system," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO'02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 3–10.

[300] J. M. Shapiro, G. B. Lamont, and G. L. Peterson, "An evolutionary algorithm to generate hyper-ellipsoid detectors for negative selection," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '05. New York, NY, USA: ACM, 2005, pp. 337–344.

[301] Jungwon Kim and P. J. Bentley, "Towards an artificial immune system for network intrusion detection: an investigation of clonal selection with a negative selection operator," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 2, May 2001, pp. 1244–1252 vol. 2.

[302] Zhou Ji and D. Dasgupata, "Augmented negative selection algorithm with variable-coverage detectors," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 1, June 2004, pp. 1081–1088 Vol.1.

[303] Z. Ji and D. Dasgupta, "Real-valued negative selection algorithm with variable-sized detectors," in *Genetic and Evolutionary Computation – GECCO 2004*. Springer Berlin Heidelberg, 2004, pp. 287–298.

[304] J. Timmis, P. Andrews, and E. Hart, "On artificial immune systems and swarm intelligence," *Swarm Intelligence*, vol. 4, no. 4, pp. 247–273, Dec 2010.

[305] N. Nanas, V. S. Uren, and A. de Roeck, "Nootropia: A user profiling model based on a self-organising term network," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 146–160.

[306] C. McEwan and E. Hart, "Representation in the (artificial) immune system," *Journal of Mathematical Modelling and Algorithms*, vol. 8, no. 2, pp. 125–149, Jun 2009.

[307] F. González, D. Dasgupta, and J. Gómez, "The effect of binary matching rules in negative selection," in *Genetic and Evolutionary Computation — GECCO 2003*. Springer Berlin Heidelberg, 2003, pp. 195–206.

[308] P. K. Harmer, P. D. Williams, G. H. Gunsch, and G. B. Lamont, "An artificial immune system architecture for computer security applications," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 252–280, June 2002.

[309] J. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," *Physica D: Nonlinear Phenomena*, vol. 22, no. 1, pp. 187 – 204, 1986, proceedings of the Fifth Annual International Conference.

[310] J. Kim and P. J. Bentley, "An evaluation of negative selection in an artificial immune system for network intrusion detection," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO*. Morgan Kaufmann, 2001, pp. 1330–1337.

[311] J. Balthrop, S. Forrest, and M. Glickman, "Revisiting LISYS: parameters and normal behavior," in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*. IEEE, 2002.

[312] S. A. Hofmeyr, "An immunological model of distributed detection and its application to computer security," Ph.D. dissertation, The University of New Mexico, 1999, aAI9926862.

[313] X. Z. Gao, S. J. Ovaska, and X. Wang, "Genetic algorithms-based detector generation in negative selection algorithm," in *2006 IEEE Mountain Workshop on Adaptive and Learning Systems*, July 2006, pp. 133–137.

[314] X. Z. Gao, S. J. Ovaska, X. Wang, and M. . Chow, "Clonal optimization of negative selection algorithm with applications in motor fault detection," in *2006 IEEE International Conference on Systems, Man and Cybernetics*, vol. 6, Oct 2006, pp. 5118–5123.

[315] S. Cayzer and J. Smith, "Gene libraries: Coverage, efficiency and diversity," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 136–149.

[316] J. Gomez, F. Gonzalez, and D. Dasgupta, "An immuno-fuzzy approach to anomaly detection," in *The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03.*, vol. 2, May 2003, pp. 1219–1224 vol.2.

[317] F. Gonzalez, J. Gomez, Madhavi kaniganti, and Dipankar Dasgupta, "An evolutionary approach to generate fuzzy anomaly (attack) signatures," in *IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop, 2003.*, June 2003, pp. 251–259.

[318] F. Esponda, S. Forrest, and P. Helman, "A formal framework for positive and negative detection schemes," *Trans. Sys. Man Cyber. Part B*, vol. 34, no. 1, pp. 357–373, Feb. 2004.

[319] X. Hang and H. Dai, "Applying both positive and negative selection to supervised learning for anomaly detection," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '05. New York, NY, USA: ACM, 2005, pp. 345–352.

[320] P. A. D. de Castro and F. J. V. Zuben, "Bais: A bayesian artificial immune system for the effective handling of building blocks," *Information Sciences*, vol. 179, no. 10, pp. 1426 – 1440, 2009, including Special Issue on Artificial Imune Systems.

[321] P. A. D. Castro and F. J. V. Zuben, "MOBAIS: A bayesian artificial immune system for multi-objective optimization," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 48–59.

[322] W. Wang, S. Gao, and Z. Tang, "A complex artificial immune system," in *2008 Fourth International Conference on Natural Computation*, vol. 6, Oct 2008, pp. 597–601.

[323] D. Dasgupta and S. Forrest, "An anomaly entection algorithm inspired by the immune syste," in *Artificial Immune Systems and Their Applications*. Springer Berlin Heidelberg, 1999, pp. 262–277.

[324] A. M. Tyrell, "Computer know thy self!: a biological way to look at fault-tolerance," in *Proceedings 25th EUROMICRO Conference. Informatics: Theory and Practice for the New Millennium*, vol. 2, Sep. 1999, pp. 129–135 vol.2.

[325] C. A. Coello Coello and N. Cruz Cortes, "A parallel implementation of an artificial immune system to handle constraints in genetic algorithms: preliminary results," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 1, May 2002, pp. 819–824 vol.1.

[326] S. Liu, T. Li, D. Wang, K. Zhao, X. Gong, X. Hu, C. Xu, and G. Liang, "Immune multi-agent active defense model for network intrusion," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 104–111.

[327] D. Dasgupta, "Immunity-based intrusion detection system: A general framework," The University of Memphis, Tech. Rep., 1999.

[328] Z. Ji and D. Dasgupta, "Revisiting negative selection algorithms," *Evolutionary Computation*, vol. 15, no. 2, pp. 223–251, Jun. 2007.

[329] L. N. D. Castro and F. J. V. Zuben, "The clonal selection algorithm with engineering applications," in *In GECCO 2002 - Workshop Proceedings*. Morgan Kaufmann, 2002, pp. 36–37.

[330] A. Watkins and J. Timmis, "Exploiting parallelism inherent in airs, an artificial immune classifier," in *Artificial Immune Systems*, G. Nicosia, V. Cutello, P. J. Bentley, and J. Timmis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 427–438.

[331] A. Ciccazzo, P. Conca, G. Nicosia, and G. Stracquadanio, "An advanced clonal selection algorithm with ad-hoc network-based hypermutation operators for synthesis of topology and sizing of analog electrical circuits," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 60–70.

[332] J. Timmis and M. Neal, "A resource limited artificial immune system for data analysis," *Knowledge-Based Systems*, vol. 14, no. 3, pp. 121 – 130, 2001.

[333] A. Watkins, J. Timmis, and L. Boggess, "Artificial immune recognition system (airs): An immune-inspired supervised learning algorithm," *Genetic Programming and Evolvable Machines*, vol. 5, no. 3, pp. 291–317, Sep. 2004.

[334] D. E. Goodman, L. Boggess, and A. Watkins, "An investigation into the source of power for airs, an artificial immune classification system," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3, July 2003, pp. 1678–1683 vol.3.

[335] L. Fang, Q. Bo, and C. Rongsheng, "Intrusion detection based on immune clonal selection algorithms," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 1226–1232.

[336] Y. Ishida, "Fully distributed diagnosis by pdp learning algorithm: towards immune network pdp model," in *1990 IJCNN International Joint Conference on Neural Networks*, June 1990, pp. 777–782 vol.1.

[337] J. E. Hunt and D. E. Cooke, "Learning using an artificial immune system," *Journal of Network and Computer Applications*, vol. 19, no. 2, pp. 189 – 212, 1996.

[338] L. N. de Castro and F. J. V. Zuben, "aiNet: An artificial immune network for data analysis," in *Data Mining*. IGI Global, 2001, pp. 231–260.

[339] C. Zhang and Z. Yi, "An artificial immune network model applied to data clustering and classification," in *Advances in Neural Networks – ISNN 2007*. Springer Berlin Heidelberg, 2007, pp. 526–533.

[340] J. Timmis, M. Neal, and J. Hunt, "An artificial immune system for data analysis," *Biosystems*, vol. 55, no. 1-3, pp. 143–150, Feb. 2000.

[341] J. Greensmith, J. Twycross, and U. Aickelin, "Dendritic cells for anomaly detection," *2006 IEEE International Conference on Evolutionary Computation*, 2006.

[342] J. Twycross, "Integrated innate and adaptive artificial immune systems applied to process anomaly detection," Ph.D. dissertation, University of Nottingham, Jan. 2007.

[343] U. Aickelin and J. Greensmith, "Sensing danger: Innate immunology for intrusion detection," *Information Security Technical Report*, vol. 12, no. 4, pp. 218–227, 2007.

[344] J. Greensmith and U. Aickelin, "Dendritic cells for real-time anomaly detection," *SSRN Electronic Journal*, 2006.

[345] J. Greensmith, "Migration threshold tuning in the deterministic dendritic cell algorithm," in *8th International Conference on the Theory and Practice of Natural Computing (TPNC'19)*, vol. 2, 2019.

[346] C.-M. Ou, C. R. Ou, and Y.-T. Wang, *Agent-Based Artificial Immune Systems (ABAIS) for Intrusion Detections: Inspiration from Danger Theory.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 67–94.

[347] R. Chao and Y. Tan, "A virus detection system based on artificial immune system," in *2009 International Conference on Computational Intelligence and Security*, vol. 1, Dec 2009, pp. 6–10.

[348] Y. Tan, G. Mi, Y. Zhu, and C. Deng, "Artificial immune system based methods for spam filtering," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2013, pp. 2484–2488.

[349] M. F. A. Gadi, X. Wang, and A. P. do Lago, "Credit card fraud detection with artificial immune system," in *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, 2008, pp. 119–131.

[350] D. Dasgupta, "An overview of artificial immune systems and their applications," in *Artificial Immune Systems and Their Applications.* Springer Berlin Heidelberg, 1993, pp. 3–21.

[351] J. Kim and P. Bentley, "An artificial immune model for network intrusion detection," *Conference on Intelligent Techniques and Soft Computing (EUFIT'99)*, 10 1999.

[352] A. Boukerche, R. B. Machado, K. R. Jucá, J. B. M. Sobral, and M. S. Notare, "An agent based and biological inspired real-time intrusion detection and security model for computer network operations," *Computer Communications*, vol. 30, no. 13, pp. 2649 – 2660, 2007, sensor-Actuated Networks.

[353] S. T. Powers and J. He, "A hybrid artificial immune system and self organising map for network intrusion detection," *Information Sciences*, vol. 178, no. 15, pp. 3024 – 3042, 2008, nature Inspired Problem-Solving.

[354] M. Drozda, S. Schaust, and H. Szczerbicka, "AIS for misbehavior detection in wireless sensor networks: Performance and design principles," in *2007 IEEE Congress on Evolutionary Computation*, Sep. 2007, pp. 3719–3726.

[355] Yang Liu, Yang Liu, and Fengqi Yu, "Immunity-based intrusion detection for wireless sensor networks," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, June 2008, pp. 439–444.

[356] L. Fasanotti, E. Dovere, E. Cagnoni, and S. Cavalieri, "An Application of Artificial Immune System in a Wastewater Treatment Plant," *IFAC-PapersOnLine*, vol. 49, no. 28, pp. 55 – 60, 2016, 3rd IFAC Workshop on Advanced Maintenance Engineering, Services and Technology AMEST 2016.

[357] M. Gong, L. Jiao, W. Ma, and J. Ma, "Intelligent multi-user detection using an artificial immune system," *Science in China Series F: Information Sciences*, vol. 52, no. 12, pp. 2342–2353, Dec. 2009.

[358] C. Laurentys, R. Palhares, and W. Caminhas, "A novel artificial immune system for fault behavior detection," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6957 – 6966, 2011.

[359] M. Zuccolotto, C. E. Pereira, L. Fasanotti, S. Cavalieri, and J. Lee, "Designing an artificial immune systems for intelligent maintenance systems," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 1451 – 1456, 2015, 15th IFAC Symposium onInformation Control Problems inManufacturing.

[360] H. Yang, M. Elhadef, A. Nayak, and X. Yang, "Network fault diagnosis: An artificial immune system approach," in *2008 14th IEEE International Conference on Parallel and Distributed Systems*, Dec 2008, pp. 463–469.

[361] D. W. Bradley and A. M. Tyrrell, "The architecture for a hardware immune system," in *Proceedings Third NASA/DoD Workshop on Evolvable Hardware. EH-2001*, July 2001, pp. 193–200.

[362] M. Kayama, Y. Sugita, Y. Morooka, and S. Fukuoka, "Distributed diagnosis system combining the immune network and learning vector quantization," in *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, vol. 2, Nov 1995, pp. 1531–1536 vol.2.

[363] W. Liu and B. Chen, "Optimal control of mobile monitoring agents in immune-inspired wireless monitoring networks," *Journal of Network and Computer Applications*, vol. 34, no. 6, pp. 1818–1826, 2011.

[364] R. Pinto, G. Gonçalves, E. Tovar, and J. Delsing, "Attack detection in cyber-physical production systems using the deterministic dendritic cell algorithm," in *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020.

[365] R. Oates, G. Kendall, and J. M. Garibaldi, "Frequency analysis for dendritic cell population tuning," *Evolutionary Intelligence*, vol. 1, no. 2, pp. 145–157, Apr. 2008.

[366] F. Gu, J. Greensmith, and U. Aickelin, "Integrating real-time analysis with the dendritic cell algorithm through segmentation," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '09. New York, NY, USA: Association for Computing Machinery, 2009.

[367] C. J. Musselle, "Insights into the antigen sampling component of the dendritic cell algorithm," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010.

[368] Z. Chelly and Z. Elouedi, "A survey of the dendritic cell algorithm," *Knowledge and Information Systems*, vol. 48, no. 3, 2015.

[369] S. Mohapatra and P. M. Khilar, *Immune Inspired Fault Diagnosis in Wireless Sensor Network*. Springer Singapore, 2020, ch. 5.

[370] R. Rizwan, F. A. Khan, H. Abbas, and S. H. Chauhdary, "Anomaly detection in wireless sensor networks using immune-based bioinspired mechanism," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, 2015.

[371] Z. Zhang, A. Mehmood, L. Shu, Z. Huo, Y. Zhang, and M. Mukherjee, "A survey on fault diagnosis in wireless sensor networks," *IEEE Access*, vol. 6, 2018.

[372] D. Cui, Q. Zhang, J. Xiong, Q. Li, and M. Liu, "Fault diagnosis research of rotating machinery based on dendritic cell algorithm," in *IEEE International Conference on Information and Automation*, 2015.

[373] E. Alizadeh, N. Meskin, and K. Khorasani, "A dendritic cell immune system inspired scheme for sensor fault detection and isolation of wind turbines," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, 2018.

[374] M. Akram and A. Raza, "Towards the development of robot immune system: A combined approach involving innate immune cells and t-lymphocytes," *Biosystems*, vol. 172, 2018.

[375] N. Elisa, L. Yang, X. Fu, and N. Naik, "Dendritic cell algorithm enhancement using fuzzy inference system for network intrusion detection," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2019.

[376] J. Borgeson, S. Schauer, and H. Diewald, "Benchmarking MCU power consumption for ultra-low-power applications," Texas Instruments, White Paper E010208, Nov. 2012.

[377] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," in *Wireless Sensor Networks*, H. Karl, A. Wolisz, and A. Willig, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 307–322.

[378] D. Yuan, S. S. Kanhere, and M. Hollick, "Instrumenting wireless sensor networks – a survey on the metrics that matter," *Pervasive and Mobile Computing*, vol. 37, 2017.

[379] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "Madness: A multi-layer anomaly detection framework for complex dynamic systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 796–809, 2021.

[380] Microchip Technology Inc., *AN2447: Measure VCC/Battery Voltage Without Using I/O Pin on tinyAVR and megaAVR*, May 2019, Application Note.

[381] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[382] D. E. Knuth, *The Art Of Computer Programming, Volume 2: Seminumerical Algorithms*. Pearson Education, 1998.

[383] *ATmega1284P Datasheet*, Atmel Corporation, November 2009, 8059D–AVR–11/09. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/doc8059.pdf

[384] F. Angiulli and F. Fassetti, "Detecting distance-based outliers in streams of data," in *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, ser. CIKM '07. New York, NY, USA: Association for Computing Machinery, 2007, pp. 811–820. [Online]. Available: https://doi.org/10.1145/1321440.1321552

[385] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013, 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016314999

[386] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 2712–2721.

[387] S. F. Yilmaz and S. S. Kozat, "Pysad: A streaming anomaly detection framework in python," *CoRR*, vol. abs/2009.02572, 2020. [Online]. Available: https://arxiv.org/abs/2009.02572

[388] H. Schutte, "Bi-directional level shifter for i2c-bus and other systems," Philips Semiconductors, Application Note AN97055, 1997.

[389] S. Khan, A.-S. K. Pathan, and N. A. Alrajeh, *Wireless sensor networks: current status and future trends*. Taylor & Francis, 2013.

Dominik Widhalm

Curriculum Vitae

Education

2018 – 2022 **PhD Computer Science**, *Vienna University of Technology*.
participant in the doctoral college *Resilient Embedded Systems*

2012 – 2014 **Master Embedded Systems**, *UAS Technikum Wien*.
passed with highest distinction

2010 – 2012 **Bachelor Electronic Engineering**, *UAS Technikum Wien*.
passed with highest distinction

2004 – 2009 **Electrical Engineering/Information Technology**, *HTBLuVA St. Pölten*.
passed with highest distinction

Doctoral Thesis

Title Sensor Node Fault Detection in Wireless Sensor Networks: An Immune-inspired Approach

Supervisors Priv.-Doz. Mag. DI. DI. Dr. Karl M. Göschka
Ao.Univ.Prof. DI. Dr. Wolfgang Kastner

Ext. Reviewer Prof. Andrea Bondavalli
Prof. Davide Quaglia

Description In this thesis, a novel sensor node fault detection approach is presented. It integrates node-level diagnostics with the characteristics of the sensor data to improve the detectability of faults and, more importantly, to allow to distinguish between the effect of faults and environmental data events. The approach is inspired by the functioning of dendritic cells in the human immune system. Nevertheless, the strength of the presented approach lies in expressive node-level diagnostics rather than deeply embedding immune-related knowledge.

Master Thesis

Title *Bridging the Gap between AGENtiX and JAZZ*

Supervisors FH-Prof. DI. Dr. Martin Horauer
DI. Mag. Matthias Wenzl

Description This thesis describes the development of a novel testing framework to bridge pre- and post-silicon verification activities in mixed-signal SoC development flows. Its suitability is shown based on the verification flow of Infineon Technologies Austria AG where two main software frameworks are used, namely AGENtiX and JAZZ; hence the name of the thesis.

Achievements

2014 Kapsch Award 2014

2013 & 2014 Merit-based scholarship, Master Embedded Systems

2011 & 2012 Merit-based scholarship, Bachelor Electronic Engineering

2009 OVE GIT-Preis 2009

Experience

Vocational

2018/10 – today **Researcher**, *UAS Technikum Wien*, Austria.
Doctoral College Resilient Embedded Systems

2017/07 – 2018/10 **Researcher**, *UAS Technikum Wien*, Austria.
In a research cooperation with Elektrobit Austria GmbH

2013/04 – 2018/04 **Researcher**, *UAS Technikum Wien*, Austria.
In the Josef Ressel Center for Verification of Embedded Computing Systems (funded by CDG)

2012/09 – 2012/12 **Junior Researcher**, *UAS Technikum Wien*, Austria.
In the AC-Centrope II research project (funded by EU-EFRE)

2012/03 – 2012/05 **Electronic Engineer**, *Zizala Lichtsysteme GmbH*, Wieselburg, Austria.
Student apprentice

Teaching Academic Courses

since 2019 **Embedded Systems Software Design**, bachelor level course.

since 2017 **Wireless Communication Networks & Systems**, bachelor level course.

since 2016 **Internet of Things Applications**, *(specialization course)*, bachelor level course.

since 2016 **Embedded Software Testing**, master level course.

since 2014 **Supervisor of several bachelor/master projects & theses**.

since 2013 **C & System Programming**, bachelor level course.

Publications

Journal Publications

[1] D. Widhalm, K. M. Goeschka, and W. Kastner, "An open-source wireless sensor node platform with active node-level reliability for monitoring applications," *Sensors*, vol. 21, no. 22, 2021.

[2] M. Horauer, D. Widhalm, S. Tauner, and S. Mirtl, "Verification Challenges of Complex System-on-Chip Devices," *e & i Elektrotechnik und Informationstechnik*, vol. 132, no. 6, pp. 269–273, 2015.

Conference Publications

[3] D. Widhalm, K. M. Goeschka, and W. Kastner, "Sensor node fault detection in wireless sensor networks utilizing node-level diagnostics," in *The 20th ACM Conference on Embedded Networked Sensor Systems (SenSys'22)*, (in review), Boston, United States, 2022.

[4] ——, "Undervolting on wireless sensor nodes: A critical perspective," in *2022 23rd International Conference on Distributed Computing and Networking (ICDCN)*, 2022.

[5] ——, "Is arduino a suitable platform for sensor nodes?" In *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–6.

[6] ——, "Node-level indicators of soft faults in wireless sensor networks," in *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, 2021, pp. 13–22.

[7] ——, "Sok: A taxonomy for anomaly detection in wireless sensor networks focused on node-level techniques," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES '20, Virtual Event, Ireland: Association for Computing Machinery, 2020.

[8] D. Widhalm, S. Tauner, and M. Horauer, "Augmenting Pre-Silicon Simulation by embedding a Scripting Language in a SystemC Environment," in *Mechatronic and Embedded Systems and Applications (MESA), 2016 IEEE/ASME 12th International Conference on*, Aug. 2016.

[9] S. Tauner, D. Widhalm, and Horauer, "Synchronization Approaches for Testing Mixed-Signal SoCs under Real-Time Constraints using On-Chip Capabilities," in *Proceedings of the 2015 IEEE Austrian Workshop on Microelectronics (AUSTROCHIP)*, Sep. 2015, pp. 36–41.

[10] D. Widhalm, S. Tauner, M. Horauer, A. Schumacher, and A. Haggenmiller, "A Common Platform for Bridging Pre- and Post-Silicon Verification in Mixed-Signal Designs," in *Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International*, May 2015, pp. 1584–1589.

Conference Poster Presentations

[11] S. Tauner, D. Widhalm, and M. Horauer, *Unification of Pre- and Post-Silicon Verification Flows in Mixed-Signal Designs*, Microelectronic Systems Symposium (MESS'16), Apr. 2016.

Theses

[12] D. Widhalm, "Sensor node fault detection in wireless sensor networks: An immune-inspired approach," PhD thesis, Vienna University of Technology, Doctoral College Resilient Embedded Systems, Sep. 2022.

[13] ——, "Bridging the gap between agentix and jazz," Master's thesis, University of Applied Sciences Technikum Wien, Jun. 2014.

SW/HW Engineering Skills

Languages	C/C++, Assembly (ARM/AVR), Python, Perl, Java, VHDL, SystemC, PHP, AJAX, Javascript, HTML, CSS, LaTeX, AsciiDoc, Markdown
Software	Code Blocks, Eclipse, AVR/Atmel Studio, Visual Studio, Matlab/Simulink, LabView, Altera Quartus & Modelsim, Protel/Altium, Proteus, Mathcad, AutoCAD, ePlan, KiCad

Languages

German	native
English	excellent command
Russian	basic communication skills
Std. Chinese	basic communication skills

Interests

Research	sensor networks, fault tolerance, anomaly detection, artificial immune systems
Electronics	embedded systems, internet of things applications, home automation
Activities	running, swimming, hiking, sport climbing, fitness training, ballroom dancing