UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Roberto Milton Scheffel

Fault Tolerance in Wireless Sensor Networks with a Multi-Sink Protocol and Data Confidence Attribution

Florianópolis
2021

Roberto Milton Scheffel

Fault Tolerance in Wireless Sensor Networks with a Multi-Sink Protocol and Data Confidence Attribution

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Doutor em Ciência da Computação.
Orientador: Prof. Antônio Augusto Medeiros Fröhlich, Dr.

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Scheffel, Roberto Milton
Fault Tolerance in Wireless Sensor Networks with a Multi-Sink Protocol and Data Confidence Attribution / Roberto Milton Scheffel ; orientador, Antônio Augusto Medeiros Fröhlich, 2021.
133 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2021.

Inclui referências.

1. Ciência da Computação. 2. Tolerância a Falhas. 3. Detecção de Falhas. 4. Atribuição de Confiança. 5. Desvio de Vazios em Roteamento Geográfico. I. Fröhlich, Antônio Augusto Medeiros. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Roberto Milton Scheffel

Fault Tolerance in Wireless Sensor Networks with a Multi-Sink Protocol and Data Confidence Attribution

O presente trabalho em nível de doutorado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Doutor em Ciência da Computação.

ICP-Edu
Documento assinado digitalmente
Vania Bogorny
Data: 14/02/2021 07:03:34-0300
CPF: 684.586.570-15

Profª. Vania Bogorny, Drª
Coordenadora do Programa

ICP-Edu
Documento assinado digitalmente
Antonio Augusto Medeiros Frohlich
Data: 12/02/2021 21:45:39-0300
CPF: 500.696.400-68

Prof. Antônio Augusto Medeiros Fröhlich, Dr.
Orientador

Florianópolis, 2021.

Este trabalho é dedicado à minha família. Sem o apoio e a compreensão de vocês, não seria possível. Amo vocês com todo meu coração!

ACKNOWLEDGEMENTS

Primeiramente agradeço a Deus, por me dar perseverança e força para completar esta importante etapa da minha vida. Com muita saudade, agradeço ao meu pai Aldino e à minha mãe, Adelyra (*in memorian*), que sempre procuraram mostrar aos seus filhos a importância do conhecimento e do estudo. A todos os familiares, por seu apoio e palavras de incentivo. E especialmente, agradecer à minha amada esposa Osnilde e aos meus amados filhos Felipe e Bruno por (pacientemente) ouvirem meus lamentos durante este período, além de entenderem quando eu estive muito ocupado e não pude dar-lhes a devida atenção.

Meu agradecimento aos colegas da Universidade Tecnológica Federal do Paraná (UTFPR) - Campus Toledo, que de alguma forma me auxiliaram a vencer este desafio.

Gostaria de agradecer também aos colegas do Laboratório de Integração Software/Hardware (LISHA), pela ajuda em diversas tarefas e pelas produtivas discussões nos mais variados assuntos.

Meu especial agradecimento ao meu orientador, prof. Guto, por sua paciência, suas sugestões, dicas preciosas e merecidas reprimendas, que tanto me ajudaram. Sinceramente, obrigado!

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

If you find that you're spending almost all your time on theory, start turning some attention to practical things; it will improve your theories. If you find that you're spending almost all your time on practice, start turning some attention to theoretical things; it will improve your practice.

Knuth, D.

RESUMO

Os sensores têm sido empregados para fins de monitoramento em vários campos de aplicação ao longo de décadas, e a identificação de falhas, sejam causadas por mau funcionamento, interferência ou intrusão, é de grande relevância para a tolerância a falhas em sistemas. Redundância e diversidade de sensores são uma das principais abordagens para lidar com falhas. Comparar medidas distintas de sensores que observam o mesmo fenômeno é uma maneira natural de obter confirmação. Entretanto, os sistemas tolerantes a falhas geralmente resolvem o problema com modelos estáticos, baseados em leis da física ou estatísticas sobre a operação do sensor. Esses modelos são específicos e não se adaptam bem a ambientes dinâmicos, onde se espera que sensores sejam adicionados dinamicamente ao sistema, seja para substituir aqueles com falha ou para adquirir dados adicionais sobre seu comportamento.

Este trabalho propõe uma abordagem para determinar a exatidão dos dados detectados usando preditores que exploram a correlação de dados para atribuir um nível de confiança a cada dado produzido por sensores. A variação nos níveis de confiança permite a identificação de sensores com defeito, além de fornecer feedback sobre grupos de sensores. O mecanismo de atribuição de confiança proposto pode ser aplicado a qualquer cenário no qual conjuntos de sensores monitoram fenômenos correlacionados. Neste trabalho, é aplicado para aumentar a tolerância a falhas em Redes de Sensores Sem Fio (RSSF), visto que estas naturalmente têm que lidar com sensores com falha em ambientes dinâmicos. As RSSF também podem tirar proveito da natureza distribuída do mecanismo de atribuição de confiança, com uma sobrecarga muito pequena nas mensagens originais, sem mensagens de diagnóstico ou votação.

As RSSF geralmente usam algoritmos de roteamento geográfico totalmente reativos para suportar nós móveis e falhas de comunicação, uma vez que tais algoritmos não requerem procedimentos de construção e manutenção de rotas. Este trabalho contribui para este campo, explorando redundância de gateways e algoritmos de desvio de vazio. As soluções propostas aumentam a disponibilidade e a confiabilidade da comunicação entre os sensores e o mundo externo. O protocolo proposto, denominado FT-TSTP, usa um "modo de recuperação"para encontrar rotas alternativas para os pacotes ao enfrentar vazios. Também entrega mensagens para todos os gateways, ao contrário dos protocolos que escolhem um deles, reduzindo assim o tempo de entrega e o consumo de energia, enviando pacotes apenas para o gateway mais próximo.

As soluções propostas foram avaliadas através de simulações. O protocolo FT-TSTP alcançou taxas de entrega acima de 97% nos seis cenários avaliados. O consumo de energia apresentou um crescimento linear de até 150% com 3 gateways, com estabilização para mais de 3 gateways. O mecanismo de atribuição de confiança foi avaliado em quatro cenários diferentes, identificando cerca de 90% das falhas dos sensores. Uma análise dos parâmetros do algoritmo foi realizada para mapear sua sensibilidade para tipos de erros específicos. Ao rotular os dados com confiança, também acelera a identificação de mudanças no ambiente sempre que um conjunto de sensores correlacionados mostram alterações simultâneas nos níveis de confiança.

Palavras-chave: Tolerância a Falhas. Detecção de Falhas. Atribuição de Confiança. Desvio de Vazios em Roteamento Geográfico. Redes de Sensores sem Fio.

RESUMO ESTENDIDO

Introdução

Os avanços na tecnologia na área de sistemas embarcados permitiu a uso de Rede de Sensores sem Fio (RSSF) em uma variada gama de aplicações. Edifícios e cidades inteligentes, monitoramento industrial, monitoramento ambiental e diversos outros tipos de aplicações podem ser monitorados e controlados por dispositivos cada vez menores e mais econômicos, equipados com sensores e atuadores, comunicando-se através de rádio. As RSSF são base para a infra-estrutura de comunicação em locais remotos, onde redes comuns não estão disponíveis. A tolerância a falhas é um requisito fundamental em diversos domínios, que necessitam operar de maneira confiável e segura. Estas aplicações executam, muitas vezes, em ambientes sujeitos a falhas e interferências no sensoriamento e na comunicação. Desta forma, incrementar a tolerância a falhas das RSSF, tanto no aspecto da comunicação entre os nodos, como nos dados obtidos pelos sensores, permite melhorar a qualidade e a confiabilidade dos dados coletados, bem como aumentar a corretude das atuações deste tipo de sistemas.
Redundância e diversidade de sensores são uma das principais abordagens para lidar com falhas. Comparar medidas distintas de sensores que observam o mesmo fenômeno é uma maneira natural de obter confirmação. Entretanto, os sistemas tolerantes a falhas geralmente resolvem o problema com modelos estáticos, baseados em leis da física ou estatísticas sobre a operação do sensor. Esses modelos são específicos e não se adaptam bem a ambientes dinâmicos, onde se espera que sensores sejam adicionados dinamicamente ao sistema, seja para substituir aqueles com falha ou para adquirir dados adicionais sobre seu comportamento.
Em RSSF que utilizam protocolos de roteamento geográfico reativo, há uma deficiência de algoritmos de desvio de vazios, pois os algoritmos atualmente apresentados na literatura dependem de informações sobre posicionamento dos nodos vizinhos, ou então de algoritmos de construção de rotas. Outro aspecto que compromete a tolerância a falhas nas arquiteturas tradicionais é o uso de um único *gateway*. Este componente concentra toda a comunicação entre uma RSSF e a estrutura de TI, configurando um ponto único de falha. Sua inatividade ou sua tomada por um invasor malicioso pode comprometer a operação de toda a RSSF.

Objetivos

Este trabalho propõe uma abordagem para determinar a exatidão dos dados detectados usando preditores que exploram a correlação de dados para atribuir um nível de confiança a cada dado produzido por sensores. A variação nos níveis de confiança permite a identificação de sensores com defeito, além de fornecer feedback sobre grupos de sensores. O mecanismo de atribuição de confiança proposto pode ser aplicado a qualquer cenário no qual conjuntos de sensores monitoram fenômenos correlacionados. O objetivo deste trabalho é demonstrar que a tolerância a falhas, em redes de sensores sem fio que utilizam protocolos de roteamento geografico reativo, pode ser incrementada pela combinação de um protocolo de roteamento com múltiplos *gateways* e um algoritmo de atribuição de confiança. Desta forma espera-se resolver o problema de falta de dados devido à vazios na rede e o problema da configuração de um ponto único de falha representado pelo *gateway*. Espera-se também fornecer um mecanismo para que os nodos e a aplicação possam determinar a corretude dos dados produzidos pelos sensores.

Metodologia

Para atingir o objetivo proposto, a primeira etapa foi o projeto e a implementação do protocolo para RSSF denominado Fault-Tolerant Trustful Space-Time Protocol (FT-TSTP), com suporte a

múltiplos *gateways* e com um mecanismo de roteamento que permite contornar regiões vazias. Este protocolo foi então avaliado, no tocante às taxas de entrega de pacotes e em relação ao consumo de energia pelos nodos, em diferentes tamanhos de redes, frequências de sensoriamento e configurações de áreas vazias, além de diferentes números de *gateways*. Para tanto foi utilizada uma ferramenta de simulação de RSSF, denominada OMNet++ (versão 4.6) (OPENSIM, 2017), sobre o *framework* Castalia (versão 3.3) (BOULIS, 2017).
Numa segunda etapa, um mecanismo de atribuição de confiança foi desenvolvido, baseado em preditores que devem executar localmente nos nodos. O preditor de cada nodo, utilizando como entrada os dados dos nodos próximos, deve calcular o valor esperado para o sensor em questão. Com base nestas informações, o mecanismo atribui um *nível de confiança* ao dado lido do sensor. Este nível de confiança pode ser utilizado por outros algoritmos para decidir se o dado pode ser utilizado ou deve ser descartado em uma eventual tomada de decisão. Após o desenvolvimento do mecanismo, dados de sensoriamento reais foram empregados para sua avaliação. Com a injeção de diferentes tipos de erros — *outliers*, *peak*, *stuck-at*, e *noise* — as variações dos parâmetros de configuração do mecanismo foram avaliadas, produzindo dados para a análise da eficácia do mesmo.

Resultados e Discussão

O protocolo FT-TSTP mostrou-se eficaz em atingir os objetivos para os quais foi projetado, apresentando altas taxas de entrega nos diferentes cenários avaliados. Em relação ao protocolo anterior (denominado Trustful Space-Time Protocol (TSTP)), mostrou-se resiliente a diversos tamanhos e formatos de vazios de comunicação na rede. Em cenários com diferentes números de *gateways*, o protocolo também mostrou-se capaz de propiciar altas taxas de entrega dos dados para todos os *gateways* da rede, com taxas de entrega acima de 97% nos seis cenários com vazios avaliados. Desta forma, é possível concluir que a tolerância a falhas de comunicação das RSSF foi consideravelmente incrementada. Aliada ao uso de um protocolo de consenso bizantino entre os gateways, a solução proposta também oferece uma solução para a detecção de intrusos que tentem alterar dados, seja pela invasão de alguns nodos individuais, seja pela invasão de algum dos *gateways* envolvidos.
O mecanismo de atribuição de confiança também mostrou-se eficiente na tarefa de identificar sensores cujas leituras não estão de acordo com os valores preditos pelos modelos gerados. O algoritmo desenvolvido permite, sob determinadas condições, identificar os nodos em que o dado lido está incorreto, através da substituição dos valores incorretos lidos pelos preditos. Isto previne a interferência entre os resultados dos nodos, de modo que apenas o sensor em desacordo atribua um seu nível de confiança baixo ao seu dado. A solução foi avaliada em quatro diferentes cenários, com a identificação de aproximadamente 90% as falhas em sensores, dependendo do tipo de erro. Foram avaliadas as combinações dos parâmetros do algoritmo, com uma análise do impacto de cada um nos resultados obtidos.
A atribuição de confiança também fornece um dado adicional para a aplicação, que pode analisar a variação nos valores da confiança em diferentes nodos, permitindo a identificação de *Concept Drifts* através da análise da variação da confiança dos dados, além do efeito de diferentes fenômenos. Além disso, há indícios de que a variação nos níveis de confiança dos sensores possa ser também utilizada como indicadores para a detecção de alteração de dados por intrusos.

Considerações Finais

Os resultados obtidos nos experimentos validaram a hipótese de que é possível aumentar a tolerância a falhas em RSSF, através de um protocolo *multi-gateway* em roteamento geográfico

reativo, além da atribuição de confiança por um mecanismo que permite aos nodos uma avaliação dos dados gerados pelos sensores, sem a necessidade de mensagens adicionais.

Palavras-chave: Tolerância a Falhas. Detecção de Falhas. Atribuição de Confiança. Desvio de Vazios em Roteamento Geográfico. Redes de Sensores sem Fio.

ABSTRACT

Sensors have been employed for monitoring purposes in several application fields over decades, and failures, either due to malfunction, interference, or intrusion, is of major relevance to fault-tolerance systems. Sensor redundancy and diversity is one of the main approaches to deal with failures. Comparing distinct measurements from sensors that are observing the same phenomenon is a natural way to achieve confirmation. Nevertheless, fault-tolerant systems often address the problem with static models, based on physics laws or statistics about sensor operation. These models are specific, and do not adapt well to dynamic environments in which sensors are expected to be dynamically added to the system, either to replace failing ones or to acquire additional data about its behaviour.

This work proposes an approach to determine the correctness of sensed data using predictors that exploit correlation in data to assign a confidence level to each piece of data produced by sensors. The variation in confidence levels enables the identification of faulty sensors, and also provides feedback about sensor groups. The proposed confidence attribution mechanism can be applied to any scenario in which sets of sensors monitor correlated phenomena. In this work, it is applied to increase the fault-tolerance on Wireless Sensors Networks (WSN), as they naturally have to deal with faulty sensors in dynamic environments. WSN can also take advantage of the distributed nature of the confidence attribution mechanism, with a very small overhead in the original messages, without diagnose or voting messages.

WSN often use fully reactive geographical routing algorithms to support mobile nodes and communication faults, since such algorithms do not require route construction and maintenance procedures. This work contributes to this field by exploiting gateway redundancy and void detour algorithms. The proposed solutions increase the availability and the confiability of the communication between sensors and the external world. The proposed protocol, named FT-TSTP, uses a ‘recovery mode’ to find alternative routes for the packets when facing voids. It also delivers messages to all gateways, in contrast to the protocols that choose one of them, thus reducing delivery time and energy consumption, by sending packets to the nearest gateway.

The proposed solutions were evaluated through simulations. The FT-TSTP protocol achieved delivery rates above 97% in the six evaluated scenarios. The energy consumption showed a linear growth up to 150% when using 3 gateways, with a stabilization for more than 3 gateways. The confidence attribution mechanism was evaluated in four different scenarios and was able to identify around 90% of sensor faults. An analysis of the algorithm’s parameters was performed to map their sensibility for specific error types. By labeling data with confidence, it also speeds the identification of changes in the environment whenever a set of sensors show correlated rates of changes simultaneously.

Keywords: Fault Tolerance. Fault Detection. Confidence Attribution. Void Detour in Geography Routing. Wireless Sensor Networks.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Several application fields rely on sensors for monitoring purposes over decades, and failures, either due to malfunction, interference, or intrusion, is of major relevance to fault-tolerance systems. Sensor redundancy and diversity is one of the main approaches to deal with failures. Fault-tolerant systems often address the problem of verifying sensors' data integrity with static models based on statistics about the sensor's operation, or on models built based on physics laws. Besides, redundancy is a solution widely used in fault-tolerance also applied to sensors, as comparing distinct measurements from the same phenomenon can provide a required level of confirmation. Fault detection solutions, in such scenarios, use these static models to evaluate the data produced by a specific type of sensor. It also assumes that the environment and the expected behavior of the sensor are well known. This allows identifying a faulty sensor if it does not behave as reported by the model, reinforcing the decision by comparing values of other sensors.

When running on a centralized device connected to all sensors, such fault detection methods perform well, as models can be applied and results are easily compared. However, the use of Wireless Sensor Network (WSN) in application in several fields turns the problem of ensuring fault-tolerance on sensors into a distributed problem. In this work, the sensor fault detection will be applied to the WSN field, in order to improve the fault tolerance in applications that use this type of sensor infrastructure. Another problem arises when considering the dynamic nature of many of the WSN application fields, where sensors can be dynamically added to the system, either to replace the failing ones or to improve the system with additional data from new sensors.

WSN are typically deployed in wide areas, with no direct connection to a network. Therefore, the devices have to establish a self-organized network, forwarding data from each other, and the communication protocols between the WSN nodes must attend to some restrictions on connectivity and energy. Many times the WSN deployment occurs in harsh environments, where nodes are subject to unexpected interference, signal propagation obstacles, nodes displacement, among other problems. The communication between the WSN devices and the internet infrastructure is done through special nodes called gateways or sinks. Despite many applications that use sensors that can communicate through standard internet infrastructure, using the Internet of Things (IoT) paradigm, there are fields where the access is not available, demanding a WSN infrastructure. Applications of WSN are diverse, involve monitoring, tracking, and controlling in different areas, as risk areas as volcanoes and mountains, field data acquisition in solar and wind power farms, smart cities monitoring devices, military field surveillance, agricultural fields monitoring, healthcare, industries, among others (Kassim; Harun, 2016), (GLORIA et al., 2019), (AWADALLAH; MOURE; TORRES-GONZÁLEZ, 2019), (Polavarapu; Panda, 2020).

Fault-tolerance is a mandatory requirement under domains that must operate in a reliable and trustful manner. Many times these systems have to use the WSN technology, and networks are deployed in harsh environments and are highly susceptible to interferences in sensing and communication. Fault tolerance mechanisms are essential to ensure correct readings and actuation by the WSN elements. Several aspects should be addressed at the WSN node level, ensuring the correctness of the transmitted data. Different communication protocols were proposed for these systems, not always considering requirements such as timeliness, positioning, security, and fault-tolerance. If the network is correctly dimensioned, it is expected that the communication protocols deliver all collected data to the application. However, this is not always true when some nodes fail, are isolated by jamming attacks, or if intruders overtake them. Therefore, at the same time at which the sensor's data correctness must be ensured by identifying incorrect readings, it must also be ensured that the WSN provides an infrastructure that delivers data to the application in a reliable manner. So, there is a need for a fault detection mechanism at the sensor's level that does not overload the network traffic. The network has also to be resilient to communication interruptions caused by node failures or displacement that produce void regions, or by a communication interruption between the WSN and the application due to a gateway failure.

Protocols that use reactive greedy geographic routing are efficient and resilient to transient node faults when enough redundancy is provided by the WSN design. There is no need for maintaining routing tables nor to run route-building algorithms when a node fails. Once, at each hop, all nodes closer to the destination are relay candidates. This provides a natural redundancy against node failure as any of those can forward the message. Only when a local minimum is reached, when no nodes closer to the destination can be reached, a void-detour algorithm must be started. This is not trivial in reactive geographical routing, as most of the proposed solutions use control packet exchange, directed messages, and knowledge about neighbor localization to overcome a void. So, they may not be the best choice for void-aware transport protocols on reactive geographical routing, as alternative route calculations must be carried out when some node fails.

In a typical WSN architecture, a single gateway (or sink) is responsible for all the communication between the application and the WSN devices. It configures a single point of failure, compromising the entire system if the gateway comes down or is overtaken by an intruder. Also, if nodes near to the gateway stop forwarding packets for any reason, the communication of the whole network is subject to greater delivery delays, or can even be interrupted. To solve these problems, some protocols use multiple gateways to reduce the traffic and power consumption of the nodes near the gateways, as well as to provide resilience against the failure of some gateways.

Besides handling communication problems and ensuring that data is timely delivered, the data integrity is also an important aspect concerning sensors as a whole, and specifically on WSN. Incorrect readings from a sensor can be transmitted to the application, leading to incorrect actuation decisions. These errors can be caused by hardware malfunction, sensing interferences, malicious software on the nodes, or on the gateway. Additional mechanisms must be developed to ensure that a compromised gateway is not able to send wrong commands to the actuator nodes in the WSN. It demands protocols where data and commands confidence can be checked.

Therefore, we conclude that the reliability of WSN mesh networks, running reactive geographical routing protocols, still have room for improvements, at the network infrastructure level and at the data fault detection aspect. This work aims to contribute to the development of this field, by answering the research questions proposed next.

1.1 RESEARCH QUESTIONS AND GOALS

This work investigates the dependability enhancement of fully-reactive geographical routing protocols for wireless sensor networks. In the several contexts, the problem of assuring the integrity and the correctness of data arriving at the application has to be addressed. As such, this work aims to answer the following research questions: (a) *How can the single point of failure problem, represented by the WSN gateway, be avoided in terms of failure and data integrity?* (b) *How can the void-detour problem in a fully-reactive geographical routing protocol be addressed, with little or no need of control and routing messages?* (c) *How to identify a faulty or malicious node, producing incorrect data, with minimum overhead in terms of data exchange and control messages between nodes?* and (d) *how can the application perform the correctness verification of the data received from the WSN without complex protocols?*

Therefore, the main objective of this work is to demonstrate that fault-tolerance in WSN can be increased by the combination of a multi-sink routing protocol and a data confidence attribution algorithm. This will be addressed in three aspects: The first concern is to avoid gateways to be a single point of failure, either in the sense of unavailability and the sense of malicious data manipulation. The second concern is to prevent voids in a mesh network from holding messages to be delivered, by using a greedy geographically routing algorithm. The third concern is to identify sensors that are producing incorrect data, either due to external interference, hardware malfunction, or malicious software. This identification should be achieved without the need for extra messages between nodes, in a distributed way. The first and the second aspects define a fault-tolerant fully reactive routing protocol for WSN using multiple sinks. The third aspect defines a confidence attribution mechanism for sensors that can be applied in other contexts, not restricted to the WSN context. It can also be applied in a centralized architecture.

1.2 ASSUMPTIONS AND SCOPE

In terms of WSN architecture, this work focuses on mesh networks, where each node is a candidate for forwarding messages from any other node, following the rules of a fully-reactive geographical routing protocol. Each node is aware of its position through a positioning device, fixed coordinates for fixed nodes, or using a trilateration algorithm. But, except for the gateways, nodes do not have to know other nodes' positions. Regarding the gateways, every node knows the localization of each gateway, which are in fixed positions while the network is up and running. Regarding sensors, each node can be monitoring several sensors, reporting the values to an application running on a conventional network infrastructure or the Cloud. The gateways – or sinks – are responsible for managing communication between sensors and the application. The WSN nodes can also be responsible for actuation, in response to commands received from the application. The networks are expected to be correctly dimensioned in terms of the number of nodes, their positions, as well as the sensing interval to be compatible with the network throughput.

1.3 METHODOLOGY

The following steps were devised to demonstrate that the goal of this work is reached:

1. To present the actual void detour and fault detection algorithms actually in use in different layouts of WSN.
2. To discuss the problems in applying these algorithms in mesh WSN, in terms of applicability or in terms of computational and time overhead.
3. To develop a routing algorithm that is able to deliver messages to multiple gateways, to avoid the *single point of failure* problem, and that is also able to perform void detour when facing empty regions due to transient or permanent node faults.
4. To develop a model that is able to assign confidence levels to data sensed by different sensors, without the need of extra message exchange between nodes.

This work was developed in the Software/Hardware Integration Laboratory (LISHA) in the Federal University of Santa Catarina (UFSC), and based on the research conducted over the last twenty years, supervised by Professor Antônio Augusto de Medeiros Fröhlich. Several previously published works by researchers that worked in LISHA paved the way for this work. The main research areas were routing protocols, time and space synchronization, and data representation.

1.4 CONTRIBUTIONS

The contributions of this work can be enumerated as:

- An analysis of the drawbacks of traditional single-gateway architectures to fault tolerance in WSN, and a discussion of the techniques for the handling of voids in fully-reactive geographical routing algorithms;
- A discussion of the fault detection techniques used to verify the correctness of sensed data, in terms of their impact on processing and network overhead;

- The introduction of a model and a fully-reactive geographical routing algorithm that addresses the problem of transient voids in mesh WSN, as well as the problem of a single gateway;
- The introduction of a model that allows the attribution of a *confidence level* to the data sensed by the nodes, based on data sensed at correlated sensors.

1.5 STRUCTURE OF THE DOCUMENT

The remainder of this document is organized as follows: Chapter 2 reviews the fundamental concepts about data fault detection and void handling in wireless sensor networks. Chapter 3 shows the current state-of-the-art of the proposed solutions on these two aspects of fault tolerance, presenting and discussing the related works. Chapter 4 describes the Fault-Tolerant Trustful Space-Time Protocol (FT-TSTP), able to handle multiple sinks and void regions. Next, chapter 5 describes the proposal and the evaluation of the confidence attribution mechanism, with the evaluation on some real-world datasets. The conclusions are presented in chapter 6.

2 BACKGROUND

This chapter reviews some of the base concepts used in the next chapters of this document. It makes a general overview, providing references to the reader for further information. Section 2.1 reviews the basic definitions of fault tolerance. Section 2.2 reviews the void problem in WSN routing and the main approaches to address this problem in the different architectures. Section 2.3 addresses several faults that can occur on a WSN. Finally, section 2.4 reviews the main techniques used in fault detection on sensor data acquisition.

2.1 FAULT TOLERANCE

The definition of *fault tolerance* is the property of a system to handle the occurrence of faults during its operation, without leading to a failure. To better understand this, it is necessary to understand the definitions of fault, error, and failure, as outlined by Tanenbaum & Steen (2007). Therefore, in this document, a ***fault*** is any kind of defect that leads to an ***error***, which corresponds to an incorrect (undefined) system state that may lead to a ***failure***, that is the observable manifestation of an error, when the system deviates from its specification, not delivering its intended functionality. Or, as stated by Laprie (1985): "*Upon occurrence, a fault creates a latent error, which becomes effective when it is activated; when the error affects the delivered service, a failure occurs. Stated in other terms, an error is the manifestation of a fault in the system, and a failure is the manifestation of an error on the service* .". The objective of fault tolerance is to prevent faults and errors from leading to system failure (absence of service, or wrong results), so it is required to employ techniques for dealing with errors and techniques for dealing with faults.

The ***dependability*** of a system is defined as "*the quality of the delivered service such that reliance can justifiably be placed on this service*" (LAPRIE, 1985). The author also states that a dependable system is achieved by the combined utilization of methods classified into **fault-avoidance**: preventing fault occurrence, by *construction*; **fault-tolerance**: provide, by *redundancy*, service complying with the specification despite the occurrence of faults; **error-removal**: minimize the presence of latent errors, by *verification*; and **error-forecasting**: estimate the presence, the creation and the consequences of error, by *evaluation*.

In the context of WSN and Cyber-Physical Systems (CPS), to increase the dependability of a system the methods above must be applied at different stages. Still following the concepts proposed by Laprie (1985), we can state that in the context of WSN, *fault-avoidance* can be approached at design time, e.g., by correct network architecture and traffic dimensioning. Redundancy of components as sensors and gateways, as well as providing alternative routes to messages, are aspects that increase *fault-tolerance*. *Error removal* can be achieved by tools that allow the verification of properties required by the application, as data freshness and availability. Finally, *error forecasting* can be provided by tools that analyze aspects as data correctness, timeliness of packets in the network, and energy consumption by nodes.

Regarding the causes of faults, in a WSN some different sources were identified by Souza, Vogt & Beigl (2007): *Node Faults* are caused by hardware failure or damage by a harsh environment, incorrect sensing caused by interference, battery depletion, or even latent software bugs. Node failure on specific locations can isolate an entire area of the network. *Network Faults* can be caused by node displacement, interference, or obstacles for the radio signal. Packet collision or path errors can also cause message loss. *Sink Faults* can make the whole network unreachable, as it configures a single point of failure if the network architecture provides no redundancy for this component. In remote deployments, for example, severe weather conditions can break satellite communication with the sink.

When considering the temporal aspect, faults are categorized by Koushanfar, Potkonjak & Sangiovanni-Vincentell (2002) into three types: *Permanent*, that are continuous and stable by nature, as hardware faults that completely disconnect a sensor node from the others. *Intermittent*, with sporadic manifestation due to unstable hardware, or even intermittent bugs in software. *Transient* or temporary faults, which commonly are the result of some environmental impact on the sensor's hardware or interference in the radio signal.

Two categories of tasks have to be performed to provide resilience in the occurrence of faulty situations, as stated by Souza, Vogt & Beigl (2007). ***Fault detection:*** the first step is to correctly identify that a given functionality or component is (or will be) faulty; ***Fault recovery:*** the next step is to prevent or to recover from the fault, avoiding the occurrence of a failure. Components redundancy is one of the most employed techniques for system recovery.

2.2 WIRELESS SENSORS NETWORKS AND ROUTING PROTOCOLS

Wireless Sensor Networks consist of several (maybe hundreds) of Sensor Nodes (SN), deployed in a region where they are expected to measure some quantity or send information about the occurrence of certain events. Several dimensions can be analyzed when engineering a WSN, as presented by Mohamed et al. (2018). The aspects to be taken into account, cited by the authors, are:

- **Measured Data**, which defines the frequency and accuracy needed to represent the data.
- **Target** type, such as tracking a specific type of entity, monitoring some measurable value, or verify the occurrence of specific events.
- **Data Transmission** determines under which circumstance data flows on the WSN. Sensing can be done timely, answering specific queries or on the occurrence of events.
- The **Node Placement** aspect encompasses where and how the sensors will be positioned on the field (region of interest). They can cover the body of a person (BAN), be placed underwater, underground, or on the ground. The nodes' placement can be carefully predetermined or be random. The monitoring can also cover 2D or 3D spaces.

- The **Service Area** describes the field of activity of the collected data. To exemplify, it can be environmental monitoring, precision agriculture, military applications, healthcare, or industrial applications, among others. The service area determines aspects such as freshness and reliability of data.

The communication between nodes or between the gateway and a node in a WSN uses a routing protocol to deliver the messages between them. Several aspects can be considered when designing such protocols, and can be grouped under the different dimensions, as stated by Al-Karaki & Kamal (2004), Sabor et al. (2017), explained below.

The ***route processing*** dimension determines how a route is built from the origin to destination. *Proactive* protocols make route calculation in advance, and keep information in memory. *Reactive* protocols only make routing processing when messages arrive and make different decisions upon each arrival. Finally, *hybrid* protocols can use both techniques together, e.g. on nodes of several hierarchy levels.

The ***protocol operation*** dimension is concerned with how a protocol operates, as a pattern for communication, hierarchy, delivering, and next-hop calculation. *Negotiation based* routing uses high-level data descriptors to eliminate redundant data transmission. Nodes also keep track of their resources to determine how messages are delivered to optimize overall resource consumption. The *multipath based* protocols try to send messages through alternative routes, ensuring delivery. *Query based* routing sends special packets (queries) to get data and to prospect network status. Collecting information about nodes' energy and communication load is also an essential part of *QoS based* routing, which uses this information to determine routes that get a balance between energy consumption and data quality. A routing protocol is called *coherent* if a minimum data processing (such as stamping, duplicate suppression, and others) is performed at the node level, that only transmits it to their destination. Alternatively, in *non-coherent based* routing protocols data can be processed at intermediate nodes, which can perform operations like aggregation, that can suppress or change characteristics from the original data.

The dimension regarding the ***network structure*** plays an important role, as it defines the characteristics of the Base Station (BS)s and the Sensor Node (SN)s. *Flat networks* routing implies that all nodes have the same capabilities and play the same roles in the routing protocol. On the other hand, the *hierarchical network* routing makes a distinction between nodes. Sensor nodes communicate with a Cluster Head (CH), that can be responsible for, e.g., make data or packets fusion and forward to a superior node. The *location based* routing protocols make decisions about routing based on the node's location, the origin, and the destination of the transmitted packets.

These aspects can also be somehow combined in the protocol implementation, leading to hybrid protocols that can use one or another aspect while making routing decisions.

2.3 NETWORK FAULTS IN WSN

Recently, several wireless sensor network routing protocols have adopted location-based solutions. The use of protocols that use addresses, like the Internet Protocol (IP), have to handle address assignment, maintain routing tables, as well as keep track of devices' location to determine *where* the data was produced. On the other hand, in location-based protocols, a node has only to know its location and, at most, the state of its one-hop neighbors. This *information locality*, without the need for requests and state propagation messages for route building, makes this kind of protocols save bandwidth and energy (KUMAR et al., 2017).

Some WSN protocols use route building routines. Usually, these routines are started by the gateway sending a query to the network. Each node that receives it has to make some decision about the route building process, based on information like network traffic or battery power from himself and from the nodes, collected while the query goes through the network. Every node keeps the information about the routes in tables, updates the control packet, and forwards it. When the process ends, each node knows where a message must be routed, when it has to send or relay it. Network faults are easily detected by the absence of data received by the sink or by the lack of communication with the neighbors. If available, the protocol can activate redundant nodes. Another solution to the absence of a node involved in a route is to find alternative routes to overcome a path interruption. In this case, the protocol must rebuild routing tables or run a route building protocol, possibly exchanging some control messages. This rebuilding procedure demands time and energy. When transient faults occur at the end of the rebuilding algorithm, the network configuration can be changed.

Alternatively, most of the called *location-based* protocols use *greedy forward* routing algorithms. Each packet is transmitted to the next node that is closer to the destination. This leads to the problem of *black holes* or *void areas*, when a node does not find any neighbor that can help the message to make progress to the destination. When this situation occurs, it is also said that a message has reached a *local minimum*. The void can be caused by poor node placement, a node displacement (either real or virtual), natural obstacles, or when some nodes suffer hard failures or are refusing to forward messages.

Several void-detour algorithms were proposed for location-based routing protocols. They can be classified into two categories: *right-hand rule* and *back-pressure rule* (AISSANI et al., 2010). The right-hand rule is based on graph traversing algorithms and is used to detect the nodes at the border of the void. These nodes are then responsible for routing messages around the void region. This can lead to high traffic and power depletion of these nodes. On the other hand, the back-pressure rule algorithms make nodes facing a black hole send messages backward. Doing so, the precedent nodes that receive the message back have to search for other alternative routes. By "learning" that some nodes are facing voids, the precedent nodes can preventively use the alternative routes and avoid the defective route.

To increase the dependability of a WSN network protocol, mechanisms to deal with transient or even permanent node faults are mandatory. If the faulty nodes make some other nodes be overwhelmed by the extra traffic, the application can cancel some requests, restricting the sensing to a subset of critical sensors. In other words, the presence of voids can lead to the need for a complete network capacity recalculation. The traffic over the WSN has to be carefully planned in order not to exceed the nodes' transmission capabilities, which can be radically changed by the extra traffic demanded by a void occurrence.

However, only a reliable transport layer is not enough to guarantee data correctness. Data must be delivered in strict timelines, but must also be correct. To a dependable WSN, the detection of possible incorrect readings on the sensors has to be addressed, as they can be subject to faults due to natural or induced interference, hardware errors, and other causes. In many systems as CPS, inaccurate values can lead to incorrect actuation commands, with undesired behavior, economic losses, or even catastrophic consequences. These aspects are discussed in the next section.

2.4 DATA FAULT DETECTION TECHNIQUES

The previous section discussed the data and commands transporting problems in a WSN. However, faults in sensors' data reading, which can be caused by defective hardware, interference in sensing, or malicious nodes forging incorrect values, are a kind of problem harder to address. The node acts as usual in the network, sending and relaying data from other nodes. But the data it produces is useless or even dangerous, as it can lead to erroneous actuation decisions by the application. A standard solution is the deployment of redundant sensors to compare readings. We used the term *data fault* to identify incorrect data, in opposition to *sensor fault*, that can also denote a sensor that turns inoperative due to hardware failure or battery depletion, or even one that continues to read sensor data but is unable to communicate with the WSN due to a radio failure.

Fault detection techniques on sensed data can be applied into different levels and classified in three different classes, considering the involved parties. In *self-diagnosis* methods nodes can identify faults on its components, in an autonomous way. In a *group detection* mode, nodes monitor the behavior of other nodes trying to detect errors. This kind of solution is mainly used in dense deployments of homogeneous sensors. Finally, when a *hierarchical method* is used, the detection is shifted to more powerful nodes, as cluster heads or sinks, which have access to more data, and use more powerful hardware to make complex computations (SOUZA; VOGT; BEIGL, 2007).

A slightly different taxonomy is presented by Khan (2013). In a *Centralized Architecture*, the base station is responsible for collecting information and controlling the whole network. A *Distributed Architecture* uses multiple stations throughout the network, each controlling a subset of nodes. Mobile agents are used for going through the network and to perform some diagnosis and recovery tasks in a *Distributed Agent-Based Approach*. In a *Hierarchical Architecture*, lower-level nodes (starting on the sensors) send reports to higher-level nodes, in a hybrid architecture between centralized and distributed. Cluster heads act as intermediate managers in most architectures.

Sensors' data faults can present different patterns and are classified by Sharma, Golubchik & Govindan (2010) in three main types. *Short* faults are single readings with values very different from normal readings (spikes). *Noise* faults are long duration readings with values randomly varying in different ranges around the correct value. Finally, *constant* faults are readings that differ by a (relatively) constant difference between real and sensed data. The authors also classify the data fault detection methods in four classes, as follows. Rule-based methods use domain knowledge to develop heuristic rules/constraints that the sensor readings must satisfy. These rules can be originated by some scientific law, or by long-term observations previously made. *Estimation methods* use spatial correlation in measurements of different sensors to define the "normal" behavior of a sensor. *Time series analysis based methods* use time-series forecasting to predict the sensor's value, based on a model built from previous readings of the same sensor, to define if it is faulty. *Learning-based methods* use previously read data, classified as normal or faulty, to infer and train a model able to classify newly collected data.

Rule-based methods rely on the expert's knowledge and are mainly hardcoded in the algorithms. They are suitable for domains with well-known behavior and dynamics. Estimation methods use statistical analysis and direct comparison of sensor readings of the same measurement. They are relatively simple and efficient in dense networks of homogeneous sensors. Time series analysis does not need data exchange with other sensors, saving communication, but when a deviation occurs, it can be hard to determine if it is a local fault or a change in the environment.

Changes in the environment can be deterministic in data correctness analysis. A rule or model that is unaware of the impacts of environmental changes will fail if some unexpected change occurs, leading the system to an inconsistent state. This change in the read values can then be wrongly classified as a fault, while it is just an correct, but unforeseen, change in the environment. Therefore, change detection is a crucial part of prediction models applied to non-stationary (or drifting) environments. As pointed out by Ditzler et al. (2015), "*In such nonstationary environments, where the probabilistic properties of the data change over time, a non-adaptive model trained under the false stationarity assumption is bound to become obsolete in time, and perform sub-optimally at best, or fail catastrophically at worst.*"

The change detection methods can be grouped into four main families: *Hypothesis Test* uses statistical techniques to verify the classification error of a fixed-length set of readings. The variation of the classification error is compared to the error of the training dataset. *Change-Point Methods* also use a fixed-length data sequence, analyzing all partitions of the data sequence to identify the instant where the data changes its statistical behavior, called change-point. The main drawback of this method is the high computational complexity. A *Sequential Hypothesis Test* inspects each incoming sample until enough evidence that a change has occurred is found. *Change Detection Tests* are specifically designed to analyze the statistical behavior of data streams sequentially. Most of them operate by comparing the prediction of absolute error or its variance to a specified threshold, which is hard to determine at design time. Some adaptive algorithms were proposed, using cumulative errors. The use of hybrid change detection is also proposed. A change detection test can be used in a first layer, followed by a validation layer that uses a change-point method (DITZLER et al., 2015).

Model update, performed after a change is detected, consists of retraining the model with new data. The main approaches are *windowing*, *weighting* and *random sampling* (DITZLER et al., 2015). Windowing approaches, as the name implies, use a window of N most recently values to retrain the model, considering the samples that already left the window as obsolete. Therefore, when the model is retrained, it reflects the current environment dynamics. The second class of methods considers all the available samples, but assigns different weights to each one, based on its age or relevance to the accuracy obtained with the data in earlier classification models. Finally, random sampling methods collect several samples in the previously read data, randomly, to form the training dataset to update the model. As the new environmental conditions are now present in training data, it is expected that the model will be able to correctly classify it. One of the main questions is if the model has to forget the oldest rules and reinforce the new ones, or if the learning has to be cumulative. In the former case, the model is entirely retrained with new data. This implies retraining the model every time a change is detected but leads to smaller models. In the latter case, incremental learning can keep past knowledge, but models are larger, demanding more memory and processing.

2.5 ARTIFICIAL NEURAL NETWORKS AS FUNCTION APPROXIMATORS

The Artificial Neural Network (ANN)s are subject of study for several years, and many types of neural networks were proposed in the literature, and new variations are still continuously proposed, aiming to solve specific types of problems. Some of the basic definitions are kept through the different approaches and will be briefly described here to lay out some foundations.

The basic architecture of an ANN used as a predictor is illustrated in Figure 1. It is composed of an input layer, an output layer, and some hidden layers. It computes an approximation of y based on a set of input *features*, meaning $\hat{y} = f(x_1, x_2, \ldots, x_n)$. The main differences between the ANNs are about the number of hidden layers, the connections between the neurons and layers, and the activation functions used in the neuron computations. If the values of each neuron of the previous layer – or each input feature – are used as input of every neuron in the next layer, the ANN is said to be *dense*, and is said *sparse* otherwise. The activation functions are defined in a way that the output is bounded to the $[0,1]$ or to the $[-1,1]$ interval, therefore the need to do a normalization process on the input values, to get all values in a unified scale. Several types of activation functions are proposed in the literature, for different application domains. The number of hidden layers can vary when used in different kinds of computations at each layer. A more detailed definition and discussion about the characteristics and applications of ANN can be found in the literature, as Murtagh (1991), Dreyfus (2005), and Walczak (2019).

It was stated by Hornik et al. (1989), Hornik (1991) that an ANN, with a single hidden

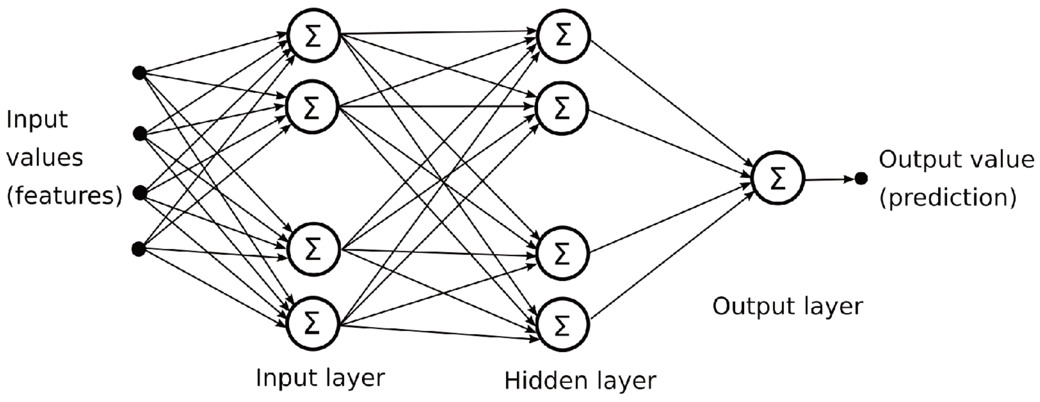


Figure 1 – Conventional Multilayer Neural Network architecture for a predictor.

layer and a feed-forward architecture, is capable of approximating any measurable function to any desired degree of accuracy, based on the Universal Approximation Theorem (CSÁJI et al., 2001). As stated by the authors, *"any lack of success in applications must arise from inadequate learning, insufficient numbers of hidden units or the lack of a deterministic relationship between input and target"* (HORNIK et al., 1989).

Multi-Layer Perceptrons (WIDROW; WINTER; BAXTER, 1988), as depicted in Figure 1, are among the most common architectures for regression and classification models. Despite the evolution of several distinct techniques for specific tasks, they are still powerful and suitable for several applications. Due to their simplicity and easy representation, they are a good choice for doing computation on resource-constrained devices.

2.6 SMARTDATA AND THE TRUSTFUL SPACE-TIME PROTOCOL (TSTP)

The *SmartData* construct proposed by Fröhlich et al. (2013), and the Trustful Space-Time Protocol (TSTP) proposed by Resner & Fröhlich (2015) are described in this sections, once they lay the basis for the proposals presented in this work. TSTP is an application-oriented, cross-layer communication protocol for CPSs on a WSN or on the Internet of Things (IoT). TSTP delivers trusted, timed, geo-referenced, SI-compliant data to applications through *SmartData*, which promotes a data-centric view of the network. First, *SmartData* and its parts are explained.

SmartData: Semantics, Interface, and Security

SmartData represents a self-contained piece of information, with metadata that defines its semantics, timing, spatial location, and trustfulness. It should be the only application-visible construct in the sensing/actuating platform, implicitly intermediating all system-level services with actuators and transducers. The modeled information is automatically updated from the network according to the parameters specified at instantiation time. Sensing and acting through SmartData is made by just reading or writing its value.

Instances of SmartData abstract local transducers or create local proxies of remote transducers. Sensed data is encoded as a SI quantity, which type is represented as a 32-bit identifier (IEEE 1451.0, 2007). Sensor readings are configured by the parameters `period` and `expiry`, defining the sampling periodicity and data's validity. For remote transducers, a region of interest for a given SI Quantity is specified as a $Space\text{-}Time\ Region(x,\ y,\ z,\ r,\ t_0,\ t_f)$, where x, y, z denote its center, r designates its radius and $[t_0, t_f]$ defines the time interval. All nodes that match the criteria specified by that region of interest will reply, producing SmartData objects in the specified time interval, at the specified periodicity.

When encapsulated in a network packet, SmartData includes the spatial coordinates and a high-resolution timestamp identifying `where` and `when` the data was produced (`Origin( x, y, z, t)`), the data's validity expressed as an absolute timestamp (`Expiry`), and a Message Authentication Code (`MAC`). Packets are signed and encrypted using the Poly1305-AES (BERNSTEIN, 2005) cryptographic MAC, with a key derived using Elliptic Curve Diffie-Hellman (ECDH) (RASO et al., 2015), demanding synchronized clocks to timestamp data. For Poly1305-AES, timestamps can serve as one of the input parameters; for ECDH, they can define a narrow time window for sensor deployment. ECDH can also use nodes' unique serial numbers or UUIDs as input. Data must be reciphered at the gateways to avoid sharing keys with the Cloud, but all WSN symmetric keys are kept in logs. Auditing a SmartData from the Cloud is as easy as recalculating and comparing the MAC.

Trustful Space-Time Protocol (TSTP)

TSTP Space and Time Synchronization

SmartData relies on nodes being synchronized in time and in space. TSTP implements the mechanisms needed for these synchronizations relying on its cross-layer design and some control information carried within its messages. TSTP achieves time synchronization via the *Speculative Precision Time Protocol*, which achieves sub-microsecond precision on an IEEE 802.15.4 WSN (RESNER; FRöHLICH; WANNER, 2016). Space synchronization within TSTP is achieved using a speculative version of the *Heuristic Cooperative Calibration Positioning System* (HeCoPs) (REGHELIN; FRöHLICH, 2006).

TSTP Medium Access Control and Routing

The TSTP Media Access Control (MAC), proposed by Resner & Fröhlich (2016) is the component within TSTP that is responsible for interfacing the protocol with the physical network in an energy-aware manner. Its design follows the general principles of RB-MAC (AKHAVAN; WATTEYNE; AGHVAMI, 2011): a long preamble composed of *microframes* is sent before each message, such that just one sender occupies the channel at every full period S; sensor nodes sleep for most of the time and, when they receive a message, nodes closer to the destination become relay candidates, using their distances to the destination to derive the time offset

$\delta(m)$ for *Clear Channel Assessment* (CCA) and retransmission. The relay candidate closest to the destination accesses the channel earlier and wins the contention, resulting in a greedy, fully-reactive, geographic routing.

Messages being routed by node i are kept in a queue Q_i. Each entry $e_Q \in Q_i$ represents a message m that is scheduled for transmission or retransmission. In addition to the message, e_Q also holds m's *Id* (extracted from the microframes), its *Expiry* T_ε (extracted from its header), its *Destination* and the message's offset $\delta(m)$ previously calculated.

TSTP uses implicit acknowledgment (ACK) to confirm the routing of messages. A node *i only* removes a message m from its queue Q_i when another message $m\prime$ with the same *Id* is overheard in the network, transmitted by a node that is closer to the destination. An explicit ACK is only used when the message reaches its final destination: that node must retransmit the same message (`Last Hop` = destination), just to acknowledge the last forwarder and any neighbors that might still have that message queued. TSTP considers that an unacknowledged message either suffered a collision or reached a geographic void. To handle this, the traditional random exponential backoff scheme is used, retransmitting the message until an ACK is received or it expires.

TSTP's transmission time offset $\delta(m)$ is made sensitive to other routing metrics by *distorting* space. A node running out of memory or consuming too much energy can *stretch* space, increasing its distance to the destination so that other nodes become more likely to win the contention to retransmit a message. Conversely, a node with a message close to expiring can *shrink* space, increasing the chances of winning the contention and transmitting it earlier. This distortion can be introduced by coefficient $\alpha \in [0,1]$ that multiplies $\delta(m)$ and defines how much any other metric influences the perceived distance, and the offset used for contention. A value of $\alpha = 0.5$ produces an offset equivalent to the real distance, an $\alpha \in [0,0.5)$ shrinks space, while an $\alpha \in (0.5,1]$ stretches space.

TSTP Greedy Forwarding Algorithm

To prevent the distortion coefficient from causing messages to be forwarded to an incorrect destination, the TSTP Greedy Forwarding Algorithm ensures that all messages queued on a node for transmission satisfy the Progress Property: there must be positive spatial progress towards the destination. This property can be written as $\forall j \forall i \{m_j i \in Q_i | D_i < D_j\}$, meaning that each message $m_j i$ from node j overheard by node i will be stored in node i's transmission queue Q_i if and only if the distance D_i from node i to the message's destination is smaller than the distance D_j from node j to the same destination. The algorithm presented by Resner & Fröhlich (2016) handles four possible cases:

Case 1: If m is already enqueued and is coming from a node closer to the destination, then m is an ACK, and e_Q is removed from Q_i and m is discarded.

Case 2: If m is already enqueued and m is coming from a node farther from the destination, then m is a retransmission and can be ignored.

Case 3: If m is a new message that came from a node more distant from the destination, then this node becomes a relay candidate for *m*. So, *m* is inserted into Q_i (creating an e_Q).

Case 4: If *m* is a new message that came from a node closer to the destination, then *m* is ignored, since the current node would not make positive progress towards the destination.

The TSTP routing algorithm is resilient to network failures due to transient nodes failures when the network design provides adequate node redundancy. Allied to the mechanism of *space distortion* upon resources restriction, it provides an adaptive routing protocol able to perform some simple void detour and let some overloaded nodes to deny packet forwarding services. But its main drawbacks are the inability to handle a *local minimum* when a void is created by node failure or displacement, as well as its dependency on a single gateway.

3 RELATED WORK

This chapter presents the related works regarding the main subjects addressed in this thesis. To increase the fault tolerance of a WSN, this work addresses two dimensions. The first dimension is to improve the delivery rate of the sensor's data on fully reactive geographical routing protocols, and also take care of *single point of failure* represented by the single gateway on traditional WSN architectures. Therefore it is mandatory to make a literature review of the void detour algorithms in the routing protocols, and also of protocols that deliver sensors' data to multiple sinks.

The second dimension is to verify the correctness of the data that arrives at the application, due to sensors' hardware or calibration problems, external interferences, or even data forging. So, a literature review on methods to identify faulty sensors in a WSN is necessary.

At the routing protocol level, the main aspects of void detour algorithms are investigated, in section 3.1. Solutions that use multiple sinks are described in section 3.2. Finally, the algorithms that are concerned with the fault detection on sensors' data are discussed in section 3.3.

3.1 VOID DETOUR

The *void detour* problem is well known in the WSN context and is addressed by several works. In geographic routing protocols, these algorithms can be classified into two groups: right-hand rule and back-pressure rule. The first makes the packet be routed through the border of the void, as in the Greedy Perimeter Stateless Routing - GPSR (KARP; KUNG, 2000). They have the side effect of high power consumption on the border nodes. Several enhancements and variations of GPRS are proposed. Among others, Wei & Yang (2010) proposed the Buffering Zone Greedy Forwarding Strategy (BZGFS) protocol to lower the priority of the nodes at the transmission range edges when recovery mode action is taken to avoid rapid power depletion. The protocol proposed by (XIAN; LONG, 2016), named EGPRS (Enhanced Greedy Perimeter Stateless Routing) divided the forward region into a set of sub-regions. Each sub-region has the same area, in the one with the highest average remaining energy is selected. The variation presented by Sun, Guo & Yao (2017), named Speed-Up GPSR (SU-GPSR), increases EGPRS's remaining energy function, calculating its *future energy* of the notes in each forward region. It also includes *mobile nodes* that move through the WSN and can carry packets in their buffers, while moving to areas with better transmission conditions due to signal strength or traffic intensity. Packets can be transmitted in a greedy mode (to still nodes) or in speed-up mode (to a moving node). The next node is selected based on a probability function that takes into account the number of nodes in each region, assigning greater weights for nodes that are mobile or that are able to do energy harvesting.

The protocol proposed by Qian & Zhang (2020) uses an enhancement of the right-hand rule to detect the boundary of a void. Then a node in the convex hull of the boundary is selected as a relay node, making the packets from the source run around the void to construct a path with less hop count.

The back-pressure rule, as in SPEED (HE et al., 2003) and FT-SPEED (ZHAO et al., 2007), send messages back to the upstream nodes, notifying the presence of a void and demanding the use of alternative paths. In their work, Aissani et al. (2010) presented three schemes in order to merge these two approaches, and the authors claim that they can be used with any geographic routing protocol. Each node has to know the neighbors' position and must be able to send messages addressed to a specific node. Special packets for *Void Detection* and *Void Maintenance* are employed in order to map the voids.

The protocol proposed by Theoleyre, Schiller & Duda (2009) uses a technique called *Reactive Deflection*. Each node, when unable to forward a message to a specific destination, updates a "blocked sector". It is described by the start and the end angle relative to the sender position, for which the node is unable to forward messages and send this information back to the sender node. When receiving this information, nodes insert it in a control list and stop sending messages to destinations that need to take a direction inside a blocked sector. If no node able to forward the message is found, then controlled flooding is started, until a viable node is reached.

Several research works have been made in Underwater Wireless Sensor Networks (UWSN). The Void Handling using Geo-Opportunistic Routing (VHGOR) protocol as proposed by Kanthimathi et al. (2017), merging geographic routing and opportunistic routing. The *closest sink* is defined by the distance and by the node's condition in terms of energy, queue length, and others. The algorithm handles convex and concave void in different ways. First, it tries to find neighbors that can find an alternative path (convex void). If it is not possible, it enters the concave void algorithm, making the packet go backward to find another way to the sink.

In the context of sparse mobile ad hoc networks (MANET), Hu & Sosorburam (2019) proposed a scheme that uses the geographic location and two-hop neighborhood information to define a forwarding node selection policy for determining the best relay candidates, which move towards target nodes. It reduces the transmission overhead and end-to-end delay against communication voids in infrastructure-less environments. The protocol proposed by Julie, Saravanan & Robinson (2019) for mobile nodes networks employs a neuro-fuzzy system that selects the next best forwarding node inside the communication area considering the residual energy, number of hops, the distance towards the sink, direction, and number of neighbors.

The aforementioned protocol types need to know the neighbor nodes' positions, to choose the next node to which a packet should be forwarded. Therefore, the algorithms do not apply to **reactive** geographic protocols. In this kind of protocol, nodes do not keep information about neighbors' positions. However, such protocols require that each node maintains information about its location. They can also require that nodes keep the system clock highly synchronized. When it listens to a message, the node decides if it can help in the message progress and, if it decides so, *reacts* to the message, forwarding it. Network packets include the last hop and the destination addresses, and it is the only information a node needs to decide about what to do with the message. No other information, like address or routing tables, is necessary.

In addition to not keeping and maintaining routing tables, these protocols also have the advantage of re-routing packets without exchanging extra control packets, as *void discovery* and *route maintenance* packets. They are also able to instantly find alternative routes (if available) when transient failures occur, and also show a fast response when additional node failures occur, changing voids' configuration.

In their work, Fan & Du (2015) proposed a solution to overcome the local minimum problem. They demonstrate that a message could make progress upon the destination node if the void boundary is a circle. Therefore, they proposed the Effective Bypass Void Routing Protocol based on the Virtual Coordinates Mapping (EBVRPVCM). Using a void discovery packet message, they calculate the void circle radius (O), based on the greatest distance between two edge nodes. This makes the edge nodes be inside the void circle, or right on the edge. In the next stage, the nodes assume a *virtual coordinate*, positioning themselves on the edge of the circle, as shown in Figure 2. After this stage, when a message reaches one of the border nodes b_n, there will always be a node closer to any destination, due to the isotropic circular geometry, and an eager geographic routing algorithm will still be able to route packets around the void.

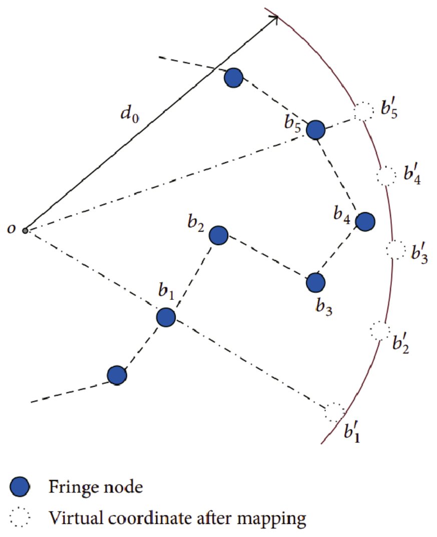


Figure 2 – Virtual coordinate mapping strategy (FAN; DU, 2015)

.

The REACT algorithm presented in (LIMA et al., 2017) assumes that the sink's signal can reach every sensor node, doing the sink-to-nodes communication in a one-hop way. The nodes don't know its exact location, but they can calculate the distance to the sink by using RSSI strength, assuming that a stronger signal means that the node is closer to the sink. The sensors' readings are transmitted to the sink in a multi-hop manner, where each node closer to the sink than the transmitter becomes a potential relay, in a self-voting mechanism. Data aggregation is used to reduce the number of messages. If a hole is detected the packet is marked as *hole-packet* and broadcasted until it reaches a node nearer than the *hole node* that initiates the process. The protocol works with one single sink, and there are no authentication or privacy mechanisms.

Discussion

Void-detour algorithms are developed mainly in protocols that have route building algorithms, and require that a node knows the location of each neighbor. Much of them also requires that the data packet has to be addressed to a specific node, instead of broadcasting it. The neighbors' addresses can be obtained by different methods, such as routing building packets, periodic beacons, or even by pig-tailing node addresses when forwarding messages. These characteristics do not apply to fully reactive geographical protocols, in which each node only knows its position and the sink's position.

Algorithms that make use of void-discovery packets have the drawback of the additional time needed to make the *void measurement*. In a situation where the void can change dynamically due to irregular interferences or transient node failures, it can be a considerable drawback, as the void configuration can change before the discovery process ends.

Unless the REACT algorithm presented by Lima et al. (2017), to our best knowledge there is no other proposal of a void-detour algorithm for fully-reactive geographical routing protocols in the literature. As a fully reactive routing protocol, the TSTP algorithm is unable to use these void-detour techniques directly. This is because every node, knowing only its position, the position of the last hop, and destination of every message, has to make local decisions if it will forward the received messages, or if it will drop it since it can not help the message to make progress.

3.2 MULTIPLE SINKS IN WSN

Several works have employed multi-sink approaches, mainly intending to provide alternative routes for packets to improve nodes' battery lifetime, or to prevent node isolation when the network suffers a partition due to communication problems. In their work, Yasotha, Gopalakrishnan & Mohankumar (2016) proposed the use of multiple mobile sinks that move to WSN regions where nodes have the most remaining energy. This aims to minimize the problem of power depletion on the nodes closer to the sinks. The sink chosen by each node is always the closest one. As sinks move through the WSN, the routing decision parameters must be constantly updated. In the solution proposed by Ozen & Oktug (2014), each node tries to send packets to the closest sink. Sinks' positions are known by all nodes. If a void is detected, a *flexible set* flag is set in the packets to enlarge the forwarder set, to circumvent the void, or to transmit packets to another sink.

In the work presented by Carlos-Mancilla, Lopez-Mellado & Siller-Gonzalez (2015) an algorithm builds clusters and backbones, connection nodes to trees with a sink as root. There are N trees, one for each sink, and each node belongs to a unique tree. Trees are rebuilt on nodes or sinks failures.

Pheromone levels (ant colony algorithm) are used by Zhang & Dong (2015) to determine multiple routes from nodes to sinks and from sinks to nodes, based on QoS parameters. Again, each node delivers its messages to the sink whose path offers the best QoS parameters. This is the same strategy of the Ant-based Dynamic Hop Optimization Protocol (ADHOP) proposed by Okazaki & Fröhlich (2011), which also uses pheronome leves determined by QoS parameters to make routing decisions. The protocol is fully reactive, able to dynamically adjust itself to network changes. However, the current version works with a single sink. The BAMBi protocol proposed by Misra, Bhattarai & Xue (2011) sends a copy to every known sink separately. Routing trees with sinks as roots are built with each node belonging at least to N trees (where N is the number of sinks). Trees are rebuilt on a node or sink failure.

The work presented by Khoufi, Minet & Laouiti (2016) addresses the relay node placement problem from a fault-tolerance perspective. The proposed solution aims at minimizing the number of relays needed to connect a set of points of interest to a sink by placing them on the vertices of a triangular lattice inscribed in a minimal rectangle that encompasses all the points. Their placement outperforms those based on straight lines or Steiner-points, being a powerful design tool for WSNs when nodes' placement can be planned.

An algorithm to optimize data aggregation in WSN with multiple sinks is presented by (Yestemirova; Saginbekov, 2018). Two data aggregation algorithms are proposed, based on Minimum Spanning Tree and the Shortest Path Tree. *Virtual Sinks* make data aggregation inside the WSN, and send the collected data to another *Super Virtual Sink*. These special nodes then send the aggregated data to all sinks of the network.

The protocol proposed by Mukherjee, Amin & Biswas (2019) uses smart three-sector antennas to get better communication between nodes. It also uses a round-robin schema to choose the neighbor to forward a message, and different routing algorithms are used by nodes up to two hops away from a sink and by the nodes that are more distant from a sink. A message is delivered when it reaches one single sink.

The Multi-sink distributed power control (MSDPC-SRMS) algorithm, proposed by Wei et al. (2019) considers each sink as a cluster head node. The optimal transmission range is negotiated among the sinks to get optimal connectivity for each node, based on the coverage quadrangles needed to cover all nodes, and nodes choose their sink using a Voronoi scoping algorithm. Each node will send data to one sink. The multi-sink protocol proposed by Gangwar, Tyagi & Soni (2020) uses main and alternative paths from each node/sink pair. Routes are built in a three-phase process and updated when nodes reach a user-defined critical battery level. Packets are considered delivered when reaching one sink. The destination sink can change while the packet is retransmitted by intermediate nodes in the route tree. Multi-sink protocols, with each node choosing a sink based on energy or timeliness criteria, were proposed by several other authors, as Carlos-Mancilla, Lopez-Mellado & Siller (2018), Carlos-Mancilla, López-Mellado & Siller (2019), Ramadan, Alreshidi & Sharif (2020), Rajput & Kumaravelu (2020), Vasavada

& Srivastava (2020). Differences are mainly about clustering algorithms and reconfiguration procedures on a node or a sink failure.

Discussion

All the mentioned solutions that use multiple sinks try to minimize the impact of nodes and gateways failures, to get a better energy balance, or to lower delivery times. They consider a message delivered if it reaches one sink. None of them proposes advanced integrity checking for messages, so they are susceptible to attacks that can modify the message content while it is transmitted over the network, or even by a sink before the data is delivered to the application. If one sink is taken over by an intruder, the application would have no mechanisms to determine which sink is the malicious one, as different values can arrive from the same node if it uses alternated sinks due to routing decisions.

Therefore, there is a lack of a solution that increases fault-tolerance on WSNs regarding redundant sinks on fully reactive geographical routing protocols. This kind of solution would prevent a subset of the nodes, or even the entire WSN, from becoming unreachable on a sink fault. Additionally, an agreement protocol between the sinks can also provide fault tolerance against data forging by a malicious sensor node, and even by a malicious gateway.

3.3 SENSORS DATA FAULT DETECTION

Sensors provide data to monitoring systems that are responsible for registering it and, most of the time, feed decision-making programs that can send commands to actuators in response to specific situations. Therefore, the correctness of the sensors' readings is crucial to avoid failures in these kinds of systems. A standard solution to increase the dependability of a wireless sensor network is the deployment of redundant sensors to compare readings. It demands a dense deployment of sensors to allow the identification of the faulty sensor among the correct ones by contrast.

To avoid the dense deployment of sensors, several alternative methods were proposed to verify the correctness of the sensors' data. These methods can be *centralized*, running on the server, *distributed*, running on the sensor nodes or *hierarchical*, when some particular nodes — like the cluster heads — collect data from a set of nodes and run the diagnosis algorithms (KHAN, 2013).

The errors in the sensor's readings were classified by Ni et al. (2009) and by Sharma, Golubchik & Govindan (2010) in four main types. *Outliers* are isolated readings that differ significantly from normal readings expected by the models. The *spike* or *peak* errors are readings that deviate too much from the normal values for a certain period of time. They are composed of at least a few data samples, and not an isolated reading as an outlier. The third type is the *"stuck-at"* error when the readings present a zero (or very little) variation for a period greater than expected. The amount of time in which the reading has to be "flat" to be considered a "stuck-at" must be determined for each type of sensor. Finally, the *high noise* or *variance* is the occurrence of unusually high variance in the sensor's readings, in such a way that it differentiates it from the usual noise which appears in many sensor types.

Several techniques were proposed to identify faults in sensors' readings. Recurrent Neural Network (RNN) were used by Moustapha & Selmic (2008) to predict sensors values, based on previous readings of the sensor and its neighbors. The authors made the assumption that all sensors are of the same type, and that the difference between the values read by neighbor nodes is bounded by a constant value of ε. After building and training the RNN, the difference between the sensed value and the predicted value is compared to a threshold η. If the difference is greater than η, the node is considered faulty.

Time series analysis combined with a voting schema is presented by Nguyen et al. (2013), assuming that all nodes sense the same phenomenon, neighbors can communicate directly, and faults occur interrelatedly. The Auto-Regressive Moving Average (ARMA) model is applied, with p auto-regressive terms and q moving-average terms. The calculation of the regression formula's parameters is done on correct (validated) readings, before the sensors' deployment. In the voting phase, readings of the neighbors are collected. Then the median of these readings is calculated and compared with the actual reading of the sensor. If the difference is larger than a threshold τ, the read value is considered faulty. Faulty values are not included in the node's history, to not disturb the moving average used in classification.

Statistical analysis of sensed values is another method applied for fault detection. The Distributed Fault Detection (DFD) algorithm, proposed by Li et al. (2015), uses the statistical technique of Grouping Test (GT), that identifies a small number of defective items in a large population. Sensors are supposed to be uniformly and independently distributed in a 2D space. A sensor is considered defective if its readings differ significantly from other sensors' readings. All nodes exchange their readings with the neighbors, run an *outlier test*, and broadcast the result. These steps are repeated for L rounds. In the last phase, every node updates its status, based on the results obtained from data exchanged in the previous rounds, by comparing to a threshold γ. The values of L and γ determine the trade-off between *false alarms* and *no detection* and are hard to determine for large networks. The authors also proposed an adaptive algorithm that dynamically determines the values of L and γ.

A modified three-sigma edit test is proposed by Panda & Khilar (2012) and by Panda & Khilar (2015), using the ratio between the current value, the median, and the normalized median absolute deviation of the last n readings. If this ratio is greater than 3, it considers the reading an outlier, and that the sensor is faulty. The spatial-temporal correlation of sensor data in agricultural systems is used by (BAE; LEE; SHIN, 2019). Data from homogeneous sensors are classified by the gateway, using statistical methods to analyze the variation of a sensor in relation to the variation of the set of correlated sensors. It assumes that more than half of the sensors are sensing correct values to be able to detect the faulty ones. Variation analysis is also used by (JIA; MA; QIN, 2019) to detect data faults locally at the sensors, avoiding message exchange for sensor's diagnose. The authors assume that homogeneous sensors, in the same area, will sense similar data.

The Distributed Bayesian Algorithm (DBA) is presented by Yuan, Zhao & Yu (2015). Sensors of the same type calculate the probability of being faulty in three steps. First, the nodes periodically exchange their values and probabilities to the other nodes in the radio range R. Sensors are in the same state (*faulty* or *not faulty*) if the difference between their readings is smaller than a specified threshold r_t. The Bayesian formula is used to calculate the fault probability of each node. In the second step, adjustments are made to avoid that a good node surrounded by faulty nodes becomes faulty (and vice versa). In the third step, nodes with a fault probability higher than a given threshold τ will send a warning message to the sink.

A distributed fault detection based on the Hidden Markov Model is presented by Saihi et al. (2015). Each node uses the difference between its value and the values of the neighbors, determining its state as *Possibly Normal* or *Possibly Faulty*. In the sequence, the probability of the node to change its state from *good* to $faulty$ (or vice versa) is calculated using a transition matrix built from the results obtained in the first step. The algorithm assumes a WSN composed by dense deployment of sensors of the same type, to directly compare readings.

A fault-tolerant algorithm for event detection in WSNs called Spatiotemporal Correlation Based Fault-Tolerant Event Detection (STFTED) is proposed by Liu et al. (2015). The presented scheme uses a Location-Based Weighted Voting Scheme (LWVS) to get a decision from the involved nodes. It explores the spatiotemporal correlation between sensor nodes, assuming a dense deployment of sensors of the same type, to detect events. It also assumes a mean value m_n representing a normal reading (or absence of event) and a mean value m_e representing the presence of an event. At the node level, the readings of the neighbors are weighted based on their distance. Closer nodes have a higher influence on the estimation function and vice versa. In this step (LWVS), an estimator R_n for the state node n is calculated, which can be inaccurate. So, the second step (STFTED) uses the Bayesian formula to calculate the probability of a node to be faulty, based on the estimation of other nodes in the same *fault range*. If the majority of nodes have a high likelihood of normal readings, abnormal readings are considered faulty, and the contrary is also true.

In the work presented by Titouna, Aliouat & Gueroui (2016), an algorithm named Fault Detection Scheme (FDS) is proposed, based also on a local step, carried out on sensor nodes, and a second decision step that runs on the cluster head nodes. On the local step, each node calculates the probability of node i being faulty (Joint Probability, or PJ_i), using a Bayesian Network that uses the energy level and the sensed data of node i (EL_i and SD_i). If the probability of being faulty exceeds a threshold δ, the node classifies itself as *Possibly Faulty - PF*. Otherwise, it classifies itself as *Possible Normal - PN*. Each node sends PJ_i and its decision to the CH, which executes the second step of the scheme. The CH maintains a table called Probability Join Table (PJT) with the PJ of every node from the cluster. For each node i, the PJT contains the previous and the actual PJ_i. When the node decision is PF and the difference between PJ_i^{t+1}, and PJ_i^t is greater than a threshold γ_2, then the node is considered faulty. Otherwise, it is considered a false alarm. Based on FDS, the authors proposed a Distributed

Fault-Tolerant Algorithm (DFTA), where they describe a scheme to make the elimination and the recovery of faulty nodes (TITOUNA et al., 2017).

A method based on logistic regression is proposed by Zhang, Zhao & Nakamoto (2017), using a logistic regression function. The parameters of the function are obtained in the model construction (called Learning Step) executed on the sink, using data from all sensors in this step. After training the model it is sent to the nodes, where it is executed. The value predicted by the model is compared to the one read by the sensor. If the difference is greater than a threshold, the node is classified as faulty, and it is classified as normal otherwise. No method to determine the threshold value is proposed, and the authors decided it is '*based mainly on experience and intuition.*'

A distributed fault detection for WSNs in Smart Grids is presented by Shao, Guo & Qiu (2017), based on credibility and cooperation among sensors. Each sensor evaluates its status as suspicious or not, based on the mean and the variance of a window containing the last k sensed values. A healthy sensor keeps the variance bounded. When a sensor detects itself as suspicious, a *Diagnostic Request* is sent to the neighbors, and the *Diagnostic Response* messages are used to determine the node's state. After receiving the response of sensors in an area determined by a radius R, the node can update its probability of being healthy or faulty.

ANNs are used by Swain & Khilar (2017) to detect and classify different faults in a WSN of homogeneous sensors. The work assumes that the sensors are uniformly distributed and with a set of anchor nodes (cluster heads) that have broad radio range and are fault free. A genetic algorithm combined with gradient descent is used to train the neural networks. These neural networks classify the state of the nodes. After node classification as secure or faulty, the last ones are disconnected from routing paths.

The use of large models, with several inputs and complex interconnections, may not be the best choice for prediction models. When data is sparse or with complex interactions, the use of ensembles can obtain better prediction results. As each classifier explores specific competence domains, an ensemble outperforms a single classifier that tries to handle all the inputs in a unique algorithm (WOŹNIAK; GRAÑA; CORCHADO, 2014). In this sense, Curiac & Volosencu (2012) used several classifiers to determine if the readings of sensors in a WSN are faulty or normal. For each sensor, the ensemble uses its historical readings and from the nearest neighbor nodes as input. All classifiers run on the gateway, in a centralized way. The outputs of the classifiers are then joined in a single decision and an estimated value, using weighted majority voting.

A centralized approach, using Kalman filter data fusion to train models on the gateway with faulty patterns in data is proposed by Biswas et al. (2019). In addition, an implementation of an Extreme Learning Machine (ELM) is used as a classifier, obtaining high prediction rates with low communication overhead. Data is considered correct or incorrect based on the distance between the read value and the predicted value, compared with a threshold χ defined by the user. The use of Non-Negative Matrix Factorization (NMF) applied to the spectral representation of data in an agricultural context was proposed by Ludeña-Choez, Choquehuanca-

Zevallos & Mayhua-López (2019). This allowed getting a good representation of data with a reduced number of features. The NMF method is applied to sensors' data at the gateway, in a centralized approach. On a hierarchical approach, Dao et al. (2020) applied Improved Multi-verse Optimizer (IMVO) and Feedforward Neural Network (FNN) on the Cluster Heads of a WSN to detect data faults.

A pure formal method, based on Timed Petri Nets was proposed by Wang, Wang & Chen (2019), using trust factor values as valid communication, data similarity, clock synchronization, history data, and remaining energy to determine the firing probability of transitions in Petri Nets built to model the network. The model is able to predict faults in the WSN, running at the gateway or at the application level.

The belief function-based decision fusion approach was proposed by Javaid et al. (2019), with the enhancements of four classification algorithms: K-Nearest Neighbor, Extreme Learning Machine, Support Vector Machine, and Recurrent Extreme Learning Machine. All classifiers were enhanced by belief functions. At the sensor nodes, the Basic Belief Assignment is made, and some belief and plausibility parameters are computed using belief function theory. The local decision and the reliability measures are sent to the fusion center, to make global decisions by merging results from different sensors, using the enhanced classifiers. This solution configures a hybrid approach, with local and centralized steps in the fault detection process.

The use of a nature-inspired approach called Improved Fault Detection Crow Search Algorithm (IFDCSA) was proposed by Gupta et al. (2019), and also evaluated Decision Trees, Random Forest, and K-Nearest Neighbor algorithms to classify the results, with better results obtained with the Random Forest algorithm. The algorithm is applied to homogeneous sensors and executed in a centralized way.

The use of the Grey Wolf Optimization Support-Vector Machine was proposed by Karmarkar, Chanak & Kumar (2020). The optimization is based on the hunting and preying of the grey wolf, to find the best position particle in the SVN algorithm. The classifier runs at Cluster Heads, to avoid communication overhead, in a hierarchical architecture. Fuzzy Logic was used to verify data faults on heterogeneous WSN by Masdari & Özdemir (2020). The coverage of the sensors is taken into account, to avoid that isolated nodes make incorrect classifications. Messages with sensor data are broadcast to neighbors in a *sensing range*. Each node checks the local value using a fuzzifier, taking into account the value difference, the distance of the data's node, and the number of coverage points. The next step is to apply the fuzzy rules and the defuzzier, which produces a score of the local value. Then a voting-based fault detection is used to determine the sensor node status.

Discussion

Most of the presented solutions for data fault detection on sensors use a model that demands a dense deployment of sensors of the same type, comparing values that should be very similar. Table 1 shows the main characteristics of the reviewed solutions, in terms of detection technique used, the architecture of the solution, the diagnostic strategy, and the type of the deployed sensors. The main difference between them is the prediction model, varying from statistical analysis, probability tables (mainly Bayesian methods), to predictors built using different approaches. The main characteristic of them – except for the solution proposed by Panda & Khilar (2012) that uses the 3σ test – use extra packet exchange in voting or diagnostic rounds. Even the method proposed by Jia, Ma & Qin (2019) uses statistical data that must be exchanged with the neighbor nodes to make the local diagnosis, and assumes homogeneous sensors.

The need for communication between nodes to diagnose a faulty node comes from the fact that when a test of the actual sensor's value reports a difference, there is always the possibility that the error is not from the sensor but the evaluation model. Extra packet transmissions in *diagnostic phases*, as in the work of Shao, Guo & Qiu (2017), exchanging test results like in the proposal of Li et al. (2015) and Jia, Ma & Qin (2019), or even in voting rounds as presented by Masdari & Özdemir (2020) result in communication overhead and introduce delays every time a node's failure occurs.

The majority of the fault detection techniques are built on WSN composed by the deployment of sensors of the same type. Although it is a possible and realistic WSN configuration, these models are not applicable when sensing a variety of values. Replication is essential to ensure reliability, but it also raises costs and can cause communication interference and network overload.

Once fully-reactive geographical protocols do not have cluster heads in their architecture, they can only use distributed or centralized architectures for fault detection. The time to send data to the sink and wait for the response can lead to problems, for example, when the *WSN* is employed in a *CPS* that needs to make a local decision and react to some event. There is a need for a solution that can make a local diagnostic better than an *outlier test* like the 3σ or any similar, but without requiring extra diagnostic or voting messages.

The fault detection scheme proposed by Titouna, Aliouat & Gueroui (2016) and Titouna et al. (2017) uses a hybrid algorithm, with a step carried out on the node, and another performed on the cluster head, which has higher processing and memory capabilities. In opposition to the work of Swain & Khilar (2017), the proposal of this work is not to automatically disconnect faulty nodes, but allow each node to determine its confidence in the sensed value. This can be used to identify a faulty node, as well as to detect data corruption by intermediate nodes. Nevertheless, the application can also cancel the interest in faulty nodes through specific commands, making them stop sensing until they can get maintenance.

Despite avoiding extra messages is an important feature for a distributed data fault detection algorithm, a complete isolated node is unable to determine the correctness of the sensed data. Some comparison or confirmation mechanism is needed, so even more complex algorithms, as the one proposed by Jia, Ma & Qin (2019) or the one proposed by Masdari & Özdemir (2020) relay on neighbors' data to decide on the status of the data read by a sensor. The approach used by Jia, Ma & Qin (2019) seems to be the most suitable, as data from closer

Table 1 – Sensor's Fault Detection Algorithms

Author	**Technique**	**Architecture**	**Diagnostic**	**Sensor Type**
Moustapha (2008)	Recurrent Neural Network	Distributed	Group / Voting	Hom.
Panda (2012)	3-sigma test	Distributed	Local	Hom.
Ngyen (2013)	Statistical (ARIMA)	Distributed	Voting	Hom.
Li (2015)	Statistical (Group Test)	Distributed	Broadcast / Compare	Hom.
Shao (2017)	Statistical (Variance Analysis)	Distributed	Local / Diagnostic Msgs.	Het.
Bae (2019)	Statistical (Variation)	Centralized	Gateway	Hom.
Jia (2019)	Statistical (Variation)	Distributed	Local / Neighbors' Data	Hom.
Yuan (2015)	Distributed Bayesian	Distributed	Diagnostic msgs.	Hom.
Saihi (2015)	Hidden Markov Model	Distributed	Broadcast / Compare	Hom.
Liu (2015)	Bayesian	Distributed	Group / Weighted Voting	Hom.
Titouna (2016)	Bayesian Network	Hierarchical	Cluster Head	Hom.
Dao (2020)	IMVO and FNN	Hierarchical	Cluster Head	Hom.
Volosencu (2012)	Classifier Ensembles (5)	Centralized	Ensemble Weighted Voting	Hom.
Zhang (2017)	Logistic Regression	Distributed	Classification	Het.
Swain (2017)	ANN / Genetic Algorithms	Hierarchical	Local / Cluster Head	Het.
Karmarkar (2020)	GWO-SVM	Hierarchical	Cluster Head	Hom.
Masdari (2020)	Fuzzy Logic	Group	Voting	Het.

nodes can be obtained in *'passively'* manner, just by listening to the neighbors' communication.

4 FAULT TOLERANCE AT THE COMMUNICATION LAYER

Parts of this chapter appeared earlier in ***Byzantine Resilient Protocol for the IoT*** (FRöHLICH et al., 2018) and ***FT-TSTP: A Multi-Gateway Fully Reactive Geographical Routing Protocol to Improve WSN Reliability*** (SCHEFFEL; Fröhlich, 2018)

This chapter describes the proposed architecture, whose objective is to improve network fault-tolerance on WSN, providing a more reliable communication layer for the IoT or CPS. A multi-sink routing protocol named Fault-Tolerant Trustful Space-Time Protocol (FT-TSTP) is presented, addressing the problems of node failures in packet forwarding on a fully reactive geographical routing protocol. By delivering messages simultaneously to redundant gateways, the problem of a *single point of failure*, represented by a single gateway on common architectures, is addressed. When running an agreement protocol between the sinks, interconnected by a reliable fast network, the proposed architecture also provides a solution against the intrusion of nodes and sinks.

The FT-TSTP protocol is an extension of the Trustful Space-Time Protocol (TSTP) proposed by Resner & Fröhlich (2015), and both protocols are based on the *SmartData* construct presented by Fröhlich et al. (2013). The new protocol is the result of a collaboration between the Software/Hardware Integration Lab (UFSC/LISHA) and the Interdisciplinary Centre for Security, Reliability, and Trust from the University of Luxembourg.

The chapter is organized as follows: first, the context and motivation are discussed in Section 4.1. Next, Section 4.2 describes the proposed protocol, followed by the evaluation on different scenarios in Section 4.3. Finally, Section 4.4 presents an evaluation and discussion of the results.

4.1 INTRODUCTION

More and more Cyber-Physical Systems (CPS) are being interconnected, particularly in the realm of Wireless Sensor Networks (WSN), Industry 4.0, and the Internet of Things (IoT). However, due to cost constraints, these systems are often being built around old Internet technology that was not designed considering requirements such as timeliness, positioning, security, fault-tolerance, and trustfulness, which are essential for the CPS domain. The proposed solution aims to provide higher data availability for CPS applications by specifically enhancing fault and intrusion tolerance of WSN architectures.

WSN architectures can exhibit failures of benign or malicious nature, occurring either at the level of individual devices (e.g. sensor, actuator, machine) or at the level of gateways that connect such devices to the traditional Information Technology (IT) infrastructure (e.g. servers, Cloud, Internet). For example, many battery-operated sensors in a WSN are deployed in areas that are physically hard to reach and maintain. As a result, many sensors and actuators e.g., in ambiental monitoring, are often left unattended. Moreover, many devices in an industrial environment are subjected to extreme conditions. Consequently, such devices are subject to failures, and even some are likely to be exploited by motivated attackers. Similar threats can also jeopardize gateways. Gateways are typically implemented on computers running an ordinary operating system such as Linux, usually connected to the Internet. Hence, a gateway is at an even higher risk of being attacked compared to individual devices.

Adding to that, typical WSN architectures, on top of which CPS applications are established, suffer from centralization. Namely, such architectures often rely on a single gateway (or sink) that connects all system devices or parts of them to the IT infrastructure. A single gateway is a component that hinders the system's availability. A failure or a spurious behavior of this gateway, relative to malicious attacks, may compromise the data from the entire sensor network.

This work addresses the above threats by presenting a fully established solution, encompassing architectural and algorithmic contributions. First, it proposes redundancy at the deployment level of both sensor nodes (devices in general) as well as gateways/sinks. Hence, the increased distribution of all network components aims to eliminate centralization.

As already presented in section 3.2, the use of multiple gateways was mainly proposed to achieve performance scalability, minimized energy consumption, and to deal with network/-gateway failures. However, messages are delivered to just **one** of the available gateways considering, for example, the route that provides better energy balance or the fastest delivery. In case of communication interruptions or gateway unavailability, these solutions re-route their traffic to an alternative gateway.

In brief, these solutions fall short in terms of resiliency: they do not provide any data guarantees when a gateway is compromised by an attacker, in which case an attacker can tamper with data before delivering it to the application. The objective of this work, unlike these existing solutions, is to enhance the system's resiliency to faults and intrusions which yields in turn better system and data availability.

To provide system robustness and resilience, this work proposes protocols that make use of the proposed sensor and gateway architectural redundancy. First, at the level of the WSN devices (sensor nodes, actuators, etc.), a routing algorithm is proposed, named *FT-TSTP*, which uses the SmartData concept. The routing protocol aims to achieve data transfer resiliency to void regions formed by interferences, malfunctioning, or displaced nodes in the WSN. FT-TSTP utilizes geographic routing to forward packets towards multiple sinks, without relying on route building techniques, announcement packets, or routing tables in the node's memory. As stated before, the FT-TSTP algorithm is built on top of the Trustful Space-Time Protocol (TSTP) proposed by Resner & Fröhlich (2015), which achieves energy-efficient and timely data transfers in single-gateway WSN architectures.

Second, at the level of gateways, which might receive different data, the proposal is the use of an intrusion-tolerant synchronization protocol. Initially, the *ByzCast* protocol presented in Fröhlich et al. (2018) was used. Any other agreement protocol can be used to verify data integrity among the several gateways. This architecture allows correct and timely actuation signals to be dispatched to actuators despite having some gateways crashed or even compromised by malicious attackers, as they can be isolated by the agreement.

4.2 FAULT-TOLERANT TRUSTFUL SPACE-TIME PROTOCOL (FT-TSTP)

FT-TSTP is a communication protocol that transports data from devices in the WSN (e.g. sensors, actuators, machines) to gateways connected to an IT infrastructure. FT-TSTP is Byzantine-resilient, i.e., it can transport data despite device and gateway failures (benign and malicious). Redundancy of nodes in a WSN is a major strategy to achieve high data availability and FT-TSTP certainly also depends on node redundancy in this sense. Nevertheless, node redundancy is not enough if the data produced on the WSN is meant to be used for network-wide CPS applications running in the Cloud or over the Internet. WSN sinks (or IoT gateways) must also be replicated to avoid being single points of failure. Therefore, FT-TSTP assumes multiple sinks and defines a novel algorithm to forward SmartData messages.

FT-TSTP's *Multi-Sink Greedy Forwarding* Algorithm (1) requires a bootstrapping slightly different from that of the original TSTP. Instead of providing newcomer nodes with the coordinates of a single sink, an array of sinks S is given. The number of sinks in this array is configurable and depends on the desired level of fault tolerance. For example, at least $3f+1$ sinks are needed to be able to tolerate f compromised sinks. So, f designates the maximum number of faulty (Byzantine) sinks that can be tolerated while still being able to reach any form of agreement on the data delivered by sinks (DOLEV, 1981). The sinks are ordered at deployment and that order is not modified during a cycle of operation, so all nodes agree on the sinks' array order. Special commands can be used to disable/enable specific gateways, and also to add new gateways to the network. However, dynamic configurations would likely be hard to be validated in terms of the network capacity dimensioning. For example, an upper limit for the number of sinks can be determined, to correctly dimension the network, and some sinks can be enabled or disabled dynamically by special commands broadcasted to the nodes, yet with the guarantee that time requirements will be met. However, in this work, we only evaluated static networks with a fixed number of sinks.

Besides modifying the bootstrap procedures, the original format of microframes and messages from TSTP was also modified. Microframes now carry the coordinates of the last hop, and therefore messages no longer need to carry the distance from the last hop to the sink, as now there are several sinks and, consequently, several distances. They now also feature a bitmap (`Sinks`) designating the sinks to which the following message is to be delivered. This bitmap has one bit for each of the N_S sinks defined at deployment-time. The resulting formats are depicted in Figures 3 and 4 (the number of bits in spatial (*sb*) and temporal (*tb*) coordinates are defined by the `Spatial Scale` field in the microframe and the `Temporal Scale` field).

With these modifications, Algorithm 1 is able to implement a greedy, fully-reactive, geographic routing policy similar to the one in the original TSTP, but with multiple destination sinks. Each node that hears any of the microframes that precedes a message m decides whether or not to wake up to listen to it after calculating its distance to ALL the destination sinks whose

Algorithm 1 FT-TSTP Multi-Sink Greedy Forwarding

```
 1: procedure MultiSink_Greedy_Forward($m$)
 2:     for each $s \in S \cap m.Sinks$ do
 3:         $queued \leftarrow false$
 4:         $isNew \leftarrow true$
 5:         $toACK \leftarrow false$
 6:         for each $e_F \in F_i^s$ do
 7:             if $e_F.id = m.id$ and $e_F.Origin = m.Origin$ then
 8:                 if $m.LastHop = \infty$ and $e_F.LastHop \neq \infty$ then
 9:                     // $m$ entered recovery mode and must be removed from $F_i^s$
10:                     // and reinserted into $Q_i^s$ (line 35)
11:                     delete $F_i^s$.remove($e_F$)
12:                     $isNew \leftarrow false$
13:                 else
14:                     // $m$ was already relayed
15:                     $queued \leftarrow true$
16:                     if $m.LastHop \neq \infty$ then
17:                         $A_i^s$.insert(m)
18:                         $toACK \leftarrow true$
19:                     end if
20:                 end if
21:             end if
22:         end for
23:         if not $queued$ then
24:             for each $e_Q \in Q_i^s$ do
25:                 if $e_Q.id = m.id$ and $e_Q.Origin = m.Origin$ then
26:                     // $m$ is being relayed
27:                     $queued \leftarrow true$
28:                     if distance($m.LastHop, s$) $\leq$ distance($here(), s$) then
29:                         // $m$ already made progress in this direction
30:                         $F_i^s$.insert($Q_i^s$.remove($e_Q$))
31:                     end if
32:                 end if
33:             end for
34:         end if
35:         if not $queued$ and $distance(m.LastHop, s) > distance(here(), s)$ then
36:             // enqueue $m$ for relay in this direction
37:             if $m.LastHop \neq \infty$ or isNew then
38:                 // $m$ is not in recovery mode
39:                 $m.LastHop \leftarrow$ here()
40:             end if
41:             $m.Retries = \beta$
42:             $Q_i^s$.insert($m$)
43:         else
44:             // $m$ will not make progress in this direction or is already in $Q_i$ or $F_i$
45:             if not $toACK$ then
46:                 delete $m$
47:             end if
48:         end if
49:     end for
50: end procedure
```

Bits: 1	11	12	N_S	2	3*sb	16
All Listen	Count	Id	Sinks	Spatial Scale	Last Hop (x,y,z)	CRC

Figure 3 – FT-TSTP microframe format

Bits: 3	1	2	8	64	3*sb + tb	64	0 or 32
Message Type	Time Request	Temporal Scale	Location Confidence	Last Hop Timestamp	Origin (x,y,z,t)	Expiry	Location Deviation

Figure 4 – FT-TSTP message header format.

bits in `Sinks` are set. If it can make the message progress toward ANY of those sinks, then it wakes up to receive m and becomes one of its potential relay nodes. The queues of messages waiting to be forwarded by each node have now a second dimension, corresponding to each of the sinks $\in S$ and is designated Q_i^s.

When a new message m is produced, its `Sinks` attribute is initialized at the origin node with all bits set, so Algorithm 1 (line 2) initially tries to forward the message to all sinks. Subsequently, relay nodes only include m in the Q_i^s corresponding to the sink for which they are relaying the message (lines 35-42), clearing the bits of m.`Sinks` associated with the other sinks. This way, messages get forwarded directionally, avoiding flooding the network with unnecessary replicas of m. The behavior of the `All Listen` bit is unaltered, so control messages and messages whose destination is not a sink are routed using TSTP's routing algorithm.

In order to avoid bounces, which may occur when a previous relay node hears a message that was already forwarded to a sink in the past and therefore was removed from the corresponding Q_i^s queue, a second bidimensional queue of recently forwarded messages, F_i^s, is kept at each node i. Messages are kept in this queue until they expire. These queues are updated at the beginning of each transmission cycle by Algorithm 2.

TSTP does not handle geographic voids. A message m is retransmitted towards its destinations until it expires. Voids can result from poor sensor placement, sensor failures, or from sensor nodes that get compromised and refuse to forward messages. Therefore, in order to achieve the availability property, FT-TSTP must handle voids natively. The acknowledgment mechanism in Algorithm 1 enables voids to be easily detected by Algorithm 2: If m is not overheard after a certain number of transmission slots (m.`Retries`, Algorithm 2, lines 23-24), then it is safe to assume that node i has no (sane) neighbors that are closer to the destination than itself. The β coefficient is also applied in this sense, so m is expected to be acknowledged (i.e. retransmitted) by a neighbor in each of the designated directions after β transmissions.

If a message m is not acknowledged after m.`Retries` retransmissions (Algorithm 2, lines 25-37), then the recovery routing mode is enabled by making $m.LastHop \leftarrow \infty$ and resetting the retransmission counter (lines 30-32). By putting the last hop position at ∞, the node triggers a reverse flooding routing strategy, since it will always be more distant from the destination than any other node. This reverse routing is exited when a node that has not heard m before (i.e. the conditions of lines 7 and 25 of Algorithm 1 do not verify) restores a real last hop (Algorithm 1, line 40).

Algorithm 2 FT-TSTP Multi-Sink Queue Update

```
 1: procedure Update_Queues
 2:     for each $s \in S$ do
 3:         for each $e_Q \in Q_i^s$ do
 4:             if $(e_Q.Expiry - \tau(e_Q)) <$ now() then
 5:                 // $e_Q$ expired without making progress
 6:                 // in this direction
 7:                 $m \leftarrow Q_i^s$.remove($e_Q$)
 8:                 delete $m$
 9:             end if
10:         end for
11:         for each $e_F \in F_i^s$ do
12:             if $e_F.Expiry <$ now() then
13:                 // $e_F$ was relayed long ago and can be forgotten
14:                 $m \leftarrow F_i^s$.remove($e_F$)
15:                 delete m
16:             end if
17:         end for
18:     end for
19:     $done \leftarrow false$
20:     while $|Q_i| \neq 0$ and not $done$ do
21:         // the element at the head of $Q_i$ is the next to be transmitted
22:         $s \leftarrow Q_i$.next_to_expire()
23:         $m \leftarrow Q_i^s$.head()
24:         $m.Retries \leftarrow m.Retries - 1$
25:         if $m.Retries < 0$ then
26:             if $m.LastHop = \infty$ then
27:                 $m \leftarrow Q_i^s$.remove($m$)
28:                 $F_i^s$.insert($m$)
29:             else
30:                 // $m$ in $e_Q$ reached a void, set recovery mode
31:                 $m.LastHop \leftarrow \infty$
32:                 $m.Retries = \beta - 1$
33:                 $done \leftarrow true$
34:             end if
35:         else
36:             $done \leftarrow true$
37:         end if
38:     end while
39: end procedure
```

False voids can be detected if a message takes two different paths simultaneously, e.g., on the edge of a previous void. The message will later converge to nodes close to a sink, and the last arriving copy will be ignored, as the algorithm doesn't forward previously seen messages. The lack of an implicit ACK will initiate the void detour algorithm, making the message be retransmitted in recovery mode and start unnecessary reverse flooding. To handle this, when a node receives a message already forwarded, the preamble of microframes will be transmitted to send the ACK. For that, the message is inserted in another queue A_i (Algorithm 1, line 16-18), which must be processed in the transmission cycle. The TSTP does not use explicit ACK messages, except for the sink nodes. In the TSTP protocol, the consequence of lacking an ACK message is that the node will retransmit it until it expires. But at the FT-TSTP protocol, the consequence would cause message flooding, overloading the WSN with unnecessary messages.

4.3 PROTOCOL EVALUATION

In order to evaluate the Fault-Tolerant Trustful Space-Time Protocol (FT-TSTP) a set of experiments was designed, focusing on the protocol's impact on the WSN message delivery rate, latency, and energy consumption.

The communication performance of the protocols proposed here was evaluated through a set of simulations on the OMNet++ 4.6 (OPENSIM, 2017) simulator using the Castalia 3.3 framework (BOULIS, 2017). The quality of the simulation models is assessed by comparing the simulation results with those obtained in the field on the Solar Smart Building, which is automated using the Embedded Parallel Operating System (EPOS) (LAB, 2017) and the original TSTP (RESNER; FRöHLICH, 2016) on EPOSMote III devices (LAB, 2017). Indeed, the source code used for the simulations is basically the same used in the real deployment, but the comparison between the real scenario and the simulated model allowed us to adjust the model's parameters to obtain realistic results.

Simulation Parameters

Table 2 – Simulation parameters.

Scenario	Grid Size (m)	Node Placement	Radio Range (m)	TX Power	Period (s)	Expiry (s)
1-Building	70x70	32 nodes, 6x6, regular	20	-10 dBm	60	60
2-Building	150x175	55 nodes, 7x9, regular	45	-5 dBm	60	60
3-Field	500x500	81 nodes, irregular	80	0 dBm	120	120

A set of WSNs was evaluated, each with different node placements and void configurations, to evaluate the impact of delivering messages simultaneously to multiple sinks and void detouring over the delivery rate, the end-to-end transmission time, and energy consumption. Three WSNs were modeled on top of the CC2420 IEEE 802.15.4 physical layer available in Castalia. Table 2 summarizes the parameters used in these simulations. Each WSN was simulated using TSTP and FT-TSTP for a simulation period of one hour (3600s), with the presence of the voids represented in Figure 5. For FT-TSTP, the β parameter (retries) was set to 1, to diminish the time overhead of retries and, consequently, diminish the risk of message expiring. FT-TSTP was also simulated with different numbers of sinks, from 1 to 4. Each simulation was executed 10 times with different random number seeds, and the results presented are the average values.

Scenarios

The scenarios were modeled to evaluate the FT-TSTP behavior under different aspects of WSN deployments, such as nodes distance and placement, radio TX power, and void configurations. The objective was to verify that the void detour algorithm can handle different void types while measuring the latency and energy overheads since there is a trade-off between reliability obtained through multi-sink transmissions and latency/power consumption. Scenarios 1 and 2 can be applied in monitoring air quality in large indoor areas, as the temperature in shopping malls or dust in big industrial sheds. Scenario 3 is typical of environmental monitoring or smart agriculture, like soil moisture in irrigation systems, or temperature/humidity in field-monitoring. In all scenarios, the data periods are not too tight, but fault tolerance is justified by economic loss on the lack of correct measurements and actuation.

The voids are modeled as concave regions (Scenarios 2 and 3), and a section on the border, causing a semi-partition in the field (Scenario 1). These layouts are similar to the presented in other works, like Ozen & Oktug (2014) and Zhang & Dong (2015). In the simulations, one SmartData update was produced by every operational node at each data period, and the message had the same time to reach the sinks before expiry. The data generation instant was randomly chosen in the first period, and from that on a new SmartData was generated in exact periodic cycles.

Table 3 – Delivery Rate

	TSTP	FT-TSTP			
Scenario	1 Sink	1 Sink	2 Sinks	3 Sinks	4 Sinks
Scenario 1	46.8434%	0.0000%	0.0000%	0.3921%	99.6079%
Scenario 2	66.4279%	0.0036%	0.5624%	9.3647%	90.0692%
Scenario 3	82.7258%	0.0119%	0.0655%	0.7917%	99.1310%

A comparison between TSTP and FT-TSTP delivery ratio in the three modeled scenarios is shown in Table 3 and Figure 6. Table 3 shows the percentage of messages that reach each number of sinks. As all scenarios have four sinks, the columns represent the number of messages that reached *only* that number of sinks. For example, in Scenario 1 all messages reached three or four sinks, with no messages reaching only one or two sinks. On Scenario 2, around 9.3% of messages reached three sink, and a very small number of messages arrived only at one sink. The numbers show that, even in the presence of faulty nodes that create a void region, FT-TSTP was able to deliver more than 99% messages to at least three sinks in all scenarios, which is enough for the ByzCast algorithm to come to an agreement about the received values. All messages were delivered to at least one sink by the FT-TSTP protocol. TSTP has delivered less than half of the messages in scenario 1, a bit more than 66% of the messages in scenario 2, and more than 82% of the messages in scenario 3. In scenario 2, as the number of nodes is larger and the data period is the same as in scenario 1, the number of messages that reached all sinks is the smallest. It happens because messages expire due to the transmission queue sizes,

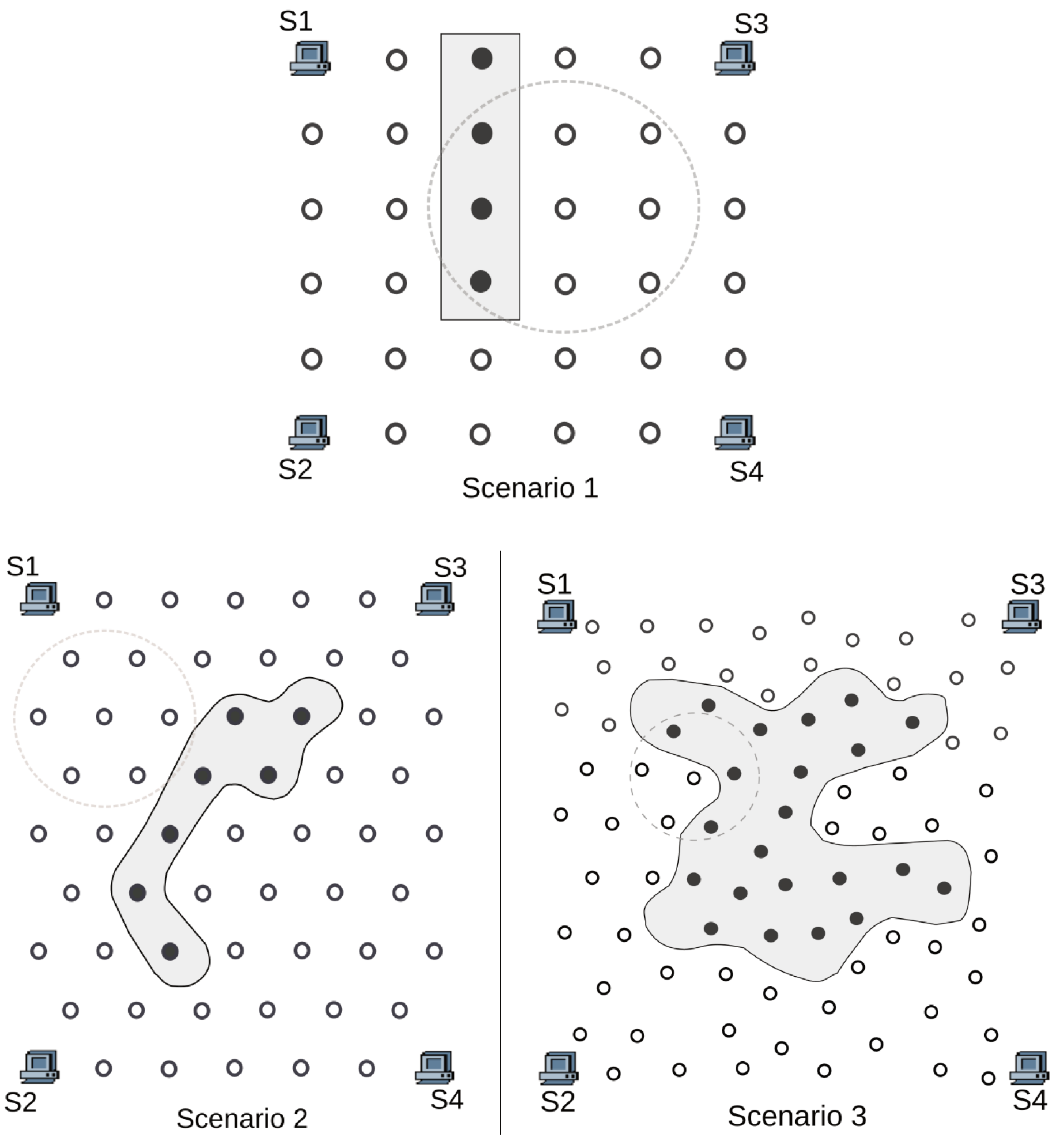


Figure 5 – Simulated scenarios. Voids are represented by the gray area. Dashed circles represent radio range.

or by exceeding the number of β retransmissions without making progress due to the *hidden node* phenomenon. In Scenario 3, the larger data period makes nodes handle messages with less competition on the transmission channels.

The next evaluations have the objective to verify the impact of increasing the number of sinks over the delivery latency and on the energy consumption. Therefore, each scenario was simulated with one (S1), two (S1, S4), three (S1, S2, S4), and the four sinks active. The different evaluations are discussed next.

Figure 7 shows the average latency time for a message to reach all the active sinks. Scenario 2 shows larger latencies with two or more active sinks because more messages are being transmitted over the WSN with the same data period, as there are more nodes than in

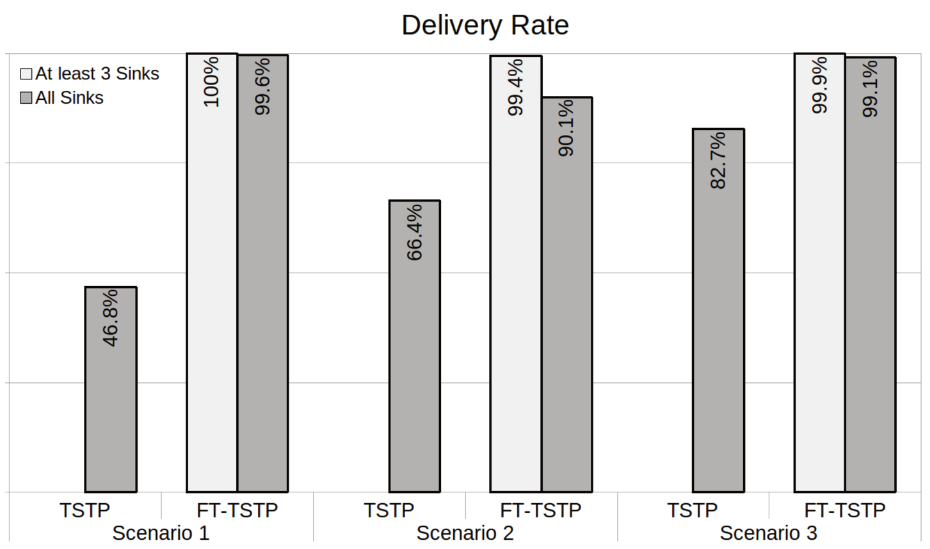


Figure 6 – FT-TSTP vs TSTP delivery ratio to at least three and to all sinks, with 4 sinks and void present.

.

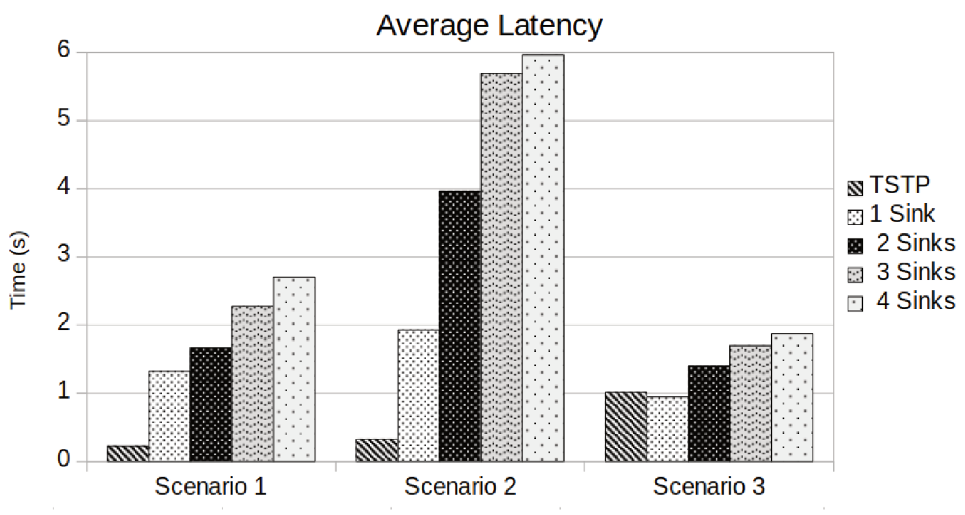


Figure 7 – FT-TSTP vs TSTP **average** end-to-end message delivery latency, TSTP with 1 sink and FT-TSTP with 1 to 4 sinks.

.

Scenario 1. This makes the transmission queues in nodes get larger, mostly in the central region of the scenario, introducing greater delays between receiving and retransmitting messages. Figure 8 shows the maximum latency observed in the simulations and shows that Scenario 2 is close to its saturation point when using 4 active sinks, as some messages are reaching the sinks very close to the expiry time (60s), and about 10% did not reach all four sinks (figure 6), but only three of the sinks. For Scenario 3, figure 8 shows that TSTP imposes long delays to some messages due to retransmissions on void borders. Messages that are not expired, even if not making progress, are retransmitted until they expire, delaying messages that can make real progress on these nodes.

Figure 9 shows the energy consumed by active nodes and sinks during the simulation

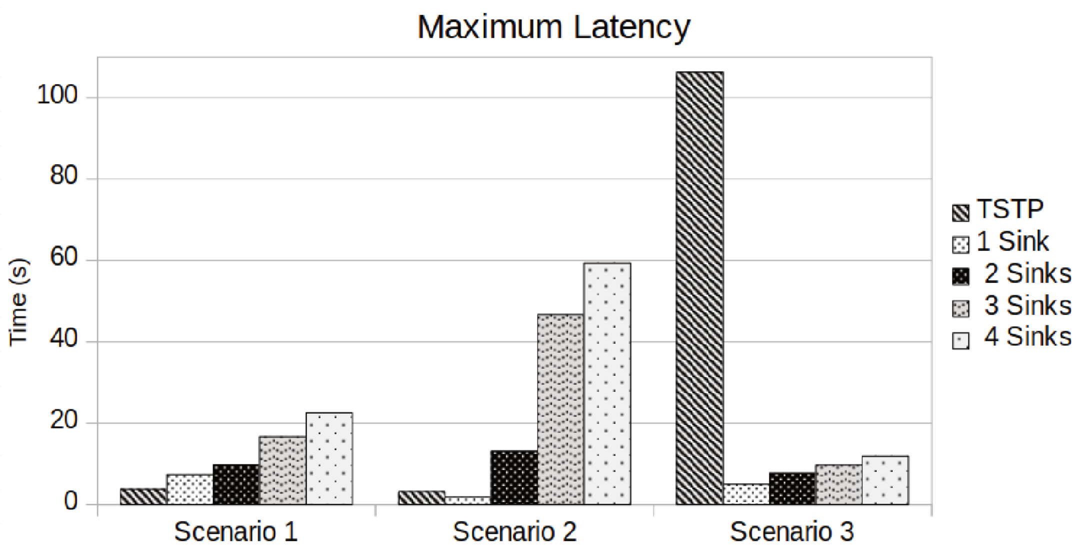


Figure 8 – FT-TSTP vs TSTP **maximum** end-to-end message delivery latency, TSTP with 1 sink and FT-TSTP with 1 to 4 sinks.

.

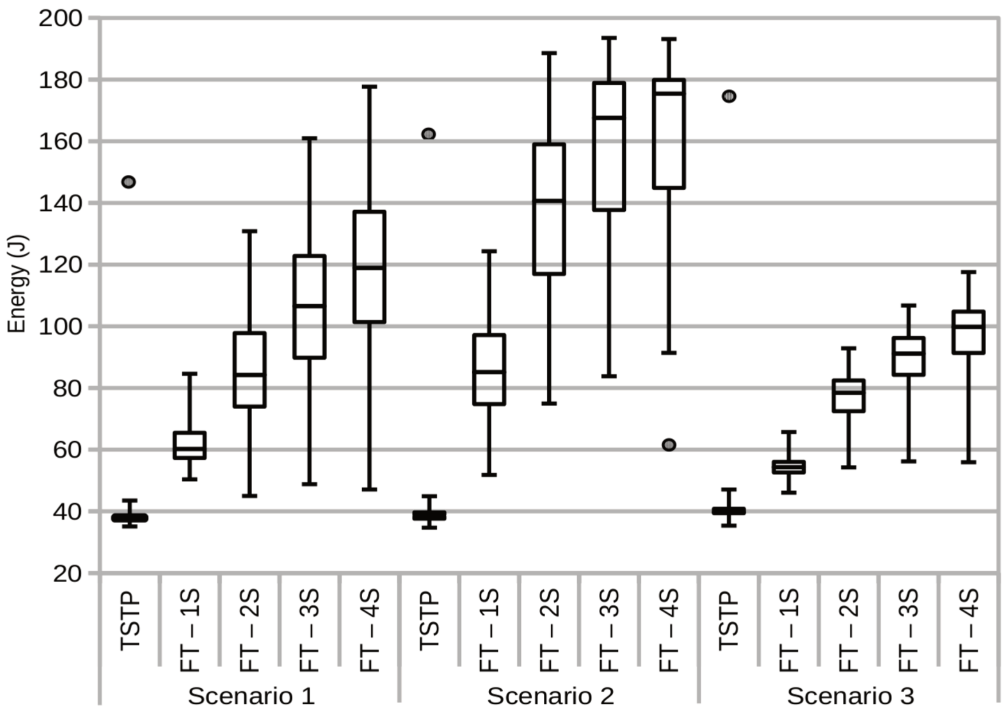


Figure 9 – TSTP vs FT-TSTP (1 to 4 sinks) energy consumption at one node in 1 hour. Circles represent outliers.

period (1 hour). The additional energy consumption arises from different aspects. First, the increase in the microframe size, from 9 to 28 bytes demands more energy to transmit the message's preamble. In conjunction with the retransmissions in *recovery mode* to circumvent the voids, it explains the energy increasing from TSTP to FT-TSTP with a single sink. The increase from 1 to 2 sinks is due to the duplication of the messages, as they have to travel in opposite directions most of the time. The energy increase for 3 and 4 sinks is smaller, as messages make

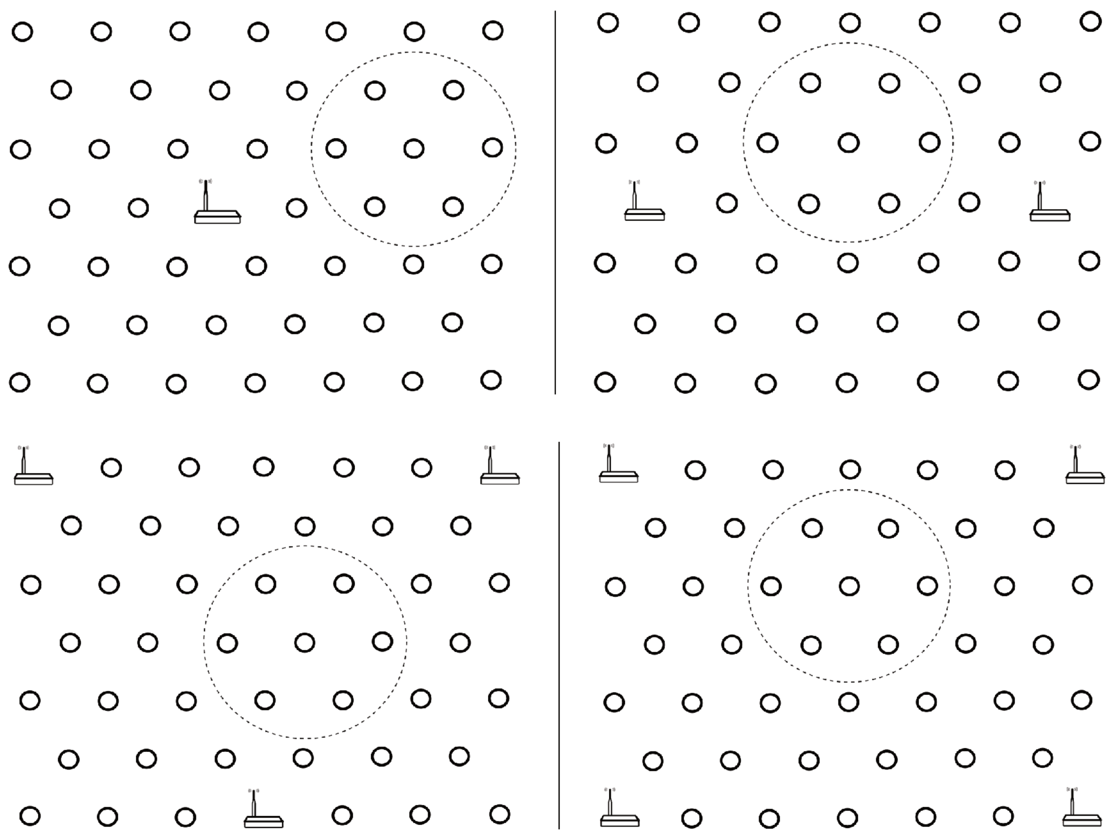


Figure 10 – Simulated scenario, varying sinks from 1 to 4. Dashed circle denotes radio range.

progress to more than one sink on most transmissions. It can be deduced that if a fifth sink were added, the energy increase would be even smaller than the increase from three to four, and so on. The outliers' values, represented by the circles, show that when TSTP reaches a *local minimum* - on a void border - it consumes much more power than the nodes in FT-TSTP with one sink. This happens because these nodes will continuously retransmit messages until their expiration. So, the deployment of a resilient WSN requires a careful dimension of data periods and duty cycle values to meet the resilience requirements, as mentioned in section 2.6.

Parameters Evaluation

Continuing the proposed protocol evaluation, another case study was set up modeling a WSN composed of 46 equidistant nodes, with some of them chosen as gateways. When using a single gateway, for a baseline comparison with TSTP, the gateway was positioned close to the field's center. When using more sinks, the gateways were positioned in a way to difficult an attacker or a failure to isolate all sinks from other nodes. The disposition of nodes and gateways is shown in Figure 10. The evaluation was done through a set of simulations on the OMNet++ simulator (version 4.6) (OPENSIM, 2017) using the Castalia framework (version 3.3) (BOULIS, 2017). As TSTP runs on EPOSMote III in several real applications, with very few adaptations from simulation codes, it is expected that FT-TSTP will also perform like the simulations' results when ported to the real motes.

Evaluation of the Retries and Duty Cycle parameters

Geographic forward algorithms without void detection diminish the *hidden terminal* problem by constantly retransmitting messages, until an ACK is received or the message expires. In the proposed approach, the *retries* (β) parameter is important as it determines when a node enters the *recovery mode* or stops transmission, dropping the packet. High network load increases the chance that a forwarding node does not receive a message due to the hidden terminal effect. Making too few retries leads to erroneous void detection and to start recovering mode unnecessarily. If the retries number is high, the End-to-End (E-T-E) time raises when voids occur, due to the delay to enter recovery mode. The MAC layer duty cycle also influences the delivery rate, as small values cause the E-T-E time to rise. If a void appears, the duty cycle has a greater impact on the delivery rates, as more packets expire and are discarded.

Figures 12 and 13 show delivery rates evolution using three or four sinks, respectively. The left graph refers to Void A and the right to Void B from Figure 11. Each line represents the FT-TSPT algorithm running with a different value of β, from 1 to 6. The algorithm was run with different values for the MAC Duty Cycle, from 0.01 (1%) to 0.5 (50%). The graphs show that the algorithm obtains delivery rates over 90% only with duty cycle values of 0.05 or larger. The results show that $\beta = 3$ has a reasonable performance in all scenarios. If too few retries are made, the hidden terminal prevents the packets from making progress and are dropped in the intermediate nodes. Otherwise, if too many retries are made when a void region is present (as simulated) packets expire and are discarded.

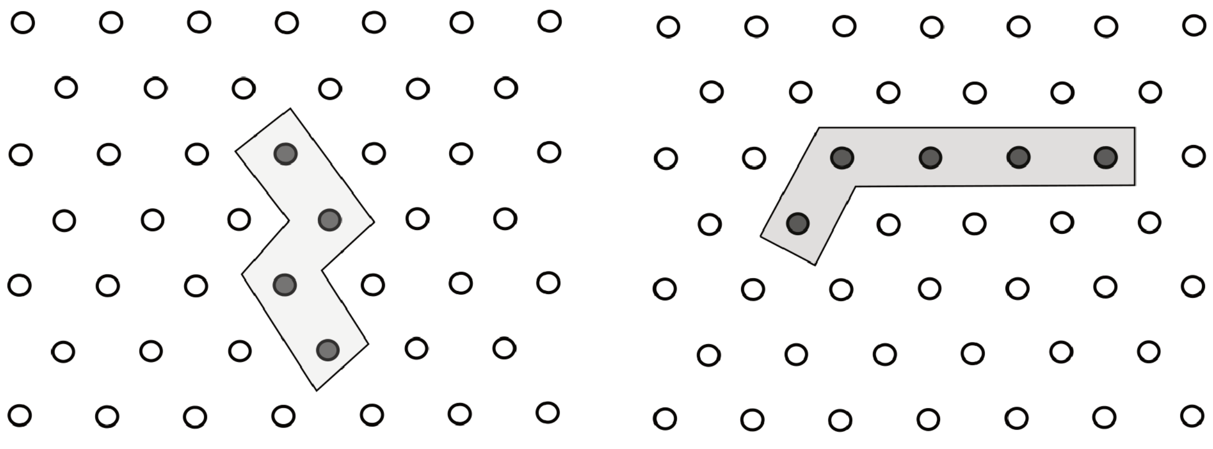

(a) Void A (b) Void B

Figure 11 – Scenarios with failed nodes.

Delivery Rates

The delivery rates in the evaluated scenarios are shown in Table 4. The number shown for 2, 3 and 4 sinks are the percentage of messages delivered to *all* sinks, in a different analysis

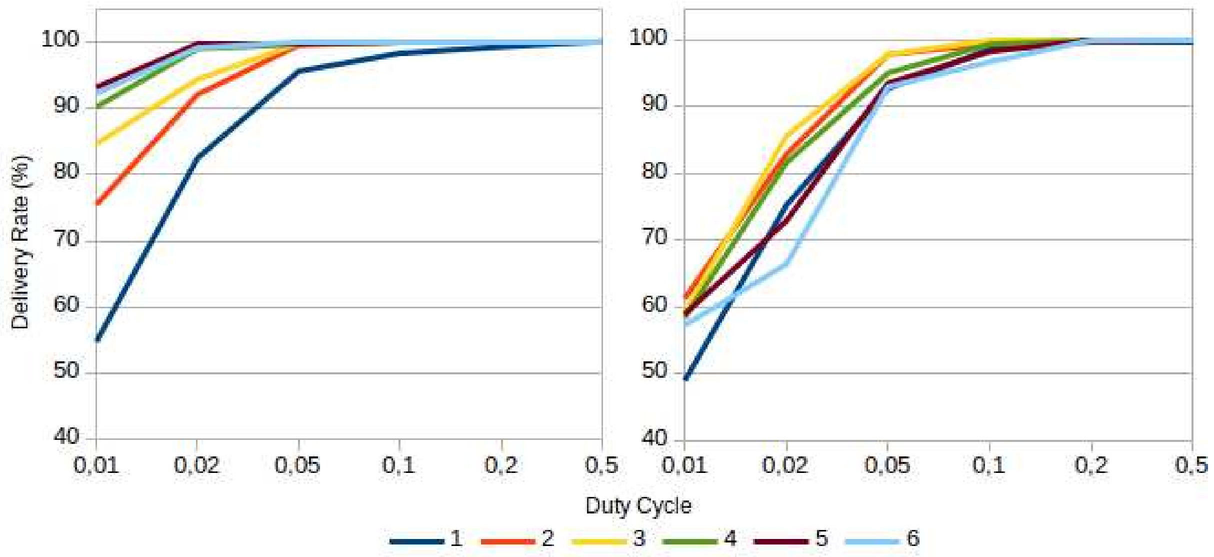


Figure 12 – Delivery Rates in voids A and B with 3 Sinks

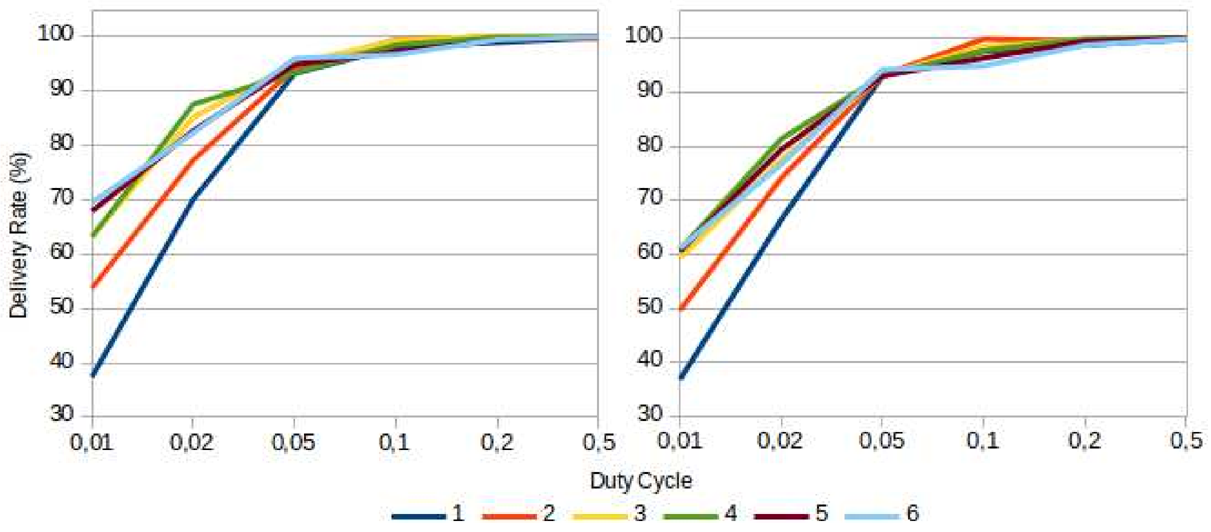


Figure 13 – Delivery Rates in voids A and B with 4 Sinks

Table 4 – Delivery Rates in different scenarios

Scenario	TSTP	FT-TSTP 1 Sink	FT-TSTP 2 Sinks	FT-TSTP 3 Sinks	FT-TSTP 4 Sinks
No Void	100%	100%	100%	99.88%	99.25%
Void A	77.38%	100%	100%	99.73%	95.01%
Void B	71.96%	99.47%	99.87%	97.78%	93.17%

from the one shown in Table 3 and Figure 6. However, the results are consistent with those obtained in the previous experiment, as this one aimed to verify the delivery rate under different number of sinks, that imply in different network configurations and message traffic. In the first two scenarios – without void and with Void A – every message was delivered to two gateways without loss. On the scenario of Void B (see Figure 11(b)) the delivery rates showed a little decrease. This can be corrected by increasing the value of the Duty Cycle, as shown in Figure 13. It can be depicted from that figure that duty cycles over 20% show deliver rates very close to 100% for different numbers of retries. If considering the use of a consensus algorithm between the sinks, Table 4 shows that more than 97% of the messages were delivered to at least three sinks, the minimum of sinks needed to reach a consensus. As expected, TSTP showed a considerable decrease in the delivery rate in the presence of failed nodes.

Energy Consumption

As each message is delivered to N sinks and the main objective is to improve data availability and reliability, an increased power demand by nodes was expected. The expectation was that the energy consumption grows linearly when the number of gateways increases, instead of growing exponentially. Figure 14 shows the average power consumption by nodes running for 60 minutes. For both protocols the duty cycle was set to 5%, and the β parameter was set to 3 in FT-TSTP. The difference on the *No Void* scenario between TSTP and FT-TSTP (from 18.35J

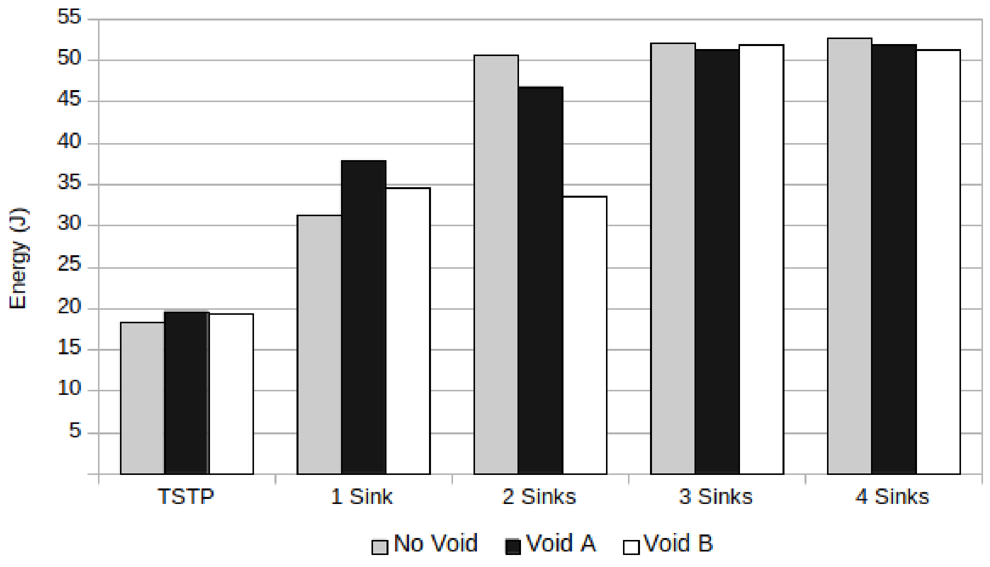


Figure 14 – Average node power consumption / 1 hour - TSTP x FT-TSTP 1-4 sinks.

to 31.33J) is due to the increment of the microframe’s size from 9 to 18 bytes, demanding more energy for transmission. The increase from 1-Sink to the 2-Sinks scenarios can be justified by the gateways’ positions. As they are in opposite directions, packets rarely show progress towards **both** gateways, splitting the message most of the time. In the 2-Sinks scenario, the energy spent when voids are present is lower than when all are operational because there are fewer sender nodes. In the *Void B* scenario, the void format does not represent an obstacle for most of the messages. So, the mean energy for all (active) nodes is lesser than the *No Void* scenario because there are less packets to forward, rarely using the void detour mechanism.

The energy increases in the 3-Sinks and 4-Sinks scenarios were as expected. The void detour algorithm consumed more energy than the 2-Sinks scenario because the gateways’ positions are now *’behind’* the voids for many nodes, demanding more retransmissions in recovery mode. But the difference between the two last scenarios (3 and 4 sinks) is low because the routing algorithm tries to give preference to the relay candidate that makes progress to **more** sinks, instead of the relay candidate **closer** to one of the sinks, which would split the message and increase traffic (Figure 15). It can be inferred that adding gateways to a WSN with $N > 3$ gateways implies little power consumption increment, instead of multiplying it by the increment factor $1 + 1/N$.

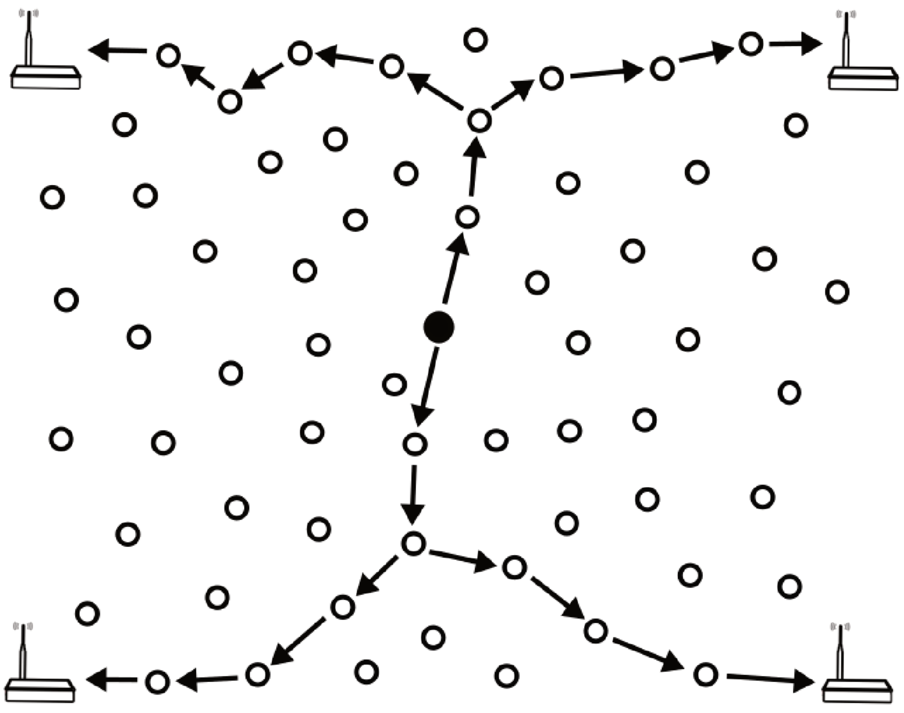


Figure 15 – FT-TSTP packet routing example

As shown by the simulations' results, the use of 3 or more sinks doubles the power consumption of the nodes compared to single-sink TSTP. The resilience against nodes and gateways failures can make this extra power consumption worth. However, it can be stated that if the sensed data is critical, sensors would be deployed with a reliable power source, as power lines or some power harvesting mechanism.

4.4 DISCUSSION

Fully-reactive geographical routing protocols are simple and yet powerful on route discovering, based on the only premise that each node knows its position, and gets the coordinates from the sender and receiver as part of the message. However, when facing voids that prevent messages from making progress upon the destination, these protocols show an inherent deficiency in circumventing the black hole formed by nodes' displacement or failure. There are very few works that try to solve this problem, like the one presented by Lima et al. (2017). Also, traditional single-sink architectures WSNs are vulnerable to come fully disconnected on a sink failure. The whole WSN trustfulness becomes compromised if an intruder takes the sink over and, for example, starts to send forged data to the application.

To overcome this limitation, in this chapter an extension of the TSTP protocol is presented. It is named FT-TSTP, and increases the original protocol with two new features that provide fault-tolerance at the network level. The fist is the ability to perform void-detour when reaching a local minimum while routing a message to a specific coordinate. Like any other fully-reactive geographical routing protocol, the TSTP is already able to find alternative routes when the network provides enough node density. But, depending on the shape and the size of the void, it is frequently impossible to overcome the black hole. The main idea is that a node, when trying to transmit a message, assumes that it is on the border of a void area if it does not receive an ACK after β retries. Then it assumes a fake *infinity* position and starts to transmit the message that will be relayed by any other node that did not see that message from *infinite* before. This causes a controlled backward flooding until progress upon the destination can be made.

As the protocol uses neither void-discovery nor route-building packets, when facing a void area, immediately it starts to route packets through alternative relay nodes, by entering the *recovery mode*. When the missing nodes come back to the original route, by a positioning correction or by recovering from a transient failure, the original routing recovers instantly. This configures the protocol as highly adaptive and resilient to transient failures.

The second feature is the support of multiple sinks by the routing protocol. It provides an important fault-tolerance upgrade, by requiring that every message sent by a node must be delivered to **every** sink. The routing algorithm is enhanced to make a message be relayed by the node that can provide progress to <u>more</u> sinks, instead of only to the <u>nearest</u> sink. This avoids that a message is split into N messages, each progressing to a specific sink, and helps to control the number of messages competing for the radio signal.

In addition to the elimination of the architectural single point of failure, the FT-TSTP increases the security against intruders on the gateways and the nodes. At the gateway level, running different operating systems and using different tools reduces the exposure to known vulnerabilities explored by intruders. If, even so, a sink or node is compromised, the solution improves security against message forging by malicious code. At the level of the sinks, the application can require a consensus between the sinks, through an agreement protocol, before the data is accepted. So, the intruder would have to take over the majority of sinks to successfully forge messages at the sinks.

On the energy consumption aspect, there is the obvious trade-off between delivering messages to several sinks and energy consumption. The results showed that the energy consumption increases to one limit if the sinks are well-positioned. It happens because the protocol makes a node that makes progress to more sinks on the same message will wait less time to start transmission. Nodes that are at the border of voids can also decide to wait more time before relaying messages, given to other nodes the chance to collaborate with the message progression. This time shrinking or stretching mechanism is inherited from the TSTP protocol.

Another possible situation is the intruder to take over the identity of a WSN node, despite the protocol's authentication and security mechanisms. If the intruder has success, there are two possible situations. If the sender node is compromised and sends forged data, then nothing can be done, as all sinks will receive the same forged data. If the compromised node tries to change data from other nodes while acting as a relay, it has to change a message with the majority of the sinks bits set, to win the agreement protocol. However, in the proposed protocol, messages make progress through several concurrent paths and are dropped when they reach nodes that already retransmitted successfully the same message. So, it is unlikely that the forged message will reach the sinks before all other messages containing the original data. The intruder will have success only when it is the unique relay between a border node and the WSN.

5 SENSORS' DATA CONFIDENCE ATTRIBUTION

Parts of this chapter appeared earlier in ***WSN Data Confidence Attribution Using Predictors*** (SCHEFFEL; Fröhlich, 2018) and ***Increasing sensor reliability through confidence attribution*** (SCHEFFEL; FRÖHLICH, 2019).

In this chapter, the problem of identifying data faults in a WSN is addressed. As a *sensor fault* can encompass a range of problems, the term *data fault* is used to denote the specific problem of a sensor producing incorrect data. The chapter is organized as follows: first, the context and motivation are discussed in Section 5.1. Next, Section 5.2 describes the proposed mechanism. The concept drift aspect is discussed in Section 5.3. Some case studies, with the evaluations of the proposed solutions are presented in Section 5.4. Section 5.5 makes a discussion of the obtained results and the conclusion.

5.1 INTRODUCTION

Several modern applications of WSN are getting deployed in different application areas. It is essential that they operate in a reliable and trustful manner. Many times, these applications must operate in harsh environments and are susceptible to interference in sensing and communication. Therefore, as in any other application that uses sensors, fault tolerance mechanisms are essential to ensure correct readings and actuations by the WSN elements. Faults in a WSN can range from incorrect sensor readings, communication failure caused by environmental or intentional interference, to nodes and gateways intrusion by attackers to forge sensor readings or send incorrect commands to actuators.

As in centralized applications, the use of redundant sensors increases the fault tolerance in sensed data, as comparison with values from other sensors is crucial to get data to confirm a diagnostic about the state of a suspicious sensor. When applied in the context of a WSN, the fault detection must use some strategy that allows nodes to access raw data or some diagnosis information from other nodes. Several solutions were proposed in the literature, as presented in Section 2.1. The majority of these proposals use special messages to decide if there was an error and to determine the source. This leads to overhead in terms of latency, bandwidth, and energy consumption in the WSN. Hierarchical or centralized architectures try to minimize this overhead but are subject to errors if the data received is altered by malicious or defective nodes while it is transmitted. It also avoids safe local decisions that must be made in a short time interval and must rely on correct data. Wait for the decision from voting rounds, or for the response of a central node about the data correctness can lead to the timeout of the needed reaction.

The solution proposed in this work provides self-diagnosis capabilities to the sensor nodes, based on data gathered from correlated neighbors, incurring little communication overhead. To accomplish this, a predictor is built off-line for every type of sensor node in an interest area, based on the readings from other correlated sensors in the same area. This model is transferred to the sensor. At runtime, each sensor listens to the data transmitted by other nodes in the interest area and uses the model to predict its own value. Comparing the sensed value with the predicted one, each node can calculate its confidence level and is also able to determine if the error is caused by the local sensor or by a faulty neighbor. Extra messages are avoided by transmitting the sensed value, the prediction result, and the confidence level of every node as normal data, thus providing information about its current state to the application and the other correlated nodes.

Although applied to WSN in this work, the proposed solution can be used in any context, with different types of sensors monitoring correlated data. Data can arrive from direct connections, from a standard network, or recovered from a database. The assumptions that make this solution fit to WSN are two. The first is the use of correlated data, originated from close nodes inside a region of interest, even with nodes having more than one sensor. So, nodes can obtain the needed input just by extracting data from neighbors' packets in the network. The second assumption is that data has a constant periodicity, which is needed to create model that predict values of a round T based on data from the previous round $T-1$. This periodicity is guaranteed by the protocol defined in Chapter 4, but the proposed solution can also be applied to any other protocol that meets this requirement.

5.2 CONFIDENCE ATTRIBUTION USING PREDICTORS

Decentralized fault detection approaches try to minimize the message and time overhead inherent to centralized approaches. On the other hand, they have to deal with limited input sets and less computational power to perform their work. Therefore, models used to perform fault detection at the sensor level have to take into account the resource constraints. The current solutions do this by statistical analysis, mainly watching the variance of the sensor's readings. It works well on sensors that show slow variations in their readings. Domains in which sudden data changes can be observed can lead the algorithm to classify abrupt variation as faulty. The only way to verify if the variation is a correct reading is to compare it with the readings of other sensors, incurring in communication overhead.

Fault detection with no need to make extra packet transmissions, as in voting and checking protocols, is desirable on a WSN. This is because radio communication is the most power-consuming resource in such systems. Also, packet collision is a problem to be addressed in WSNs with a large number of sensors or with high sampling rates. In such scenarios, extra messages for diagnosis or voting in fault detection algorithms should be avoided, as some failures can lead to a network communication overload.

Regarding the architecture of fault-diagnosis schema, in the WSN perspective, a good solution would be distributed and use a self-diagnosis approach, with all nodes being able to autonomously determine the correctness of their sensors' values. As some comparisons are required to ensure the correctness of sensed values, group detection approaches are also an alternative, but with a minimum communication overhead. The adoption of a *"speculative* *mode"*, with no specific diagnosis or control messages, using only messages that are already transmitted by the WSN, provides a solution with enhanced data confidence with a very little communication overhead.

The assumption made in this work is that the network is composed of different types of sensors, monitoring several aspects of the same phenomenon or different interconnected phenomena. This assumption ensures that each node produces data with some correlation with the data produced by some other nodes. The network has to be designed in a way that different interest areas contain a set of sensors producing correlated values. The size of this set is not fixed and can vary concerning the correlation between the sensed physical quantities. Another assumption is that every node can read the data transmitted by the other nodes. If the communication is ciphered, some global or group key schema has to be used. Messages encrypted with a group key allow other authenticated nodes to read the transmitted data. For authenticity, messages can be signed with the node's private key, avoiding the message content from being altered by other nodes.

Thus, the proposed solution aims to provide nodes with self-diagnosis capabilities, based on data gathered from correlated neighbors, without exchanging extra messages, causing minimal communication overhead. For every node type in an *interest area*, a predictor model is built off-line, based on historical data from the different types of sensors in this area. This work makes use of Artificial Neural Networks, but any other type of predictor could be used. The only requirement is that the model has to be able to be transmitted over the network and demand a limited amount of memory and processing capabilities. This model (or the new parameters, when updating a model) is then transmitted to the sensors, where it is stored and executed.

At runtime, each sensor listens to the data transmitted by other nodes and uses the model to predict its value. Each node calculates its *confidence level* — or probability of correctness — comparing the sensed value with the predicted one. When the confidence reaches a lower bound, a second step is performed in order to determine the source of the fault. This step is necessary to determine the cause of the discrepancy: if it is an erroneous reading, or if it is caused by an incorrect input from another faulty sensor. To accomplish this, every node transmits the read value, the predicted value, and the confidence level, to provide information about its current state to the application and the other nodes. There is no extra packet transmission, only an increment in the size of the transmitted packet. Assuming a 64-bit value as a sensor reading, the increase is 9 bytes: the predicted value plus one byte for the confidence (a value between 0 and 100). If more confidence levels are needed, more bytes can be added to the packet. It's a design decision whether more granularity in the confidence levels is worth the increment on the network packet size.

The whole process is depicted in Figure 16. The firsts steps, namely **Feature Selection** and **Model Building and Training** are performed off-line, using historical data from the sensors of an interest region stored in a database. This implies that a *new* WSN will not have a reliable predictor on its first deployment. Therefore, the WSN has to run without confidence attribution at the beginning. After some time running, the data collected by the sensors can be used to train a primary model for each sensor and deliver it to the sensors. Afterwards, an update can be performed periodically or every time a model drift is detected, in order to produce a more accurate model.

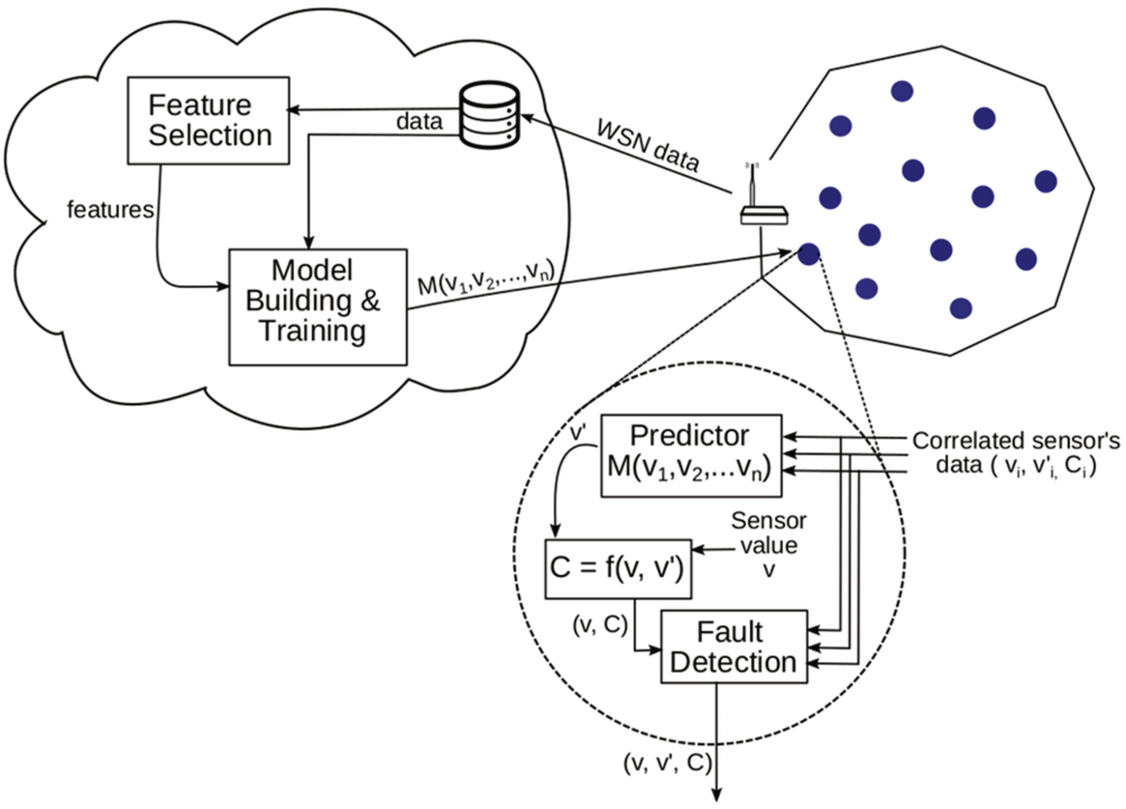


Figure 16 – Model learning and confidence attribution scheme.

The Feature Selection process searches the smallest input set for a predictor that produces the best results concerning accuracy and model size. This process can be resource and time-consuming, as many combinations of the inputs have to be evaluated. For large input datasets, or when the correlation between the variables is unknown, automatic attribute selection techniques can be employed. There are several algorithms for this, with different approaches. These algorithms were grouped into three classes by Visalakshi & Radha (2014). *Filter Methods* use statistical measures to assign a score to each feature and create a ranking. The lower ranked features are removed from the dataset. The methods are often univariate and consider each feature independently, or about the dependent variable. Some examples of some filter methods include the Chi-squared test, information gain, and correlation coefficient scores. *Wrapper Methods* consider the selection of the feature set as a search problem. Different combinations are prepared, evaluated and compared to each other.

A predictive model is used to evaluate a combination of features and assign a score based on model accuracy. The search process may be methodical as a best-first search, stochastic as a random hill-climbing algorithm, or heuristic-based like forward and backward passes to add and remove features. An example of a wrapper method is the recursive feature elimination algorithm. *Embedded Methods* learn which features increase the model accuracy while it is created. The most common type of embedded feature selection methods are the regularization methods. Regularization methods — also called penalization methods — introduce additional constraints into the optimization of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (fewer coefficients). Examples of regularization algorithms are the Least Absolute Shrinkage and Selection Operator (LASSO), Elastic Net, and Ridge Regression.

There are many machine learning tools available that can efficiently execute this task, as the Weka tool (FRANK; HALL; WITTEN, 2016) or scikit-learn machine learning toolkit (PEDREGOSA et al., 2011). Different techniques for feature selection can be used, as a filter method or a wrapper method. As communication is costly in WSN s, it is necessary to impose an extra restriction: the input set must be selected from the sensors close to the target sensor. The notion of *close* can vary, based on the communication layer. In a single-gateway architecture, it can denote nodes inside the radio range, so the node can listen to their transmissions. In multi-gateway architectures, *close* can mean nodes two or three hops away, once the routing makes their packets be re-transmitted by a neighbor that can be listened to. TSTP and FT-TSTP, discussed in chapter 4, have mechanisms that make a message reach all nodes inside an interest region. In this context, *close* is a node whose data can be observed by the current node, which needs the values as its predictor input. This also implies that the application has to know the location of the sensors, which is a requirement of many routing protocols.

After identifying the set of sensors that are more correlated to the target sensor, a model for each node can be built automatically. In the experiments carried out in this work, Multilayer Perceptrons were used. The number of layers and neurons in each layer can be determined by heuristics and by rules. The input layer has one neuron for each input value. An extra input can be used for backpropagation or bias. The output layer has one neuron, corresponding to the predicted value. Rules like the proposed by Trenn (2008) can determine the number of neurons in the hidden layer. Pruning methods shown by Thomas & Suhner (2015) can determine the optimal number of neurons in the hidden layer. Once built and trained, the models are transmitted to the respective nodes.

When the predictor is received, sensors can start to evaluate the confidence of their readings. The *confidence level* C of a node's value is a function $C = f(v, \hat{v})$ that evaluates the difference between the sensed value v and predicted value $\hat{v}$, as described in equation 5.1. In this work, the *Mean Absolute Error (MAE)* (see equation 5.2) obtained in the training process is used to calculate the result of the function. Comparing it to the Root Square Mean Error (RSME), Willmott & Matsuura (2005) state that MAE is a natural measure of average error and is unambiguous, being widely used for model-performance evaluation.

$$f(v, \hat{v}) = \begin{cases} 1, & \text{if } |v - \hat{v}| \leq \beta \times MAE \\ 1 - \frac{|v-\hat{v}| - \beta \times MAE}{\alpha \times MAE}, & \text{otherwise} \end{cases} \tag{5.1}$$

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |v_i - \hat{v}_i| \tag{5.2}$$

Since a ANN can be used as a universal approximator for any function, and the approximation theorem defined by Hornik et al. (1989), it is expected that, for a set of well fitted ANNs, the value of every sensor i being predicted shall not be greater than ε_i. The starting value for ε can be the MAE calculated on the training phase. But as the MAE is a mean, if the values being sensed do not present a "smooth" behavior, this value can be too restrictive. To allow a fine-grained tuning, the value can be multiplied by a factor β. Therefore, for every sensor S_i, it is expected that:

$$|v_i - \hat{v}_i| < \beta \times MAE_i \tag{5.3}$$

where $v_i, \hat{v}_i$ and MAE_i are the read value, the predicted value and the MAE of the sensor i, and β is a predefined constant, that acts as a calibration factor used to adjust the sensitivity of the confidence attribution function. The interval $[\hat{v} - \beta \times MAE, \hat{v} + \beta \times MAE]$ defines the *full confidence interval*, that means that the sensor read value can be considered 100% correct, and it takes in consideration the implicit error of the ANN approximation denoted by MAE. If the model has a good accuracy, the MAE value is expected to be small, and also a small value of β can be used to detect little variations in the readings. Otherwise, if the monitored value presents high variability, larger values can be assigned to this constant in order to accommodate the variations.

When evaluating the correctness of sensor values, a discrete binary classification as *correct* or *incorrect* for each value can lead to ambiguous situations, as two similar values can be labeled in opposite ways if they lay around the line that divides the classes. Therefore, this work opted to define a ***confidence level*** to be assigned to sensors' values, taking in account how distant they are from the expected values in the current context. Using another factor, called α, it is possible to define a ***tolerance range*** in which the read value loses its confidence level, which decreases from 100% to 0%. This tolerance range is also defined in terms of the calculated MAE for every sensor, in order to capture the characteristics of every measured unit. The α factor defines the velocity at which a value loses its confidence level as it gets away from the expected value. The smaller the value of α, the faster the confidence level decreases. If the predicted and read values are too distant from each other, function 5.1 may result in a negative value, in which case the confidence assumes 0 (zero).

The intervals around the correct value, defined by the parameters β and α and are illustrated in Figure 17. The black line denotes the correct sensor's values. The green area around the correct values denote the tolerance region, defined by the model's MAE and the β parameter. If the read value is inside this area, its confidence level is maximum, 100%. The red area around the green region is the area in which values loose their confidence. A value right next the green area has a confidence level of 99%, and it decreases to 0 is it moves away. The smaller this area, determined by the parameter α, the faster the sensor's readings lose their confidence.

The prediction models will show circular references (the model to predict variable S_j depends on variable S_k, and the model to predict S_k depends on S_j) since the correlation

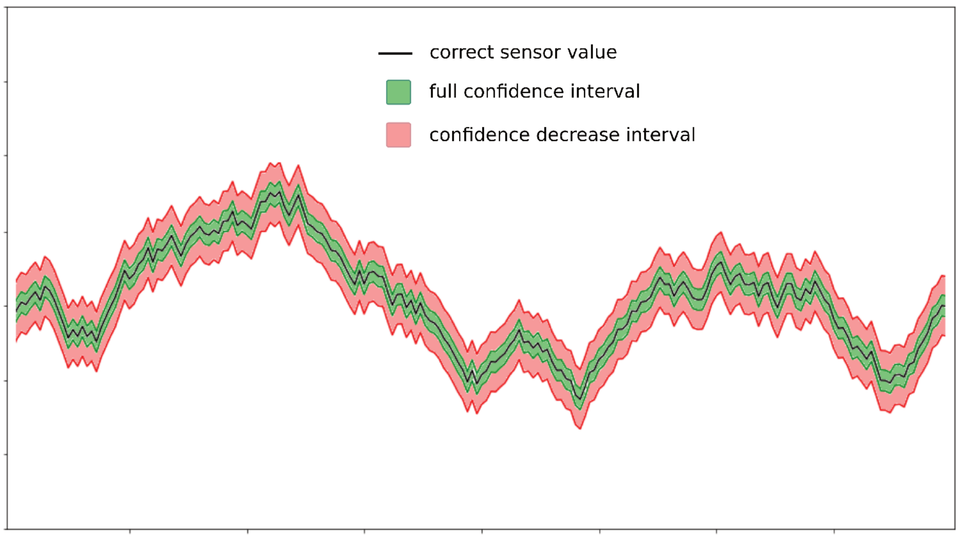


Figure 17 – Thresholds around the correct value defined by α and β.

is reflexive. The problem that arises here is the freshness of the models' inputs. Monitoring applications of WSN perform periodic sampling of the sensor values, in the opposite way of the *event triggered* events, that report data only when an event of interest occurs. So, it is assumed that it is possible to determine a period P, in which at last one reading of each sensor type is transmitted to the application.

The majority of WSN communication protocols do not guarantee the order between messages while they are retransmitted by the intermediate nodes. Also, the exact instant when a message arrives on a node is not determined only by the instant it is produced. It is also highly influenced by communication delays and network load. The protocols only guarantee that data is delivered at each period T, without ordering guarantees. So, it is only possible to assume that the data from the previous period is available. Therefore, models are trained using data from the previous period $t-1$ to predict the values of the actual period t. Each node will passively listen to the data sent by other nodes, collecting the inputs for its predictor. Data produced by close sensors required as the model's input will be buffered. This data will then be used by the local node to make the prediction of its own value at the next period $t+1$ (equation 5.4).

$$\hat{v}_i^{t+1} = P_i(v_1^t, v_2^t, v_3^t, \ldots, v_n^t), \text{ for all } n \text{ inputs} \tag{5.4}$$

Fault Identification and the Interference Problem

The use of the raw values from the correlated sensors to predict the current value on a specific sensor leads to a problem called here as the *interference problem*. When the difference between the sensed value v and the predicted value $\hat{v}$ is greater than the defined threshold, it is hard to make a precise statement about the correctness of the sensed value. There is no way to state that the sensor is faulty, or if an input of another defective sensor caused a wrong prediction. Also, a faulty value sent by a node will cause a prediction error on all other nodes that use it as model input. The magnitude of this interference is determined by the correlation strength and by the number of features used as inputs by the predictor. As a result, there will be several sensors that will classify their values as faulty, without a clear identification of the fault's source. To solve this problem, many fault detection algorithms use voting or joint probability tables to identify the fault source. This kind of solution leads to extra message exchange.

To solve this problem, this work proposes proposes the use of three values, to be transmitted by every node i: the sensed value v_i, the value $\hat{v}_i$ predicted by its model, and the confidence level c_i, assigned by the issuing node. By knowing these three values, received from every correlated sensor node, each node can autonomously verify the correctness of its readings, without extra messages. The process is explained next.

On every node i, when an input value v_k arrives from a correlated node k it checks the confidence lower against a predefined threshold γ. If it is greater than the required limit (i.e. $c_k > \gamma$), then the local predictor will use the read value v_k as input to calculate its own $\hat{v}_i$ and to calculate its confidence level c_i. Otherwise (i.e. $c_k < \gamma$), the current node will use the predicted value $\hat{v}_k$ to do its computations. The fact that $c_k < \gamma$ means that a significant difference between the sensed and the predicted values was detected by the node where the data was originated. Assuming that all inputs of this sensor are sane, it is reasonable to assume that the validated model has produced a accurate prediction, and it can be used as a good approximation of the correct value. So, by replacing the low confidence sensor's value by its predicted value, the sensor's failure will no longer affect the results of other sane sensors and, consequently, not interfere in their confidence level. It also enables the identification of the faulty node, as it will be only one to show lower confidence. At every node, the confidence attribution process, as shown in the highlighted circle in figure 16, is described in algorithm 3.

As stated earlier, the proposed solution assumes that a correlation between the values of a set of sensors S exists. As they are not independent, it is possible to build a predictor $P_i(s_1, s_2, \ldots, s_j)$ for every sensor $s_i \in S$. This predictor is able to calculate an accurate approximation of the actual value of s_i based on the most recent values of other j correlated sensors. During the training process, the MAE is calculated for every sensor. The MAE is used to define a threshold ε_i, for every sensor: $\varepsilon_i = \beta_i \times MAE$. Therefore:

$$\forall s_i,\ \hat{v}_i = P_i(s_1, s_2, \ldots, s_j) \implies |\hat{v}_i - v_i| \leq \varepsilon_i$$

This assumption must be evaluated for the domain in which the solution will be applied. If no accurate models can be found, with acceptable values for the threshold ε, the proposed solution does not fit. Take as an example some sensors that measure unrelated phenomena, and therefore have completely independent readings.

Let us consider that each sensor s_i reads its value v_i, and knows only the values of the other nodes on the previous period of time, then it can calculate a prediction $\hat{v}_i = P_i(s_1, s_2, \ldots, s_n)$. The following situations can occur:

- The error is less than the threshold: $|v_i - \hat{v}_i| \leq \varepsilon_i$. The node assumes that the reading is correct.

Algorithm 3 Confidence Attribution Routine

```
1: procedure Confidence_Attribution
2:     // Calculates the predicted value and confidence
3:     $\hat{v}_o \leftarrow model(values)$
4:     $diff \leftarrow |y - \hat{v}_o|$
5:     if $diff \leq (\beta * MAE)$ then
6:         $conf_o \leftarrow 100$
7:     else
8:         $conf_o \leftarrow (|y - \hat{v}_o| - \beta \times MAE)/(\alpha \times MAE) \times 100$
9:         if $conf_o < 0$ then
10:            $conf_o \leftarrow 0$
11:        end if
12:    end if
13:    // Changes low confidence readings to the predicted values
14:    for each $c_i \in confidences$ do
15:        if $(c_i < \gamma)$ then
16:            $values[i] \leftarrow predictions[i]$
17:        end if
18:    end for
19:    $\hat{v}_p \leftarrow model(values)$
20:    $diff \leftarrow |y - \hat{v}_p|$
21:    if $diff \leq (\beta * MAE)$ then
22:        $conf_p \leftarrow 100$
23:    else
24:        $conf_p \leftarrow (|y - \hat{v}_p| - \beta \times MAE)/(\alpha \times MAE) \times 100$
25:        if $conf_p < 0$ then
26:            $conf_p \leftarrow 0$
27:        end if
28:    end if
29:    if $(conf_o > conf_p)$ then
30:        return $(v, \hat{v}_o, conf_o)$
31:    else
32:        return $(v, \hat{v}_p, conf_p)$
33:    end if
34: end procedure
```

- The error is greater than the threshold: $|v_i - \hat{v}_i| > \varepsilon_i$. Nothing can be stated about the correctness of the read value, because the error can be due to a faulty value received from a correlated sensor, used as the predictor's input.

- Assuming that all model's inputs are correct, then a large error indicates that the current node is really faulty. But in the next round – data period – the node's incorrect value v_i will be used as input by some other sensors' predictors. This probably will impact every predicted value $\hat{v}_j$, making $|\hat{v}_j - v_j| > \varepsilon_j$ for every node v_j that uses v_i as input.

For some nodes, the weight of v_i in the predictor can be very small, so an error causes just a slight deviation on the predicted value, keeping the prediction error bounded. By using just the local predictor's output and the thresholds ε fol all sensors, an error will rapidly propagate to the sensors that rely on the faulty sensor's value as input. This makes it impossible to distinguish the faulty sensor from the sane ones. The only information available at each node is that there is a noticeable difference between the predicted value $\hat{v}$ and the read value v.

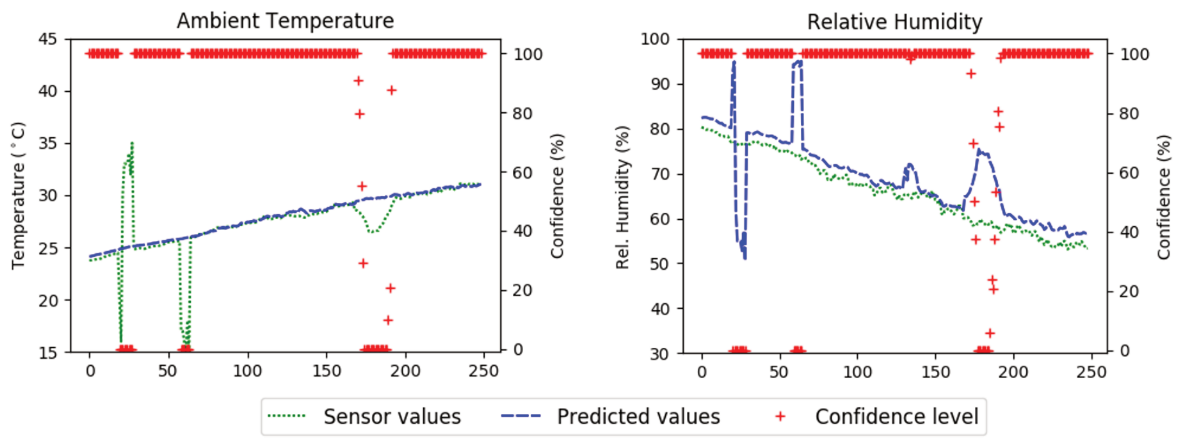
If the nodes transmit their predicted value $\hat{v}$ and their confidence level c along with the read value v, then all nodes get enough information to improve the decision about their own data. Lets us now consider that each sensor s_i, that now knows $v_j, \hat{v}_j$ and c_j of every correlated sensor s_j, reads its value v_i and calculates a prediction $\hat{v}_i = P_i(s_1, s_2, \ldots, s_n)$. The following situations can occur:

- All input values have high confidence assigned by their origin nodes: $\forall s_j \in S \mid c_j > \gamma$. As all inputs have high confidence assigned to their values, then the confidence level assigned by the equation 5.1 can be trusted. The local node s_i can assume that its sensed value is correct if the $c_i \leq \beta \times MAE_i$. If the read value is faulty, it is expected that the confidence assigned will decrease, as the model's inputs are all correct.
- Some input values have low confidence assigned by their origin nodes: $\exists s_j \in S \mid c_j < \gamma$. The local node uses discards the read values v_j and uses the predicted values $\hat{v}_j$ to calculate its own prediction. This will lead to a better prediction, isolating the low confidence only to the nodes that are really faulty.

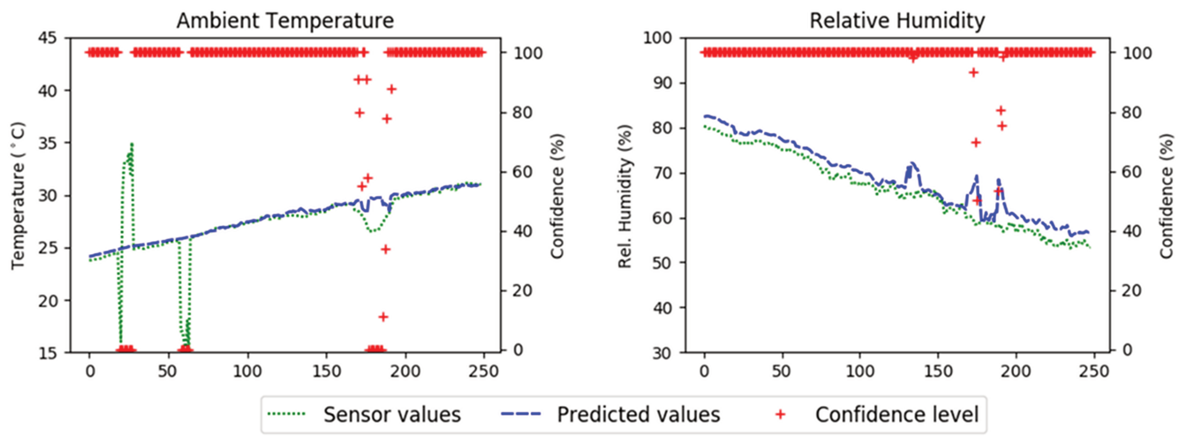
The interference problem and the solution proposed is illustrated in figure 18. The figure 18a shows how the use of the raw sensor values as the predictor's input, leads to an error propagation through the models. Deviant data in the first sensor (ambient temperature) makes the predictor of the second sensor (relative humidity) show also a deviant result, making the difference between the read value and the predicted value be large. The direction and the amplitude of the prediction errors show clearly that there is a strong inverse correlation between these two variables. Both sensors have their confidence level drop to zero, as it is impossible, using only the raw data read from the sensors, to determine the error source. Each node only detects a large difference between the read and the predicted value, and the confidence of both drops to zero.

Figure 18b shows how the proposed algorithm can identify and isolate the error of a specific sensor. When the first model calculates its prediction with correct inputs, the result will be distant from the (wrong) reading, getting a low confidence level. It then searches for low confidence values in the model input, without success. When the second model makes its prediction with a wrong input, it also gets a low confidence level. It then finds an input (from the first sensor) with low confidence. It replaces this input with its prediction, the new prediction will be near to the read value, and the node's confidence is recomposed. As the substitution begins only when the confidence falls below the γ parameter, when the sensor deviates slowly from the correct readings, the confidence level starts to decrease on both sensors, as it can see on the lower graphs, between samples 150 and 200. But as soon as the confidence level goes under γ, set to 40 in this example, substitutions are made and the confidence level of the correct sensor is promptly restored.

It is assumed that only one sensor, or at least a small set of sensors, will be faulty at any time. If several correlated nodes are faulty at the same time and in the same region, the



(a) Confidence Attribution only by sensed values.



(b) Confidence Attribution replacing low confidence input by it's predicted value.

Figure 18 – Fault isolation by algorithm 3. In (a), the predictions of Relative Humidity are distorted by Ambient Temperature. In (b), the algorithm isolates the distortion when confidence goes below $\gamma = 40$.

proposed solution is not able to identify the nodes with incorrect values. As a consequence, all nodes will start assigning low confidence levels to their readings. But this behavior can also be observed as a consequence of the *concept drift* phenomenon, explained next.

5.3 CONCEPT DRIFT AND MODEL UPDATE

In several monitored environments, the measured values can change in range, and their correlations may vary over time, characterizing many WSNs as *non-stationary environments*. As described by (WEBB et al., 2016), a *Concept* is the classification or prediction result of a vector of values α. If the result changes over time, i.e. $P_t(\chi) \neq P_u(\chi)$, then a *Concept Drift* occurs, as the same input set χ produces different results at times t and u, and both are correct. Similarly, (GAMA et al., 2014) defines it as $\exists \chi : P_{t_0}(\chi, y) \neq P_{t_1}(\chi, y)$, meaning that the joint distribution of a set of input variables χ and the target variable y may vary from time t_0 to time t_1.

In their work, O'Reilly et al. (2014) present a process flow for anomaly detection under such conditions, composed of five sub-processes, enumerated below:

1. *Change detection*: achieved through data monitoring to detect changes in the data distribution. If a significant change occurs, then a model update must be performed.
2. *Training set formation*: the data vector for model construction and training is formed using sliding windows techniques, discarding n oldest samples and adding n new samples to the training set.
3. *Model selection*: the optimal parameter set is determined for the new training dataset.
4. *Model construction*: with the parameters determined in the previous step, a new model is constructed. It can be done in *batch mode*, discarding the $model(t-1)$, and then building a new $model(t)$ from scratch, or in an *incremental mode*, where the $model(t-1)$ and n new data vectors are used to build the $model(t)$.
5. *Anomaly detection*: the new $model(t)$ is used as the new anomaly detector for fresh data $X_{t+1}, X_{t+2}, \ldots$.

The learning framework proposed by Ditzler et al. (2015) shown in figure 19 applies to such environments. When a change is detected, the model is updated and sent to the classifier. When applying this framework to WSN s, the *Feature Extraction*, *Change Detector* and *Adaptation* processes can run on a dedicated server or the Cloud, in a centralized approach. Once the model is built or updated, it can be transmitted to the WSN nodes and used to assign confidence to their readings. It can be the sensor node itself or intermediate nodes, in a hierarchical architecture.

As the environment changes, the *Change Detector* process presented in figure 19 has to detect the changes and trigger the *Adaptation* process. Change detection methods can be grouped into four main families (DITZLER et al., 2015):

- Hypothesis Test: uses statistical techniques to verify the classification error of a fixed-length set of readings. The variation of the classification error is compared to the error of the training dataset.
- Change-Point Methods: also uses a fixed-length data sequence, analyzing all partitions of the data sequence to identify the instant when the data changes its statistical behavior, called change-point. The main drawback of this method is the high computational complexity.
- Sequential Hypothesis Test: instead of analyzing a fixed-length window of data, this method inspects each incoming sample, until they have enough evidence that a change has occurred or not.
- Change Detection Tests: are specifically designed to analyze the statistical behavior of data streams sequentially. Most of them operate by comparing the prediction of absolute

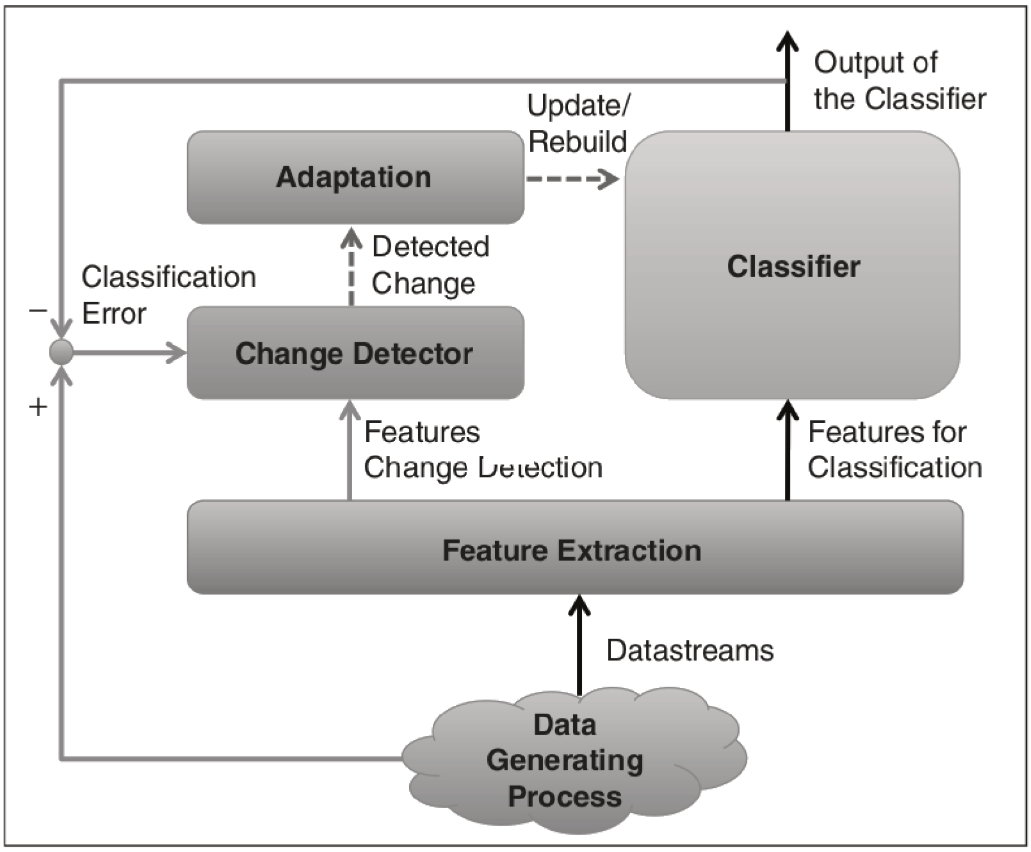


Figure 19 – Model learning and change detection scheme (DITZLER et al., 2015).

error or its variance to a specified threshold. The threshold is hard to determine at design time. Some adaptive algorithms were proposed, using cumulative errors.

The authors also propose the use of hybrid change detection. A Change Detection Test can be used in a first layer, followed by a validation layer that uses a Change-Point method.

The Model Update process, performed after a change is detected, consists in retraining the model with new data. The main approaches are *windowing*, *weighting* and *random sampling*. At this step, one of the main questions is if the model has to forget the oldest rules and reinforce the new ones, or if the learning has to be cumulative. In the former case, the model is entirely retrained using new data. This implies retraining the model every time a change is detected but can lead to smaller models. The latter case, incremental learning can keep past knowledge, but models tend to be larger, demanding more memory and processing.

In the proposed solution, a newly deployed WSN has to run without confidence attribution for some time, to gather enough data to build and train models for the different sensors. Models are then built for each type of sensor in the same region, and can be sent to them in a *group communication*. The application can control the model transmission frequency to avoid network flooding. Model updates are expected only when the correlation between the sensed values changed. In a *training phase* some overhead will happen when some models have to be updated. After some time – that depends on the environment dynamics – it is expected to occur very rarely. The compressed size of the model (around 3 KB) makes it feasible to be transmitted over the WSN. The proposed solution does not fit well to environments where the correlations are changing continuously at a high rate. The use of cumulative learning, although producing larger models, was chosen in the first experiments as it is expected that environments with cyclic changes will *stabilize* over time in the learning process, demanding much less (or even none) updates after running for a period.

Concept Drift Detection

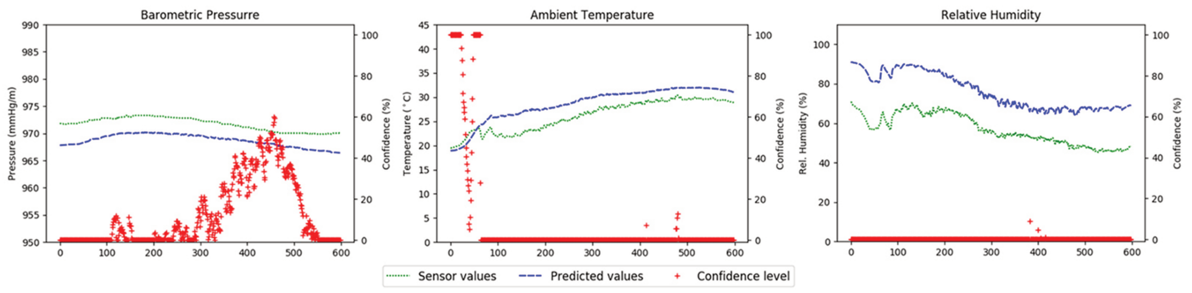


Figure 20 – General confidence drop on concept drift.

Concept drifts are not easily detected as they may be confused with long-lasting erroneous readings. Several methods for concept drift detection were proposed, mainly applying statistical analysis of errors and correlations. The first attempt in concept drift detection was the Exponentially Weighted Moving Average (EWMA) chart method proposed by (ROSS et al., 2012), which verifies the changes in the moving average over the last *N* readings, with no need to keep buffers. However, when applied to a single time series, the algorithm classifies long error sequences as concept drifts, so it is mandatory to first classify if the readings are reliable, before applying the formulas. This makes it harder to apply this method directly into the proposed solution, as concept drifts may also be classified as errors, or at least as readings with lower confidence.

When a concept drift occurs the predictors built at the server with outdated data will not be able to correctly handle that changes. It is expected that the majority of the models start to assign lower confidence levels to their data. For example, if the training is made with data obtained in summer days, then the model will assign low confidence to Relative Humidity, Ambient Temperature and Barometric Pressure when the data is collected in winter days, they show correlations slightly different (figure 20). As shown by the figure, the predicted values are clearly following the sensor readings, but are not close enough to be considered correct.

At the server side – where enough computational power is available – more elaborated algorithms can be used to verify if some sequences are real concept drifts or long lasting errors, like sensors slowly deviating their readings from the correct ones. Models comparing readings variations of each sensor, and comparing them to the variations in other sensors must be built, mixing statistical analysis and predictions results. These *ensembles* can use some voting mechanism, for example, to decide if a concept drift has occurred. Another solution is use the outputs of all the different detection mechanisms and build and train a classifier to verify if the data streams should be considered errors or concept drifts. The algorithm then has to decide if a set of deviations, occurring for a long enough period of time, is an error or a concept drift.

As the predictors' models are expected to reflect well the correlation between the sensors' data, it is expected that, if they change due to a Concept Drift, then all – or at least the majority – of the predictors will change the confidence assigned to the sensors' values. So, the confidence level assigned by the algorithm can also be a valuable source of information for the *Change Detector* algorithms.

5.4 CASE STUDIES

In this section, a set of real datasets were used to evaluate the proposed solution for fault detection on the data. When the dataset is labeled, with *good* and *faulty* samples correctly identified, they are used in the evaluation. Otherwise, it is assumed that data is correct, reinforcing this assumption with manual inspection of the dataset.

The evaluations were made using Python3 scripts, using the scikit-learn (PEDREGOSA et al., 2011), Theano (BERGSTRA et al., 2010; AL-RFOU et al., 2016), Tensorflow (ABADI et al., 2016) and Keras (CHOLLET, 2015) frameworks.

For every test case, the number of features (input set) was chosen in an exploratory way, starting from a limited number of sensors (5) and increasing the number of sensors while evaluating the MAE on a restricted training/test data set. Some manual inspection on graphs comparing real/predicted values was also carried out. As the objective was to evaluate the whole mechanism's efficiency, it was assumed that all sensors were close enough to others, and could be used. However, some corrections had to be made, making evident that the feature selection had to be made with more sophisticated algorithms, which was not the main objective of this work. The methods used in this work were the *SelectKBest* selector, from the scikit-learn framework, and Pearson's Correlation Coefficient analysis, to group the sensors with data with stronger correlation coefficients.

The model building and training phases were done with a subset of data, of a representative period. The training and test procedure followed the standard procedure for machine learning algorithms: around 80% of data was randomly chosen for training and the other 20% for the test. Usually, 5 rounds were made, with new train/test datasets chosen at every round. In the end, the models were applied to the entire selected dataset, to calculate the MAE for every sensor data.

Photovoltaic Solar Energy

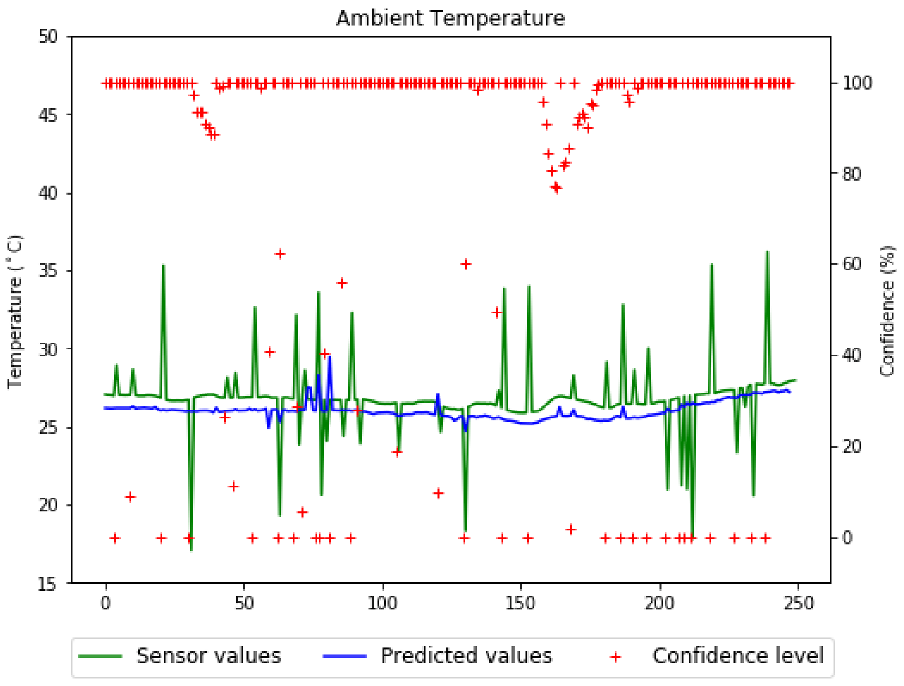
The first experiment used data from one year of readings from nine environment monitoring sensors from a photovoltaic solar energy plant. All data had the timestamp of the readings, taken in a 1-minute interval. In the experiment, the data from the first 15 days of January was used to perform the features selection using the *SelectKBest* selector from the scikit-learn framework, simulating the first data obtained by a new deployed WSN.

For each sensor, four other sensors with the highest correlation were selected to be the predictor's input. The sensors and the output of the feature selection process for each are shown in Table 5. Along with the selected sensors, the hour and minute of each reading instant were selected as relevant inputs for the predictors. It is assumed that date and time can always be used as predictors' input, as they are available from the device's internal clock. It is also assumed that the WSN devices are running some protocol to keep their clocks synchronized at a reasonable level.

Table 5 – Feature Selection Result

Target	**Selected Predictor Inputs**
Diffuse Solar Irradiation	Direct Solar Irradiation, Global Solar Irradiation, Barometric Pressure, Datalog Internal Temperature
Direct Solar Irradiation	Diffuse Solar Irradiation, Global Solar Irradiation, Ambient Temperature, Datalog Internal Temperature
Global Solar Irradiation	Diffuse Solar Irradiation, Direct Solar Irradiation, Barometric Pressure, Datalog Internal Temperature
Barometric Pressure	Diffuse Solar Irradiation, Global Solar Irradiation, Ambient Temperature, Datalog Internal Temperature
Rainfall Index	Diffuse Solar Irradiation, Barometric Pressure, Ambient Temperature, Datalog Internal Temperature
Wind Direction	Diffuse Solar Irradiation, Barometric Pressure, Ambient Temperature, Datalog Internal Temperature
Ambient Temperature	Global Solar Irradiation, Barometric Pressure, Datalog Internal Temperature, Relative Humidity
Datalog Internal Temperature	Global Solar Irradiation, Barometric Pressure, Ambient Temperature, Relative Humidity
Relative Humidity	Global Solar Irradiation, Barometric Pressure, Ambient Temperature, Datalog Internal Temperature

After the feature selection, an Artificial Neural Network (ANN) was built for every sensor, using the Multilayer Perceptron architecture. All ANNs have the same structure, with six neurons in the input layer (hour, minute and the four selected sensors), five neurons in the hidden layer and one neuron in the output layer. The Keras API (CHOLLET, 2015) and the Theano library (AL-RFOU et al., 2016) were used to build, train and evaluate the ANNs. The small size of the model makes it suitable to be transmitted to the sensor nodes and executed locally to make the sensed value predictions. After the ANNs were built, the Mean Absolute Error (MAE) was calculated for every sensor, and the values are shown in Table 6. To illustrate again the meaning of MAE in the confidence formula, the MAE value for the ambient temperature, if $\beta = 1.0$, makes a sensor reading v be considered having 100% confidence if it lies in the range $[\hat{v} - 0.41697, \hat{v} + 0.41697]$, where $\hat{v}$ is the predicted value.



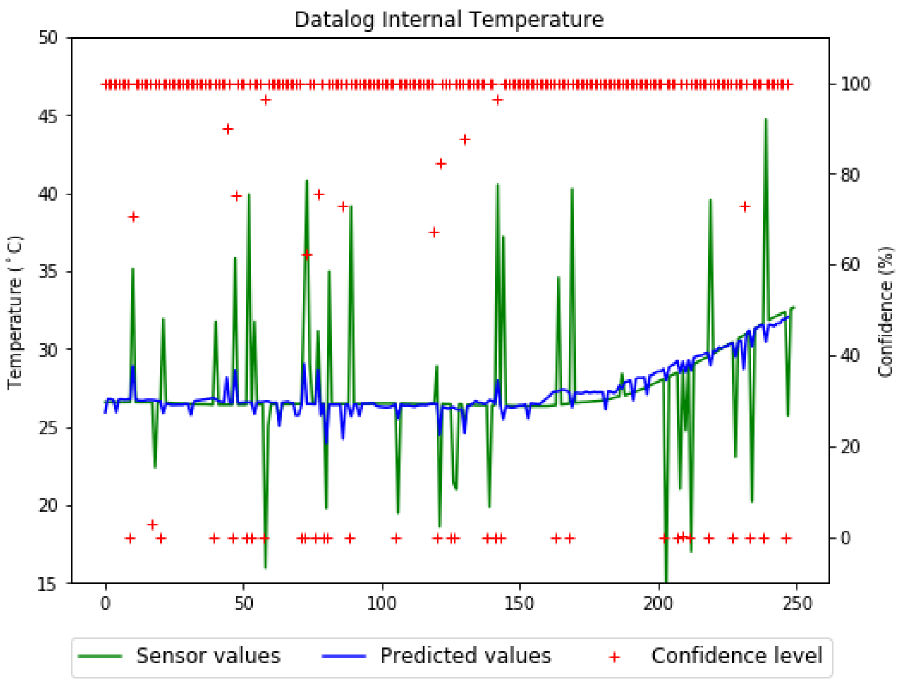


Figure 21 – Error injection for evaluation of algorithm 3.

Once the model building phase was finished, several simulations were carried out to verify the accuracy of the proposed confidence attribution scheme. Five different data chunks were extracted from the original data set left out of the training process, each one with 1440 samples (one-day sampling). The evaluation was then carried out in 2 steps. First, to evaluate the model's efficiency in identifying and isolating errors in the defective sensor (its origin), several errors of the *outlier* type (SHARMA; GOLUBCHIK; GOVINDAN, 2010) were injected. An outlier is a single reading that shows a value distant from the real value, with the sensor returning to normal values in the next readings. Figure 21 shows the error injection result for the

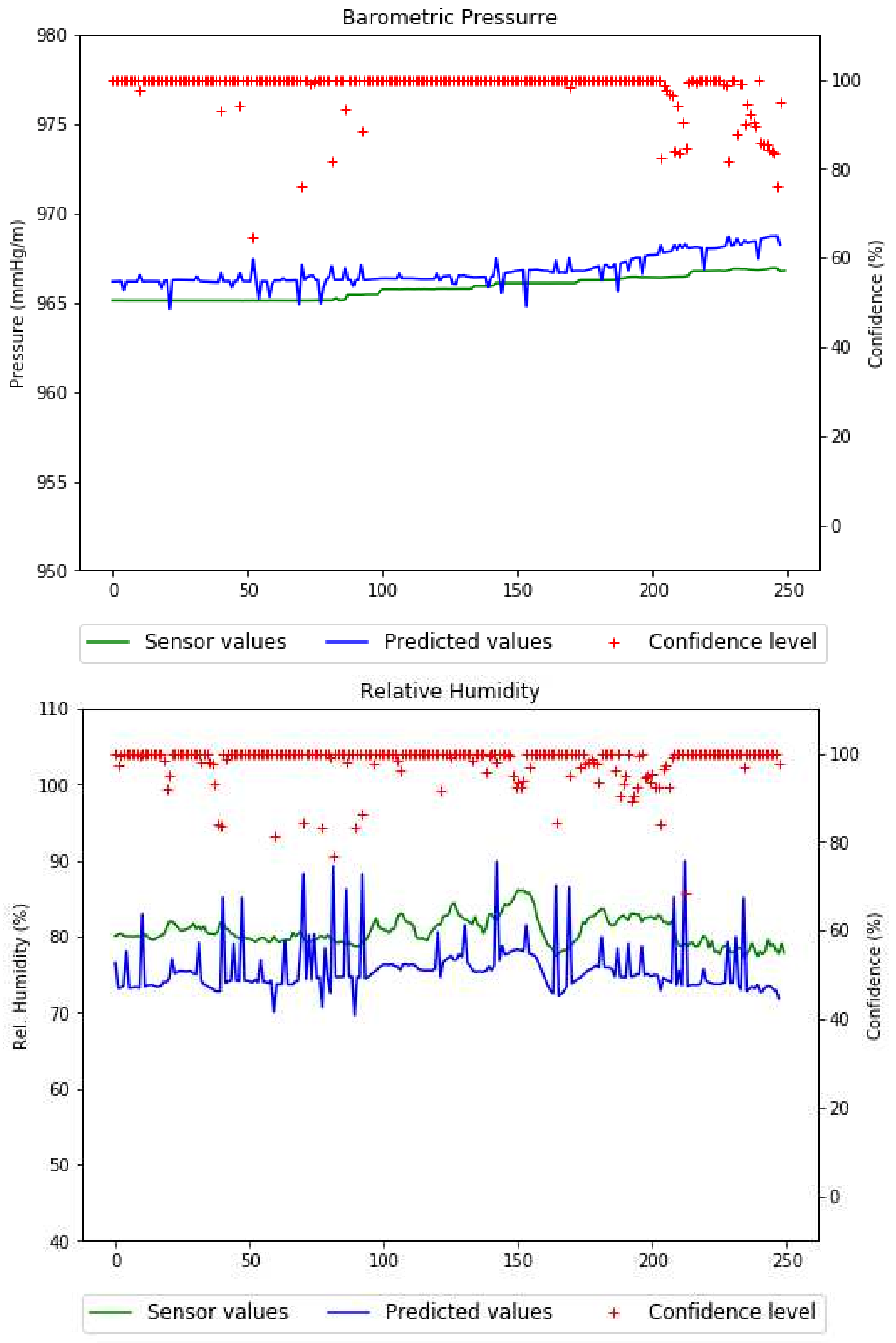


Figure 22 – Error injection reflected on other sensors.

Ambient Temperature and the Datalog Internal Temperature sensors. The green lines show the sensor data, with the errors visible as the peaks in the graph. The blue lines show the predicted value. It can be seen that at some point also the predicted values show a small peak, meaning that the prediction was affected by other sensors' values, but not enough to make the confidence decrease as much as needed to perform substitution of the read value by the predicted one. The algorithm 3 was executed on the data chunks, measuring the detection and false positives rates. Several experiments were carried out, combining different values of the three algorithm parameters - namely β, α, and γ - in order to verify their influence on the algorithm efficiency.

Table 6 – MAEs calculated for predictors.

Sensor	**Model's MAE**
Diffuse Solar Irradiation	2.5799
Direct Solar Irradiation	15.8076
Global Solar Irradiation	16.2823
Barometric Pressure	0.7605
Rainfall Index	0.0017
Wind Direction	28.9098
Ambient Temperature	0.41697
Datalog Internal Temperature	0.7318
Relative Humidity	3.3265

In the second set of experiments, the other three types of error defined by (SHARMA; GOLUBCHIK; GOVINDAN, 2010) — *peaks*, *"stuck-at"* and *noise* — were injected in the sensors in order to verify the algorithm's sensitiveness to each type of error. The errors were randomly injected in sequences of 15 values showing one type of error, in one or in two sensors. The sequences are injected in random positions of the dataset until reaching the desired number of errors. It is important to make the remark that the objectives of this work do not include error classification, only to identify the point where a sensor reports a faulty value. Therefore, when different types of error appear, causing a significant difference between the read and the predicted value, the algorithm will only assign low confidence to the data, no matter what kind of error occurred.

Algorithm Parameters Evaluation

To evaluate how the algorithm parameters' influence on the results, several errors of *outlier* type were randomly injected in each data chunk. The amplitude of the error was also randomly defined. So, it is expected that not *every* error is detected with confidence below γ, as some generated values can fall inside the range of acceptable values. The Ambient Temperature and the Datalog Internal Temperature sequences were chosen to have their values changed, as they are input for all other predictors, and also show high interdependence. The errors were injected as described previously. The process was repeated ten times with each data chunk, and the mean values of the detection and false positive rates were calculated.

The *Error Detection Rate (EDR)* (Equation 5.5) is the percentage of the injected errors that are correctly identified by the proposed algorithm. An *Interference Error* occurs when a node with no error injection presents a confidence level below the γ parameter (algorithm 3, line 15), at the next period where an error was injected in another sensor. As explained in formula 5.4

the next period is when each read value is used as input of the predictors. This implies that low confidence on this sensor results from a wrong prediction caused by the injected error, which is unexpected and the algorithm tries to eliminate or, at least, to minimize. As some errors may occur in the sample data – as it is data from real sensors –, the confidence of all sensors *before* the error injection was calculated and stored. For each low confidence found *after* the error injection, it was compared with the recorded value. Therefore, it is possible to correctly identify the interference errors. The *Interference Error Rate* (IER - Equation 5.6) is calculated based on the number of injected errors, making it possible to compare the results of experiments made with different sizes of input set and different numbers of injected errors. The formulas are defined below.

$$EDR = \frac{detected\_errors \times 100}{injected\_errors} \tag{5.5}$$

$$IER = \frac{interference\_errors \times 100}{injected\_errors}. \tag{5.6}$$

The experiment's results are shown on Tables 17 and 18, varying the minimum confidence (γ) from 90 down to 20 in intervals of 10, and α and β varying from 1.0 to 4.0 in a step of 0.5, making a total of 392 evaluated combinations. Due to the table size, it was moved to the appendix A. The data is also plotted in figure 23. As stated before, small values of β means that the current reading can show only a little discrepancy from the predicted one to be considered 100% correct. The opposite is true for larger values of β. The α parameter determines the width of a range of values where the confidence decreases from 100% to 0%. The wider the range, the slower the confidence decreases. So, small values of α make the algorithm drop confidence very fast, and vice versa. Finally, the γ parameter defines the confidence level below which the algorithm tries to replace the read values by the predicted ones, searching for a possible sensing error. The lower the value of this parameter is, the more *"lenient"* is the algorithm with the investigation of possible errors.

As the objective is to maximize the number of detected errors and, at the same time, to minimize the number of injected false positives, the best combination of parameters will be the one who gets the biggest difference between these two results. To show this relation, the difference (*Diff*) between the EDR and the IER was added to the table, as an additional column for each combination of parameters. The highest EDRs were obtained using the most restrictive combination of parameters (β = 1.0, α = 1.0 and γ = 90), detecting 94.2% of the injected errors. But this combination also shows the highest IER, as high as 79.5% of the injected errors. This is expected because any reading that deviates a bit more than the MAE (Mean Absolute Error) from the model will be considered an error and tested against the predicted value. The IER decreases as the value of γ decreases, and also decreases from left to right in the tables, as β increases, and in each test group of γ, from upper to lower lines, as α increases. On the right side of the table, where β is bigger, the EDR and IER values show a noticeable decrease,

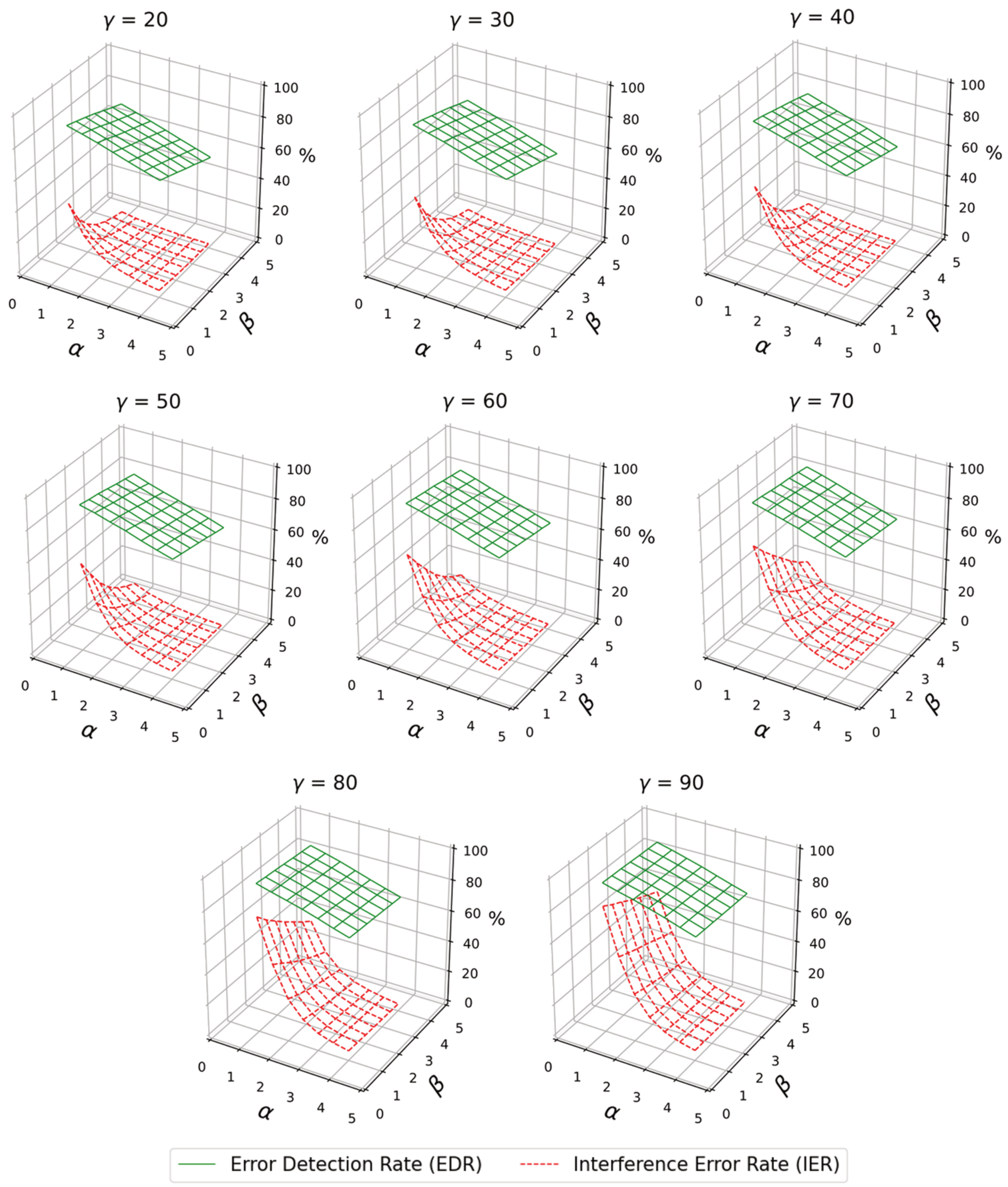


Figure 23 – α, β and γ influence on Error Detection Rates and Interference Error Rates - outlier errors

mainly the IER. Analyzing the numbers on tables 17 and 18, it is possible to see that average values for the parameters show the most balanced results.

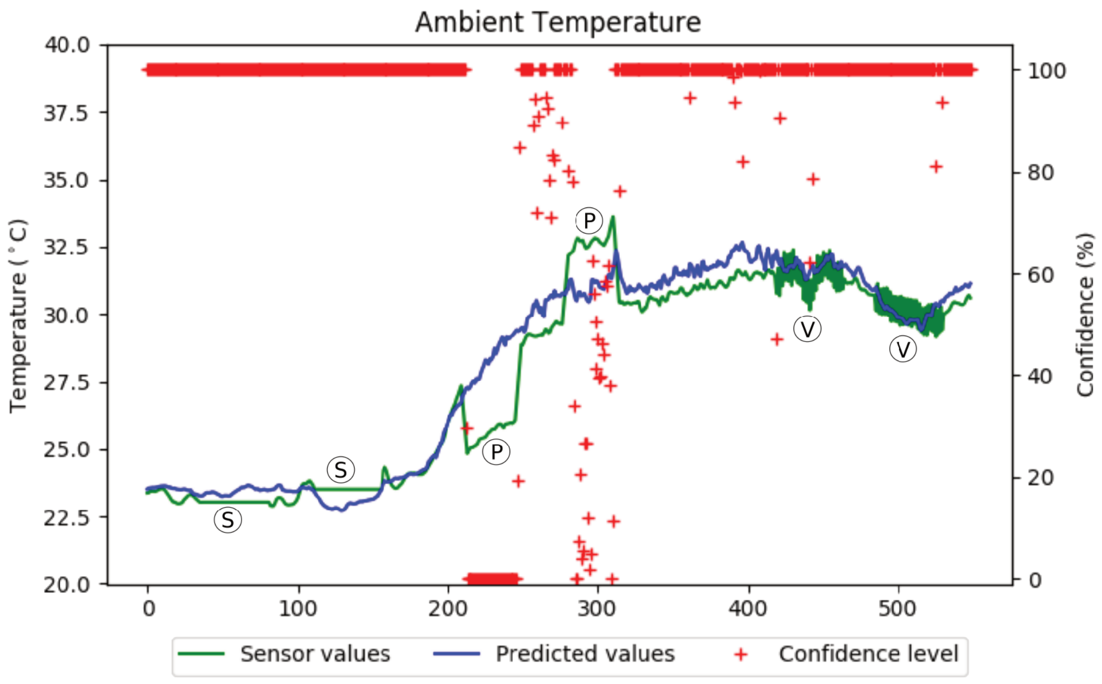
The parameters can be individually adjusted for different scenarios and application needs. Therefore, a reasonable starting point is setting α and β parameters to a value around 2.5 and γ to 50, and tuning them until the desired detection and false-positive rates were obtained. As just three numbers must be adjusted, thinking on a WSN, it would be necessary only to broadcast a single message to adjust the algorithm's parameters on all sensors, or a subset of sensors of the same type. As the MAE value is expected to be larger for models of sensors with highly varying readings, the parameters can also be set individually for some sensors.

Error Type Coverage

A second set of evaluations was executed in order to identify the algorithm's effectiveness on detecting the other three types of errors defined by Sharma, Golubchik & Govindan (2010): *peak*, *stuck-at* and *noise*. The error injections mechanism was adjusted to generate several sequences of errors in the same data chunks used in the previous evaluation. Examples of the injected errors and their correspondent confidence levels are illustrated in figure 24. In these graphs, each type of error was injected twice. The different error types are marked on the graphs with the letters: **S** for *stuck-at* errors, **P** for *peak* errors and **V** for *variance* (or *noise*) errors. The parameters used to create the graphs of figure 24 were $\alpha = 2.5$, $\beta = 2.5$ and $\gamma = 50$.

The peak errors are very similar to the outlier errors, used in the first evaluation. So, the confidence level drops as soon as the sensor's values deviate enough from the usual (predicted) readings. A remarkable situation occurs when some difference exists between the predicted values and sensor readings. In this situation, a peak can make the sensor's readings go into the directions of the predictions. It makes the values remain in the range with confidence high enough, and the sequence may not be classified as an error. The second peak of the first graph in Figure 24 illustrates this, as the sensor readings get various confidence levels, from 0 to 100, when the peak error is occurring (around sample 300).

The proposed solution does not identify directly the noise error, as the readings do not exceed the *"thresholds"* defined by the α and β parameters. The graphs of figure 24 shows that high variance errors (**V** marks) around the normal value are not detected as an error. Only when the noise is too high, the sensor's confidence starts to decrease. If the noise makes the sensed values fall too far from the expected ones, the proposed solution will detect them as it does with the outliers. The noise can be an indicator of a malfunction of the sensor or some interference, but while it does not cause readings too distant from real values, the confidence attribution scheme will ignore this kind of behavior, as they are not in the scope of this work. Simple solutions of variance analysis can be used at the server to indicate sensors with noise in their readings, indicating the need for some maintenance. Noise errors can also be detected if they produce readings outside the $\beta \times MAE$ threshold, and the sensor starts to produce sequences of readings with different levels of confidence. Again, this sensitiveness can be adjusted by setting the α and β parameters. The *stuck-at* is an error type that can show a different impact on the sensor's confidence, depending on *when* it occurs and *how long* it lasts. When the readings of the sensor *freezes* on a value where the normal readings do not show variations, the error is not detected. This is a situation similar to the noise error and is illustrated in the first graph of figure 24. But when the sensor's normal readings are ascending or descending – like the temperature in a heating or in a cooling process – then readings of a constant value will occur in an interval, making the deviation to be detected. This will make the sensor's confidence fall as the expected (predicted) readings vary. This is well illustrated in the second graph of figure 24.



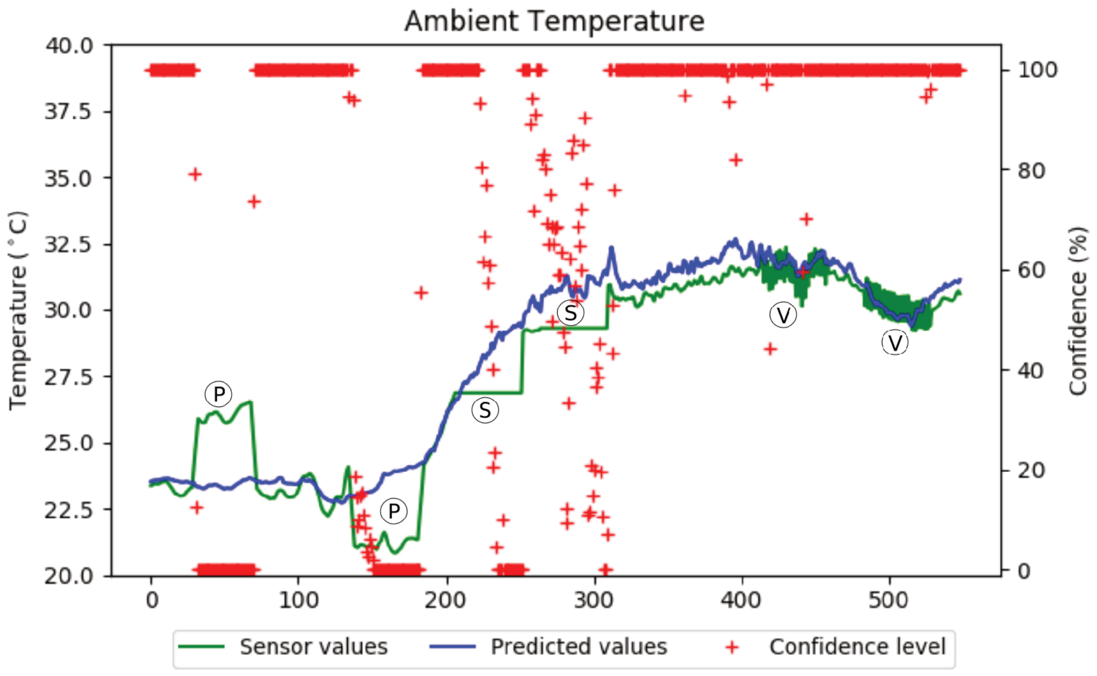


Figure 24 – Different error types injection.

Obviously, if an error makes the ADC (Analog/Digital Converter) be stuck at a zero or at a distant value (like the maximum value of the register), then the algorithm will drop the sensor's confidence instantly.

Table 7 shows the results for some combinations of the algorithm parameters values. As already discussed, the EDR and IER for the *stuck-at* and *noise* errors are significantly lower than for *outlier* and *peak* errors. The peak errors also show an EDR lower than the outlier errors, mainly by the fact that the duration of peaks that produce values that will approximate to the sensor readings, and consequently not dropping confidence, have a more significant statistical

Table 7 – Parameters Evaluation Results

			Outlier			**Peak**			**Stuck-At**			**Noise**		
γ	α	β	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**
20	1.0	1.0	90.59	40.99	49.60	61.72	27.24	34.48	19.56	14.21	5.34	17.27	14.37	2.90
	1.0	4.0	73.30	2.60	70.70	57.59	7.02	50.57	12.91	1.96	10.94	2.62	0.83	1.80
	2.0	2.0	80.32	6.87	73.45	64.16	12.97	51.19	15.61	5.18	10.43	6.95	3.29	3.66
	3.0	3.0	69.81	1.63	68.19	54.30	4.61	49.70	11.62	1.32	10.30	1.85	0.55	1.30
	4.0	1.0	76.70	4.30	72.40	61.21	10.41	50.80	14.75	3.72	11.03	4.73	2.56	2.17
	4.0	4.0	58.38	0.67	57.70	42.86	2.50	40.36	8.16	0.43	7.73	0.41	0.10	0.31
50	1.0	1.0	92.12	55.03	37.09	64.07	33.91	30.16	22.09	20.10	1.99	22.91	19.24	3.67
	1.0	4.0	75.10	3.30	71.80	59.53	8.75	50.79	13.86	2.73	11.13	3.57	1.48	2.09
	2.0	2.0	84.24	11.24	73.00	68.03	19.89	48.13	18.21	10.21	8.01	10.91	7.74	3.17
	3.0	3.0	75.10	3.35	71.75	60.07	9.07	51.00	14.62	2.84	11.78	4.23	1.89	2.34
	4.0	1.0	84.2	11.20	73.00	69.07	21.49	47.58	19.29	11.17	8.12	11.51	9.35	2.16
	4.0	4.0	65.79	0.99	64.81	50.53	3.21	47.32	10.76	0.97	9.79	1.36	0.38	0.98
90	1.0	1.0	94.16	79.54	14.62	66.61	44.54	22.06	25.49	29.03	—	31.24	26.80	4.44
	1.0	4.0	77.30	4.60	72.70	61.78	10.97	50.81	14.91	4.07	10.84	5.14	2.96	2.18
	2.0	2.0	88.56	25.84	62.72	72.63	33.54	39.09	22,82	22.39	0.43	18.22	18.89	—
	3.0	3.0	82.26	8.64	73.62	67.50	18.14	49.36	18.73	7.97	10.76	10.03	7.09	2.94
	4.0	1.0	92.70	60.50	32.20	79.61	51.20	28.42	30.60	38.34	—	31.68	33.83	—
	4.0	4.0	75.63	3.63	71.99	60.83	9.57	51.27	15.37	3.19	12.18	4.81	2.33	2.48

impact than one outlier that does the same. If the simulation is adjusted to produce only large deviations when generating peak errors, the results would be very close to those obtained with the outlier errors. But this would artificially inflate the EDR, and would not follow the original definition of the error type. The *stuck-at* and *noise* errors show much lower EDRs, as the error values were mainly too close to the predicted values.

The results' analysis showed that the algorithm can identify sensors' errors that are greater than some *threshold*, and ignores errors that do not deviate too much from the *"normal"* sensor readings. It does not detect *noise* and *stuck-at* errors when they lay around the normal readings. These types of errors are easily detected by algorithms that use statistical analysis on a window of the *N* last readings. These types of detection and classification are out of the scope of this work. We are interested in identifying errors – from failures or intentionally injected by an intruder – that could lead to wrong decisions, e.g. in a Cyber-Physical System.

The proposed algorithm labels every reading with a confidence level, adjustable by the parameters. So, even if not classified as an error by the algorithm (falling below γ), every variation in the sensors' readings can reflect in a variation in the confidence level assigned to them. A small (or even 0) value of β and a reasonable value of α labels different readings with different confidence levels, that can be used to identify *noise* errors, for example.

Grand Saint Bernard SensorScope Dataset

The public dataset from the SensorScope project (BARRENETXEA et al., 2008) was also used to evaluate the proposed solution. It was chosen because it is a public dataset, and allows the comparison to another fault detection technique. The dataset consists of environmental data collected at the Grand Saint Bernard pass between Switzerland and Italy in 2007. It contains samples of temperature, humidity, and solar irradiation collected over 43 days with a temporal resolution of 2 minutes. The results obtained by applying the confidence attribution schema on this dataset and the results are compared to the results obtained by the Hybrid Fault Detection method proposed by Nguyen et al. (2013). They selected 10 sensors among the most faulty ones to evaluate the accuracy of fault detection and classification, and also applied time-series analysis and neighbor voting algorithms to detect four types of failure.

In the first step, the data samples were aligned in time. Two sensors that failed to produce data (their work focus was on sensor failure, this work focus is on data confidence) were excluded from the comparison. The resulting dataset contains 20180 samples. Subsequently, discrepant data points were manually tagged as faulty, in ranges. After that, sequences with only good data were selected at five different periods of the time series, resulting in a training set of 6800 records. The feature selection procedure was applied next, identifying the relationship between the variables across different sensors.

The ANNs were built following the process described in Section 5.4, determining, this way, the MAE for every sensor. The models were then evaluated for every sensor with average values: $\gamma = 50$, $\alpha = 2.5$, and $\beta = 2.5$. A node was considered faulty when the confidence reached 0. In their evaluation, Nguyen et al. (2013) counted errors by occurrence in each test interval T of 30 minutes, no matter how long the error remained. As our proposed algorithm evaluates each data point individually, a direct comparison of error counts is meaningless. So, the Error Detection Rate (which is called Success Ratio in their work) will be used for comparison. In their work, *Neighbour Voting (NV)* and *Time Series Analysis (TS)* were combined to achieve the best results ($NV \cup TS$), which is the basis for the comparison with our mechanism summarized in Table 8.

Table 8 – Compared Algorithm Evaluation

Sensor	**NV ∪ TS**	**Confidence Attribution**
Sensor 6	97.4%	75.1%
Sensor 7	93.3%	99.8%
Sensor 9	94.1%	90.7%
Sensor 15	93.1%	96.7%
Sensor 17	91.8%	93.4%
Sensor 18	92.5%	74.1%
Sensor 19	92.5%	83.9%
Sensor 20	97.6%	74.2%

The results show that for some sensors the EDR is significantly lower than the compared technique. Inspecting the data, it is possible to verify that the causes explained in Sections 5.4 are present in the evaluated dataset. Taking as example sensor 18 it is visible that the errors are mainly of the *suck-at* type, as shown in Figure 25. For a matter of clarity, confidence (red crosses) is only marked at 100 (no error) and 0 (error) to not hide data lines on the graph. The sensor reported -1 for a long time, and as the real temperature was around that value for a significant period (around sample 3000), it was not detected as a faulty value, once it was in the range of acceptable values defined by β, α and γ. Almost all of the undetected errors are due to the fact that *stuck-at* errors are not identified when they are close to the real values. On sensors where mostly peak and outlier errors occur, as for sensor 7, 15 and 17, the error detection rate of the algorithm proposed in this work outperforms the compared solution. Figure 26 shows the detection of this type of errors on sensor 15. On the other hand, Nguyen et al. (2013) had demonstrated that the peak errors (named *drift* by the authors) is the one with lowest detection rates, reaching from 76.9% to 84.6% of accuracy.

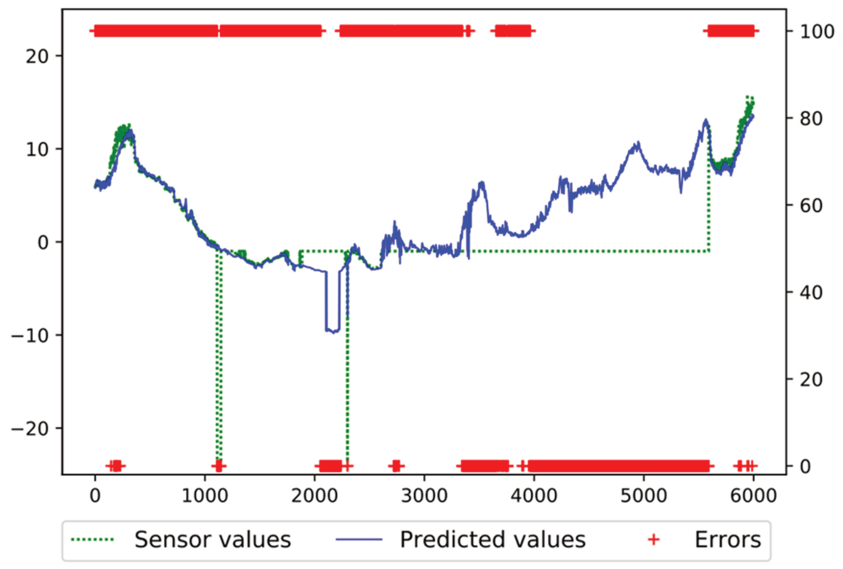


Figure 25 – Sensor 18 stuck at -1 for a long time.

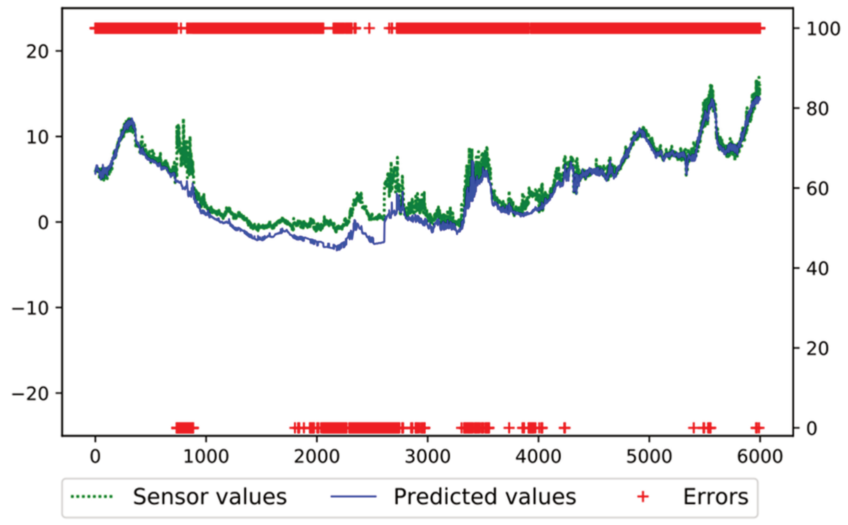


Figure 26 – Sensor 15 with peak and outlier erros.

This experiment helped to determine that the *type* of the error, combined with the sensed values *behavior*, are determinant for the algorithm's EDR. *Stuck-at* and *noise* errors, if not deviating from the expected values, can make the algorithm report high confidence for the data. It is also determined by the predictor's accuracy, that can report a large MAE for a given sensor, making the error range with high confidence be larger than expected. This can happen due to lack of model training, a weakness of the model to represent the relation between the inputs, or even the lack of a deterministic relationship between the inputs and the target, predicted value (HORNIK et al., 1989). A *stuck-at* error will be detected much easier when the sensor's data shows a certain degree of variation. As opposite, *noise* error will be easier to detect if the sensor's data shows mostly a stable behavior.

Hydraulic Test Rig Instrumentation

Other public available dataset used to evaluate the proposed solution was the sensors' data of a hydraulic test rig. It consists of a primary working and a secondary cooling-filtration circuit, connected via an oil tank (HELWIG; PIGNANELLI; SCHÜTZE, 2015). The system cyclically repeats constant load cycles of 60 seconds, and measures the process values, while the conditions of four hydraulic components (cooler, valve, pump and accumulator) are quantitatively varied. The dataset is composed of six pressure sensors, four temperature sensors, two volume flow sensors, one vibration sensor, two cooling indicators (power and efficiency), the motor power indicator, and a efficiency factor indicator. The 17 sensors are described in Table 9, along with the unit of measure and sampling frequencies.

Table 9 – Hydraulic Test Rig Sensors

Sensor	**Physical Quantity**	**Unit**	**Sampling rate**
PS1	Pressure	bar	100 Hz
PS2	Pressure	bar	100 Hz
PS3	Pressure	bar	100 Hz
PS4	Pressure	bar	100 Hz
PS5	Pressure	bar	100 Hz
PS6	Pressure	bar	100 Hz
MPW	Motor power	W	100 Hz
FS1	Volume flow	l/min	10 Hz
FS2	Volume flow	l/min	10 Hz
TS1	Temperature	°C	1 Hz
TS2	Temperature	°C	1 Hz
TS3	Temperature	°C	1 Hz
TS4	Temperature	°C	1 Hz
VS1	Vibration	mm/s	1 Hz
CE	Cooling efficiency (virtual)	%	1 Hz
CP	Cooling power (virtual)	kW	1 Hz
SE	Efficiency factor	%	1 Hz

The dataset consists of 13,230,000 data samples, obtained under different conditions of pressure, leaking and motor power. In order to select features and train the predictors, only data obtained under good conditions: with no severe leakages, no significant pressure reduction, and with valves with optimal behavior or with a small lag. The training dataset formed this way has 48,000 data points.

As the dataset was composed of a considerable set of sensors, to create models with a smaller number of inputs and produce closed subsets of sensors to apply the proposed algorithm, the Pearson Correlation Coefficient between the sensors was calculated. This process grouped the devices with a correlation factor greater than 0.6 or lower than -0.6, which indicates a medium to a strong correlation between them. In doing so, two groups are identified, which are: {PS1, PS2, PS3, MPW, FS1} and {PS4, PS5, PS6, FS2, TS1, TS2, TS3, TS4, VS1, CE, CP, SE}. These groups are clearly identified in the Pearson Correlation Coefficient heat map shown in Figure 27. The more blue or yellow a square is, the stronger is the correlation between the sensors' values. The yellow color denotes a direct correlation, while the blue color denotes an inverse correlation.

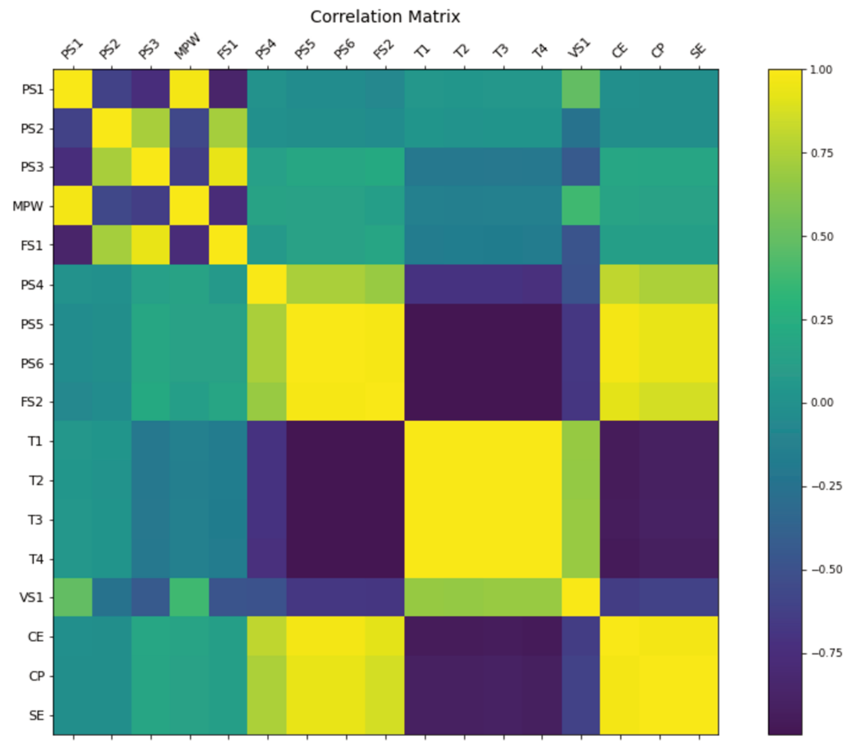


Figure 27 – Pearson Correlation Coefficients - Hydraulic Test Rig

The MAEs calculated for each variable are shown in Table 10, and were used in the evaluation of the sensor failures. The evaluation is done by doing error injection at random points. The experiments took data not used in the training process, and several portions of errors of each type were injected randomly in different points of the data. At each round, two sensors of each group were randomly selected to be "defective". Every fault was injected in one of the sensors, or in both, with the same probability (33% for each sensor and 33% for both together). As the stuck-at error with the sensor "freezed" at the last read value is hard to detect, in this experiment the stuck-at errors were simulated with the sensor reporting 0 (zero).

Table 10 – Hydraulic Test Rig Sensors MAEs

Sensor	MAE
PS1	0.20585
PS2	0.26896
PS3	0.03950
MPW	3.68210
FS1	0.03328

Sensor	MAE
PS4	0.00045
PS5	0.01813
PS6	0.01256
FS2	0.00622
TS1	0.03749
TS2	0.02540
TS3	0.02303
TS4	0.02289
VS1	0.00165
CE	0.02794
CP	0.00110
SE	0.00095

In this experiment, the variations of the three parameters were made in a slightly different manner. The γ parameter was evaluated with the values 100% (not evaluated in the previous experiments), 90%, 80%, 60% and 40%. The γ value was included to evaluate the behavior of the proposed solution if no deviation from the tolerance range defined by β was allowed. Other values were omitted because the curve's tendency is the same as the obtained in the previous experiment.

The results are shown in Figure 28 and in Tables 19 and 20 in Appendix A. As expected, the highest EDR were obtained with this value of γ, but also the number of false positives was very high. With values of β and α close to 1.0, very little variation around the expected value was allowed, dropping the confidence level for any value that deviates very little from the prediction. But the number of false positives is also very high. The injected errors produced the same amount of low confidence on other sensors. With β value of 1.0, only when γ lowers to 80% and α is greater than 2.5 the number of false positives is lower than the number of injected errors. For values of β greater than 1.0, the Interference Error Rate (IER) decreases fast. Once again, the values of β and α around 2.0 and 2.5 show a good relation between the EDR and the IER.

The evaluation, as mentioned before, was made with the *stuck-at* errors reporting the sensor's value as 0. This makes the EDR raise, as they do not show the behavior reported in Section 5.4. The tests made with the *stuck-at* errors repeating the last valid value resulted in a nearly zero detection of these errors. And a *stuck-at* error with an arbitrary value is equivalent to a *peak* error. Therefore, this type of error was removed from Table 11, showing only the EDRs for the *peak* (P) and *noise* (N). As expected, the greater the values of β, the lower is the value of EDR. The peak errors consistently show lower detection rates, while the noise error got a good detection rate, with a lower decrease when the values of α and β increase. This can

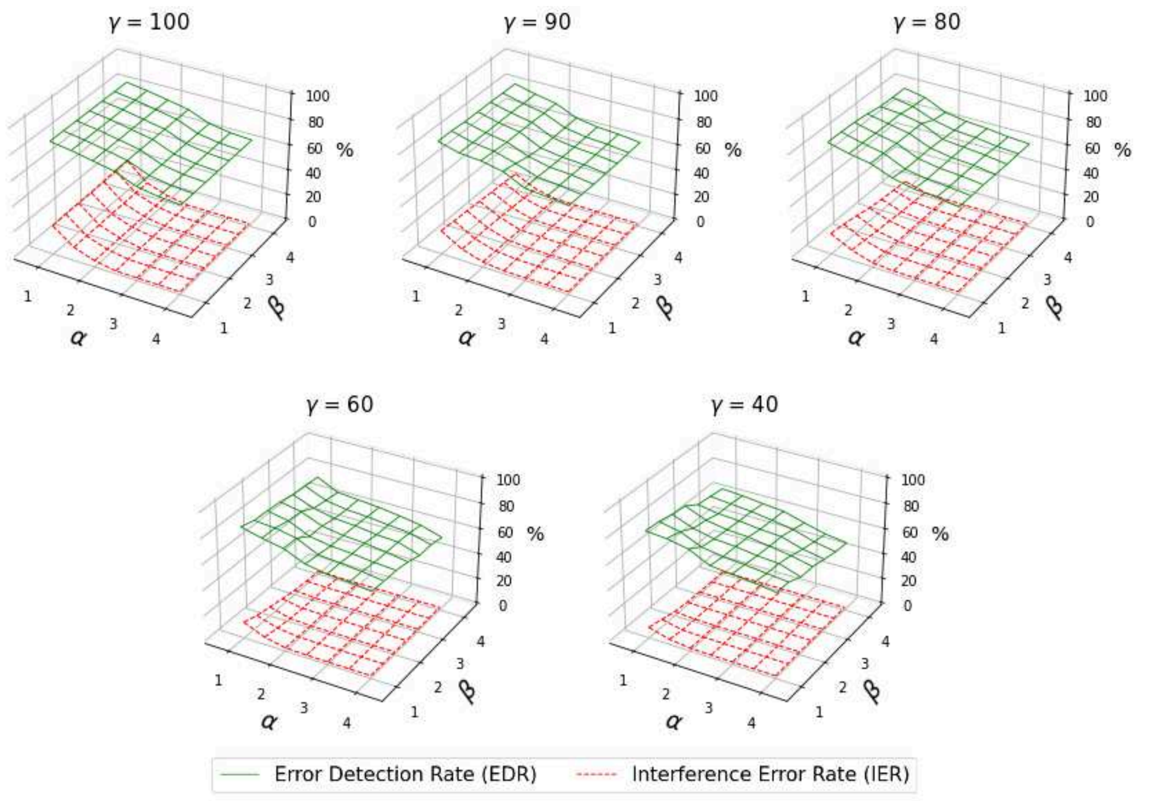


Figure 28 – EDR and IER for different γ values - Hydraulic Test Rig

be explained by the fact that the peak errors can displace the read data in the direction of the predicted values, making them not be detected. On the other side, noise errors, as they vary in both directions, will increase the error at least when the variation goes in one direction.

In this experiment the False Positive Rate (FPR) of the confidence attribution algorithm was also evaluated, being considered a *False Positive* any sensor that reported a confidence level under the limit defined by γ, when no error was injected at that point. As in the previous experiments, all combinations of α and β values, between 1 and 4, where used, and also varying γ in the set $\{100, 90, 80, 60, 40\}$, and the results are shown in Table 12. When the γ parameter assumes the value 100, all the FPR values are the same for all values of α, as *any* variation of the confidence level will be detected as a positive value. The FPR decreases fast when the β value increases, meaning that a larger variation around the predicted value is accepted.

However, values of β greater than 2.5 show a lower decrease of the confidence level when β increases. This means that fewer readings are more than $2.5 \times MAE$ away from the predicted value. As previously stated, the data reinforces that the best start values for α and β are around 2.0 and 2.5, and for γ is around 60 and 50. The *trade-off* between EDR and IER/FPR is a matter of decision by the application, as well as it is dependent on the prediction model's accuracy.

Table 11 – Peak (P) and Noise (N) EDR - Hydraulic Test Rig

		β													
		1.0		**1.5**		**2.0**		**2.5**		**3.0**		**3.5**		**4.0**	
γ	α	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**
	1.0	72.5	93.0	63.8	94.0	55.0	93.0	51.3	88.0	33.8	85.0	30.0	81.0	28.8	79.0
	1.5	72.5	93.0	63.8	94.0	55.0	93.0	51.3	88.0	33.8	85.0	30.0	81.0	28.8	79.0
	2.0	72.5	93.0	63.8	94.0	55.0	93.0	51.3	88.0	33.8	85.0	30.0	81.0	28.8	79.0
100	**2.5**	72.5	93.0	63.8	94.0	55.0	93.0	51.3	88.0	33.8	85.0	30.0	81.0	28.8	79.0
	3.0	72.5	93.0	63.8	94.0	55.0	93.0	51.3	88.0	33.8	85.0	30.0	81.0	28.8	79.0
	3.5	72.5	93.0	63.8	94.0	55.0	93.0	51.3	88.0	33.8	85.0	30.0	81.0	28.8	79.0
	4.0	72.5	93.0	63.8	94.0	53.8	90.0	35.0	88.0	33.8	83.0	30.0	80.0	46.3	85.0
	1.0	71.3	94.0	57.5	94.0	53.8	93.0	46.3	88.0	31.3	84.0	30.0	81.0	28.8	78.0
	1.5	70.0	95.0	58.8	93.0	52.5	90.0	40.0	86.0	31.3	82.0	30.0	80.0	28.8	77.0
	2.0	67.5	95.0	57.5	93.0	52.5	90.0	32.5	86.0	31.3	82.0	30.0	80.0	27.5	76.0
90	**2.5**	67.5	95.0	57.5	93.0	52.5	90.0	32.5	86.0	30.0	82.0	28.8	80.0	27.5	76.0
	3.0	67.5	95.0	57.5	93.0	52.5	89.0	32.5	86.0	30.0	81.0	28.8	79.0	26.3	76.0
	3.5	66.3	95.0	56.3	93.0	52.5	88.0	32.5	86.0	30.0	81.0	28.8	79.0	26.3	76.0
	4.0	66.3	95.0	56.3	93.0	52.5	88.0	32.5	86.0	30.0	81.0	28.8	79.0	43.8	82.0
	1.0	67.5	95.0	57.5	93.0	52.5	90.0	32.5	86.0	31.3	82.0	30.0	80.0	27.5	76.0
	1.5	67.5	95.0	57.5	93.0	52.5	89.0	32.5	86.0	30.0	81.0	28.8	79.0	26.3	76.0
	2.0	66.3	95.0	56.3	93.0	52.5	88.0	32.5	86.0	30.0	81.0	28.8	79.0	26.3	75.0
80	**2.5**	63.8	94.0	55.0	93.0	51.3	88.0	33.8	85.0	30.0	81.0	28.8	79.0	25.0	75.0
	3.0	57.5	94.0	53.8	90.0	40.0	88.0	31.3	82.0	30.0	80.0	28.8	78.0	25.0	75.0
	3.5	57.5	93.0	52.5	90.0	32.5	86.0	31.3	82.0	30.0	80.0	27.5	76.0	25.0	75.0
	4.0	57.5	93.0	52.5	89.0	37.5	86.0	30.0	81.0	28.8	79.0	26.3	76.0	43.8	81.0
	1.0	66.3	95.0	56.3	93.0	52.5	88.0	32.5	86.0	30.0	81.0	28.8	79.0	25.0	75.0
	1.5	57.5	94.0	53.8	90.0	35.0	88.0	31.3	82.0	30.0	80.0	28.8	78.0	25.0	75.0
	2.0	57.5	93.0	52.5	89.0	32.5	86.0	30.0	81.0	28.8	79.0	26.3	75.0	23.8	72.0
60	**2.5**	53.8	90.0	40.0	88.0	33.8	83.0	30.0	80.0	28.8	79.0	25.0	75.0	21.3	66.0
	3.0	52.5	90.0	32.5	86.0	31.3	82.0	30.0	80.0	27.5	76.0	25.0	75.0	21.3	64.0
	3.5	52.5	88.0	32.5	86.0	30.0	81.0	28.8	79.0	25.0	75.0	22.5	71.0	17.5	64.0
	4.0	52.5	91.0	47.5	84.0	30.0	83.0	28.8	79.0	26.3	76.0	25.0	74.0	41.3	76.0
	1.0	57.5	94.0	53.8	90.0	35.0	88.0	31.3	82.0	30.0	80.0	28.8	78.0	25.0	75.0
	1.5	55.0	90.0	41.3	88.0	32.5	84.0	30.0	80.0	28.8	79.0	25.0	75.0	22.5	66.0
	2.0	52.5	90.0	32.5	86.0	31.3	82.0	30.0	80.0	27.5	75.0	23.8	73.0	21.3	64.0
40	**2.5**	35.0	88.0	33.8	83.0	30.0	80.0	28.8	79.0	25.0	75.0	21.3	66.0	15.0	63.0
	3.0	32.5	86.0	30.0	81.0	28.8	79.0	25.0	75.0	22.5	72.0	17.5	64.0	8.8	62.0
	3.5	31.3	82.0	30.0	80.0	28.8	78.0	25.0	75.0	21.3	66.0	12.5	62.0	6.3	60.0
	4.0	47.5	86.0	45.0	82.0	26.3	79.0	26.3	75.0	25.0	72.0	21.3	68.0	2.5	29.0

3D Printer Monitoring

The proposed algorithm for faulty sensor detections was also applied in the context of the CAPES-PRINT INCANTO project, in order to monitor the sensors that report different variables on a 3D Printer, while some object is being built. To achieve this objective, a 3D Printer was built at the University of Parma - Italy. It was then equipped with accelerometers monitoring the vibrations on different points, that are able to capture the movement patterns while the printer process goes on. Three EPOSMote III (LAB, 2017) devices, equipped with ST'LSM330 3-axis accelerometer, were employed to do this task. A set of encoders was also

Table 12 – False Positive Rate (FPR) - Hydraulic Test Rig

		β						
γ	α	**1.0**	**1.5**	**2.0**	**2.5**	**3.0**	**3.5**	**4.0**
100	**1.0**	37,53	20,58	11,02	6,93	5,48	4,01	2,81
	1.5	37.53	20.58	11.02	6.93	5.48	4.01	2.81
	2.0	37.53	20.58	11.02	6.93	5.48	4.01	2.81
	2.5	37.53	20.58	11.02	6.93	5.48	4.01	2.81
	3.0	37.53	20.58	11.02	6.93	5.48	4.01	2.81
	3.5	37.53	20.58	11.02	6.93	5.48	4.01	2.81
	4.0	37.53	20.58	11.02	6.93	5.48	4.01	2.81
90	**1.0**	31.52	18.28	9.68	6.55	5.20	3.70	2.68
	1.5	29.95	17.08	9.10	6.34	5.05	3.54	2.67
	2.0	28.49	15.81	8.73	6.24	4.93	3.35	2.66
	2.5	27.13	14.93	8.36	6.12	4.75	3.27	2.58
	3.0	25.87	14.13	7.98	6.00	4.59	3.12	2.53
	3.5	24.49	13.33	7.62	5.89	4.46	3.04	2.51
	4.0	23.20	12.68	7.37	5.72	4.29	2.98	2.48
80	**1.0**	28.49	15.81	8.73	6.24	4.93	3.35	2.66
	1.5	25.87	14.13	7.98	6.00	4.59	3.12	2.53
	2.0	23.20	12.68	7.37	5.72	4.29	2.98	2.48
	2.5	20.58	11.02	6.93	5.48	4.01	2.81	2.44
	3.0	18.28	9.68	6.55	5.20	3.70	2.68	2.36
	3.5	15.81	8.73	6.24	4.93	3.35	2.66	2.28
	4.0	14.13	7.98	6.00	4.59	3.12	2.53	2.27
60	**1.0**	23.20	12.68	7.37	5.72	4.29	2.98	2.48
	1.5	18.28	9.68	6.55	5.20	3.70	2.68	2.36
	2.0	14.13	7.98	6.00	4.59	3.12	2.53	2.27
	2.5	11.02	6.93	5.48	4.01	2.81	2.44	2.25
	3.0	8.73	6.24	4.93	3.35	2.66	2.28	2.21
	3.5	7.37	5.72	4.29	2.98	2.48	2.26	2.14
	4.0	6.55	5.20	3.70	2.68	2.36	2.25	2.06
40	**1.0**	18.28	9.68	6.55	5.20	3.70	2.68	2.36
	1.5	12.68	7.37	5.72	4.29	2.98	2.48	2.26
	2.0	8.73	6.24	4.93	3.35	2.66	2.28	2.21
	2.5	6.93	5.48	4.01	2.81	2.44	2.25	2.10
	3.0	6.00	4.59	3.12	2.53	2.27	2.17	1.97
	3.5	5.20	3.70	2.68	2.36	2.25	2.06	1.89
	4.0	4.29	2.98	2.48	2.26	2.14	1.95	1.80

integrated in the printer, allowing to determine the exact place where the moving components are at each moment. The temperatures at the injection nozzle and at the printing table are also monitored. Therefore, there are 15 data points available, with different sensing frequencies, as shown in Table 13. The EPOSMote used to capture the printing's vibrations are installed at specific points: at the printer head, to capture its moves, at the printer frame, and under the table to capture the structure vibrations while printing. A simplified schema representing the printer structure and the point at which the sensors are installed are shown in Figure 30. Pictures of the sensors installed on the printer head and the printer frame are also shown in Figure 31.

Once instrumented, the data collected by the sensors were transmitted to the Hardware/Software Integration Laboratory (LISHA) IoT Platform, where they are stored and avail-

Table 13 – Sensors on 3D Printer

Sensor	Sensed Value	Unit	Frequency
Printer Head - X axis	Gravity Accel.	g	200Hz
Printer Head - Y axis	Gravity Accel.	g	200Hz
Printer Head - Z axis	Gravity Accel.	g	200Hz
Printer Frame - X axis	Gravity Accel.	g	200Hz
Printer Frame - Y axis	Gravity Accel.	g	200Hz
Printer Frame - Z axis	Gravity Accel.	g	200Hz
Printer Bed - X axis	Gravity Accel.	g	200Hz
Printer Bed - Y axis	Gravity Accel.	g	200Hz
Printer Bed - Z axis	Gravity Accel.	g	200Hz
Position Encoder - X axis	Encoder value	Position	30Hz
Position Encoder - Y axis	Encoder value	Position	30Hz
Position Encoder - Z axis	Encoder value	Position	30Hz
Extruder Encoder	Encoder value	Position	30Hz
Printer Nozzle	Temperature	°C	10Hz
Printer Bed (Table)	Temperature	°C	10Hz

Table 14 – Models' features and MAEs - 3D Printer

Device	Name	Predictor's Inputs	MAE
0	Bed X axis	0, 3, 4, 5, 7, 8, 10	0.00270
1	Bed Y axis	1, 3, 4, 5, 8, 10, 11	0.00767
2	Bed Z axis	2, 5, 6, 7, 8, 10, 11	0.00401
3	Head X axis	1, 3, 6, 7, 8, 10, 11	0.00718
4	Head Y axis	0, 1, 4, 6, 8, 10, 11	0.03290
5	Head Z axis	1, 2, 5, 7, 8, 10, 11	0.01945
6	Frame X axis	1, 3, 4, 5, 6, 10, 11	0.00639
7	Frame Y axis	2, 3, 4, 5, 7, 10, 11	0.00741
8	Frame Z axis	1, 3, 4, 5, 8, 10, 11	0.00533
9	Encoder – X axis	1, 2, 4, 5, 7, 9, 10	13.21807
10	Encoder – Y axis	2, 4, 5, 7, 9, 10, 11	8.25537
11	Encoder – Z axis	1, 2, 4, 5, 7, 10, 11	4.13072

able to be accessed at any time through the LISHA's API for SmartData. The platform also offers a graphical interface for visualizing the stored data, as shown in figure 29.

When analyzing the collected data, it was found that there is no strong correlation between the values of the different sensors, as shown in figure 32. The feature selection was made through the built-in `SelectKBest` function, using the `mutual_info_regression` method. As the extruder only measures the material consumption, and the temperatures are completely independent of the printer movement, these sensors were excluded from the evaluation. Therefore, in the experiment, only the values from the accelerometers and the axis positions were considered. The number of features selected for each sensor was seven, as more sensors did not

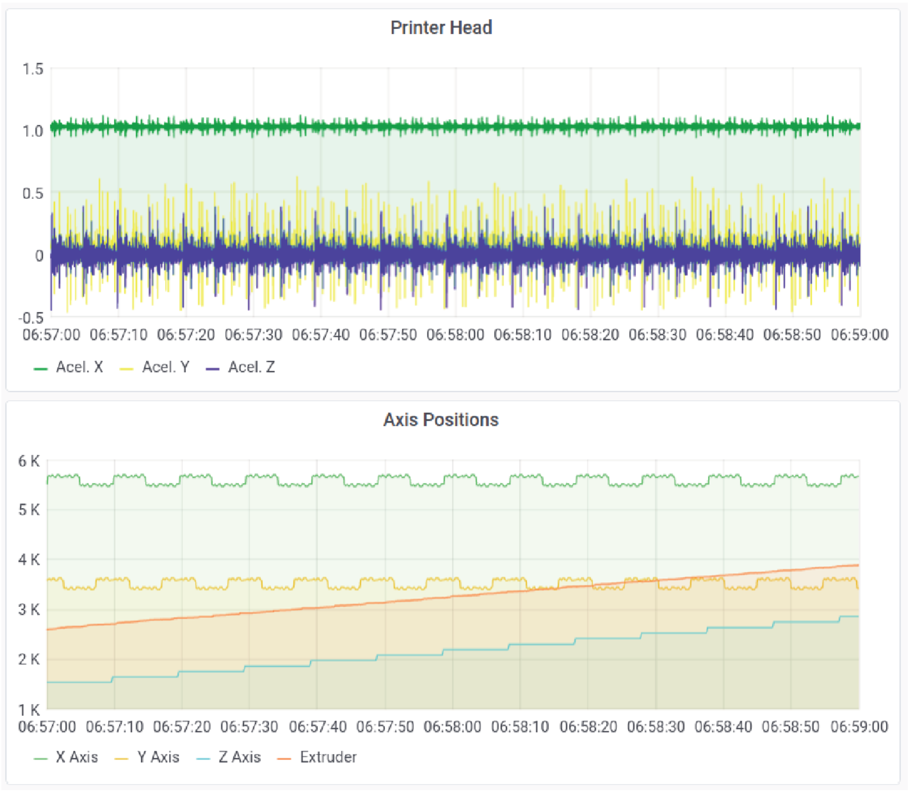


Figure 29 – LISHA's IoT Platform Dashboard - Printing monitoring

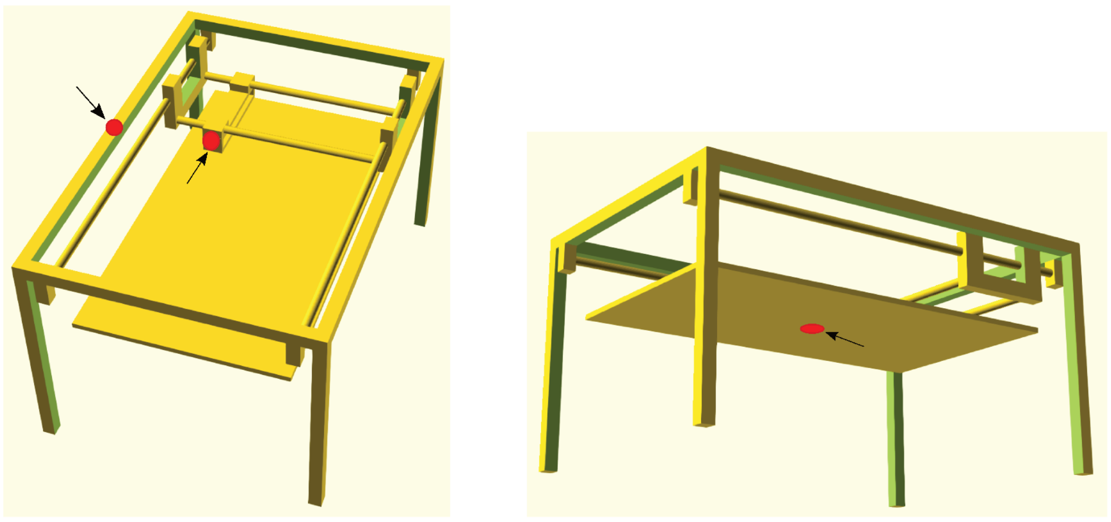


Figure 30 – INCANTO Project Printer Schema - Accelerometers' installation.

decrease the MAE values (accuracy) of the models. The sensors used as input for each model and the respective MAE are shown in Table 14. The maximum value considered for the encoders (extracted from the collected data) was 1.5 for the accelerometers, and 13000 for the encoders. This gives a parameter to evaluate the MAE values.

After training the models on a subset of a print job's data, as in the previous experiments, it could be noted that the models did not produce good results when tested on unseen data. This can be explained by the fact that the movement patterns are specific for each part of the printing, and therefore can not be well predicted. When training the models on a whole

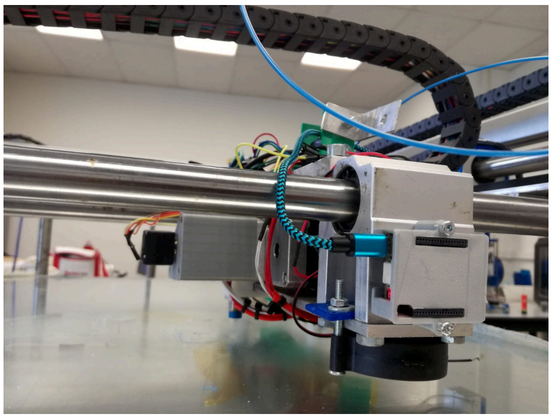
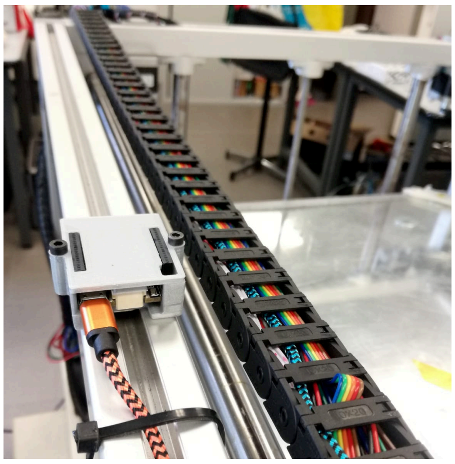



Figure 31 – INCANTO Project Printer - EPOS-Mote installed on printing head and Frame.

print job, and then running against the data of other printing of the same object, the models performed better, as they learned the movement patterns of that printing.

The second characteristic observed in this experiment was the fact that some models, even presenting very good MAE values at the training set, do not perform well when making their predictions. This happened with the encoders' positions. They were selected as input on every accelerometer, and it can be explained by the fact that their movement will cause the accelerometers' variations. But the reverse predictions do not work well, and the confidence level assigned was always low. This caused the substitution by the predicted value, which makes the other sensors lose their confidence. In the end, the error propagation caused a general failure of the proposed solution.

Removing the encoders from the model was not possible because, as said before, their position sequences determine the accelerometers' behaviors. So, they are kept as models' inputs, but their data was always considered correct. Doing so, the models recovered their normal function, and the model stabilized in the expected behavior. In the context of the INCANTO Project it was viable, as the printer's head position is constantly monitored, via encoders' values, by another software. By comparing the encoders' readings with the expected trajectory obtained from the commands sent to the printer, any deviation can be signalized by that software.

After these adaptations, the proposed model was then applied to the sensors' data, and the results are shown in Table 21 and 22 in Appendix A.

The results are also shown in Figure 33. The EDR presented higher values with higher values of γ and stricter values of β and γ. As in the other experiments, the *stuck-at* errors showed a low detection rate, once the accelerometers' values always oscillate around a subset of values. With the most restrictive values ($\gamma = 100$, $\beta = 1.0$, $\gamma = 1.0$) the EDR for *stuck-at* errors reached at most 54%, and for more relaxed conditions the EDR dropped fast to zero. As it can be seen on Table 15, the EDR for the most restrictive configuration, without the *stuck-at* error would be around 93%, while the general EDR with the same configuration dropped to around 83% when including the *stuck-at* error in the statistics.

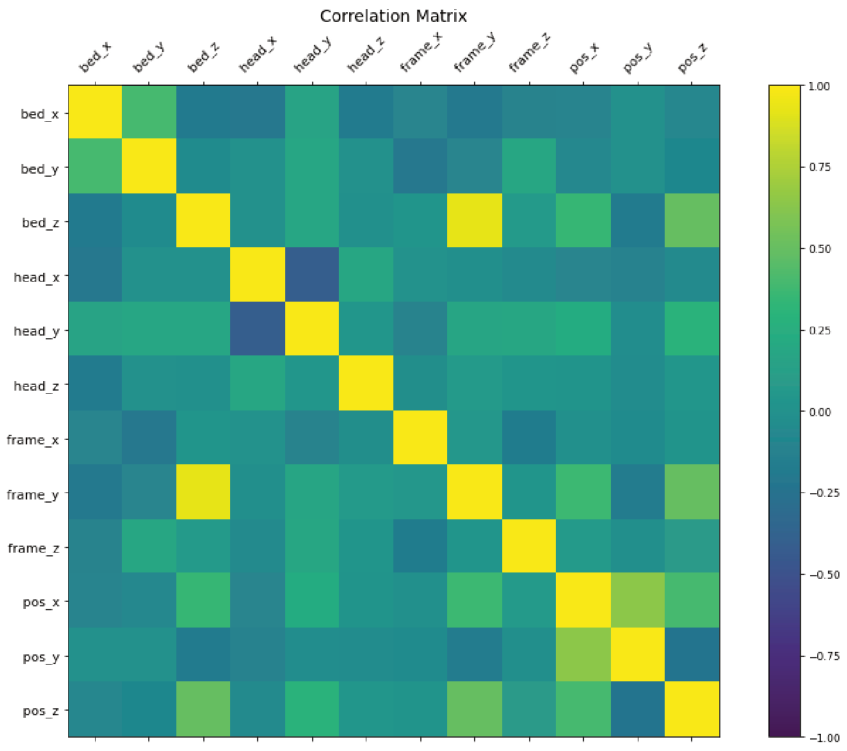


Figure 32 – Pearson's Correlation between sensors on 3D Printer

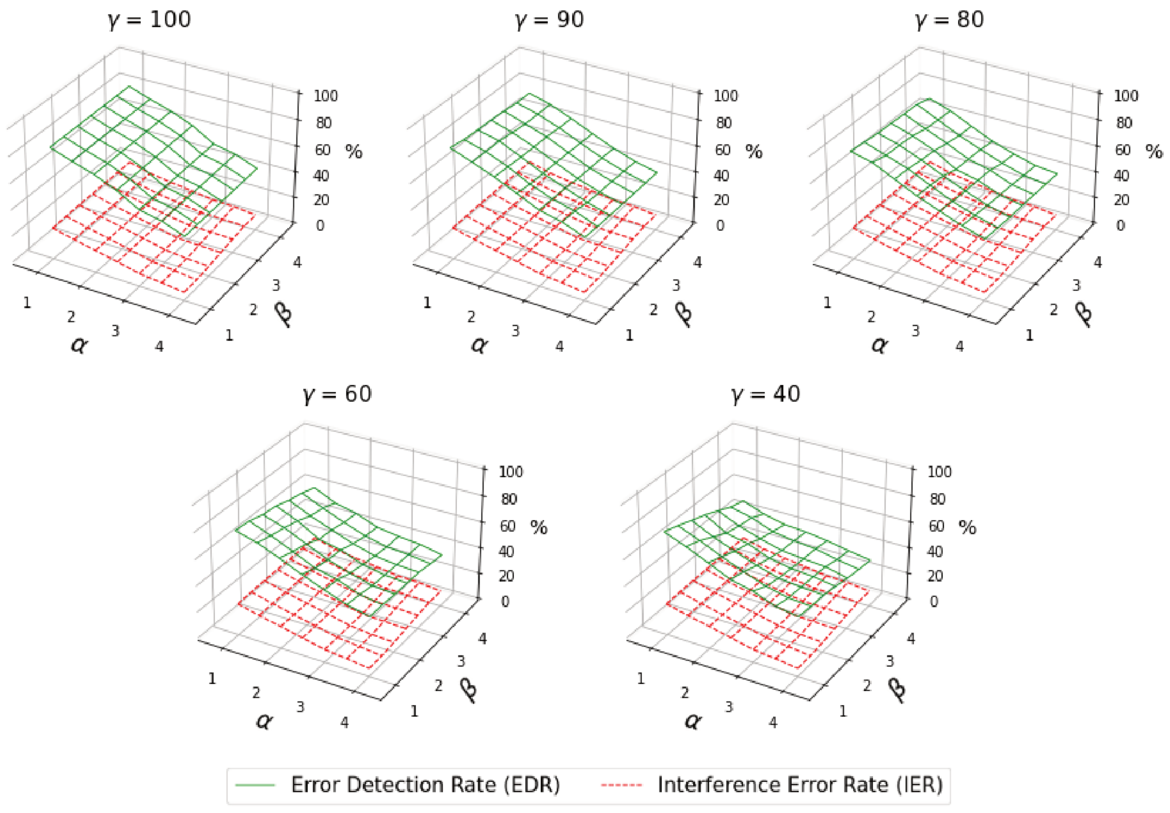


Figure 33 – EDR and IER for different γ values - 3D Printer

This experiment also shows some new patterns in the results, concerning the algorithm parameter's values. The γ and β parameters maintained their influence in the EDR and IER

Table 15 – Peak and Noise EDR - 3D Printer

		β													
		1.0		**1.5**		**2.0**		**2.5**		**3.0**		**3.5**		**4.0**	
γ	α	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**
	1.0	95.6	100	90.0	97.5	87.8	95.8	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5
	1.5	95.6	100	90.0	97.5	87.8	95.8	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5
	2.0	95.6	100	90.0	97.5	87.8	95.8	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5
100	**2.5**	95.6	100	90.0	97.5	87.8	95.8	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5
	3.0	95.6	100	90.0	97.5	87.8	95.8	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5
	3.5	95.6	100	90.0	97.5	87.8	95.8	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5
	4.0	95.6	100	90.0	97.5	87.8	95.8	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5
	1.0	95.6	99.2	87.8	97.5	87.8	95.0	78.9	90.0	73.3	85.8	73.3	76.7	68.9	70.8
	1.5	94.4	98.3	87.8	96.7	87.8	95.0	77.8	90.0	73.3	85.0	73.3	75.8	68.9	68.3
	2.0	93.3	98.3	87.8	96.7	85.6	95.0	77.8	89.2	73.3	84.2	71.1	75.8	68.9	67.5
90	**2.5**	93.3	98.3	87.8	96.7	85.6	94.2	76.7	89.2	73.3	80.8	71.1	75.0	68.9	66.7
	3.0	92.2	97.5	87.8	96.7	85.6	93.3	75.6	89.2	73.3	80.0	71.1	74.2	68.9	65.8
	3.5	91.1	97.5	87.8	96.7	85.6	90.8	74.4	88.3	73.3	79.2	71.1	73.3	67.8	65.8
	4.0	91.1	97.5	87.8	96.7	83.3	90.8	73.3	88.3	73.3	77.5	70.0	73.3	66.7	65.8
	1.0	93.3	98.3	87.8	96.7	85.6	95.0	77.8	89.2	73.3	84.2	71.1	75.8	68.9	67.5
	1.5	92.2	97.5	87.8	96.7	85.6	93.3	75.6	89.2	73.3	80.0	71.1	74.2	68.9	65.8
	2.0	91.1	97.5	87.8	96.7	83.3	90.8	73.3	88.3	73.3	78.3	70.0	73.3	66.7	65.8
80	**2.5**	90.0	97.5	87.8	95.8	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5	66.7	65.8
	3.0	87.8	97.5	85.6	95.0	78.9	90.0	73.3	85.8	73.3	76.7	68.9	70.8	66.7	65.0
	3.5	87.8	96.7	85.6	95.0	77.8	89.2	73.3	84.2	71.1	75.8	68.9	67.5	66.7	64.2
	4.0	87.8	96.7	85.6	93.3	75.6	89.2	73.3	80.0	71.1	74.2	68.9	65.8	66.7	63.3
	1.0	91.1	97.5	87.8	96.7	83.3	90.8	73.3	88.3	73.3	78.3	70.0	73.3	66.7	65.8
	1.5	87.8	97.5	87.8	95.0	78.9	90.0	73.3	85.8	73.3	76.7	68.9	70.8	66.7	65.8
	2.0	87.8	96.7	85.6	93.3	75.6	89.2	73.3	80.0	71.1	74.2	68.9	65.8	66.7	63.3
60	**2.5**	87.8	95.8	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5	66.7	65.8	66.7	62.5
	3.0	85.6	95.0	77.8	89.2	73.3	84.2	71.1	75.8	68.9	67.5	66.7	64.2	66.7	61.7
	3.5	83.3	90.8	73.3	88.3	73.3	78.3	70.0	73.3	66.7	65.8	66.7	62.5	66.7	56.7
	4.0	78.9	90.0	73.3	85.8	73.3	76.7	68.9	70.8	66.7	65.8	66.7	61.7	64.4	54.2
	1.0	87.8	97.5	87.8	95.0	78.9	90.0	73.3	85.8	73.3	76.7	68.9	70.8	66.7	65.8
	1.5	87.8	96.7	83.3	90.8	73.3	88.3	73.3	78.3	70.0	73.3	66.7	65.8	66.7	62.5
	2.0	86.7	95.0	77.8	89.2	73.3	84.2	71.1	75.8	68.9	67.5	66.7	64.2	66.7	61.7
40	**2.5**	81.1	90.8	73.3	86.7	73.3	77.5	68.9	72.5	66.7	65.8	66.7	62.5	66.7	55.8
	3.0	75.6	89.2	73.3	80.0	71.1	74.2	68.9	65.8	66.7	63.3	66.7	60.0	64.4	51.7
	3.5	73.3	85.8	73.3	76.7	68.9	70.8	66.7	65.8	66.7	61.7	64.4	54.2	64.4	46.7
	4.0	73.3	78.3	70.0	73.3	66.7	65.8	66.7	62.5	66.7	56.7	64.4	50.0	64.4	43.3

results. But the α parameter showed an opposite effect over the IER as in the previous experiments. The value of the IER increased for larger values of α. This can be explained as an effect over the other sensors' models that make their confidence drop faster as the confidence of the sensor where the error is injected. So, a larger value of α makes the algorithm classify the output where the error is injected as good. But, in another model where the input causes an important impact on the confidence level, the confidence can drop faster and cause a false positive. It is also noticeable that the increment of the IER is not very huge, being always around 5% to 10%, while dropping around 40% in several cases in the Photovoltaic Solar Energy case study.

Table 16 – False Positive Rate (FPR) - 3D Printer

		β						
γ	α	**1.0**	**1.5**	**2.0**	**2.5**	**3.0**	**3.5**	**4.0**
100	**1.0**	31.98	22.77	18.81	8.81	6.48	5.38	4.77
	1.5	31.98	22.77	18.81	8.81	6.48	5.38	4.77
	2.0	31.98	22.77	18.81	8.81	6.48	5.38	4.77
	2.5	31.98	22.77	18.81	8.81	6.48	5.38	4.77
	3.0	31.98	22.77	18.81	8.81	6.48	5.38	4.77
	3.5	31.98	22.77	18.81	8.81	6.48	5.38	4.77
	4.0	31.98	22.77	18.81	8.81	6.48	5.38	4.77
90	**1.0**	28.67	21.75	17.19	7.88	6.19	5.17	4.65
	1.5	27.50	21.25	16.56	7.65	6.17	5.13	4.54
	2.0	26.67	20.79	15.98	7.46	6.04	5.06	4.50
	2.5	25.92	20.54	14.96	7.23	5.98	4.98	4.46
	3.0	25.29	20.21	14.02	7.15	5.73	4.96	4.44
	3.5	24.48	19.79	12.65	6.98	5.65	4.88	4.40
	4.0	23.73	19.54	11.56	6.81	5.56	4.88	4.31
80	**1.0**	26.67	20.79	15.98	7.46	6.04	5.06	4.50
	1.5	25.29	20.21	14.02	7.15	5.73	4.96	4.44
	2.0	23.73	19.54	11.56	6.81	5.56	4.88	4.31
	2.5	22.77	18.81	8.81	6.48	5.38	4.77	4.23
	3.0	21.75	17.19	7.88	6.19	5.17	4.65	4.19
	3.5	20.79	15.98	7.46	6.04	5.06	4.50	4.15
	4.0	20.21	14.02	7.15	5.73	4.96	4.44	4.10
60	**1.0**	23.73	19.54	11.56	6.81	5.56	4.88	4.31
	1.5	21.75	17.19	7.88	6.19	5.17	4.65	4.19
	2.0	20.21	14.02	7.15	5.73	4.96	4.44	4.10
	2.5	18.81	8.81	6.48	5.38	4.77	4.23	4.00
	3.0	15.98	7.46	6.04	5.06	4.50	4.15	3.98
	3.5	11.56	6.81	5.56	4.88	4.31	4.04	3.81
	4.0	7.88	6.19	5.17	4.65	4.19	3.98	3.65
40	**1.0**	21.75	17.19	7.88	6.19	5.17	4.65	4.19
	1.5	19.54	11.56	6.81	5.56	4.88	4.31	4.04
	2.0	15.98	7.46	6.04	5.06	4.50	4.15	3.98
	2.5	8.81	6.48	5.38	4.77	4.23	4.00	3.73
	3.0	7.15	5.73	4.96	4.44	4.10	3.92	3.50
	3.5	6.19	5.17	4.65	4.19	3.98	3.65	3.17
	4.0	5.56	4.88	4.31	4.04	3.81	3.48	2.88

The FPR was also evaluated for this experiment, with the results shown in Table 16. The FPR is a bit lower when the algorithm is applied with the more restrictive values of γ (100 and 90), when compared to the values obtained in the Hydraulic Test Rig experiment. However, the values do not decrease as much as the previous experiment when the values of γ decrease to 60 or 40, for example. This can be explained by the nature of the data. Vibration data show high variability, and therefore the predictor then to be less exact. This implies more false positives, as wrong predictions can cause the confidence of the data points to get lowered. As also stated in other analysis, the use of more restrictive values for the parameters will result in higher false positives, as less variation in data is accepted by the algorithm before the confidence is lowered when β is small, and the confidence drops faster when the value of α is small.

Another aspect to be taken into account is that a data point is classified as a False Positive if it shows a confidence below γ, but not necessarily this means that the data is faulty. The final decision can be taken by the application, analyzing how far from γ the actual confidence is, how fast it dropped from the last reading, and how other sensors' data is behaving.

5.5 DISCUSSION

Most of the solutions proposed in the related works (Section 2.1) rely on extra messages between nodes to do voting rounds, or exchange diagnostic requests and responses, to confirm if the data of a suspicious node is correct or not. Other solutions use centralized approaches, making the fault detection at the CH, at the WSN sink or even at another centralized node running extensive data analysis algorithms.

The solution in this work tries to overcome the need for extra special messages, performing a distributed fault detection. To do this, each node labeling its data with a *confidence level* allows other nodes and the application, without extra message exchange, to know "how much" the sender node relies on the sensor's reading.

The first assumption is that there are several sensors collecting data that show some correlation. It means that a prediction model can be built for every sensor, taking as input the readings of a set of other sensors. This is coherent with the other solutions proposed in the related works. Some of them require that there are other sensors of the same type, like Moustapha & Selmic (2008), Yuan, Zhao & Yu (2015), Saihi et al. (2015), Jia, Ma & Qin (2019), and Karmarkar, Chanak & Kumar (2020). Others also use data from heterogeneous sensors, like Zhang, Zhao & Nakamoto (2017), Shao, Guo & Qiu (2017), Swain & Khilar (2017) and Curiac & Volosencu (2012), but on hierarchical or centralized architectures. This work assumes that sensors can listen to the data from close sensors when the protocol can "broadcast" messages inside small "interest regions" and the WSN design provides it. The authenticity and confidentiality can be ensured by the use of group-based cipher mechanisms. Therefore, it can be assumed that the required data to run the predictors will be available when needed.

The second assumption is that the model will produce predictions within a bounded error. As a unique value would be hard to be defined, the proposed solution makes use of the MAE obtained in the training process to define it, in conjunction with a multiplier factor called β. As shown in the experiments, different models will produce different MAEs. So, instead of defining several thresholds, the β parameter will define a tolerance interval for every sensor, that is proportional to the data behavior in its domain, as well as to the model's accuracy. The experiments' results also presented a consistent behavior on the error detection and false-positive ratios over the values of β, showing that a reasonable value for it can be found, no matter the specific data domain on which the proposed solution is applied.

The parameter α defines a "transition area" where the confidence level of the sensor's data degrades from 100% to zero. It avoids the binary classification of correct and incorrect values and shows the error behavior while the read and predictor values start to get away from each other. The experiments' results show that varying α has almost the same effect over the EDR and IER than varying β. This is because they determine the point at which the confidence level goes under the limit defined by the third parameter γ, that defines when, in the experiments, the data is considered faulty.

The use of the two parameters, instead of just one, can be justified by the flexibility they provide to the algorithm's output. If an application wants to do a binary classification with a given threshold around the expected value, it can define a reasonable specific value to β and a very little (or even zero) value to α. Otherwise, if an application wants to observe the "degradation" of the sensors' data under certain circumstances, it can set β to a value close (or equal) to zero, and set α to a larger value. This will show the difference between the predicted and read values as a variation of the confidence level assigned to it.

The error type detection was also an aspect evaluated at the experiments. The results showed that the behavior of the data at each type of error is not a feature that determines its detection. The only aspect the algorithm takes into account is the error's amplitude: how far is the sensor reading from the value that the model predicts. Therefore, if an error does not make the sensor's value exceed the threshold defined by the γ, β and α values, it will not be detected by the proposed solution. But with the correct combination of β and α parameters, the data's confidence level will show how the read values is deviating from the value expected by the predictor. The objective of this work is only to identify the occurrence of errors, with no classification.

The error type analysis showed that the *stuck-at* error is hard to detect when the sensor freezes on a valid value. If the data, in its domain, changes smoothly or remains around a specific value for an extended period, the solution proposed in this work hardly can identify it. This kind of behavior was also detected by other authors, like Shao, Guo & Qiu (2017), that suggested the use of a *watchdog* that marks a sensor as suspicious if its state does not change over a predefined time. Statistical analysis can also identify periods with zero variance. Such a type of mechanism can be easily integrated in the proposed solution without impact, as it does not require extra messages and is based only on the analysis of a window with the *N* most recent readings from the sensor.

The influence of one sensor over others can also introduce some undesired consequences, as an increase in the IER, as shown in the 3D Printer case study. This happens when the error on a sensor's value drops the confidence of other sensors – by modifying the model output – faster than its confidence drops. This can be a consequence of the model chosen in this work (ANN), a specificity of the domain, or due to a lack of better tuning the training process. To overcome this, one possible solution could be a restriction like "no model can deviate its prediction in a rate greater than the variation of any input". But it can be too restrictive and would be very hard to verify. At this point, when applying the proposed model, it must be observed that an increment on α can lower or raise the IER. The former is usual in a model with several sensors with strong correlation coefficients.

However, not directly signaling an error when variations do not exceed the minimum confidence level does not restrict the use of the proposed solution to make some advanced analysis. For example, when some variation on a given sensor occurs, the confidence level assigned to it and to the correlated sensors can provide useful information. If the value and confidence levels of one sensor show some variations, and the other sensors show only smaller confidence variations, it can be a signal that the sensor is going through some interference or other kind of problem that affects its readings. Otherwise, if all sensors show variations, the models are expected to reflect these variations and do not show relevant changes in the confidence levels.

This chapter described a confidence assignment mechanism able to stamp every data produced by the sensors of a WSN with a confidence level. This information determines how much the node believes the data is correct, based on the output of a predictor running on the node. In this work, the prediction model employed was Artificial Neural Networks. The use of this specific type of predictor is not obligatory. Any other prediction model could be used if it is able to make predictions based on the data from the previous data period defined by the WSN. The measurement error should also be available after the training phase, to allow the calculation of the MAE, which is used in formula 5.1 to assign the confidence level on each sensor node.

The experiments' results demonstrated that the proposed solution is viable, with good sensitivity to the readings' variations and also a good resilience to the interference problem. Failures of an individual sensor and failures in pairs of sensors were evaluated, with coherent results. Of course, if several sensors fail together, all sensors will inevitably face a confidence level drop, which is inevitable in any solution. But the proposed solution will still present a signal that something wrong is happening with the sensors, yet not being able to identify exactly which sensors are facing problems. When the correlation coefficient between the sensors' data is not very strong, the variation of a sensor can cause interference on other sensors, increasing the number of *false positive* results. This reinforces the requirement of strong correlations between the sensors' to get better results from the proposed solution.

A mechanism for fine-tuning specific types of sensors, or even sensors under specific conditions, is provided by the three parameters β, α, and γ. They can be used to specify different levels of sensitivity for the confidence attribution mechanism, as well as the information level about the sensors' behavior under different conditions. The models' behavior

The confidence level assigned by the solution proposed in this work does more than just classify sensors' readings as *correct* or *incorrect*. It provides valuable additional information to the node, allowing a self-diagnosis mechanism that can be useful when taking local decisions, as well as in analysis carried out at the application level.

6 CONCLUSIONS

Several application fields have been employing sensors for monitoring purposes over decades. Sensor failures, either due to malfunction, interference, or intrusion, is a relevant concern to fault-tolerant systems. Use several sensors to provide redundancy and diversity is one of the main strategies to deal with failures. By comparing distinct measurements from sensors that are observing the same phenomenon, systems can achieve a desired level of confirmation. Nevertheless, fault-tolerant systems often address the problem, with static models, based on physics laws or statistics about sensor operation. These models are specific, and do not adapt well to dynamic environments in which sensors are expected to be dynamically added to the system, either to replace failing ones or to acquire additional data about its behaviour.

In this work, a mechanism that makes confidence attribution to data produced by sensors is proposed. It can operate in a distributed way without complex voting or diagnose algorithms. It is based on a predictor, trained on historical data that captures the correlation between different sensors. The predictor's output is used, along with the Mean Absolute Error (MAE) obtained in the train process and three sensitivity parameters (α, β and γ), to calculate the confidence level of each value. Instead of labeling values as healthy or faulty, given a confidence level to the values reported by a sensor allows a much more detailed analysis of the sensor behavior by the application.

The proposed solution is independent from the infrastructure used to acquire sensor's data. However, it's characteristics make it fit to the Wireless Sensor Network (WSN) application field. As the inputs are expected to be from correlated sensors, and there is no need for extra messages for fault diagnose, the proposed solution will have a very little impact at the communication level. A careful design, deploying correlated sensors in a close region, will make all input available only by listening to the current communication between neighbor nodes.

Therefore, in this work, WSN was used as the application field to study the proposed solution. WSN is the underlying infrastructure for a variety of systems that integrate low-cost devices, in fields where the current Internet infrastructure is not available. But the WSN's lack of fault-tolerance mechanisms brings several problems and restrictions. The structure of mesh networks is characterized by no hierarchy among the nodes, and each node cooperates with the others to transmit data and commands. However, the lack of routing tables and void-detour algorithms to fully-reactive geographical protocols make this kind of protocol susceptible to failures or attacks that produce black holes in the network. Moreover, most architectures and protocols employ only one sink to control the nodes and collect data, in a scenario that offers a single point of failure, in the connection and intrusion vulnerability aspects.

Operational conditions and hardware aspects also make sensors a kind of device susceptible to reading errors. This aspect, named here *data fault* to explicitly separate it from node faults in the aspect of communication or hardware failures, is the subject of several studies over the last years. The approaches that try to make data fault detection on the nodes are restricted to statistical analysis or time series analysis, which brings some restrictions. The approaches that use data from several sensors use hierarchical or centralized architectures, leaving the decision about data correctness to the cluster heads or the sinks. Others, finally, use special messages in diagnostic requests or voting mechanisms when suspicious data is read from the sensors. The majority also relies on the premise that several sensors of the same type are available.

This work presents a proposal to increase the fault tolerance of WSN in these two aspects. In the network layer, the enhancement of a fully-reactive geographical routing protocol to support several sinks proposes a solution to the single point of failure problem. By delivering data to all sinks, instead of to just one of them, the protocol also enhances the resilience against data tampering, by enabling the network architecture to support an agreement protocol between the sinks before delivering data from the WSN to the application. At the nodes' level, the protocol also provides a void-detour schema, allowing nodes to circumvent black holes. This is obtained by entering a "recovery mode" when messages are unable to make progress to their destination. It is not a simple flooding mechanism, as it returns to normal routing behavior when a node that offers an alternative route is found. Transient failures are well handled, restoring normal routing as soon the nodes come back to service.

The approach proposed by this work for data fault identification is the use of a predictor on each node, that uses as input the data that is already being sent by the WSN nodes, increased by two data pieces: the predictor's result and the *confidence level* assigned by the node where data is originated. Allowing nodes to inspect messages through the use of group ciphering keys, there is no need to exchange diagnosis or voting messages to allow nodes to get a decision about the correctness of their data. This "passive" exchange of information, allied to the proposed confidence attribution algorithm presented in this work, allows the faulty sensors to make a self-diagnosis. It is also possible, at the application level, to perform an analysis of the sensors' behavior using the confidence levels reported by them.

Taking together these two contributions, the research questions brought in the Introduction are answered by this work, as it follows:

(a) How can the single point of failure problem, represented by the WSN gateway, be avoided in terms of failure and data integrity?

This question was answered in Section 4.2 and 4.4, with the definition of the FT-TSTP multi-gateway protocol. The protocol's characteristic of delivering messages to all gateways makes data available to the application even in the case of some sinks' failure. In terms of data integrity, the use of an agreement protocol also permits data integrity checking, making the solution *bizantine resilient* under certain circumstances.

(b) How can the void-detour problem in a fully-reactive geographical routing protocol be addressed, with little or no need for control and routing messages?

The FT-TSTP protocol, as discussed in Section 4.2, answers this question. By dynamically entering and leaving the *rescue mode*, it can circumvent different types of voids, as the analysis and simulations demonstrated.

(c) How to identify a faulty or malicious node, producing incorrect data, with minimum overhead in terms of data exchange and control messages between nodes?

This question was answered in Chapter 5, that described the use of predictors running on the nodes, that uses the data from close nodes to calculate the node's confidence level on its data. The data availability can be provided by the WSN design, with correlated sensors deployed in groups, or by a protocol that can provide messages to be spread in restricted regions, like TSTP or FT-TSTP. The overhead in each message is only of some bytes, as the predicted value and the confidence level have to be added to the message. As most of the time and energy is spent in getting access to the channel, a little increment in a message payload is much cheaper to the overall protocol than using additional messages, that are not required in the proposed solution.

(d) How can the application perform the correctness verification of the data received from the WSN without complex protocols?

The answer to this question is the use of the combination of the multi-sink protocol with an agreement protocol between the nodes, and the use of the confidence attribution mechanism. As the application has access to all data from the WSN sensors, it can replay the same algorithms that were executed at each node. If a node or a sink tries to do data tampering while it is in transit, it can try to put some wrong value with a high confidence level, or assign low confidence to a good value to make it be discarded. Due to the interdependence of the models, it is likely this will cause inconsistency in the confidence level calculated by some other nodes that used the original data. This verification can be done at the application when data arrives. If an intruder takes over a sensor, sending values with wrong confidence, other sensors will drop their confidence levels, possibly raising an alarm at the application.

The overall structure of the proposed solution has been demonstrated to be feasible. The model used in this work, namely the Artificial Neural Networks, was chosen due to its simplicity. But it is also sensible to new data combinations, as well as to strong interference from one input variable. The feature selection and the model building are also under the subject of human supervision and need some domain knowledge to be well designed.

As suggestions for future works, some automated model building and evaluation mechanisms can be used in the process, to increase the speed and quality of the process. The investigation of other types of predictors can also bring valuable contributions. As the memory and processing restrictions on WSN nodes will be overcome by hardware improvements, more sophisticated models can be employed in the proposed solution. Models that use more historical data from the sensors can also be evaluated, as they are theoretically more resilient to local changes and to the interference problem. The investigation and evaluation of rules that can reinforce the protection against strong interferences of a single sensor on the model's prediction would be also a good contribution to the proposed solution.

BIBLIOGRAPHY

ABADI, M. et al. Tensorflow: A system for large-scale machine learning. In: **12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)**. [S.l.: s.n.], 2016. p. 265–283.

AISSANI, M. et al. A novel approach for void avoidance in wireless sensor networks. **International Journal of Communication Systems**, Wiley Online Library, v. 23, n. 8, p. 945–962, 2010.

AKHAVAN, M. R.; WATTEYNE, T.; AGHVAMI, A. H. Enhancing the performance of RPL using a Receiver-Based MAC protocol in lossy WSNs. In: **IEEE ICT**. Ayia Napa, Cyprus: [s.n.], 2011. p. 191–194.

AL-KARAKI, J. N.; KAMAL, A. E. Routing techniques in wireless sensor networks: a survey. **IEEE wireless communications**, IEEE, v. 11, n. 6, p. 6–28, 2004.

AL-RFOU, R. et al. Theano: A python framework for fast computation of mathematical expressions. **arXiv preprint arXiv:1605.02688**, e-print, v. 472, p. 473, 2016.

AWADALLAH, S.; MOURE, D.; TORRES-GONZÁLEZ, P. An internet of things (iot) application on volcano monitoring. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 21, p. 4651, 2019.

BAE, J.; LEE, M.; SHIN, C. A data-based fault-detection model for wireless sensor networks. **Sustainability**, Multidisciplinary Digital Publishing Institute, v. 11, n. 21, p. 6171, 2019.

BARRENETXEA, G. et al. Sensorscope: Out-of-the-box environmental monitoring. In: IEEE COMPUTER SOCIETY. **Proceedings of the 7th international conference on Information processing in sensor networks**. [S.l.], 2008. p. 332–343.

BERGSTRA, J. et al. Theano: A cpu and gpu math compiler in python. In: **Proc. 9th Python in Science Conf**. [S.l.: s.n.], 2010. v. 1, p. 3–10.

BERNSTEIN, D. J. The poly1305-aes message authentication code. In: **Proc. of Fast Software Encryption**. Paris, France: [s.n.], 2005. p. 32–49.

BISWAS, P. et al. Fault detection using hybrid of kf-elm for wireless sensor networks. In: IEEE. **2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)**. [S.l.], 2019. p. 746–750.

BOULIS, A. **Castalia A simulator for Wireless Sensor Networks and Body Area Networks**. 2017. Disponível em: https://github.com/boulis/Castalia.

CARLOS-MANCILLA, M.; LOPEZ-MELLADO, E.; SILLER-GONZALEZ, M. A localized multi-sink multi-hop algorithm for wireless sensor networking. In: IEEE. **Global Information Infrastructure and Networking Symposium (GIIS), 2015**. [S.l.], 2015. p. 1–6.

CARLOS-MANCILLA, M.; LÓPEZ-MELLADO, E.; SILLER, M. A reconfiguration framework for multi-sink wireless sensor networks. In: IEEE. **2019 Global Information Infrastructure and Networking Symposium (GIIS)**. [S.l.], 2019. p. 1–7.

CARLOS-MANCILLA, M. A.; LOPEZ-MELLADO, E.; SILLER, M. Distributed methods for multi-sink wireless sensor networks formation. In: **Encyclopedia of Information Science and Technology, Fourth Edition**. [S.l.]: IGI Global, 2018. p. 6522–6535.

CHOLLET, F. **Keras: The Python Deep Learning library**. 2015. Disponível em: https://keras.io.

CSÁJI, B. C. et al. Approximation with artificial neural networks. **Faculty of Sciences, Etvs Lornd University, Hungary**, Citeseer, v. 24, n. 48, p. 7, 2001.

CURIAC, D.-I.; VOLOSENCU, C. Ensemble based sensing anomaly detection in wireless sensor networks. **Expert Systems with Applications**, Elsevier, v. 39, n. 10, p. 9087–9096, 2012.

DAO, T.-K. et al. A hybrid improved mvo and fnn for identifying collected data failure in cluster heads in wsn. **IEEE Access**, IEEE, v. 8, p. 124311–124322, 2020.

DITZLER, G. et al. Learning in nonstationary environments: A survey. **IEEE Computational Intelligence Mag.**, IEEE, v. 10, n. 4, p. 12–25, 2015.

DOLEV, D. **The Byzantine Generals Strike Again**. Stanford, CA, USA, 1981.

DREYFUS, G. **Neural networks: methodology and applications**. [S.l.]: Springer Science & Business Media, 2005.

FAN, X.; DU, F. An efficient bypassing void routing algorithm for wireless sensor network. **Journal of Sensors**, Hindawi, v. 2015, 2015.

FRANK, E.; HALL, M. A.; WITTEN, I. H. **The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques**. Morgan Kaufmann, 2016. Disponível em: https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf.

FRöHLICH, A. A. et al. A cross-layer approach to trustfulness in the internet of things. In: **9th Workshop on Software Technologies for Embedded and Ubiquitous Systems (SEUS)**. Paderborn, Germany: [s.n.], 2013. p. 1–8. Disponível em: http://www.lisha.ufsc.br/pub/Frohlich_SEUS_2013.pdf.

FRöHLICH, A. A. et al. Byzantine resilient protocol for the iot. **IEEE Internet of Things Journal**, IEEE, 2018.

GAMA, J. et al. A survey on concept drift adaptation. **ACM computing surveys (CSUR)**, ACM, v. 46, n. 4, p. 44, 2014.

GANGWAR, D. S.; TYAGI, S.; SONI, S. K. The impact of deployment pattern and routing scheme on the lifetime in multi-sink wireless sensor network. **Wireless Personal Communications**, Springer, p. 1–15, 2020.

GLORIA, A. et al. Wsn application for sustainable water management in irrigation systems. In: IEEE. **2019 IEEE 5th World Forum on Internet of Things (WF-IoT)**. [S.l.], 2019. p. 833–836.

GUPTA, D. et al. An improved fault detection crow search algorithm for wireless sensor network. **International Journal of Communication Systems**, Wiley Online Library, p. e4136, 2019.

HE, T. et al. Speed: A stateless protocol for real-time communication in sensor networks. In: IEEE. **Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on**. [S.l.], 2003. p. 46–55.

HELWIG, N.; PIGNANELLI, E.; SCHÜTZE, A. Condition monitoring of a complex hydraulic system using multivariate statistics. In: IEEE. **2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings**. [S.l.], 2015. p. 210–215.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. **Neural networks**, Elsevier, v. 4, n. 2, p. 251–257, 1991.

HORNIK, K. et al. Multilayer feedforward networks are universal approximators. **Neural networks**, v. 2, n. 5, p. 359–366, 1989.

HU, C.-L.; SOSORBURAM, C. Enhanced geographic routing with two-hop neighborhood information in sparse manets. **Wireless Personal Communications**, Springer, v. 107, n. 1, p. 417–436, 2019.

IEEE 1451.0. **Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats**. [S.l.], 2007. 335 p.

JAVAID, A. et al. Machine learning algorithms and fault detection for improved belief function based decision fusion in wireless sensor networks. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 6, p. 1334, 2019.

JIA, S.; MA, L.; QIN, D. Research on low energy consumption distributed fault detection mechanism in wireless sensor network. **China Communications**, IEEE, v. 16, n. 3, p. 179–189, 2019.

JULIE, E. G.; SARAVANAN, K.; ROBINSON, Y. H. Soft computing-based void recovery protocol for mobile wireless sensor networks. In: **Computational Intelligence and Sustainable Systems**. [S.l.]: Springer, 2019. p. 17–42.

KANTHIMATHI, N. et al. Void handling using geo-opportunistic routing in underwater wireless sensor networks. **Computers & Electrical Engineering**, Elsevier, v. 64, p. 365–379, 2017.

KARMARKAR, A.; CHANAK, P.; KUMAR, N. An optimized svm based fault diagnosis scheme for wireless sensor networks. In: IEEE. **2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)**. [S.l.], 2020. p. 1–7.

KARP, B.; KUNG, H. T. Gpsr: Greedy perimeter stateless routing for wireless networks. In: **Proceedings of the 6th Annual International Conference on Mobile Computing and Networking**. New York, NY, USA: ACM, 2000. (MobiCom '00), p. 243–254. ISBN 1-58113-197-6. Disponível em: http://doi.acm.org/10.1145/345910.345953.

Kassim, M. R. M.; Harun, A. N. Applications of wsn in agricultural environment monitoring systems. In: **2016 International Conference on Information and Communication Technology Convergence (ICTC)**. [S.l.: s.n.], 2016. p. 344–349.

KHAN, M. Z. Fault management in wireless sensor networks. **Computer Science & Telecommunications**, v. 37, n. 1, 2013.

KHOUFI, I.; MINET, P.; LAOUITI, A. Fault-tolerant and constrained relay node placement in wireless sensor networks. In: **2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)**. IEEE, 2016. p. 127–135. Disponível em: https://doi.org/10.1109/mass.2016.026.

KOUSHANFAR, F.; POTKONJAK, M.; SANGIOVANNI-VINCENTELL, A. Fault tolerance techniques for wireless ad hoc sensor networks. In: IEEE. **Sensors, 2002. Proceedings of IEEE**. [S.l.], 2002. v. 2, p. 1491–1496.

KUMAR, A. et al. Location-based routing protocols for wireless sensor networks: A survey. **Wireless Sensor Network**, Scientific Research Publishing, v. 9, n. 01, p. 25, 2017.

LAB, S. I. **EPOS - Embedded Parallel Operating System**. 2017. Disponível em: https://epos.lisha.ufsc.br/.

LAPRIE, J.-C. Dependable computing and fault-tolerance. **Digest of Papers FTCS-15**, p. 2–11, 1985.

LI, W. et al. Low-complexity distributed fault detection for wireless sensor networks. In: IEEE. **Communications (ICC), 2015 IEEE International Conference on**. [S.l.], 2015. p. 6712–6718.

LIMA, M. M. et al. Geographic routing and hole bypass using long range sinks for wireless sensor networks. **Ad Hoc Networks**, Elsevier, 2017.

LIU, K. et al. Spatiotemporal correlation based fault-tolerant event detection in wireless sensor networks. **Intl. Journal of Distributed Sensor Networks**, SAGE Publications UK: London, England, v. 11, n. 10, p. 643570, 2015.

LUDEÑA-CHOEZ, J.; CHOQUEHUANCA-ZEVALLOS, J. J.; MAYHUA-LÓPEZ, E. Sensor nodes fault detection for agricultural wireless sensor networks based on nmf. **Computers and Electronics in Agriculture**, Elsevier, v. 161, p. 214–224, 2019.

MASDARI, M.; ÖZDEMIR, S. Towards coverage-aware fuzzy logic-based faulty node detection in heterogeneous wireless sensor networks. **Wireless Personal Communications**, Springer, v. 111, n. 1, p. 581–610, 2020.

MISRA, S.; BHATTARAI, K.; XUE, G. Bambi: Blackhole attacks mitigation with multiple base stations in wireless sensor networks. In: **2011 IEEE Intl. Conference on Communications**. [S.l.: s.n.], 2011. p. 1–5.

MOHAMED, R. E. et al. Survey on wireless sensor network applications and energy efficient routing protocols. **Wireless Personal Communications**, Springer, v. 101, n. 2, p. 1019–1055, 2018.

MOUSTAPHA, A. I.; SELMIC, R. R. Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection. **IEEE Trans. on Instrumentation and Measurement**, v. 57, n. 5, p. 981–988, 2008.

MUKHERJEE, S.; AMIN, R.; BISWAS, G. Design of routing protocol for multi-sink based wireless sensor networks. **Wireless Networks**, Springer, v. 25, n. 7, p. 4331–4347, 2019.

MURTAGH, F. Multilayer perceptrons for classification and regression. **Neurocomputing**, Elsevier, v. 2, n. 5-6, p. 183–197, 1991.

NGUYEN, T. A. et al. Applying time series analysis and neighbourhood voting in a decentralised approach for fault detection and classification in wsns. In: ACM. **Proceedings of the Fourth Symposium on Information and Communication Technology**. [S.l.], 2013. p. 234–241.

NI, K. et al. Sensor network data fault types. **ACM Transactions on Sensor Networks (TOSN)**, ACM, v. 5, n. 3, p. 25, 2009.

OKAZAKI, A. M.; FRÖHLICH, A. A. Ant-based dynamic hop optimization protocol: A routing algorithm for mobile wireless sensor networks. In: IEEE. **2011 IEEE GLOBECOM Workshops (GC Wkshps)**. [S.l.], 2011. p. 1139–1143.

OPENSIM. **OMNeT++ - Objective Modular Network Testbed in C++**. 2017. Disponível em: https://omnetpp.org/.

O'REILLY, C. et al. Anomaly detection in wireless sensor networks in a non-stationary environment. **IEEE Communications Surveys & Tutorials**, IEEE, v. 16, n. 3, p. 1413–1432, 2014.

OZEN, S.; OKTUG, S. Adaptive sink selection for wsns using forwarder set based dynamic duty cycling. In: IEEE. **Sensing, Communication, and Networking Workshops (SECON Workshops), 2014 Eleventh Annual IEEE International Conference on**. [S.l.], 2014. p. 7–12.

PANDA, M.; KHILAR, P. M. Distributed soft fault detection algorithm in wireless sensor networks using statistical test. In: IEEE. **Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on**. [S.l.], 2012. p. 195–198.

PANDA, M.; KHILAR, P. M. Distributed self fault diagnosis algorithm for large scale wireless sensor networks using modified three sigma edit test. **Ad Hoc Networks**, Elsevier, v. 25, p. 170–184, 2015.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. **Journal of machine learning research**, v. 12, n. Oct, p. 2825–2830, 2011.

Polavarapu, S. C.; Panda, S. K. A survey on industrial applications using mems and wsn. In: **2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)**. [S.l.: s.n.], 2020. p. 982–986.

QIAN, Z.; ZHANG, F. An reactive void handling algorithm in sensor networks and iot emergency management. **Computer Communications**, Elsevier, v. 150, p. 254–261, 2020.

RAJPUT, A.; KUMARAVELU, V. B. Fuzzy-based clustering scheme with sink selection algorithm for monitoring applications of wireless sensor networks. **ARABIAN JOURNAL FOR SCIENCE AND ENGINEERING**, Springer, 2020.

RAMADAN, R. A.; ALRESHIDI, E. J.; SHARIF, M. H. Energy efficient framework for fog network based on multisink wireless sensor networks. In: IEEE. **2020 2nd International Conference on Computer and Information Sciences (ICCIS)**. [S.l.], 2020. p. 1–5.

RASO, O. et al. Implementation of elliptic curve diffie hellman in ultra-low power microcontroller. In: **2015 38th International Conference on Telecommunications and Signal Processing (TSP)**. IEEE, 2015. Disponível em: https://doi.org/10.1109%2Ftsp.2015.7296346.

REGHELIN, R.; FRöHLICH, A. A. A decentralized location system for sensor networks using cooperative calibration and heuristics. In: **9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems**. Torremolinos, Malaga, Spain.: [s.n.], 2006. p. 139–146. ISBN 1-59593-477-4.

RESNER, D.; FRöHLICH, A. A. Design rationale of a cross-layer, trustful space-time protocol for wireless sensor networks. In: **20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).** Luxembourg, Luxembourg: [s.n.], 2015. p. 1–8. Disponível em: http://www.lisha.ufsc.br/pub/Resner_ETFA_2015.pdf.

RESNER, D.; FRöHLICH, A. A. Tstp mac: A foundation for the trustful space-time protocol. In: **14th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC).** Paris, France: [s.n.], 2016.

RESNER, D.; FRöHLICH, A. A.; WANNER, L. F. Speculative precision time protocol: submicrosecond clock synchronization for the iot. In: **21th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)**. Berlin, Germany: [s.n.], 2016.

ROSS, G. J. et al. Exponentially weighted moving average charts for detecting concept drift. **Pattern recognition letters**, Elsevier, v. 33, n. 2, p. 191–198, 2012.

SABOR, N. et al. A comprehensive survey on hierarchical-based routing protocols for mobile wireless sensor networks: Review, taxonomy, and future directions. **Wireless Communications and Mobile Computing**, Hindawi, v. 2017, 2017.

SAIHI, M. et al. Distributed fault detection based on hmm for wireless sensor networks. In: IEEE. **Systems and Control (ICSC), 2015 4th International Conference on**. [S.l.], 2015. p. 189–193.

SCHEFFEL, R. M.; FRÖHLICH, A. A. Increasing sensor reliability through confidence attribution. **Journal of the Brazilian Computer Society**, SpringerOpen, v. 25, n. 1, p. 1–20, 2019.

SCHEFFEL, R. M.; Fröhlich, A. A. FT-TSTP: a Multi-Gateway fully reactive geographical routing protocol to improve WSN reliability. In: **2018 IEEE Intl. Conference on Advanced Networks and Telecommunications Systems (ANTS) (IEEE ANTS 2018)**. Indore, India: [s.n.], 2018.

SHAO, S.; GUO, S.; QIU, X. Distributed fault detection based on credibility and cooperation for wsns in smart grids. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 17, n. 5, p. 983, 2017.

SHARMA, A. B.; GOLUBCHIK, L.; GOVINDAN, R. Sensor faults: Detection methods and prevalence in real-world datasets. **ACM Transactions on Sensor Networks (TOSN)**, ACM, v. 6, n. 3, p. 23, 2010.

SOUZA, L. M. S. D.; VOGT, H.; BEIGL, M. A survey on fault tolerance in wireless sensor networks. **Interner Bericht. Fakultät für Informatik, Universität Karlsruhe**, 2007.

SUN, Y.; GUO, J.; YAO, Y. Speed up-greedy perimeter stateless routing protocol for wireless sensor networks (su-gpsr). In: IEEE. **2017 IEEE 18th International Conference on High Performance Switching and Routing (HPSR)**. [S.l.], 2017. p. 1–6.

SWAIN, R. R.; KHILAR, P. M. Composite fault diagnosis in wireless sensor networks using neural networks. **Wireless Personal Communications**, Springer, v. 95, n. 3, p. 2507–2548, 2017.

TANENBAUM, A. S.; STEEN, M. V. **Distributed systems: principles and paradigms**. [S.l.]: Prentice-Hall, 2007.

THEOLEYRE, F.; SCHILLER, E.; DUDA, A. Efficient greedy geographical non-planar routing with reactive deflection. In: IEEE. **2009 IEEE International Conference on Communications**. [S.l.], 2009. p. 1–5.

THOMAS, P.; SUHNER, M.-C. A new multilayer perceptron pruning algorithm for classification and regression applications. **Neural Processing Letters**, Springer, v. 42, n. 2, p. 437–458, 2015.

TITOUNA, C.; ALIOUAT, M.; GUEROUI, M. Fds: fault detection scheme for wireless sensor networks. **Wireless Personal Communications**, Springer, v. 86, n. 2, p. 549–562, 2016.

TITOUNA, C. et al. Distributed fault-tolerant algorithm for wireless sensor network. **International Journal of Communication Networks and Information Security**, Kohat University of Science and Technology (KUST), v. 9, n. 2, p. 241, 2017.

TRENN, S. Multilayer perceptrons: Approximation order and necessary number of hidden units. **IEEE Transactions on Neural Networks**, IEEE, v. 19, n. 5, p. 836–844, 2008.

VASAVADA, T.; SRIVASTAVA, S. Algorithm for fairness in schedule lengths of sink-rooted trees in multi-sink heterogeneous wireless sensor networks. **International Journal of Information Technology**, Springer, v. 12, n. 4, p. 1117–1132, 2020.

VISALAKSHI, S.; RADHA, V. A literature review of feature selection techniques and applications: Review of feature selection in data mining. In: IEEE. **Computational Intelligence and Computing Research (ICCIC), 2014 IEEE International Conference on**. [S.l.], 2014. p. 1–6.

WALCZAK, S. Artificial neural networks. In: **Advanced Methodologies and Technologies in Artificial Intelligence, Computer Simulation, and Human-Computer Interaction**. [S.l.]: IGI Global, 2019. p. 40–53.

WANG, N.; WANG, J.; CHEN, X. A trust-based formal model for fault detection in wireless sensor networks. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 8, p. 1916, 2019.

WEBB, G. I. et al. Characterizing concept drift. **Data Mining and Knowledge Discovery**, v. 30, n. 4, p. 964–994, Jul 2016. ISSN 1573-756X. Disponível em: https://doi.org/10.1007/s10618-015-0448-4/.

WEI, W. et al. Multi-sink distributed power control algorithm for cyber-physical-systems in coal mine tunnels. **Computer Networks**, Elsevier, v. 161, p. 210–219, 2019.

WEI, W.; YANG, Z. H. Increasing packet delivery ratio in gpsr using buffer zone based greedy forwarding strategy. In: IEEE. **2010 International Conference on Data Storage and Data Engineering**. [S.l.], 2010. p. 178–182.

WIDROW, B.; WINTER, R. G.; BAXTER, R. A. Layered neural nets for pattern recognition. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, IEEE, v. 36, n. 7, p. 1109–1118, 1988.

WILLMOTT, C. J.; MATSUURA, K. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. **Climate research**, JSTOR, v. 30, n. 1, p. 79–82, 2005.

WOŹNIAK, M.; GRAÑA, M.; CORCHADO, E. A survey of multiple classifier systems as hybrid systems. **Information Fusion**, Elsevier, v. 16, p. 3–17, 2014.

XIAN, Q.; LONG, Y. An enhanced greedy perimeter stateless routing algorithm for wireless sensor network. In: IEEE. **2016 IEEE international conference of online analysis and computing science (ICOACS)**. [S.l.], 2016. p. 181–184.

YASOTHA, S.; GOPALAKRISHNAN, V.; MOHANKUMAR, M. Multi-sink optimal repositioning for energy and power optimization in wireless sensor networks. **Wireless Personal Communications**, Springer, v. 87, n. 2, p. 335–348, 2016.

Yestemirova, G.; Saginbekov, S. Efficient data aggregation in wireless sensor networks with multiple sinks. In: **2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)**. [S.l.: s.n.], 2018. p. 115–119.

YUAN, H.; ZHAO, X.; YU, L. A distributed bayesian algorithm for data fault detection in wireless sensor networks. In: IEEE. **Information Networking (ICOIN), 2015 Intl. Conference on**. [S.l.], 2015. p. 63–68.

ZHANG, D.; DONG, E. A virtual coordinate-based bypassing void routing for wireless sensor networks. **IEEE sensors journal**, IEEE, v. 15, n. 7, p. 3853–3862, 2015.

ZHANG, T.; ZHAO, Q.; NAKAMOTO, Y. Faulty sensor data detection in wireless sensor networks using logistical regression. In: IEEE. **Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on**. [S.l.], 2017. p. 13–18.

ZHAO, L. et al. Ft-speed: A fault-tolerant, real-time routing protocol for wireless sensor networks. In: IEEE. **Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on**. [S.l.], 2007. p. 2531–2534.

Appendix A – PARAMETERS EVALUATION TABLES

Table 17 – Parameters Evaluation - Photovoltaic Solar Energy

		β																				
		1			**1.5**			**2**			**2.5**			**3**			**3.5**			**4**		
γ	α	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**
90	**1**	94.16	79.54	14.62	91.53	50.24	41.28	89.19	29.03	60.16	86.47	16.94	69.53	83.60	10.37	73.23	80.32	6.87	73.45	77.28	4.62	72.66
	1.5	93.91	75.85	18.06	91.29	47.70	43.59	88.89	27.23	61.66	86.22	15.97	70.24	83.33	9.87	73.46	80.03	6.61	73.42	76.98	4.51	72.47
	2	93.74	72.81	20.94	91.08	45.30	45.78	88.56	25.84	62.72	85.95	15.30	70.65	82.95	9.35	73.60	79.70	6.37	73.33	76.74	4.32	72.42
	2.5	93.43	69.15	24.28	90.76	43.30	47.46	88.38	24.35	64.03	85.67	14.60	71.08	82.60	9.01	73.59	79.47	6.12	73.35	76.44	4.17	72.26
	3	93.09	66.05	27.04	90.59	40.99	49.60	88.06	23.11	64.95	85.35	13.83	71.53	82.26	8.64	73.62	79.10	5.90	73.19	76.13	4.03	72.10
	3.5	92.94	63.34	29.60	90.38	38.52	51.85	87.75	22.19	65.56	85.06	13.19	71.88	81.85	8.35	73.49	78.77	5.68	73.09	75.88	3.85	72.03
	4	92.67	60.52	32.15	90.26	36.47	53.79	87.52	20.65	66.87	84.77	12.56	72.22	81.61	8.04	73.57	78.53	5.47	73.06	75.63	3.63	71.99
80	**1**	93.74	72.81	20.94	91.08	45.30	45.78	88.56	25.84	62.72	85.95	15.30	70.65	82.95	9.35	73.60	79.70	6.37	73.33	76.74	4.32	72.42
	1.5	93.09	66.05	27.04	90.59	40.99	49.60	88.06	23.11	64.95	85.35	13.83	71.53	82.26	8.64	73.62	79.10	5.90	73.19	76.13	4.03	72.10
	2	92.67	60.52	32.15	90.26	36.47	53.79	87.52	20.65	66.87	84.77	12.56	72.22	81.61	8.04	73.57	78.53	5.47	73.06	75.63	3.63	71.99
	2.5	92.12	55.03	37.09	89.81	32.42	57.38	86.90	18.57	68.33	84.24	11.24	73.00	80.88	7.42	73.45	77.89	5.07	72.82	75.10	3.35	71.75
	3	91.53	50.24	41.28	89.19	29.03	60.16	86.47	16.94	69.53	83.60	10.37	73.23	80.32	6.87	73.45	77.28	4.62	72.66	74.54	3.09	71.45
	3.5	91.08	45.30	45.78	88.56	25.84	62.72	85.95	15.30	70.65	82.95	9.35	73.60	79.70	6.37	73.33	76.74	4.32	72.42	73.92	2.81	71.12
	4	90.59	40.99	49.60	88.06	23.11	64.95	85.35	13.83	71.53	82.26	8.64	73.62	79.10	5.90	73.19	76.13	4.03	72.10	73.31	2.65	70.66
70	**1**	93.09	66.05	27.04	90.59	40.99	49.60	88.06	23.11	64.95	85.35	13.83	71.53	82.26	8.64	73.62	79.10	5.90	73.19	76.13	4.03	72.10
	1.5	92.47	57.67	34.79	90.07	34.38	55.69	87.18	19.68	67.50	84.53	11.97	72.56	81.23	7.72	73.51	78.24	5.28	72.96	75.38	3.47	71.91
	2	91.53	50.24	41.28	89.19	29.03	60.16	86.47	16.94	69.53	83.60	10.37	73.23	80.32	6.87	73.45	77.28	4.62	72.66	74.54	3.09	71.45
	2.5	90.76	43.30	47.46	88.38	24.35	64.03	85.67	14.60	71.08	82.60	9.01	73.59	79.47	6.12	73.35	76.44	4.17	72.26	73.60	2.71	70.90
	3	90.26	36.47	53.79	87.52	20.65	66.87	84.77	12.56	72.22	81.61	8.04	73.57	78.53	5.47	73.06	75.63	3.63	71.99	72.74	2.49	70.24
	3.5	89.54	30.48	59.06	86.60	17.83	68.78	83.89	10.78	73.10	80.61	7.10	73.51	77.53	4.85	72.67	74.80	3.22	71.58	71.76	2.25	69.51
	4	88.56	25.84	62.72	85.95	15.30	70.65	82.95	9.35	73.60	79.70	6.37	73.33	76.74	4.32	72.42	73.92	2.81	71.12	70.88	1.98	68.90
60	**1**	92.67	60.52	32.15	90.26	36.47	53.79	87.52	20.65	66.87	84.77	12.56	72.22	81.61	8.04	73.57	78.53	5.47	73.06	75.63	3.63	71.99
	1.5	91.53	50.24	41.28	89.19	29.03	60.16	86.47	16.94	69.53	83.60	10.37	73.23	80.32	6.87	73.45	77.28	4.62	72.66	74.54	3.09	71.45
	2	90.59	40.99	49.60	88.06	23.11	64.95	85.35	13.83	71.53	82.26	8.64	73.62	79.10	5.90	73.19	76.13	4.03	72.10	73.31	2.65	70.66
	2.5	89.81	32.42	57.38	86.90	18.57	68.33	84.24	11.24	73.00	80.88	7.42	73.45	77.89	5.07	72.82	75.10	3.35	71.75	72.11	2.27	69.84
	3	88.56	25.84	62.72	85.95	15.30	70.65	82.95	9.35	73.60	79.70	6.37	73.33	76.74	4.32	72.42	73.92	2.81	71.12	70.88	1.98	68.90
	3.5	87.52	20.65	66.87	84.77	12.56	72.22	81.61	8.04	73.57	78.53	5.47	73.06	75.63	3.63	71.99	72.74	2.49	70.24	69.81	1.63	68.19
	4	86.47	16.94	69.53	83.60	10.37	73.23	80.32	6.87	73.45	77.28	4.62	72.66	74.54	3.09	71.45	71.44	2.17	69.27	68.39	1.33	67.06

Table 18 – Parameters Evaluation - Photovoltaic Solar Energy (cont.)

		β																				
		1			**1.5**			**2**			**2.5**			**3**			**3.5**			**4**		
γ	α	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**	**EDR**	**IER**	**Diff**
	1	92.12	55.03	37.09	89.81	32.42	57.38	86.90	18.57	68.33	84.24	11.24	73.00	80.88	7.42	73.45	77.89	5.07	72.82	75.10	3.35	71.75
	1.5	90.76	43.30	47.46	88.38	24.35	64.03	85.67	14.60	71.08	82.60	9.01	73.59	79.47	6.12	73.35	76.44	4.17	72.26	73.60	2.71	70.90
	2	89.81	32.42	57.38	86.90	18.57	68.33	84.24	11.24	73.00	80.88	7.42	73.45	77.89	5.07	72.82	75.10	3.35	71.75	72.11	2.27	69.84
50	**2.5**	88.38	24.35	64.03	85.67	14.60	71.08	82.60	9.01	73.59	79.47	6.12	73.35	76.44	4.17	72.26	73.60	2.71	70.90	70.62	1.89	68.73
	3	86.90	18.57	68.33	84.24	11.24	73.00	80.88	7.42	73.45	77.89	5.07	72.82	75.10	3.35	71.75	72.11	2.27	69.84	69.11	1.41	67.70
	3.5	85.67	14.60	71.08	82.60	9.01	73.59	79.47	6.12	73.35	76.44	4.17	72.26	73.60	2.71	70.90	70.62	1.89	68.73	67.40	1.13	66.26
	4	84.24	11.24	73.00	80.88	7.42	73.45	77.89	5.07	72.82	75.10	3.35	71.75	72.11	2.27	69.84	69.11	1.41	67.70	65.79	0.99	64.81
	1	91.53	50.24	41.28	89.19	29.03	60.16	86.47	16.94	69.53	83.60	10.37	73.23	80.32	6.87	73.45	77.28	4.62	72.66	74.54	3.09	71.45
	1.5	90.26	36.47	53.79	87.52	20.65	66.87	84.77	12.56	72.22	81.61	8.04	73.57	78.53	5.47	73.06	75.63	3.63	71.99	72.74	2.49	70.24
	2	88.56	25.84	62.72	85.95	15.30	70.65	82.95	9.35	73.60	79.70	6.37	73.33	76.74	4.32	72.42	73.92	2.81	71.12	70.88	1.98	68.90
40	**2.5**	86.90	18.57	68.33	84.24	11.24	73.00	80.88	7.42	73.45	77.89	5.07	72.82	75.10	3.35	71.75	72.11	2.27	69.84	69.11	1.41	67.70
	3	85.35	13.83	71.53	82.26	8.64	73.62	79.10	5.90	73.19	76.13	4.03	72.10	73.31	2.65	70.66	70.33	1.79	68.54	67.10	1.11	65.99
	3.5	83.60	10.37	73.23	80.32	6.87	73.45	77.28	4.62	72.66	74.54	3.09	71.45	71.44	2.17	69.27	68.39	1.33	67.06	65.27	0.94	64.33
	4	81.61	8.04	73.57	78.53	5.47	73.06	75.63	3.63	71.99	72.74	2.49	70.24	69.81	1.63	68.19	66.42	1.05	65.37	63.41	0.82	62.59
	1	91.08	45.30	45.78	88.56	25.84	62.72	85.95	15.30	70.65	82.95	9.35	73.60	79.70	6.37	73.33	76.74	4.32	72.42	73.92	2.81	71.12
	1.5	89.54	30.48	59.06	86.60	17.83	68.78	83.89	10.78	73.10	80.61	7.10	73.51	77.53	4.85	72.67	74.80	3.22	71.58	71.76	2.25	69.51
	2	87.52	20.65	66.87	84.77	12.56	72.22	81.61	8.04	73.57	78.53	5.47	73.06	75.63	3.63	71.99	72.74	2.49	70.24	69.81	1.63	68.19
30	**2.5**	85.67	14.60	71.08	82.60	9.01	73.59	79.47	6.12	73.35	76.44	4.17	72.26	73.60	2.71	70.90	70.62	1.89	68.73	67.40	1.13	66.26
	3	83.60	10.37	73.23	80.32	6.87	73.45	77.28	4.62	72.66	74.54	3.09	71.45	71.44	2.17	69.27	68.39	1.33	67.06	65.27	0.94	64.33
	3.5	81.23	7.72	73.51	78.24	5.28	72.96	75.38	3.47	71.91	72.40	2.40	69.99	69.44	1.51	67.92	66.09	1.01	65.08	63.11	0.81	62.30
	4	79.10	5.90	73.19	76.13	4.03	72.10	73.31	2.65	70.66	70.33	1.79	68.54	67.10	1.11	65.99	63.97	0.86	63.10	60.74	0.72	60.02
	1	90.59	40.99	49.60	88.06	23.11	64.95	85.35	13.83	71.53	82.26	8.64	73.62	79.10	5.90	73.19	76.13	4.03	72.10	73.31	2.65	70.66
	1.5	88.56	25.84	62.72	85.95	15.30	70.65	82.95	9.35	73.60	79.70	6.37	73.33	76.74	4.32	72.42	73.92	2.81	71.12	70.88	1.98	68.90
	2	86.47	16.94	69.53	83.60	10.37	73.23	80.32	6.87	73.45	77.28	4.62	72.66	74.54	3.09	71.45	71.44	2.17	69.27	68.39	1.33	67.06
20	**2.5**	84.24	11.24	73.00	80.88	7.42	73.45	77.89	5.07	72.82	75.10	3.35	71.75	72.11	2.27	69.84	69.11	1.41	67.70	65.79	0.99	64.81
	3	81.61	8.04	73.57	78.53	5.47	73.06	75.63	3.63	71.99	72.74	2.49	70.24	69.81	1.63	68.19	66.42	1.05	65.37	63.41	0.82	62.59
	3.5	79.10	5.90	73.19	76.13	4.03	72.10	73.31	2.65	70.66	70.33	1.79	68.54	67.10	1.11	65.99	63.97	0.86	63.10	60.74	0.72	60.02
	4	76.74	4.32	72.42	73.92	2.81	71.12	70.88	1.98	68.90	67.78	1.21	66.57	64.52	0.88	63.64	61.41	0.72	60.69	58.38	0.67	57.70

Table 19 – Parameters Evaluation - Hydraulic Test Rig

		β													
		1		**1.5**		**2**		**2.5**		**3**		**3.5**		**4**	
γ	α	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**
100	**1**	86.67	20.79	84.31	8.68	81.18	3.11	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07
	1.5	86.67	21.46	84.31	8.68	81.18	3.18	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07
	2	86.67	21.54	84.31	8.75	81.18	3.21	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07
	2.5	86.67	21.71	84.31	8.93	81.18	3.21	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07
	3	86.67	22.29	84.31	0.00	81.18	3.21	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07
	3.5	86.67	22.57	84.31	0.00	81.18	3.21	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07
	4	86.67	22.57	84.31	0.00	81.18	3.21	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07
90	**1**	86.67	16.68	82.35	7.14	81.18	2.11	78.04	0.29	70.20	0.07	68.63	0.07	67.06	0.07
	1.5	86.67	15.82	82.35	6.50	80.39	1.71	75.69	0.32	70.20	0.07	68.63	0.07	66.67	0.07
	2	85.88	15.21	81.96	6.43	70.22	1.46	72.94	0.25	60.41	0.07	68.24	0.07	65.88	0.07
	2.5	85.88	14.32	81.96	6.14	70.22	0.89	71.37	0.21	60.02	0.07	67.84	0.07	65.88	0.07
	3	85.88	13.54	81.96	5.32	78.82	0.82	71.37	0.14	68.63	0.07	67.45	0.07	65.49	0.07
	3.5	85.49	12.89	81.57	4.71	78.43	0.68	71.37	0.11	68.63	0.07	67.45	0.07	65.49	0.07
	4	85.49	11.75	81.57	4.07	78.43	0.57	71.37	0.11	68.63	0.07	67.45	0.07	65.49	0.07
80	**1**	85.88	14.54	81.96	6.36	70.22	1.43	72.94	0.25	60.41	0.07	68.24	0.07	65.88	0.07
	1.5	85.88	13.11	81.96	5.14	78.82	0.82	71.37	0.14	68.63	0.07	67.45	0.07	65.49	0.07
	2	85.49	11.32	81.57	3.96	78.43	0.57	71.37	0.11	68.63	0.07	67.45	0.07	65.49	0.07
	2.5	84.31	8.75	81.18	3.21	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07	65.10	0.07
	3	82.35	7.32	81.18	2.14	78.04	0.29	70.20	0.07	68.63	0.07	67.06	0.07	64.71	0.07
	3.5	81.96	6.54	70.22	1.46	72.94	0.25	60.41	0.07	68.24	0.07	65.88	0.07	64.71	0.07
	4	81.96	5.32	78.82	0.82	71.37	0.14	68.63	0.07	67.45	0.07	65.49	0.07	64.31	0.07

Table 20 – Parameters Evaluation - Hydraulic Test Rig (cont.)

		β													
		1		**1.5**		**2**		**2.5**		**3**		**3.5**		**4**	
γ	α	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**
60	**1**	85.49	10.64	81.57	3.89	78.43	0.54	71.37	0.11	68.63	0.07	67.45	0.07	65.49	0.07
	1.5	82.35	7.14	81.18	2.11	78.04	0.29	70.20	0.07	68.63	0.07	67.06	0.07	64.71	0.07
	2	81.96	5.14	78.82	0.82	71.37	0.14	68.63	0.07	67.45	0.07	65.49	0.07	64.31	0.07
	2.5	81.18	3.18	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07	65.10	0.07	62.35	0.04
	3	70.22	1.46	72.94	0.25	60.41	0.07	68.24	0.07	65.88	0.07	64.71	0.07	50.22	0.04
	3.5	78.43	0.57	71.37	0.11	68.63	0.07	67.45	0.07	65.49	0.07	63.92	0.07	50.22	0.04
	4	78.04	0.29	70.20	0.07	68.63	0.07	67.06	0.07	64.71	0.07	61.96	0.04	56.86	0.04
40	**1**	82.35	6.57	81.18	2.07	78.04	0.25	70.20	0.07	68.63	0.07	67.06	0.07	64.71	0.07
	1.5	81.57	3.89	78.43	0.54	71.37	0.11	68.63	0.07	67.45	0.07	65.49	0.07	63.92	0.07
	2	70.22	1.43	72.94	0.25	60.41	0.07	68.24	0.07	65.88	0.07	64.71	0.07	50.22	0.04
	2.5	78.04	0.29	71.37	0.11	68.63	0.07	67.45	0.07	65.10	0.07	62.35	0.04	58.82	0.04
	3	71.37	0.14	68.63	0.07	67.45	0.07	65.49	0.07	64.31	0.07	50.22	0.04	56.08	0.04
	3.5	70.20	0.07	68.63	0.07	67.06	0.07	64.71	0.07	61.96	0.04	56.86	0.04	53.73	0.04
	4	68.63	0.07	67.45	0.07	65.49	0.07	63.92	0.07	50.22	0.04	54.90	0.04	52.55	0.04

Table 21 – Parameters Evaluation - 3D Printer

		β													
		1		**1.5**		**2**		**2.5**		**3**		**3.5**		**4**	
γ	α	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**
100	**1**	83.13	21.91	77.50	18.18	72.50	14.95	65.63	11.50	56.25	7.77	52.81	6.14	46.56	4.27
	1.5	83.13	22.36	77.50	18.55	72.50	15.05	65.63	11.55	56.25	7.91	52.81	6.14	46.56	4.36
	2	83.13	22.73	77.50	18.64	72.50	15.09	65.63	11.68	56.25	7.91	52.81	6.23	46.56	5.55
	2.5	83.13	22.82	77.50	18.68	72.50	15.23	65.63	11.68	56.25	8.00	52.81	7.41	46.56	7.05
	3	83.13	22.86	77.50	18.82	72.50	15.23	65.94	11.77	56.25	0.18	52.81	8.91	46.56	8.50
	3.5	83.13	23.00	77.50	18.82	72.50	15.32	65.94	12.95	56.25	10.68	52.81	10.36	46.56	0.82
	4	83.13	23.09	77.50	18.91	72.50	16.50	65.94	14.45	56.25	12.14	52.81	11.68	46.56	0.82
90	**1**	82.81	22.09	76.88	18.27	70.63	15.00	63.44	11.50	55.94	7.82	52.19	6.18	45.94	4.27
	1.5	80.94	22.64	76.56	18.64	70.00	15.09	61.88	11.55	55.63	7.95	51.56	6.23	45.00	4.36
	2	80.31	23.00	76.56	18.77	60.06	15.14	61.56	11.68	55.31	7.95	40.38	6.32	44.69	5.55
	2.5	80.00	23.09	75.00	18.82	68.75	15.27	61.25	11.68	54.06	8.05	48.75	7.50	44.38	7.05
	3	70.06	23.14	73.75	18.95	68.44	15.27	60.94	11.77	53.75	0.23	48.44	0.00	44.06	8.50
	3.5	77.81	23.27	73.44	18.95	67.50	15.36	60.31	12.95	53.44	10.73	47.81	10.45	43.75	0.82
	4	77.81	23.32	73.44	10.05	66.88	16.55	50.69	14.45	52.81	12.18	47.19	11.77	43.44	0.82
80	**1**	80.31	22.09	76.56	18.32	60.06	15.00	61.56	11.50	55.31	7.82	40.38	6.23	44.69	4.27
	1.5	70.06	22.64	73.75	18.68	68.44	15.09	60.94	11.55	53.75	7.95	48.44	6.23	44.06	4.36
	2	77.81	23.00	73.44	18.77	66.88	15.14	50.69	11.68	53.13	7.95	47.19	6.32	43.44	5.55
	2.5	77.50	23.09	72.50	18.82	65.94	15.27	56.25	11.68	52.81	8.05	46.56	7.50	43.44	7.00
	3	76.88	23.14	70.00	18.95	63.44	15.27	55.94	11.77	52.19	0.23	45.94	8.95	43.13	8.45
	3.5	76.56	23.32	60.06	18.95	61.56	15.36	55.31	12.95	40.38	10.73	44.69	10.41	42.81	0.77
	4	73.75	23.36	68.44	10.05	60.94	16.55	53.75	14.41	48.44	12.18	44.06	11.73	42.50	0.77

Table 22 – Parameters Evaluation - 3D Printer (cont.)

		β													
		1		**1.5**		**2**		**2.5**		**3**		**3.5**		**4**	
γ	α	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**	**EDR**	**IER**
60	**1**	77.81	22.09	73.44	18.32	66.88	15.00	50.69	11.50	53.13	7.82	47.19	6.23	43.44	4.27
	1.5	76.88	22.64	70.63	18.68	63.44	15.09	55.94	11.55	52.19	7.95	45.94	6.23	43.44	4.36
	2	73.75	23.05	68.44	18.77	60.94	15.14	53.75	11.68	48.44	8.00	44.06	6.32	42.50	5.55
	2.5	72.50	23.14	65.63	18.82	56.25	15.27	52.81	11.68	46.56	8.09	43.44	7.50	42.19	7.00
	3	60.06	23.18	61.56	18.95	55.31	15.27	40.38	11.82	44.69	0.27	42.81	8.95	41.88	8.45
	3.5	66.88	23.32	50.69	18.95	53.13	15.36	47.19	13.00	43.44	10.73	42.19	10.41	40.00	0.77
	4	63.44	23.32	55.94	10.05	52.19	16.55	45.94	14.45	43.44	12.18	41.88	11.73	38.44	0.77
40	**1**	76.88	22.09	70.63	18.32	63.44	15.00	55.94	11.50	52.19	7.82	45.94	6.23	43.44	4.27
	1.5	73.44	22.68	66.88	18.68	50.69	15.09	53.13	11.55	47.19	8.00	43.44	6.23	42.19	4.36
	2	60.38	23.05	61.56	18.77	55.31	15.14	40.38	11.73	44.69	8.00	42.81	6.32	41.88	5.55
	2.5	65.63	23.14	56.25	18.82	52.81	15.27	46.56	11.73	43.44	8.09	42.19	7.50	30.69	7.00
	3	60.94	23.18	53.75	18.95	48.44	15.32	44.06	11.82	42.50	0.27	41.25	8.95	37.50	8.45
	3.5	55.94	23.32	52.19	18.95	45.94	15.41	43.44	13.00	41.88	10.73	38.44	10.41	35.63	0.77
	4	53.13	23.32	47.19	10.09	43.44	16.59	42.19	14.45	40.00	12.18	36.88	11.73	34.38	0.77