Autonomous Fault Detection and Diagnosis in Wireless Sensor Networks using Decision Trees

Angeliki Laiou[1], Christina M. Malliou[1], Sotirios-Angelos Lenas[1]and Vassilis Tsaoussidis[1]
[1] Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi 67100, Greece
Email: laios.angela@gmail.com; {cmalliou, slenas, vtsaousi}@ee.duth.gr

Abstract —Reliable communication in wireless sensor networks constitutes an essential factor in maintaining critical systems operational. Despite this, wireless sensor networks are known to be volatile and prone to faults disrupting their normal working state. Particularly in open environments, wireless sensor networks must be able to detect arising faults to minimize subsequent failures of the network. This study deals with the detection and identification of faults in wireless sensor networks, notably faults that occur due to externally driven events, affecting network services, such as data transfers and communication between nodes. Faults commonly occurring due to such factors are loss of connectivity because of faulty node interfaces, disrupted connectivity due to obstacles, and extreme packet loss because of increased noise conditions or congestion. Detection, identification, and recovery of sensor network faults have been studied extensively in the literature. In this paper, a Machine Learning approach is used to detect and diagnose these faults. A decision tree algorithm was used to train the model. The produced model is consistently able to identify the faults on test data with an overall accuracy of 96.46%. Results also include high precision and recall values for each separate fault case, thus producing a successful fault identification model.

Index Terms—autonomous networks, decision tree, fault diagnosis, wireless sensor networks

I. INTRODUCTION

With the advancement of Wireless Sensor Networking (WSN) many systems, including IoT devices being increasingly used in recent years, rely on the information collected from small sensor nodes. While the advantages of WSN are many, the fact remains that the individual nodes of a network are extremely sensitive and frequently exhibit faults. These faults disrupt the normal working state of the network on various levels; from the actual nodes, to corrupted data (collected or produced), as well as faults impairing the network's services. The research to minimize such faults has been pursued extensively in the literature covering the multitude and diversity of the faults. The work presented here accounts for faults regarding the disruption of the communication in the network. Specifically targeting faults due to external events, i.e. events that the user does not have control over. Such faults are:

1) The loss of connectivity between nodes, either randomly due to obstacles or complete loss of connectivity because of a faulty interface.
2) Extreme packet loss caused by:
   a) Increased noise due to extreme weather conditions.
   b) Disturbances in the network services, for instance, sensor nodes taking more measurements and producing more packets as a result of an external trigger, notably causing congestion and thus loss of packets or discovery messages.

Faults that interrupt the communication services of the network, especially in sensor networks, can compromise the entire network. The detection of the faults, and subsequently their correct identification are the only ways to work towards any actions concerning the recovery of the network. Regarding the faults mentioned, as they are primarily triggered by external events, it is desirable that the nodes themselves can contribute to their recovery, particularly in the case where users are absent, or the nodes are placed in remote locations. In this context, our contributions include the detection and identification of communication faults encountered in WSN. Benefitting from machine learning techniques, the required accuracy and precision of these two actions can be enhanced to further advance fault management, especially autonomous recovery actions.

By using machine learning algorithms, correlations that cannot be otherwise detected are taken into account, thus producing an accurate model for fault detection and identification. Ideally, every fault would have a specific set of symptoms describing the problematic situation. However, in most cases these symptoms are difficult to discern by humans and machines alike, given the variety of sensor networks and their applications. Difficulties developing a detection and identification scheme are apparent in empirical systems, especially in edge cases. On the other hand, when using machine learning techniques, a general set of network metrics can be used to best describe the faults whilst simultaneously not targeting specific cases.

The faults covered in this paper are not the foreground of most existing fault detection schemes. Our work is realized by means of an application regarding structural health monitoring (SHM), specifically a wireless sensor network deployed on a bridge. In applications such as described in [1], the communication in the network is

Manuscript received January 30th, 2019; revised February 24th, 2019.
This work was supported by the EC H2020 project SENSKIN under Grant No. 635844.
Corresponding author email: slenas@ee.duth.gr.
doi:10.12720/jcm.v.n.p-p

affected by the faults, especially given the open environment plan of most SHM deployments.

This application field is ideal for the problem at hand, especially considering that autonomous fault management schemes and the correct diagnosis of faults can be valuable assets in this area. Given the faults examined in this study, their close connection to the communication services of the network, and potential self-recovery steps, the necessary dataset for this work is generated via the simulation of a realistic network. The dataset includes all the faults mentioned above and are simulated with respect to the structural health monitoring aspect of the network and the physical restraints of sensor nodes deployed on a realistic bridge topology. Through the process of simulating each fault case in the network, samples of data were collected using various network metrics. By maintaining a subset of the most beneficial network metrics, features for the problem were extracted and labels for each fault case were defined, as determined by supervised machine learning algorithms. The problem is represented as a multiclass classification problem with categorical labels for each fault case.

The tools used in the proposed solution include the Opportunistic Networking Environment simulator (ONE) [2], and the Machine Learning and Statistics Toolbox [3] in MATLAB for the subsequent training of the model using a Decision Tree algorithm. Python scripts are used for data manipulation, to extract beneficial information from the simulator's report files. Preprocessing of the data is completed in MATLAB before passing the final dataset to the toolbox. By testing the produced model, each type of fault including the normal state of the network is classified with high accuracy, recall, and precision. The model is able to clearly distinguish between the fault cases, while only misclassifying samples from fault classes with known similarities. Overall, the model has an accuracy of 96.46% across six classes.

The rest of the paper is organized as follows. Chapter 2 presents the related work regarding the various categories of faults in wireless sensor networks (WSNs) whilst including the machine learning techniques used to solve the problems stated. Chapter 3 showcases the formulation of the problem, describing the faults the paper sets to successfully detect and identify. Chapter 4 makes way for the experimental aspect of the paper, entailing the generation of the dataset through the simulation of a realistic network. Chapter 5 presents the proposed solution highlighting feature extraction. Chapter 6 evaluates the model and presents the results. Finally, Chapter 7 concludes the paper and presents the future work.

II. Related Work

In the process of eliminating or at least mitigating faults and failures in WSNs many studies have provided schemes with effective solutions for various applications using numerous techniques. The selection of works presented here offers insight to the problem domain, highlighting positive attributes from each study, while comparing the methods used to our own work and goals. It includes similar methods such as an empirical decision tree, as well as other machine learning techniques. As the faults in WSN can be approached in many different ways and with different goals in mind, the faults, while similar in all cases, are diagnosed with varying identification titles referring usually to the potential recovery steps each study hopes to act upon.

To begin with, in [4], the tool developed, Sympathy, relies on the amount of data reaching the sink to infer failures in the network. Using a selection of metrics, the sources of faults are identified and subsequently divided into certain fault categories to be reported with a level of urgency to the network's users. The goal in this work is to restore data collection in the network via user notification, thus pinpointing the offending section of network is essential to the recovery of the network. However, Sympathy uses an empirically developed decision tree to determine the most likely cause of data loss in the network, which in turn triggers insufficient data collection at the sink. Given the unreliable nature of WSN and the magnitude of applicable fields, a rule-based method slowly eliminating the fault causes via basic node health, connectivity, and communication status, could reasonably not take into consideration useful metrics or values. Important correlations could be overlooked, thus substituting empirically developed methods with machine learning techniques constitutes fewer chances to miss such correlations in the data. Such an approach was chosen in our work, in hopes of generating a holistic method so that, given set of metrics, faults can be accurately detected in sensor networks. A final aspect which is considered is the amount of user notification. As faults are considerably common in sensor networks, our goal is to be able to identify faults, which not only disrupt the communication status of the network but can also be rectified by the sensor nodes themselves, eliminating cost heavy notification actions in the network.

Another study which uses machine learning techniques to discover faults in sensor networks is described in [5]. Here, the problem is directed at uncovering unknown faults which disrupt the network, thus the study focuses on unsupervised machine learning techniques to produce results. The authors' argument against supervised techniques is the lack of sufficient domain knowledge to learn all the rules describing the problem. Although in some cases this is true, the sheer degree of diversity in WSN applications gives way to developing solutions which are based on the same principles but highly customizable. Our work exhibits a first study-case of applying machine learning techniques to structural health monitoring problems targeting a set of faults specifically regarding communication in the network. As it is anticipated, the authors of [5] focus on the problem at hand, which benefits from detecting unknown faults.

Their work, Agnostic Diagnosis, is tested on a real-life network (i.e. GreenOrbs) with positive results. This motivates the physical implementation of our own work to further study the promising simulated results. Even though this study uses machine learning techniques, the use of the same database to compare experimental results with our own work was not possible as different aspects of fault diagnosis were highlighted in each study. This shows the amount of diversity in the field regarding faults in WSN, including the wide range of viable solutions for each application domain. Their study uses both temporal and spatial detection; temporal detection is achieved through a cumulative sum with change point detection for the time series, whereas the algorithm K-MEANS is used for spatial detection, thus highlighting the use of unsupervised machine learning techniques. Even though a complex paradigm was put in place for Agnostic Diagnosis the authors admit that through the selection of metrics used, some faults can be easily detected, whereas others require agnostic detection. This further motivates our work in building a more focused set of faults, not only ensuring detection between normal and faulty states but also identifying specific faults which can be subsequently dealt with by the nodes themselves using recovery actions.

At this point it is worthy to note that both [4] and [5] use a similar selection of network metrics to diagnose faults in the network. This reinforces the fact that faults in WSN can be detected through metric changes and notably the selection of the most meaningful metrics in combination with the most robust technique for each application will ultimately produce the best results.

The work in [6] interprets fault diagnosis in WSN as a pattern classification problem, taking advantage of the inherent characteristics of WSNs. Specifically, the notion of a node's nearest neighbours is taken into account to increase the robustness of the introduced Neighbourhood Hidden Conditional Random Field (NHCRF) model. The method models the WSN as a graph and is able to detect faulty sensors as well as faulty transmission paths, by using transmission time based features, such as signal strength and signal delay. Knowing that a node or a transmission path is faulty, while desirable, does not give an accurate amount of information regarding the fault. While detection and identification are the primary steps in the fault management process, poorly identified faults may hinder secondary but essential recovery steps in subsequent works. Overall, the NHCRF model presents impressive comparative performance in regard to popular machine learning algorithms and various study cases, however, the only performance metric used is accuracy. A more comprehensive assessment of the experimental results would be useful as accuracy has the tendency to produce a bias towards the majority class of the dataset.

In [7] the authors' goal is to target reliability complications in WSNs. Firstly, by providing a model to assess link quality and secondly, a way to improve resource efficiency due to storage constraints in WSNs. Both link quality and storage constraints contribute to packet loss in WSNs, thus optimizing these issues in the network will have beneficial outcomes. While this study covers only an aspect of the work we present in this paper, i.e. the link quality, the methods used are the same. Machine learning techniques, specifically supervised algorithms and more to the point decision trees and additionally rule learners are used to solve the problem at hand. As in our case, the authors cite the advantages of automatic correlation discovery, not relying on application specific heuristics and unreliable environmental factors. Another point of reference is the similar network metrics used as features. This shows that using a subset of metrics common to WSNs can be used to solve different problems with various applications, while remaining accessible to any wireless sensor network, as is also noticeable in [4] and [5]. However, the point of view of the study differs greatly from our own. The authors use supervised learning to essentially, detect and identify disruptive elements and take subsequent preventative actions to better routing and reliability in WSNs. Our work takes into account the challenges that arise in link quality analysis and to perform detection and identification. This is merely a part of the work presented here, as a variety of faults is targeted rather than gaining specialist insight to only one aspect. The positive results and successful outcome of the authors' work highlight the advantages of machine learning algorithms and motivates our own work accordingly.

Another fault detection scheme [8] uses both supervised and unsupervised techniques to build a system for autonomous decision making in WSNs. Using optimized versions of linear discriminant analysis (LDA) and clustering techniques the authors have produced a system to collect data, and infer decisions through knowledge gained from correlations in the data. The faults, or anomalies, as they are referred to, include physical jamming, collisions, misdirection, and selective forwarding. The faults and their features, which include carrier sensing count, packet send ratio, retransmission count, packet drop ratio, and misdirection count, can be found in WSNs regardless of the field of application, making this system easily transferrable. However, the faults selected have little potential in regard to recovery techniques. As most fault detection schemes will attest, their future goal is the ability to recover the network from serious implications disrupting its services. In any case, the authors provide insight to the mentioned machine learning techniques, specifying that LDA requires fewer samples to learn correlations between data compared to clustering yet has higher learning complexity, whereas clustering has lower overhead.

Moving in a different direction, many schemes provide fault detection with support from machine learning algorithms by using the sensors' actual readings. By comparing the output of the sensors to previous instances or to neighbouring sensors' outputs, faults are discovered through the discrepancies in the data. In such cases,

effective modelling of the network while benefitting from machine learning techniques increases successful detection. For instance, in [9] the network is modelled as a modified recurrent neural network (MRNN), concentrating on the sensor node, its dynamics and the interconnections with other nodes to build an accurate model for sensor fault detection. This takes into account neighbouring nodes' outputs as well. We also see works where techniques such as SVM classifiers [10] are used to detect faults in WSNs. Here, the authors concentrate on a precise solution to identifying faulty nodes whilst simultaneously minimizing loss. Failures detected are broadly identified as hardware, software, or communication faults all of which are closely related to the sensor operations. Therefore, using sensor data is the most suitable way of predicting faults in coherence with powerful machine learning techniques.

Finally, the work presented in [11] is similar to our study in that it uses machine learning techniques for structural health monitoring. Here, a data-driven approach is followed to achieve a decentralized and autonomous fault detection system as a way to produce more reliable and accurate structural health monitoring. Having a reliable sensor network is a key component to SHM as sensor readings can be affected by faults. This work is closely related to our own with the exception of the type of data utilized and machine learning algorithm selected to solve the problem. The authors use the sensor data in combination with an artificial neural network to infer faults in the network, in contrast to our work in which network metrics are used as features for a decision tree algorithm. Taking advantage of the redundant information obtained by sensors, the faults detected are bias and precision degradation with regard to the sensor data, whereas our work focuses on communication fault. In neither case is there a requirement of any application specific data or information.

In conclusion, the works evidently present the wide range of possible solutions to this problem and the diversity of techniques which can be used.

III. Problem Statement

Faults can disrupt working networks to the point of a complete halt; for instance, in the case of SHM, the network can no longer transfer monitoring data because of a faulty interface. In other cases, the reported results may not be accurate due to missing data caused by packet loss, as a result of either excessive noise or congestion leading to degraded communication. In any case, these issues are not desirable, but in the event that they occur, it is, first and foremost necessary to diagnose the faults to be able to take any remedial actions. Possible ways of rectifying the type of faults we set to diagnose include increasing signal strength in cases of increased noise and to bypass the obstructing object, and backup interfaces can start working in the case the primary interface is faulty. Methods referring to recovery actions are not further discussed, as they are not the focal point of this paper and are frequently associated with the application field of the network.

To detect faults in WSN, the normal working state of the network needs to be precisely defined. From there, any potential faulty states arise from discrepancies within the normal state. The goal is to be able to analyze the symptoms of each fault and its corresponding characteristics. In this way, a fault profile can be formed containing the network metrics, which are affected by the fault, so that coherent features can be extracted from the data.

To begin with, faults that disrupt the network can occur for longer or shorter time periods, randomly within the working lifetime of a network. In the case of faults occurring due to external events, depending on the type, they can severely impact the network, by losing connectivity between nodes or less so, for instance, by introducing higher noise levels.

Regarding the faults studied in our work, connectivity and its quality is shown as a multifaceted issue. Firstly, a node can be cut off from its neighbours due to a faulty interface. In this case, the node has no way to contact either neighbouring nodes or system administrators to report the fault or indeed to rectify it, at least not immediately. Thereby, it is taken as a given that this sort of fault lasts a long time period. Another side effect of this fault is the fact that neighbouring nodes also face a fault as they themselves lose a single neighbour, the link to the faulty interfaced node. Thereby, declaring faulty neighbour nodes as another fault in the case study. Symptoms of this failure are the inability to transfer packets to their next hop node or their destination. In severe situations, a network may be split into smaller networks without the ability to communicate, if the node is acting as the connecting node. Secondly and predominately in our application area, SHM, obstacles can affect connectivity between nodes for shorter intervals (with regard to faulty interfaces). Here, the link between two nodes is lost, so nodes can still communicate with their remaining neighbours until the object is no longer obstructing the communication. Such an instance could be a truck passing upon a bridge whilst in traffic. In response to these faults the network will exhibit the following symptoms: Neighbour lists will drop nodes, especially if the interface is faulty, or drop the nodes and then reinstate them to the list if an obstacle is obstructing connectivity. Packet loss is expected to increase particularly in the case where connectivity is lost during packet transfer. Other symptoms include very low signal strength for lost connections and limited packets being transferred when the interface is faulty.

Moving forward, an increased level of noise due to extreme weather conditions accentuates difficulties in transferring packets in the network. Rain, heavy fog, and even dust storms can affect noise levels. In this case, the quality of communication is affected, not allowing packets to be transferred with the same ease compared to

normal conditions and even more so in critical situations, when packets may be lost completely due to the weather conditions. Symptoms which can be observed in this case include low signal strength, increased packet loss and delay. The number of packets sent in such conditions is expected to be larger than in normal conditions due to retransmissions as packets are less likely to be delivered at the destination.

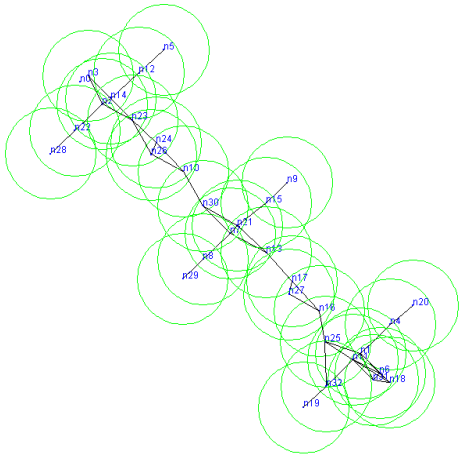

Fig. 1. Topology of the simulated WSN.

Finally, in the case that sensor nodes produce more measurements, communication within the network is affected. Most commonly, initial stages of congestion can be detected or in more extreme cases, as a result, network discovery packets can be lost in the network. In the latter case, nodes mistakenly cannot communicate with their neighbours, as the packets pertaining to their availability are lost. These conditions are more often than not, aftereffects of unpredictable events (emergency or otherwise). Events can be considered of emergency status if they result in damaging the structure the network is set to monitor. For instance, natural disasters such as earthquakes fit this category. The resulting packets generated and sent within the network are significant enough to cause disturbances in the network. The excess of generated packets can be cumbersome to the networks' services due to the sheer magnitude of packets sent into the network, resulting in congestion. This will affect the network's ability to transfer packets as a result of increased buffer sizes and larger packet delays.

IV. Dataset Generation

A dataset incorporating the above faults and the normal state is essential for investigating the problem. Of course, a dataset produced by a real network containing these faults would be ideal. However, this would require a lengthy amount of time to set up and test said network, even more so to ensure data from all fault cases are collected. To alleviate these restraints a network simulation was preferred. Each network state was simulated as realistically as possible to gauge the effect the faults have on the network. Common network metrics obtainable in all WSNs are used to identify faults. These are reported by the simulator to ultimately discern the status of the network.

To properly simulate all the fault cases, the simulator was modified to incorporate extra features concerning the faults. The simulator chosen was the Opportunistic Networking Environment (ONE). Given its easy to modify structure, ONE was ideal for simulating the network at hand and incorporating new features. Such features include the received signal strength indicator (RSSI) to simulate the noise levels used in the extreme weather conditions case, the loss of connectivity between two nodes, thus removing one link from each corresponding node, while ensuring the remaining links are intact, and finally simulating the loss of the node's interface.

A static network was used, based upon the actual topology of a WSN deployed on a bridge [12], which incorporates sensor nodes on the bridge pylons and the road, as shown in Fig. 1. The nodes are represented as dots with the appropriate name tags (i.e. n1), their corresponding interface range and connections to other nodes. In total, 33 nodes make up the network and one node acts as the sink node. Each node generates packets to be sent to the sink node, whereas the sink node does not generate any packets. The network operates with a duty cycle; nodes are active for twenty-minute intervals 4 times a day, and in those intervals one packet is generated and sent into the network. The simulated network uses an epidemic routing algorithm. As for the interface IEEE 802.15.14 is used at 20kbps and at a frequency of 868MHz. The 868MHz frequency was chosen as there are fewer power requirements for data transfer, while less noise is apparent in link connection [13].

The experiments from the simulated network cover the faults mentioned, firstly by simulating the base case, *Normal*, i.e. the normal working state of the network. The rest of the cases simulated are the *No Interface*, *Weather*, *Obstacle*, and *External Event* cases referring to the faults explained in section III. Included in the *No Interface* case are the faults due to the loss of the interface, regarding both the faulty node and its neighbouring nodes.

For the *Obstacle* case a single node in the network experiences a fault, whereas in the *Normal* and *Weather* cases all nodes of the network experience the same effects of the scenario. In the *No Interface* case one node is considered as having a faulty interface, while all its neighbours will also lose the corresponding connection, thus being faulty as an aftereffect. Finally, in the *External Event* case one node is denoted as faulty, i.e. the node which produces excessive measurements, while the rest of the network becomes active to ensure the packets can be transferred throughout the network. This scenario is more likely to occur in emergency events.

For all cases, the faults are simulated as random periods during the simulation time of each experiment and the point at which the fault starts to take effect is randomly chosen.

TABLE I: FEATURE SELECTION

Feature Name	Corresponding Metric
conTime	Connection Time per Node
signalStrength	RSSI value
xmitPkts	Transmitted Packets
buffer	Buffer Size
buffer_diff	Buffer Size Difference
pktLoss	Packet Loss

For the *Normal* case in particular, the RSSI values are between -50dBm and -100dBm, whereas in the *Weather* case each link has a random value between -80dBm and -100dBm for the faulty periods [14]. Each experiment lasts 7 days. Faults occur separately; practically, nodes suffer a single fault per experiment or are in normal conditions. Approximately 1000 experiments were run, covering all fault cases, for each node of the network. Thus, ensuring the dataset has sufficient information regarding both each fault case and each of node in the network, bearing in mind that the generated model is common for all the nodes.

V. PROPOSED SOLUTION

Through the analysis of each fault and its corresponding symptoms it is clear that WSNs depend on the reliability of the network in order to perform the tasks each application requires. Hence, by using network metrics as attributes for the machine learning algorithm and selecting those common to all WSNs, the solution is independent of the application field. In essence, the features used in this problem are based on commonplace network metrics readily available in WSNs, which collectively describe the symptoms which the faults exhibit in the network.

The features are selected by processing the reports from the simulator which contain information regarding events that take place in the network for each study case, such as the creation of a packet, the transmission and delivery of a packet given the simulation time, connectivity status, buffer occupancy, and packets which have been aborted during their transfer. For the simulated time, samples are taken in 100s intervals and each sample is labelled with the corresponding fault tag. Faults are considered only if they occur for at least 30% of the time interval given. This value is selected depending on the characteristics of the network and the sensitivity the fault diagnosis system is set to achieve. A lower value would consider faults almost immediately, whereas larger values would take longer to consider a fault. A middle ground was reached for this value.

The features selected are given in Table I and explained below.

- *conTime*: The connection time of a node with its neighbours for the duration of the sample time. This feature is expressed as a value between 0 and 1. A value of 1 represents a node demonstrating no connectivity problems with any of its neighbours. Any values below 1 indicate that at least one link has been affected by a fault, thus a link has been lost during the sample time.
- *signalStrength*: The mean RSSI value for the node, taking into account the quality per link for each neighbour for the sample time.
- *xmitPkts*: The number of transmitted packets per node for the sample time.
- *buffer*: The buffer size of the node.
- *buffer_diff*: The increase in buffer size per sample.
- *pktLoss*: The number of lost packets as result of a fault. This includes aborted packets due to excessive noise or a lost connection and dropped packets due to overflowing buffer queues.

In the proposed solution a distributed logic is applied when making predictions. In this way, in subsequent work each node can participate in its own recovery actions.

For the modelling, instead of using separate models for each node, data is collected to produce a general fault model incorporating instances from every node in the network. This is why features such as *conTime* are included, expressing the connectivity of each node without directly focusing on the number of its neighbours.

While more characteristics were obtained from the simulation reports, metrics such as packet delay had many missing values as a delay value could only be described when a packet was sent into the network. In combination with the network's duty cycle the values did not possess any complementary information and were not useful for determining fault patterns.

TABLE II: LABELS DEFINITIONS

Label	Fault Case	Network Status
Normal	Normal	Normal
No_Intf	No Interface	Loss of Node Connectivity – Faulty Interface
Weather	Weather	Extreme Weather Conditions
Obstacle	Obstacle	Loss of Link Connectivity – Obstacle
Faulty_Nbr	No Interface	Faulty Neighbour due to faulty interface
Ext_Event	External Event	Excessive Packet Generation

The samples are labeled manually based on the time periods exhibiting faults, as multiple simulated seconds are recorded in a single sample. The faults labels are defined in Table II with the corresponding faults cases and the status of the network.

Upon completion of the dataset, it is then subjected to preprocessing techniques to be fit for training. As the network is expected to be working normally for most of its lifetime, the faults only take up a small percentage of the dataset which is commonly seen in such problems. Randomly undersampling the majority class is used to balance each class of faults, which is essential when training a new model. This step is then followed by the separation of the dataset into a training set and a testing set, resulting in a 70% - 30% split respectively. Finally, the training dataset is normalized using the SoftMax function.

A decision tree algorithm is used to train the model from the Machine Learning and Statistics Toolbox in MATLAB. Decision trees are ideal training algorithms for the problem at hand as they can classify both binary and multiclass datasets and are able to work well with categorical labels, thus being a good choice when predicting our selected fault labels which are multiclass categorical labels. Given the large dataset at our disposal, the fast prediction speed of the algorithm is desirable, benefitting both prediction accuracy and training time. Decision trees are well suited for imbalanced datasets and are preferred in such cases, as the problem by default does not generate balanced classes and undersampling is used for the majority class. The training dataset is fed to the app and the model is produced. It is then exported so that the model's performance can be assessed through the test set.

VI. Results

For the evaluation of the model, the testing set undergoes the same processing steps as the training set. The metric used to primarily measure the model's performance is accuracy. In our case the formula of balanced classification accuracy (BCA) is used to make the classes further comparable [15], just as the majority class was undersampled, the performance metric accounts for unbalanced data as well. Other metrics used include sensitivity and specificity, as well as the positive and negative predictive value, and finally the F-Score [16].

$$BCA_i = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right). \quad (1)$$

$$BCA = \frac{1}{L}\sum_{i=L}^{L} BCA_i. \quad (2)$$

The model's overall balanced accuracy is 96.46% and has a misclassification rate of 3.54% across the six classes. In combination with a high overall accuracy value, each fault class also has a high balanced accuracy. Namely, the *Weather* class has the highest of the values at 99.36% followed by the *Normal* class with a value of 98.47%. The *External Event* class has a value of 96.01% while the *Obstacle* class is at 95.93%. Finally, the *No Interface* class has a value of 95.8% and the *Faulty Neighbour* class a value of 93.2%.

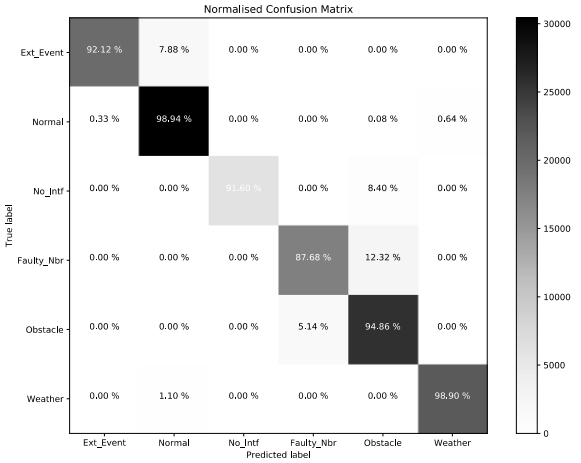

Fig. 1. The model's accuracy in the form of a confusion matrix.

Supplementary to this evaluation and depicting more so the quality of the model are the metrics sensitivity and precision and their negative counterparts, specificity and negative predictive value. Sensitivity and specificity indicate the positive and negative hit rates, i.e. how often the model predicts positive and negative samples, whereas precision and negative predictive value indicate the predictive value of the model, essentially how believable the classes it predicts for each sample are. These are detailed in Table III. High values are observed for each metric in every class.

The model's F-Score offers insight to the model's quality as the harmonic mean of sensitivity and precision. Here, the *Faulty Neighbour* class has the lowest F-Score, as a result of the lowest sensitivity value, indicating the low positive hit rate for the class. The *Obstacle* class has the second lowest F-Score value expressing the lowest precision value of the classes, which is expected as the *Obstacle* class misclassifies samples from all the classes regarding connectivity as well as the *Normal* class. In both instances, the symptoms arising in these classes have similar attributes which makes the distinction between the two classes both less than ideal and responsible for the lower values in regard to the rest of the faulty classes.

The confusion matrix of the tested data is depicted in Fig. 2 giving the number of correctly classified samples for each class including the samples which are misclassified. As shown, the number of incorrectly classified samples is low for each class and misclassifications occur between similar faults rather than across the board. For instance, the fault cases *Faulty Neighbour* and *Obstacle* have completely different causes, which would make recovery efforts different, but have similar fault symptoms thus easier to misclassify. Keeping this in mind, both faults are distinguishable, however improvements can be made.

TABLE III: RESULTS

	Balanced Accuracy	Sensitivity	Specificity	Precision	Negative Predictive Value	F- Score
Ext_Event	0.9601	0.9212	0.9990	0.9949	0.9842	0.9566
Normal	0.9847	0.9894	0.9800	0.9398	0.9966	0.9640
No_Intf	0.9580	0.9160	1	1	0.9954	0.9562
Faulty_Nbr	0.9320	0.8768	0.9872	0.9265	0.9775	0.9010
Obstacle	0.9593	0.9486	0.9699	0.8936	0.9861	0.9203
Weather	0.9936	0.9890	0.9981	0.9911	0.9977	0.9900

This is also observed for the cases *Weather-Normal* and *External Event-Normal*. Samples from the *Weather* and *External Event* classes are misclassified only with regard to the *Normal* case. Each class is presented distinctly as thereare no misclassified samples between the *Weather* and *External Event* classes, or the remaining classes.

The *No Interface* class misclassifies samples only in relation to the *Obstacle* class. No samples from other classes are mistakenly predicted as *No Interface*, as shown with the specificity metric, depicting that this particular class is well defined within the problem.

Finally, it is worth noting that the model misclassifies samples to multiple classes with regard to the true *Normal* class. Also, the predicted class *Obstacle* contains incorrectly classified samples which originate from various classes. This shows that the *Obstacle* case may need to be more precisely defined in the future, whereas the *Normal* case is the most general case, as it defines the normal state of the network and all fault cases arise from this state, which is expected.

VII. CONCLUSIONS

In conclusion, fault management constitutes a key feature for the reliable operation of a WSN. Emphasis is given to detection and identification as WSNs are susceptible to various faults and without these steps any form of recovery is not possible. In this paper, faults disrupting the communication process were examined, particularly originating from external factors in which network users have little or no control over. Such faults need to be identified as precisely as possible so that recovery actions can be considered or take place immediately.

The faults examined in this paper are lack of connectivity between nodes due to a faulty interface, obstacles hindering communication, and packet loss due to extreme weather conditions which increase noise levels or as a result of the effects of excessively generated packets. In total, the above cases were considered as well as a normal case regarding a normally functioning network and a faulty neighbour case which takes into account the status of the remaining neighbours in the faulty interface case.

Using a topology that corresponds to a realistic WSN deployed on a bridge, the network is simulated to generate data referring to each study case, to be then used to train the machine learning model. While our work focuses on a specific application, namely a WSN deployed for SHM, the holistic use of data from all the network nodes ensures a more general model which can be applicable to various domains.

Using machine learning techniques detection and identification of these faults was achieved. A successful model was produced classifying each fault with high accuracy and precision. To reiterate, the model's overall accuracy is 96.46%. All faults are detectable specifically, the *Weather* class has an accuracy value of 99.36% followed by the *Normal* class with a value of 98.47%. The *External Event* class has a value of 96.01% while the *Obstacle* class is at 95.93%. Finally, the *No Interface* class has a value of 95.8% and the *Faulty Neighbour* class a value of 93.2%. More importantly the faults are presented as distinct cases as shown by the hit rates and predictive values of each class. Even though similar symptoms can be observed in the fault cases, such as *Obstacle* and *Faulty Neighbour*, the model is able to distinguish each class.

Future work of this paper primarily includes the improvement of the dataset. Here, the work is evaluated using simulated data, which demonstrates the potential of the study. However, using a more realistic environment, such as implementing a small-scale testbed to produce a non-synthetic dataset would be ideal for a more representative performance evaluation of the system. This would also be complemented with more diverse study cases, scenarios and experiments.

Moreover, a better understanding of the fault symptoms and how they are expressed in the network might provide a better selection of features, thus being able to better describe the faults to classify the data samples. Therefore, highlighting the enhancement of the feature subset.

With regard to the machine learning approach, a decision tree was the best option to initially solve the problem, however there are many algorithms and techniques which might improve the extracted model and the results obtained. For instance, incorporating the time series elements of the problem to its solution. As all events in the network are time based (packet transfers, packet drops and so forth), an algorithm which also considers this factor will potentially enhance the detection and identification of the faults.

Finally in the future work and to complete the fault management scheme, recovery steps to alleviate the faults described in this paper would move in the right direction in delivering a reliable system capable of dealing with its own faults.

REFERENCES

[1] Konstantinos Loupos et al, "Structural Health Monitoring for bridges based on skin-like sensor", IOP Conf. Ser.: Mater. Sci. Eng., 236 012100, 2017

[2] A. Keränen, J. Ott and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Rome, Italy, 2009.

[3] "Statistics and Machine Learning Toolbox™ User's Guide," The MathWorks, Inc., 2017.

[4] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler and D. Estrin, "Sympathy for the sensor network debugger," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, New York, NY, USA, 2005.

[5] X.Miao, K.Liu, Y.He, D.Papadias, Q.Ma and Y.Liu, "Agnostic Diagnosis: Discovering Silent Failures in Wireless Sensor Networks," *IEEE Transactions on Wireless Communications,* vol. 12, no. 12, pp. 6067-6075, December 2013.

[6] P. Tang and T. W. S. Chow, "Wireless Sensor-Networks Conditions Monitoring and Fault Diagnosis Using Neighborhood Hidden Conditional Random Field," *IEEE Transactions on Industrial Informatics,* vol. 12, no. 3, pp. 933-940, 2016.

[7] Y. Wang, M. Martonosi and L. s. Peh, "Supervised Learning in Sensor Networks: New Approaches with Routing, Reliability Optimizations," in *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, Reston, VA, 2006.

[8] S. Krishnamurthy, G. Thamilarasu and C. Bauckhage, "MALADY: A Machine Learning-Based Autonomous Decision-Making System for Sensor Networks," in *2009 International Conference on Computational Science and Engineering*, Vancouver, BC, 2009.

[9] A. I. Moustapha and R. R. Selmic, "Wireless Sensor Network Modeling Using Modified Recurrent Neural Networks: Application to Fault Detection," *IEEE Transactions on Instrumentation and Measurement,* vol. 57, no. 5, pp. 981-988, May 2008.

[10] S. Zidi, T. Moulahi and B. Alaya, "Fault Detection in Wireless Sensor Networks Through SVM Classifier," *IEEE Sensors Journal,* vol. 18, no. 1, pp. 340-347, 2018.

[11] K. Smarsly, K. Dragos and J. Wiggenbrock, "Machine learning techniques for structural health monitoring.," in *Proceedings of the 8th European Workshop on Structural Health Monitoring (EWSHM 2016)*, Spain, Bilbao, 5–8 July 2016.

[12] S. A. Jang and B. F. J. Spencer, "Structural Health Monitoring for Bridge Structures using Smart Sensors," Newmark Structural Engineering Laboratory. University of Illinois at Urbana-Champaign., 2015-05.

[13] M. Woehrle, M. Bor and K. Langendoen, "868 MHz: A noiseless environment, but no free lunch for protocol design," in 2012 Ninth International Conference on Networked Sensing (INSS), Antwerp, 2012.

[14] K. Srinivasan, P. Dutta, A. Tavakoli and P. Levis, "An empirical study of low-power wireless," ACM Transactions on Sensor Networks , vol. 6, no. 2, pp. 1-49, 2010.

[15] K. H. Brodersen, C. S. Ong, K. E. Stephan and J. M. Buhmann, "The Balanced Accuracy and Its Posterior Distribution," in *2010 20th International Conference on Pattern Recognition*, Istanbul, 2010.

[16] D. M. W. Powers, "Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation," *Journal of Machine Learning Technologies,* vol. 2, no. 1, pp. 37-63, 2011