DEGREE PROJECT

Detecting and identifying radio jamming attacks in low-power wireless sensor networks

John Kanwar

Information Security, master's level (120 credits)
2021

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

Abstract

Wireless sensor networks (WSNs) are used in all kinds of different sectors ranging from agriculture, environment, healthcare and the military. Embedded systems such as sensor nodes are low-power and consist of low memory, which creates a challenge for its security. One of WSN's worst enemies is interference radio jamming attacks. They are easy to construct and execute, but hard to detect and identify.

In this thesis, we tackle the problems of detecting, but most importantly identifying, and distinguishing the most commonly used interference radio jamming attacks. Presenting SpeckSense++, a firmware that makes it possible for low-power embedded systems to detect, identify and distinguish interference radio jamming attacks and unintentional interference such as Bluetooth and WiFi to a certain degree. Showing an accuracy of 96 to 90 % for proactive jammers, 89 % for reactive, and 92 to 85 % for unintentional interference.

Acknowledgements

I would like to thank my supervisors at LTU Christer Åhlund and Saguna Saguna for the advice I have got under the course, the guidance of the thesis, and for the support. I would also like to thank my industrial supervisors Thiemo Voigt and Joakim Eriksson at RISE for all the feedback, discussions, tips, and guidance I have got under the way. Thanks to the STACK group meeting with Nicklas Finne, Saptarshi Hazra, Joel Höglund, and Nicolas Tsiftes for valuable feedback.
Thanks to Anastasios Agrianidis for valuable information during the opposition, which helped to improve this master's thesis.
Last but not least do I thank Lina Näsström for the overwhelming support.

Contents

1 Introduction

The World Wide Web became public in 1989 and thus a new era was born, the Internet era. The Internet has given a lot to mankind and accelerated civilization's knowledge with rocket speed. It has had a great impact on development as it lets people around the world communicate and share knowledge. As early as the beginning of the 90s when the Internet started have people tried to connect things to the Internet for different reasons. The famous Trojan room coffee pot was a coffee machine that was seated at the University of Cambridge in a computer laboratory. To prevent people from walking up to the coffee pot and their disappointment find it empty, was a camera installed with the purpose to broadcast the coffee machine and send the images to the office networks.[9] John Romkey connected a toaster to the Internet that allowed the toaster to start and stop through the Internet.[25] This is considered the first "device" connected to the Internet. Kevin Ashton coined the term "Internet of things" when he was working at Procter&Gamble in 1999[1], and after that did multiple companies start to launch smart devices. Today we have everything from smart fridges that are considered the internet of things (IoT) to robot vacuum cleaners. Everything to make the everyday quality easier and more enjoyable.
IoT devices are also used in other more critical areas as well, from everything between health care to saving the environment. This leads to the devices needing protection. For example, a wireless sensor network that operates outdoors to gather data from the environment does not only lack physical security as they are most of the time unattended. The embedded system's devices do also generally have little space for memory, lack of battery power, computing resources, and thus lack technical controls that other more robust systems could contain, such as heavy encryption algorithms.[2]
Embedded systems devices that are part of a wireless sensor network run the risk of getting attacked by radio jamming attacks.
This thesis will tackle the problem of radio jamming attacks and contribute to research about detecting and identifying radio jamming interference in low-power wireless sensor networks.

1.1 Radio jamming interference

Over the years have multiple types of radio jamming emerged, some being more advanced than others. Radio jamming does share a common goal, which is to interrupt the signal transmission of the target. In wireless sensor networks would it be to interrupt the radio frequencies used by the nodes in the network and by doing so jam a specific node or multiple nodes.
There are more or less unlimited amounts of radio jamming attacks out there, but the most common jammers that multiple papers have stated as the most effective and popular are the constant, deceptive, random, and reactive jammers. [32][19][10][21][35] There are a lot of different types of jamming attacks and the tweaking one can apply to modify a jammer is endless, therefore it would be impractical to list all different jammers in this thesis. These four are the

main jammers that are being used and they are the four that are proven to be effective.[38] In this thesis, we are going to focus on detecting and identify these four jamming attacks.

1.2 Aim of the report

As wireless sensor networks are increasing at a rapid rate is it most likely that attacks towards wireless sensor networks will increase as well. This report aims to detect, identify and classify the most common radio interference attacks. The thesis work created is built upon SpeckSense [14] an interference detector capable of distinguishing between different types of network technologies. The purpose of this thesis is to contribute to attack detection and identification by performance and interference monitoring. It can also be used as a foundation in order to avoid specific interference radio jamming attacks.
This project is commissioned by the company Research Institutes of Sweden (RISE), to contribute to the Secure Interoperable Full-Stack Internet of Things for Smart Home projects.
This thesis seeks to answer the question:

- How can we detect and identify radio jamming attacks in low-power wireless sensor networks?

The aim of the report is to:

- Detect radio jamming interference attacks.
- Identify and distinguish between radio jamming interference attacks.
- Implement different radio jamming interference attacks.

1.3 Gap & and contribution

Most research focuses on the possibility to detect radio jamming attacks, but not on identifying and distinguishing the radio jamming attacks. By being able to identify the radio jamming attack does it allow applying specific countermeasures towards the different jamming attacks. Getting the knowledge about what type of jamming attack does also generate data that can be used for other things as well, such as forensics.
This thesis contributes with a solution that makes it not only possible to detect radio jamming attacks in low-power wireless sensor networks but also the ability to identify them.

1.4 Scope

There are more or less an infinite amount of different types of jamming attacks, as every jamming attack can be customized and tweaked. Therefore, are the scope on identifying the four most commonly used jamming attacks.

1.5 Research methodology

This thesis uses a quantitative approach. The research method is carried out with multiple experiments, which purpose is to test the possibility to detect, identify and distinguish unintentional interference and intentional interference such as radio jamming attacks. Weekly meetings have been held with supervisors both at RISE and LTU in order to update the current status of the work, and discuss if any problems or major obstacles have occurred. The work cycle has mostly consisted of researching, discussion and implementation.
The literature review carried out for this thesis uses the Webster and Watson [36] approach. The reason for that is to get a solid base of knowledge that is modern. It is proactive research that contributes with a solution to identify the four most commonly used jamming attacks in real-time.

1.6 Outline

Chapter 2 *Related work* goes through how the objective has been tackled before and what type of ways have been used in the past. Chapter 3 *Theory* describes a basic understanding of the different algorithms and tools that are used in this thesis and gives an overview of how SpeckSense++ is assembled.
Chapter 4 *Method* goes through in detail how the different algorithms and tools from the *theory* chapter were implemented and how the experiments were carried out.
Chapter 5 *Results* goes through the result of what the experiment yields.
Chapter 6 *Analyses and discussion* discusses the different approaches made and reasons for pros and cons of what did go right, and what did go wrong.
Chapter 7 *Conclusion* reasons for future work and concludes the whole thesis.

2 Related work

2.1 Methods for detection

Intrusion is defined as: "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource".[41] With that definition can an intrusion detection system be very broad as it points to any of the elements in the Confidentiality, Integrity and Availability (CIA) triad. In this thesis is the intrusion detection system focused on the availability trait, more specifically radio interference jamming.
It is possible to categorize intrusion detection systems (IDS) into three different categories. Anomaly-based detection, misuse-based detection, and specification-based detection.

2.1.1 Misuse-based detection

Misuse-based detection is more or less rule-based like a firewall. For example, to prevent a brute force attack is it possible to insert the rule "*after 5 wrongfully password inputs is the user going to get a timeout*". The advantages of implementing a misuse-based detection on a WSN would be that it would be great at detecting hard-coded patterns. This would also be its Achilles heel, it wouldn't be able to detect other attacks that are not profiled.

2.1.2 Specification-based detection

Specification-based detection contains a set of constraints and specifications that characterize the programs, or protocol's right operations. The specification and constraints do then make sure that the execution of the program is monitored. It is a combination of misuse & anomaly-based detection in the sense that misuse-based detection tries to recognize bad behaviour that is known, and anomaly-based detection tries to detect the effect of the bad behaviour.[31] The disadvantage is that it is very time-consuming.

2.1.3 Anomaly-based detection

Anomaly-based detection is based on statistical behaviour modelling. The concept is to compare the normal operations of members and a certain amount of deviation is going to be flagged as an anomaly in case it deviates from the normal behaviour. The advantage of this model is that it tends to get a high score on detection accuracy, and it is also very consistent as longs as the static behavioural patterns are followed.[15] It is also good at detecting attack patterns that haven't been discovered yet. The disadvantage is that it is important to update the normal profiles periodically. The network behaviour may change rapidly, and it is, therefore, a must.

Anomaly-based detection is possible to break down into three subcategories.
Statistical-based, Knowledge (Data mining-based), and Machine learning-based.

[4]. Figure *1* shows an overview of the whole sub-category tree.

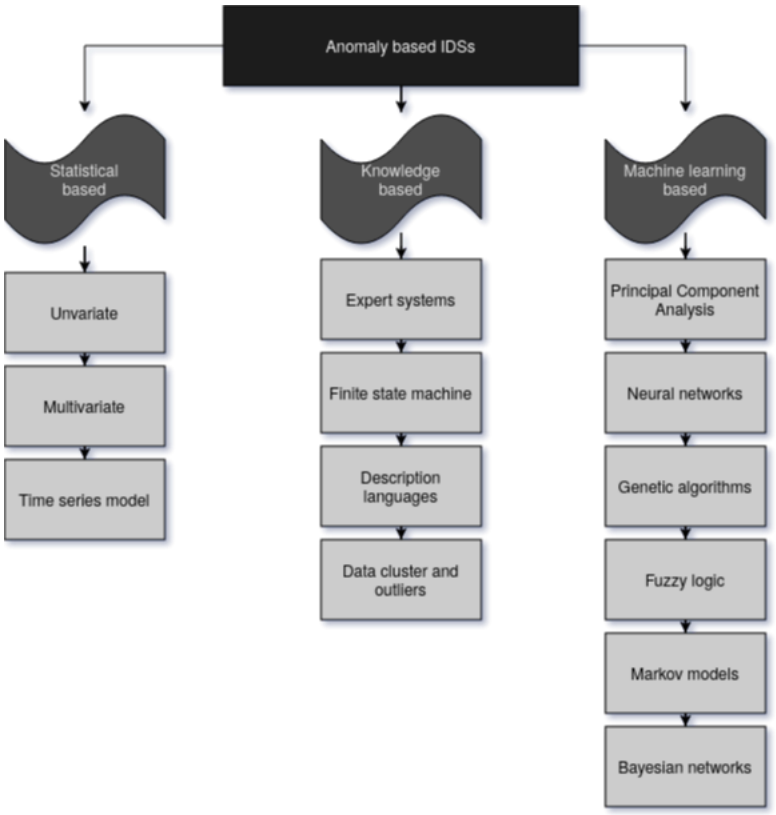

Figure 1: Anomaly based IDSs approaches

Statistical-based is when a data package is captured and then a profile representing its stochastic behaviour is generated. A reference profile is created, and the network is monitored. By comparing the reference profile against the anomaly score that is generated periodically is it possible to discover unwanted packages in case the score is higher or equal to a specific threshold. There are three different statistical-based sub-categories, *univariate*, *multivariate*, and *time series model* [4].

Knowledge-based anomaly IDSs use previous data and knowledge to detect. For example are *data clustering and outlier detection*, *finite state machine*, *description languages*, and *expert systems* knowledge-based methods.[4] It is possible to say that knowledge-based is more or less data mining based.

Machine learning-based anomaly is based on machine learning. A pattern is

generated that is either implicit or explicit, and this prevents attacks later on. They can be updated regularly with data in order to grow and keep updated. For example are *principal component analysis*, *neural networks*, *genetic algorithms*, *fuzzy logic*, *Markov models*, and *Bayesian networks* machine learning-based. [4]

2.2 Related approaches

Colin C. Murphy and CO [22] uses machine learning to classify and detect intrusion in WSNs. They simulated ZigBee transmission in a Monte Carlo MatLab simulator to show the benefits of machine learning and the use of quadrature signals samples to prove their cause. The machine learning technique used was the random forest, which is a supervised machine learning algorithm. Random forest algorithms consist of different decision trees, which all have the same nodes. Different data will however lead to different leaves in the tree. To produce a result does it merge multiple tree's decisions, which in the end represents the average. It is however only simulated, and the authors say that for it to work on low-power WSN edge devices does it need to be optimized.

Jamming on the Internet of Things: A Game-Theoretic Perspective [20] looks at how intended interference such as DDoS, radio jamming, and side-channel attacks could be solved from a game-theoretic perspective. The authors propose an anti-jamming technique for Orthogonal frequency-division multiplexing (OFDM)-based IoT systems that protects IoT devices against malicious attacks by enabling an IoT controller. The game theory approach uses the Colonel Blotto game [24] with asymmetric resources, which interacts between the jammer and the controller node. The defender in this game is the IoT controller that tries to prevent a malicious jamming attack. It does that by dispenses the power in the network between the subcarriers. By smartly doing that does it allow to decrease the aggregation error bit rate that the jammer causes. The goal of the attacker (the jammer) is to interfere with the IoT network by allocating power in the frequency band. Each player (the attacker and the defender) explores the maximal payoff by selecting a randomized strategy to allocate its power to the subcarriers.

The authors solve this game by using evolutionary and genetic algorithms. By the combination of the Nash equilibrium and the Blotto game does it find a diverse strategy that is successful in maintaining the bit error rate over the acceptable threshold, which results in a protected network.
The authors of Jamming Mitigation via Hierarchical Security Game for IoT Communications [33] use a similar approach to solve the problem. They implement a hierarchical game that is specifically used to mitigate reactive jamming attacks.

Sensor Network Interference Classification (SoNIC) [12] is a system that enables resource-limited embedded devices to detect interference and classify that interference. It operates on the 2.4 GHz frequency band and aims to identify

cross-technology interference such as WiFi, Bluetooth devices, microwaves, etc. It doesn't actively sample the spectrum for values to detect interference but does instead spectate corrupted 802.15.4 packets that the nodes in the network have received. Corrupted 802.15.4 packets could be packets where the payload's checksum doesn't match.

The Technology-Independent Interference Mitigation solution (TIIM) makes it possible to detect, quantify and react to cross-technology unintentional interference. [13] It uses two different approaches in order to detect the interference. Capturing the variation of the energy by the use of reading the RSSI values and an alternative approach which is to exploit the use of per-packet channel metrics are calculated in radio that is off the shelf.

In the paper *Detection and Classification of Radio Frequency Jamming Attacks using Machine learning* [16] do the authors use different machine learning algorithms such as k-nearest neighbours, decision tree, gradient boosting and random forest to classify different radio jamming attacks. In order to detect the attacks do they measure the received signal strength and the packet delivery ratio. They use the simulator ns-3 to simulate jamming attacks, collect the data which consists of packet delivery ratio and received signal strength. The data is then labelled with the different jamming attacks, preprocessed, divided into a training set and testing set, and lastly are the algorithms trained.

2.2.1 Similarities

What most researchers have in common when detecting and mitigating interference in IoT devices that are energy-based are two things. 1. They use signal features that are extracted from RSSI samples. 2. A classification method is used to classify the methods, which is either a technology-specific (heuristic techniques) threshold [17] [5] [40] or machine learning [14] [8]. And to store more data on memory, different compression algorithms are used.

3 Theory

3.1 SpeckSense

SpeckSense is an algorithm that makes it possible to distinguish between different types of unintentional interference in the 2.4 GHz spectrum with the help of machine learning.[14] From previous research that targets interference from selected sources does SpeckSense contribute with solutions to mitigate and avoid interference from multiple sources. The algorithm consists of three major elements, being the RSSI-sampler, interference detection, and interference classification.

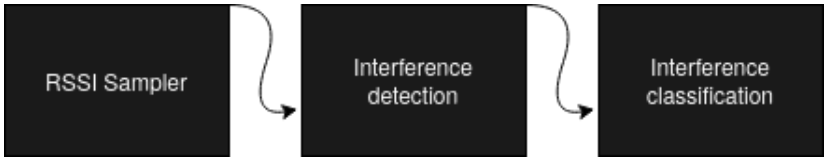

Figure 2: Speck Sense workflow

3.1.1 SpeckSense - RSSI-sampler

The RSSI-sampler does three things. It reads in the different rssi-samples from the air, quantizes those values and run-length encode them. The first reason is to save memory as storing raw values takes a lot of memory, and the second being that it removes slight variations in emissions as slight variations aren't relevant.
Run-length-encoding(RLE) is a form of compressing data. Instead of storing multiple identical rssi-values in a row, does the RLE compress the data and bundle together the values that are identical in a sequence and call that duration. So for example, after quantization and an RLE of the RSSI values -95 - 95 -95 -92 - 87 would it create 2d-vectors that could look like this: (3,3) (3,1) (2,1). Where the first value is the power level and the second value is the duration, which means multiple identical rssi-values in a row over time.
After the quantization and RLE are bursts extracted from the 2d-vectors. A burst is a continual subsequence where the channel activity is active, meaning that it is not idle. The power level needs to be greater than one. A burst is created by taking the weighted mean power level and the total duration of the subsequence. A burst is divided when the channel stops being idle. For example, the 2d-vectors (3,3) (3,1) (2,1) would be:

$$((3 \times 3 + 3 \times 1 + 2 \times 1) \div (3 + 1 + 1), (3 + 1 + 1))$$

3.1.2 SpeckSense - Interference detection and classification

The interference detection uses the unsupervised machine learning algorithm K-means clustering, which clusters on the features *power level* and *duration* that are extracted from the bursts. The K is not known beforehand as it is impossible to know the number of interference sources that are in the air. The K is therefore calculated in real-time and is increased in case if the cost difference at termination is less than the empirical chosen value of 0.001.
Interference classification is done by comparing the numbers of detected interference clusters and the average interval between burst separation with a predetermined threshold. By doing so is it possible to detect if there is interference on the different channels and by comparing it with pre-determined thresholds is it possible to distinguish and identify what type of interference it is. Based on the number of interference clusters that are detected and the average inter-burst separation is it possible for SpeckSense to distinguish between different channel activity.

3.2 SpeckSense++

SpeckSense++ makes it possible to detect radio jamming attacks in low-power wireless sensor networks as well as unintentional interference to a certain degree. It is built upon SpeckSense and consists of the same core structure which is the RSSI-sampler, interference detection and interference classification. The RSSI-sampler is more or less the same as SpeckSense except that it doesn't sample RSSI values that are very low, and it doesn't create bursts. The detection part uses the same K-means algorithm that decides the K in real-time, but they still differ.

3.2.1 SpeckSense++ - Interference detection

The 2D vectors that are creating from the RSSI-sampler are directly sent to the K-means algorithm. The K-means clustering algorithm clusters equally to what the amount of K represents. It decides the K in real-time as the original SpeckSense does. The K is iteratively increased depending on if the cost difference at termination is less than a specific value. The K-means cluster algorithm uses the weighted Euclidean distance algorithm to calculate the distances between the 2D vectors and the centroids. It is a weighted Euclidean distance algorithm towards the power level, meaning that it emphasizes the power level. This makes it more likely to create centroids around 2D vectors that have a high power level.

3.2.2 SpeckSense++ - Interference classification

To classify the different jamming attacks is a *cluster sampler* process executed. After the K-means cluster algorithm has been executed and has created several clusters are the clusters sorted after a duration and compared towards the jamming attacks pre-determined thresholds. If a cluster's attributes are inside a pre-determined threshold is that cluster saved as a cluster sample. When five cluster samples have been collected under a certain amount of time is the average calculated from those cluster samples. The difference is then calculated with every sample that has been temporary stored by taking the average subtracted with every cluster sample. If the difference is in of bounds with a certain threshold resembling that the samples values are close enough to each other, is it most likely that type of jamming attack the cluster sample represents. If it is out of bounds is it most likely a random jammer, as random jammers are inconsistent. If five samples aren't collected under a certain time and a reactive SFD alert has been trigger is it most likely a reactive SFD jamming attack.

$$A = \frac{1}{n} \sum_{i=1}^{n} a_i \tag{1}$$

$$a \in cs : |A - a| \geq p \tag{2}$$

3.3 K-means clustering

K-means clustering is an unsupervised machine learning algorithm that clusters data points into groups that are similar. Unsupervised means that it aims to discover patterns in the data it is given related to its features and doesn't need to map the input to output, unlike supervised learning. The K in the K-means clustering determines how many clusters are going to be created, and a common way to determine it is by the use of the elbow method. [3] The first step of a K-means algorithm is to take in features that will be the data points. Centroids equal to K are placed among the data points. Every point is then assigned to the centroid it is closest to which will form a collection of points which is called a *cluster*. Each cluster does contain one centroid which will be updated depending on the cluster's assigned points. The steps will be repeated until no points changes cluster or the centroids don't change. The algorithm can be expressed as follows:

$$J = \sum_{j=1}^{k} \sum_{n \in S_j} |x_n - \mu_j|^2$$

where x_n is a vector of a dimension of the nth data point. μ_j is the centroids in relation to the geometry of the data points in S_j. K-means clustering clusters n data points into k clusters.

3.3.1 Euclidean distance

The Euclidean distance is an algorithm that calculates the distance between two different points in n-dimension space.The algorithm can be expressed as follows:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(x_i - p_i)^2}$$

where p and q are two points in Euclidean n-space. n is the n-space and q_i and p_i are the initial points, starting from the origin of the space are the Euclidean vectors.

3.4 Run length encoding

Run-length-encoding (RLE) is a form of a lossless data compression algorithm. It takes the sequence of identical data values, compresses and stores it as one value and a count instead. For example, the string AAAABBBCC would be run-length encoded into 2d vectors of (4,A), (3,B) and (2C).

3.5 Quantization

Quantization is the art of mapping continuous values that are infinite to a smaller set of values that are discrete finite values. All values in a range gets the same value. For example,

$$[a..b] \Rightarrow x \in \mathbb{R} : a \leq x \leq b$$

3.6 Direct test mode

Direct test mode (DTM)[28] is a mode that gives access to the operation of the radio that is on the physical level. For example, the power and sensitivity of the transmission and receiver, modulation characteristics, the offset of frequency, drift, intermodulation performance and packet error rate.

3.7 Proactive jammer

A proactive jammer is a jammer that is proactive meaning that rather than respond to a situation does it control it. A proactive jammer sends out interference signals regardless of the data communication that happens in the network, meaning ongoing, intermittent, or inactive. It transmits random or specific chosen bits on a chosen channel and occupies it.

3.7.1 Constant jammer

A constant jammer sends out a continuous flow of radio signals in the wireless medium. It transmits random bits on a chosen channel and doesn't take into consideration any rules of the MAC protocol.[37]

3.7.2 Deceptive jammer

A deceptive jammer is similar to a constant jammer as it sends out a continuous flow of radio signals. It doesn't however transmit random bits and does instead transmit packets. The deceptive jammer aims to send out regular packets in a constant flow so the target gets stuck in receiving mode and not being able to do anything else.

3.7.3 Random jammer

Random jammer is alternating between two different states, either sleeping or sending. It emits radio signals for a t amount of time and does then sleep for a t amount of time. By doing so does it save energy and become harder to detect. It can either be a constant fixed value or random and the packages sent can be either random bits or regular packages. [37]

3.8 Reactive jammer

A reactive jammer is a jammer that is reactive meaning that it reacts to its environment before it takes action. A reactive jammer starts transmitting radio signals when it senses activity on a channel that nodes are using. When it doesn't sense any activity does it not send any data. By doing so does the jammer target the reception of a message and makes it harder to detect.

3.9 Start-of-frame delimiter jammer

The beginning of an Ethernet frame consists of a preamble and a start-of-frame delimiter(SFD). The purpose of the preamble and SFD is to keep synchronization between devices that are communicating. The first bytes that the preamble and the SFD constitute are used to alert receiving nodes that there is something they can use.
An SFD interference jammer exploits that knowledge by only interfere when it recognizes an SFD.[11] By doing so does it not only keep itself more concealed to the environment by only attacking messages that are being transmitted but do also saves energy by not sending out jamming on random noise. The SFD interference jammer consists of two different states, a jamming state, and a hidden state. When it notices an SFD packet on the channel, it does send out an interference packet to corrupt the payload of the packet that is being transmitted to the receiving node. This results in the packet get dropped by the receiving node and the communication being interfered with. Figure *3* shows how it looks when the receiving packet's payload gets corrupted.

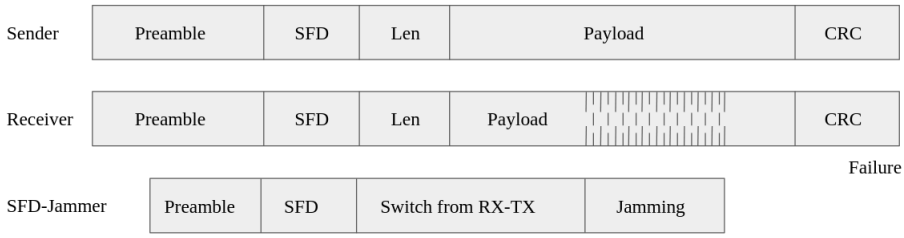

Figure 3: SFD-jammer in action corrupting packet mid-air

3.10 JamLab-NG

JamLab-NG is an open-source framework that can create deterministic WiFi interference. Allowing for customizable parameters such as transmission speed and packet length. It does also allow for WiFi interference patterns that are repeatable. [26]

3.11 Honeypot

Honeypots are tools that use deception in order to lure or bait an attacker.[18] It is often used in networking in a way to intentionally get attackers to attack a specific target, that intentionally has vulnerabilities in order to monitor the behaviour and gather the information that can help for future defences. [39]

3.12 Contiki-NG

Contiki next generation (Contiki-NG) is a newly branched version of Contiki which was originally developed by Adam Dunkels.[7] Contiki-NG is a lightweight, open-source, cross-platform operating system for IoT devices. Its focus is on memory-constrained systems, and low-power wireless embedded systems. Contiki-NG contributes to a variety of low-power standard protocols and communication. Some of those being RPL, CoAP Ipv6 over 6LoWPAN, and 6TiSCH.

3.12.1 uIP stack

The uIP (micro IP) is an open-source implementation of the TCP/IP network protocol stack, intended for small microcontrollers that are 8 and 16 bit. [6] It fits well within the limits of an 8-bit microcontroller. The setbacks are that the throughput is decreased. The TCP/IP mechanisms are however all there. The interface between the TCP/IP stack and the application is reduced compared to a normal-sized TCP/IP stack.[6]

3.12.2 ContikiRPL

Routing Protocol for Low-Power and Lossy Networks (RPL) is a routing protocol for WSN. It is generally susceptible to packet loss and has low power consumption. RPL operates on IEEE 802.15.4 and is based on distance vectors. It supports multi-hop, many-to-one communication, and one-to-one messages. ContikiRPL is an implementation of the RPL inside the uIPv6 stack. [34] It is both power efficient and lightweight.

3.13 Summary

Figure *4* and *5* shows how the different theory parts are connected and how they work together. The figures resemble how the different parts are built upon each other. In figure *4* is Contiki-NG the biggest box, everything is built inside Contiki-NG. SpeckSense++ is built upon original SpeckSense whose core are K-means clustering, quantization and run length encoding.

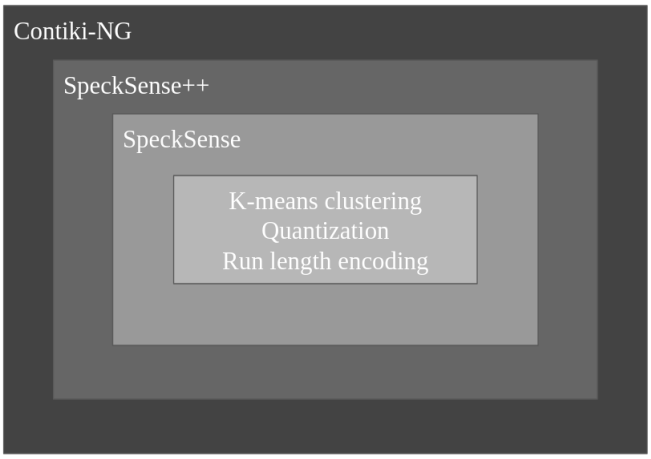

Figure 4: Summary SpeckSense++

Figure *5* shows that everything inside it is implemented in Contiki-NG. Direct test mode is used in order to modify the physical layer which is something the proactive and reactive jammers are using.
Table *1*, describes the essentials of the different WSN layers.

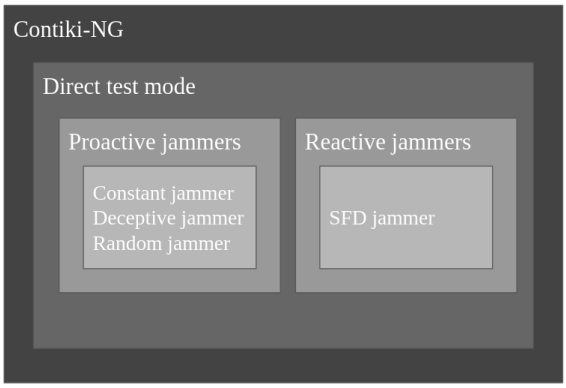

Figure 5: Summary jammers

Application layer	The application layer focuses on how data are provided and requested with individual sensor nodes and communication with an end user.
Transport layer	The transport layer focuses on data encryption and is liable for stable transport of packets. Transport protocols such as TCP and UDP belongs here.
Network layer	The network layer is in charge of how packets are forwarded and liable for assigning addresses.
Data-Link layer	The data-link layer is about multiplexing data streams, medium access control(MAC), data frame detection, error control and data encryption. Link-layer jamming is occurring in this layer and causing damage. To be more specific is it the MAC that are undergoing the most heavy damage. [6]
Physical layer	The physical layer is responsible for frequency selection, signal deflection, carrier frequency generation and modulation. The physical layer is the layer that undergo the most damage from radio jamming attacks. [6]

Table 1: WSN protocol Stack

4 Method

4.1 Design and Implementation

The design and implementation is about how SpeckSense++ is implemented and shows how it uses all the different parts from the theory chapter as its core mechanisms. Figure *6* shows an overview of the workflow of SpeckSense++, how it operates to detect, identify and distinguish different interference sources.

4.1.1 RSSI-sampler

The RSSI-sampler listens to a specifically chosen channel and samples the RSSI. The RSSI values are first offset with -45 dBm. The amount of samples varies and depends on identical signals in a row. For example, the max samples are set to a maximum value of 500, but if multiple identical signals will be read in a row, will they be run-length encoded and counted as one value. The max duration is set to 1000, so in theory, could the maximum RSSI values read be 500000.
From the RSSI values are two variables calculated, power level and duration. The power level depends on the strength of the RSSI. The RSSI is quantized to a power level between 1-36. The range between the power levels is increased by four. For example, if the dBm of the RSSI x is between a value:
$[a..b] \Rightarrow x \in \mathbb{R} : a \leq x \leq b$ where a and b will be $a+4$ and $b+4$ until 36 different power levels are represented. The duration is calculated by run-length encoding the identical signals that have the same power level in a row calculated with the duration one RSSI signal takes to receive which is 0.0052ms. The RSSI-sampler reads in values in a frequency of 26kHz. When the RSSI-sampler is done does it send the created 2D vectors that consist of power level and duration to the K-means clustering algorithm. Algorithm *1* shows the pseudocode of the RSSI-sampler.

Algorithm 1 RSSI-sampler

```
1: while samples_counter ≤ MAX_SAMPLES do
2:     Get RSSI value
3:     Offset RSSI value with -45
4:     Quantisize the RSSI value into power level
5:     Create 2d vector with power level and duration
6:     if Power level not identical to previous power level OR duration ≥ MAX_DURATION
   then
7:         Record next vector
8:         Increase samples_counter
9:     end if
10:    Increase vector duration
11:    Store vector
12: end while
```

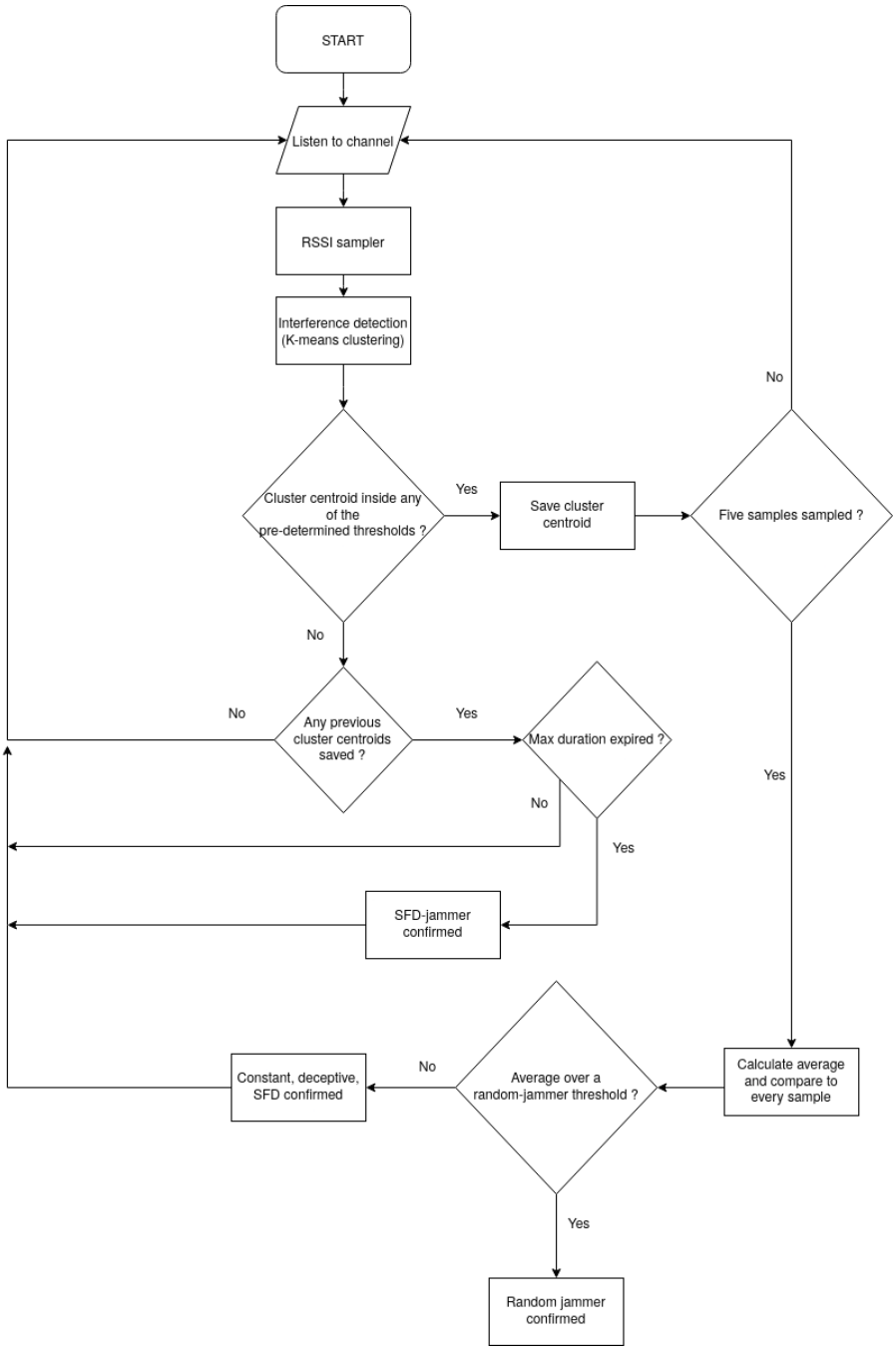

Figure 6: Flowchart interference identifier

4.1.2 K-means clustering

In order to identify the different interference on a channel is the K-means clustering algorithm implemented. Its purposes are to cluster together with the 2D vectors that seem to come from the same interferer.
The K-means cluster algorithm takes the 500 2D vectors created by the RSSI-sampler and at first, randomly initialize centroids from the 2D vectors. Each 2D vector will be assigned to the nearest cluster centroid, which will result in k amount of clusters. The K-means algorithm will recalculate the positions of the centroids until a condition is met.
When the algorithm recalculates the positions of the centroids does it use the Euclidean distance algorithm to assign the clusters. The clusters centres are updated by recalculating the average of the samples. The Euclidean distance algorithm is weighted with a value of 5000 towards the power level in order to emphasize and give more weight to 2D vectors that has a short duration and high power level.
Cluster assignment and the update of the centroids are repeated until termination is met. The termination is either that the cost of calculating the samples close to the centroids of their clusters is below a threshold of 0.001 which has been empirically decided, basically that running the algorithm further doesn't increase the accuracy of the clusters. The other termination threshold is if the numbers of k exceed the maximum of ks. The k in this algorithm is not known in advance as there can be a different amount of interference sources. The k is therefore calculated in real-time and is tested with different values of k. k starts at a value of 1 and is increased by one as long as the cost of difference of the algorithm less than 0.001. Algorithm *2* shows the pseudocode of the K-mean clustering algorithm.

Algorithm 2 K-means clustering

```
1: while clusters < MAX_CLUSTERS AND difference cost between cluster's centroids >
   STOP_THRESHOLD do
2:     Initialize centroids to random points in 2d vectors
3:     while cost difference > STOP_THRESHOLD OR iter < maximum_duration  do
4:         For each sample point in samples, connect it to the nearest centroid.
5:         Recalculate the cost difference
6:     end while
7: end while
```

4.1.3 Identification

The clusters are sorted by duration, inspected one by one, and compared against different pre-determined thresholds that are profiled earlier. If a cluster is in the range of a pre-determined threshold that is an interference jamming attack does two things happen. The number of samples variable is increased by one, and a suspicious variable is increased by two. The values of the cluster are temporarily

stored as well. If one cluster has shown signs of jamming are no further clusters investigated. In order to detect that jamming attack does SpeckSense++ take one cluster sample from every run the firmware makes. One run would be one cycle of listening to the air, sample 500 samples, prepare it with RLE and quantization, run the K-means clustering and get the clusters. If one run would yield non-suspicious clusters would it increase the suspicious variable by one. When the suspicious variable is above or equal to 10 does the firmware confirm what type of jamming attacks it was by taking the average of the calculated cluster-samples. Calculate the difference by taking the average subtracted with every cluster sample. If the summary is out of bounds with a pre-determined threshold is it ruled as a random jammer. If it isn't out of bounds is it either the reactive SFD jamming attack, proactive deceptive jammer or the proactive constant jamming attack depending on its features. The unintentional interference, i.e Bluetooth and WiFi, do not go through this extra step of detection and is confirmed directly if a cluster is in range of a pre-determined threshold that is from an unintentional interference source.
Figure *7* shows how it could look when a proactive constant jammer is identified. Five different samples have been stored. The average will be calculated, and the difference between every element and the average will be compared to a threshold that measures if the five samples are close enough to each other. Figure *8* shows how it would look if, for example, a random jammer would jam. The different samples would differ enough and over exceed the threshold which would classify it as a random jammer. Algorithm *3* and *4* shows the identification part.

999	1000	1000	1000	967

Figure 7: Five different samples of a proactive constant jammer (duration before converted to a time prefix)

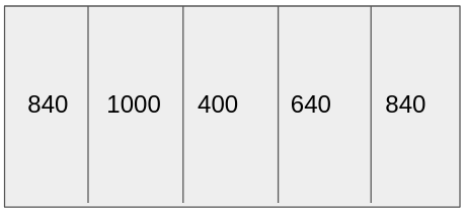

Figure 8: Five different samples of different durations (duration before converted to a time prefix)

Algorithm 3 Identification (1)

```
1: for Every cluster recently created do
2:     Sort cluster after duration
3:     if clusters duration ≥ PROACTIVE_DURATION AND
4:         clusters power level ≥ POWER_LEVEL_THRESHOLD then
5:                 Increase number of suspicious samples
6:                 Store duration into suspicious array
7:                 Proactive constant jammer notified
8:                 Break
9:     end if
10:    if clusters duration ≥ REACTIVE_DURATION_MIN AND
11:        clusters duration ≥ REACTIVE_DURATION_MAX AND
12:        clusters power level ≥ POWER_LEVEL_THRESHOLD then
13:                Increase number of suspicious samples
14:                Store duration into suspicious array
15:                Reactive SFD jammer notified
16:                Break
17:    end if
18:    if (clusters duration ≥ BLUETOOTH_DURATION_MIN AND
19:        clusters duration ≤ BLUETOOTH_DURATION_MIN) AND
20:        (clusters power level ≥ BLUETOOTH_POWER_LEVEL_MAX AND
21:        clusters power level ≤ BLUETOOTH_POWER_LEVEL_MIN) then
22:                Alarm Bluetooth interference
23:                Break
24:    end if
25:    if (clusters duration ≥ WIFI_DURATION_MIN AND
26:        clusters duration ≤ WIFI_DURATION_MIN) AND
27:        (clusters power level ≥ WIFI_POWER_LEVEL_MAX AND
28:        clusters power level ≤ WIFI_POWER_LEVEL_MIN) then
29:                Alarm WiFi interference
30:                Break
31:    end if
32: end for
```

Algorithm 4 Identification (2)

```
1: if Number of suspicious samples ≥ INTERFERENCE_NR_SAMPLES then
2:     Calculate average of suspicious array
3:     for Every element in suspicious array do
4:         Calculate difference between element and average
5:         if difference between element and average ≥ RANDOM_DURATION then
6:             Alarm proactive random jammer attack
7:             Break
8:         end if
9:     end for
10:    if Proactive_constant_jammer is notified then
11:        Alarm proactive constant jammer attack
12:    end if
13:    if Reactive_SFD_jammer is notified then
14:        Alarm Reactive SFD jamming attack
15:    end if
16: end if
```

4.1.4 RPL-UDP & SpeckSense++

When using the combination of RPL-UDP and SpeckSense++ are two different Contiki processes used, one being a basic RPL-UDP communication and the other process being SpeckSense++. If the node running RPL-UDP fails to communicate with another node does it suspect a radio jamming attack, closes the RPL-UDP process and starts SpeckSense++ for 20 seconds. SpeckSense++ does then start and tries to detect and identify the interference that is happening. When it detects and identifies the interference does it alert what type of interference it is, and it is possible to apply different specific countermeasure in order to avoid the attack.
If nothing is identified is it a possibility that it is a reactive SFD-jamming attack that is occurring. In that case does one node in the network whose whole purpose is to scan the air for attacks start to act as a honey pot to trigger the potential SFD-reactive jammer. This will make it possible for the other nodes affected by the SFD-reactive jammer to identify the jammer as the SFD-reactive jammer will get triggered by the packets that are being sent out.

4.1.5 Proactive constant jammer

The proactive constant jammer uses direct test mode to send out random bit patterns. The jammer consists of two states, jamming, and not jamming. It can choose which channel to interfere with, the length of the packets, TX power size, and packet type. It creates interference when its signal is higher than the signal-to-noise ratio of the node's communication.

Constant jammer Settings	
TX-power:	-16dBm or 8dBm
Jam time:	Constant
Packet length	254
Bit pattern	PRBS9

4.1.6 Proactive Deceptive jammer

The implementation of the deceptive jammer is equal to the constant jammer. But instead of running direct test mode, does it send out packets that consist of a preamble and SFD. It has two states, either jamming or not jamming. It can choose which channel to interfere with, the length of the packets and TX power size.

Deceptive jammer Settings	
TX-power:	-16dBm or 8dBm
Jam time:	Constant
Packet length:	254 bytes
Bit pattern:	Preamble+SFD

4.1.7 Proactive random jammer

The proactive random jammer is implemented similarly to the constant jammer. The difference is that it alters between two different states for n amount of time, where n is random and between a range of 0.2 and 2 seconds. One state being an interference state and the other being a hiding state. Under the hiding, states does the jammer not interfere and does instead conceal itself. It has the ability such as the constant jammer to choose which channel to send packets on, the length of the packets, TX power size, and packet type. It creates interference when its signal is higher than the signal-to-noise ratio of the node's communication.

Random jammer Settings	
TX-power:	-16dBm or 8dBm
Jam time:	0.2 to 2 seconds
Packet length:	254 bytes
Bit pattern:	PRBS9

4.1.8 Reactive SFD jammer

One important aspect of the reactive SFD jammer is speed. Being fast enough to go from a receiving state to a transmission state to corrupt the packets that are being sent through the air. To reach those heights does the node execute the interrupt service routine and call an interrupt directly when it sniffs an SFD

from the air. When the interrupt is called does the jamming start immediately and a real-time task timer (rtimer) is set to 312 microseconds. The jammer jams for 312 microseconds until the rtimer is executed, and it goes back to a receiver state. Under those 312 microseconds does it send out 39 bytes/312 bits.

SFD jammer Settings	
TX-power:	-16dBm or 8dBm
Jam time:	312 μs
Packet length	39 bytes
Bit pattern	PRBS9

4.2 Experimental setup

The hardware used in this project is five different nRf52840-dongles, one nRF52840-DK Development kit, and a Raspberry Pi 3 model B as can be shown in figure *9*, figure *10* and figure *11*.

Five different experiments are carried out in order to test the capabilities of SpeckSense++. The first experiment tests the possibility of detecting, identifying and distinguishing different types of radio jamming interference attacks. The second experiment is carried out to prove the same things but in a more "real-life" scenario. It is done in order to test how this work can be used as a foundation in order to apply specific countermeasures towards specific jamming attacks.
Two other experiments are also carried out to test the ability to detect and identify unintentional interference such as Bluetooth and WiFi. The last experiment tests the possibility of identifying multiple jammers simultaneously.
The answers that are sought can be asked as following:

- Is it possible to detect, identify and distinguish different radio jamming attacks ?
- Is it possible to use this thesis work as a foundation in order to apply specific countermeasures towards specific radio jamming attacks ?
- Is it possible to identify unintentional interference such as Bluetooth and WiFi
- Is it possible to detect, identify and distinguish different radio jamming attacks that are occurring simultaneously ?

Table 2, shows the values of the different variables that are used when the experiments are carried out.

SpeckSense++ variables	
Amount of rssi samples:	500
Max duration:	26 ms
Max amount of clusters	11
Amount of power levels	19
Jammers power level threshold	11
PCJ duration threshold	4.94 ms
Deceptive duration threshold	4.94 ms
SFD duration MIN threshold	0.078 ms
SFD duration MAX threshold	0.182 ms
BT power level Min threshold	4
BT power level MAX threshold	5
BT duration MIN threshold	0.5 ms
BT duration MAX threshold	3.6 ms
WiFI duration MIN threshold	2.34 ms
WiFi duration MAX threshold	4.16 ms
WiFi power level MIN threshold	6
WiFi power level MAX threshold	7

Table 2: Values of method variables

4.2.1 nRF52840-dongle

The nRF52840-dongle shown in figure *9* is a USB dongle created by Nordic. [27] Key features that it has are IEEE 802.15.4 radio support and integrated 2.4 GHz PCB antenna. The embedded processor it uses is a 64MHz Arm Cortex-M4, which supports floating points. The memory size is 1 Mb for the Flash and 256 Kb for the RAM. [30]

Figure 9: nRF52840-dongle

4.2.2 nRF52840-DK development kit

The nRF52840-DK that is shown in figure *10* is a development kit created by Nordic. [27] It contains an integrated 2.4 GHz PCB antenna. The hardware uses the same nRF52840 SoC as the nRF52840-dongle and the same 64MHz

Arm Corex-M4. It got an extended Flash memory of 64 Mb and 256 RAM. [29]

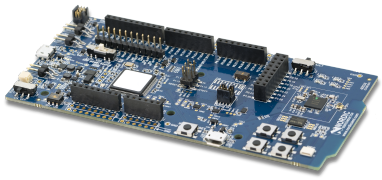

Figure 10: nRF52840-DK development kit

4.2.3 Raspberry Pi 3 Model B

The Raspberry Pi 3 Model B that figure *11* shows, is a single-board computer with wireless LAN and Bluetooth connectivity. It got a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, and 1GB RAM [23]

Figure 11: nRF52840-DK development kit

4.2.4 Pre-determined threshold

The interference sources are profiled in advance in order to identify them. The profiling is done by running the RSSI-sampler and K-means clustering while one interference source is running one at a time. While the algorithm is running does it store the unique centroid that gets created from the interference source. The average value is calculated and used as the centre point for the pre-determined threshold. The pre-determined threshold does then get a min and max value, as the interference may vary slightly in duration. It is these pre-determined thresholds that are used to identify the different jamming attacks.

4.2.5 Experiment A

The first experiment that is shown in figure *12* is carried out by placing two nodes 3 meters from each other. One node running Contiki's RPL client and the other nodes running Contiki's RPL server. The purpose of these nodes is to communicate with each other and simulate a network. A third node is placed in the middle of the two nodes and that node is running SpeckSense++. That node's job is to identify radio jamming interference attacks. A fourth embedded system is placed in the middle of the network, its job is to carry out the different attacks.
When the two nodes have established a steady connection with each other and are communicating via RPL-UDP is different jamming attacks carried out. First, is the proactive constant jammer tested. After the nodes have re-established a new stable connection are the remaining jamming attacks tested.

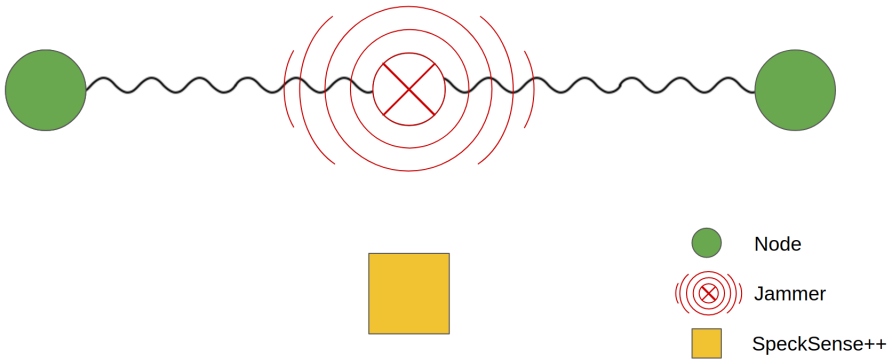

Figure 12: Experiment A, identifying multiple jamming attacks

4.2.6 Experiment B

The second experiment consists of five different nodes which are shown in figure *13*. Figure *14* shows SpeckSense++ on the nodes which represent that all nodes are running both RPL-UDP and SpeckSense++. The corner nodes are able to communicate with each other and are all running SpeckSense++. The middle node doesn't communicate with any other node and its only purpose is to scan the air for data. The jammer is placed close to a corner node and starts the jamming attacks one by one. First running the proactive jammer, then the SFD-jammer, the deceptive jammer and lastly the random jammer. When the nodes lose connection will they swap to SpeckSense++ to identify what the problem might be, and can realize there is a jamming attack carried out. When the nodes swap to SpeckSense++ to detect the SFD-jammer will they not find

anything, because it is a reactive jammer and does only trigger when it senses a packet in the air. The middle node will always run SpeckSense++ and when it senses an SFD-jammer will start to send out packets to lure the SFD-jammer to attack it, which is shown in figure *14*. By doing so will the other nodes in the network be able to identify the SFD-jammer as the middle node acts as a honey pot node and lures the SFD-jammer to attack.

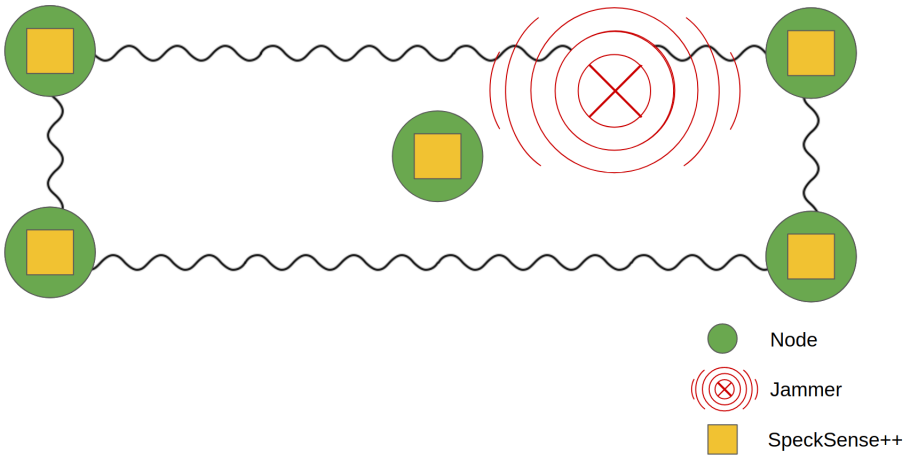

Figure 13: Experiment B, real-life scenario

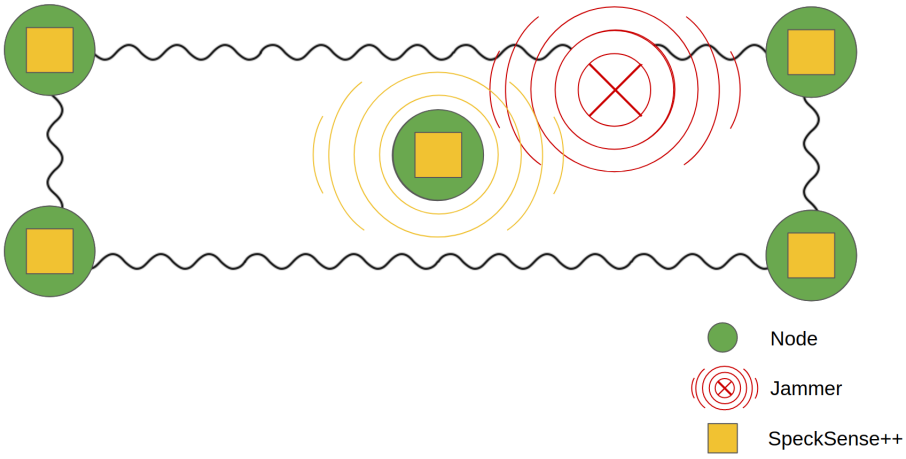

Figure 14: Experiment B setup, honeypot activated

4.2.7 Experiment C

The third experiment shown in figure *15* consists of one node running SpeckSense++ and a strong Bluetooth device. The purpose is to show if it is possible to detect unintentional jamming as the original SpeckSense. The Bluetooth will emit background noise and SpeckSense++ will try to classify the strong signal.

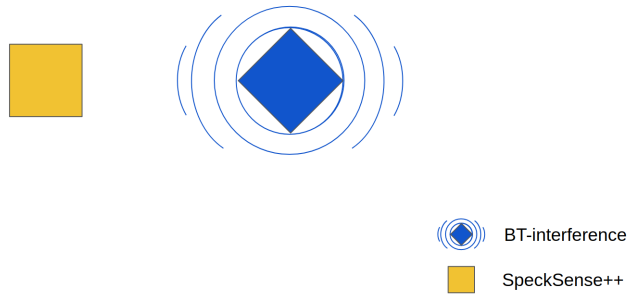

Figure 15: Experiment C, identifying Bluetooth interference

4.2.8 Experiment D

Figure *16* shows the fourth experiment which consists of one node running SpeckSense++ and another node running "JabLab++" which creates a deterministic WiFi interference. The experiment's goal is to test if it is possible to detect WiFi interference as the original SpeckSense. The WiFI will emit signals and SpeckSense++ will classify the interference.

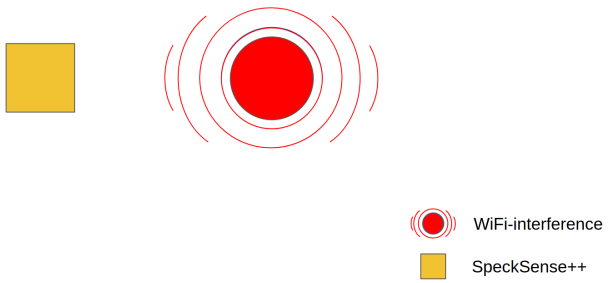

Figure 16: Experiment D, identifying WiFi interference

4.2.9 Experiment E

The fifth experiment consists of five nodes. One running SpeckSense++, the second node running the proactive constant jammer, the third node running the SFD-reactive jammer, and the two last nodes consist of an RPL-UDP client and an RPL-UDP server. Figure *17* shows the construction. The purpose of this experiment is to see if it is possible to identify and distinguish between two different jammers running simultaneously.

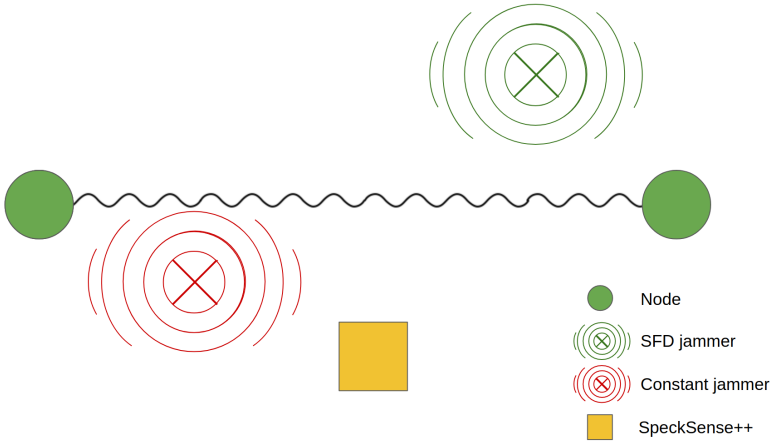

Figure 17: Experiment E, identifying multiple sources simultaneously

5 Results

The figures in the results chapter show how it looks when SpeckSense++ identifies different types of interference on two different channels and two different signal strengths. Channel 26 has low background noise and channel 13 which has higher background noise. The threshold lines outline the boundaries for identification. If a cluster is insides the coloured area is it alerted as an interference source. All the figures shown are results from *Experiment A, identifying multiple jamming attacks* as can be seen in figure *12*

Figure *18* shows the accuracy of identifying the constant jammer(96%), deceptive jammer(96%), Random jammer (90%), SFD jammer(89%), Wifi interference (85%) and Bluetooth interference(92%).

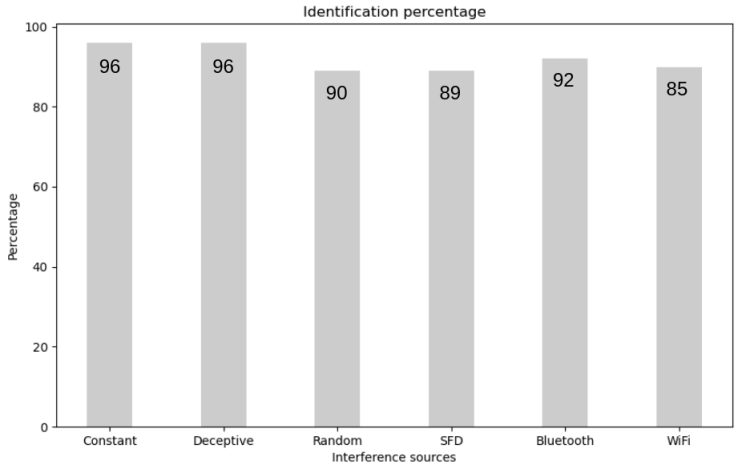

Figure 18: Identification accuracy of the different interference sources

Figure 19 shows how it looks when a constant jamming attack with a power signal of 8dBm is clustered on channel 26.

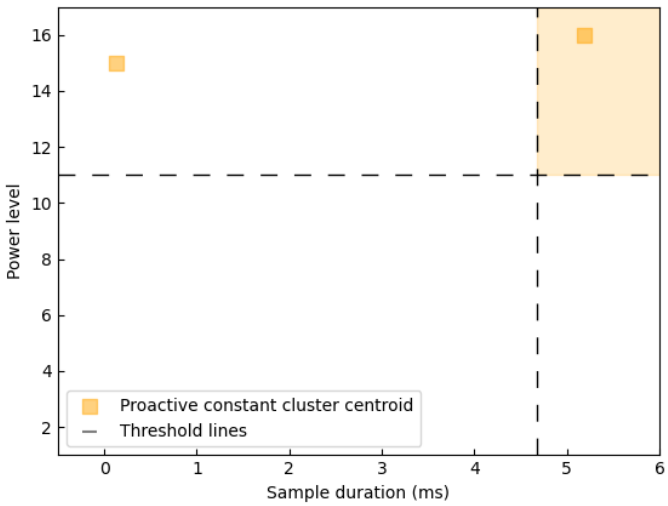

Figure 19: Clustering a constant jamming attack (8dBm) (Channel 26)

Figure 20 shows how it looks when a constant jamming attack with a negative power signal of 16dBm is clustered on channel 26.

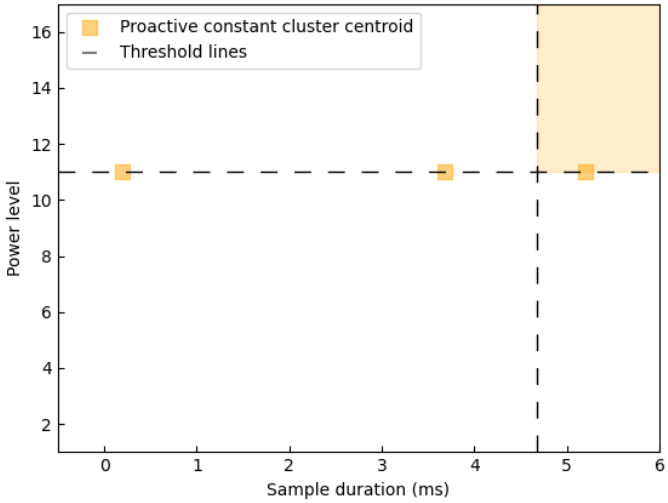

Figure 20: Clustering a constant jamming attack (negative 16 dBm) (Channel 26)

Figure 21 shows how it looks when an SFD jamming attack with a power signal of 8dBm is clustered on channel 26.

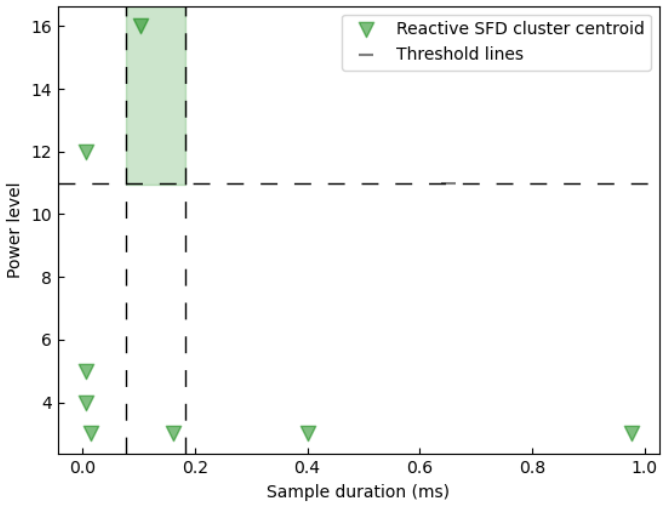

Figure 21: Clustering a reactive SFD jamming attack (8dBm) (Channel 26)

Figure 22 shows how it looks when a random jamming attack with a power signal of 8dBm is clustered on channel 26.

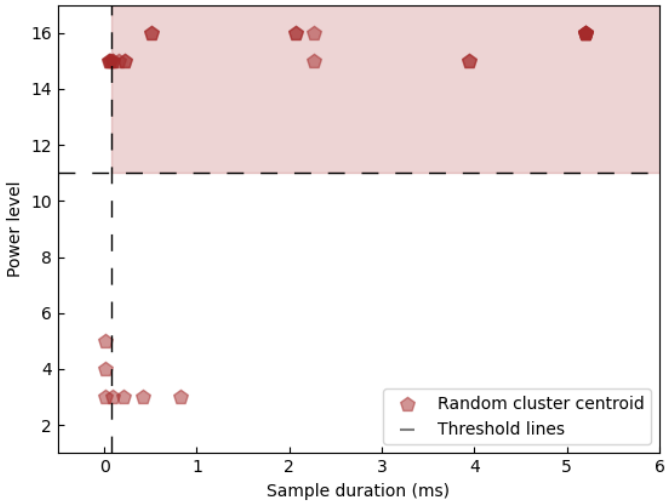

Figure 22: Clustering a random jamming attack (8dBm) (Channel 26)

Figure 23 shows how it looks when a random jamming attack with a negative power signal of 16dBm is clustered on channel 26.

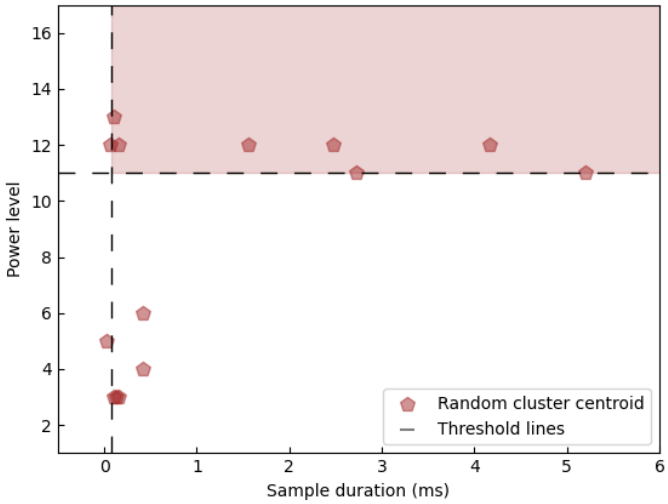

Figure 23: Cluster of random jamming attack (negative 16dBm) (Channel 26)

Figure 24 shows the result of clustering a pair of wireless Bluetooth headphones on channel 26

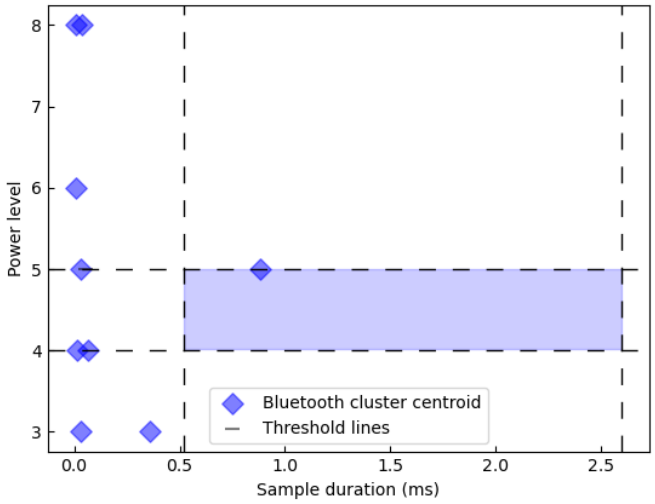

Figure 24: Clustering Bluetooth interference

Figure 25 shows how it looks when a constant jamming attack with a power signal of 8dBm is clustered on channel 13.

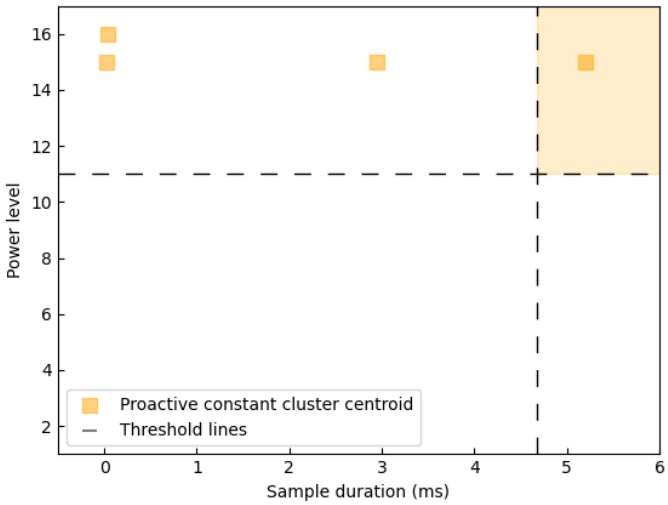

Figure 25: Clustering a constant jamming attack (8dBm) (Channel 13)

Figure 26 shows how it looks when a constant jamming attack with a negative power signal of 16dBm is clustered on channel 13.

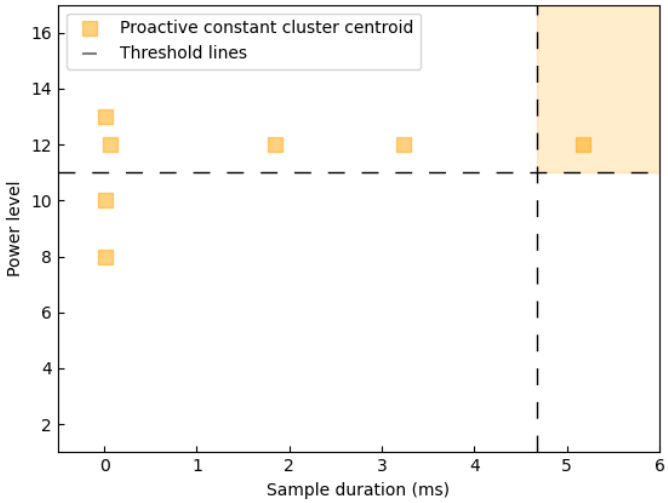

Figure 26: Clustering a constant jamming attack (negative 16 dBm) (Channel 13)

Figure 27 shows how it looks when a SFD jamming attack with a power signal of 8dBm is clustered on channel 13.

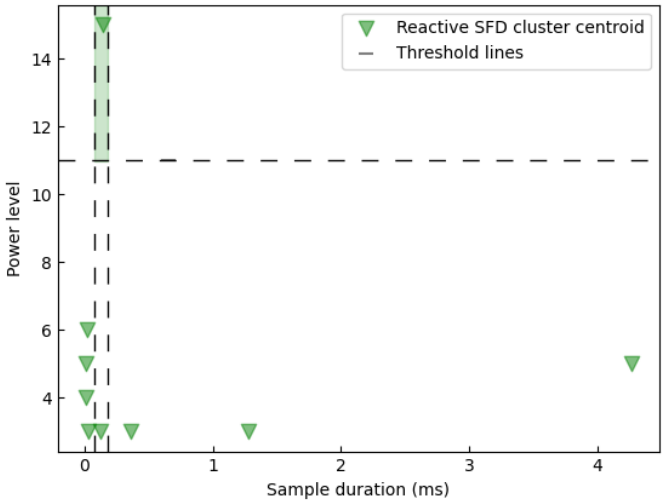

Figure 27: Clustering a reactive SFD jamming attack (8dBm) (Channel 13)

Figure 28 shows how it looks when a random jamming attack with a power signal of 8dBm is clustered on channel 13.

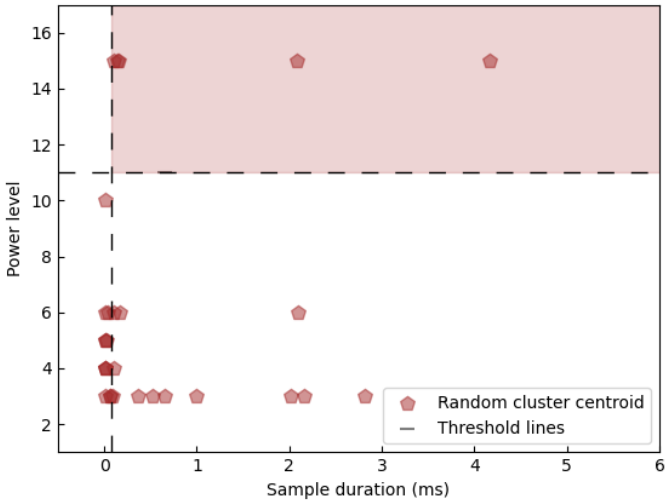

Figure 28: Clustering a random jamming attack (8dBm) (Channel 13)

Figure 29 shows how it looks when a random jamming attack with a negative power signal of 16dBm is clustered on channel 13.

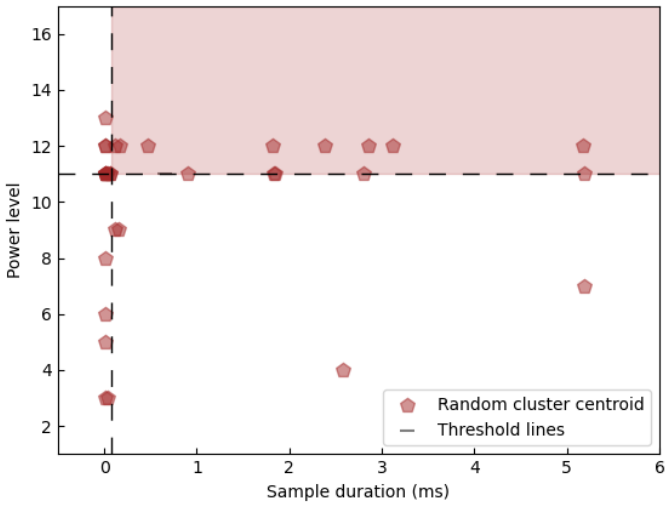

Figure 29: Clustering a random jamming attack (negative 16dBm) (Channel 13)

Figure 30 shows how it looks when a strong deterministic WiFi interference source is clustered on channel 22.

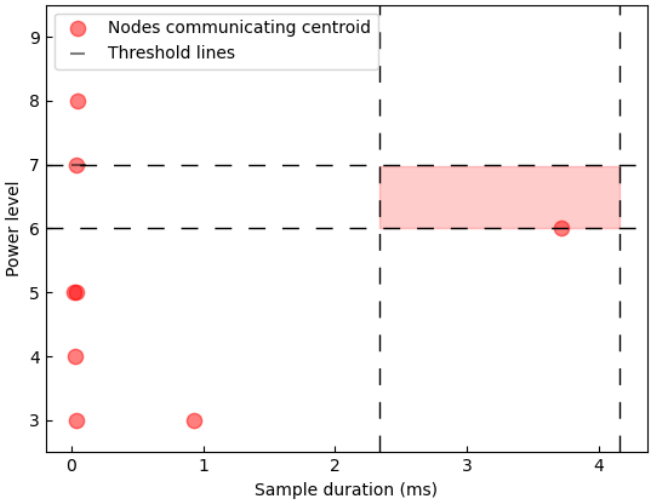

Figure 30: Clustering WiFi interference created by JamLab-NG

5.1 Evaluation

It is important to point out that when any interference jammer is creating interference and emitting signals, the signals vary slightly in dBm and are not 100% on the same level. This is expected behaviour as there is real interference that might affect. As SpeckSense++ run-length encodes its RSSI values it does create a new 2D vector as the signal changes over a specific range. This might happen multiple times when a jammer is running. For example, figure *19* shows that SpeckSense++ are detecting and identifying a proactive constant jammer. The figure shows two clusters, one that has a power level of 16 and a duration of 5.4 ms, while the other shows a power level of 15 and a duration of 0.3 ms. The cluster centroid that shows a power level of 15 and a duration of 0.3 ms is a result because of that. The Euclidean distance algorithm is also weighted which increases the clustering of outliers, which enhances that the probability of a small sample of values will get clustered.

In figure *20* one of the cluster centroids has a power level of 11 and a duration of 0.2 ms. Couldn't it as easily be that SpeckSense++ would identify that attack as a Reactive SFD jamming attack as it is inside the SFD threshold? The answer is no. The reason for that is when the clusters are created are they sorted after duration. This means that if a jamming attack shows the trait of a proactive constant jammer it can not be classified as anything else than a proactive constant jamming attack as the longest duration is checked first. The argument being is that it can't be a reactive SFD jamming attack if the duration is that long.

Figure *22* shows multiple clusters inside the threshold. It is only five centroids that are being used and that are important in the decision-making if it is a random jamming attack or not. The same reason as to why the proactive constant jammer sometimes contains multiple clusters of low value is why the figure shows multiple clusters in the threshold as well. It is however only the five centroids that have been selected from the identification algorithm that decides if it is a random jammer or not.

Figure *21* shows multiple clusters being under the power level of six. The reason for that is because the reactive SFD jamming attack is very fast, so there is a majority of background noise that is sampled. Figure *19* and figure *25* show different values of background noise as they are sampled on two different channels where one contains more traffic than the other.

5.1.1 Detecting different jammers simultaneously

Experiment E's purpose was to see if it is possible to detect multiple interference sources simultaneously. That was one of the original SpeckSense key features. It is sort of possible to detect multiple jammers simultaneously, but not how the current identification is used. For example, a test was carried out when one

reactive SFD jammer was very close to the node with a high power signal, and a proactive constant jammer running further away with a lower power signal. In that case will two different clusters be created where one can be identified as a proactive constant jammer and another as a reactive SFD jammer. However, currently how the identification is implemented will the identification instead only identify it as a proactive constant jammer as it will stop investing clusters after it finds a cluster that fits the profile. The reason why it does that is a combination of how the RSSI signals are run-length encoded and because outliers are emphasized with help of a weighted Euclidean distance.
It is however a possibility to fix and to implement the possibility to identify multiple sources simultaneously. But with the limited time and because it isn't prioritized to identify how many sources are jamming and instead what type of source it is that is jamming, the feature was not implemented.

6 Analyses and Discussion

6.1 WiFi interference

SpeckSense++ makes it possible to identify radio jamming attacks as well as unintentional interference. However, when the experiment for identifying WiFi interference was carried out, was it not always a possibility to classify it as WiFi interference. It depended a lot on the settings of JamLab-Ng. For example, if the settings in JamLab-NG had a very long length, would the signal duration be very long and be inside the range of the constant jammer. This resulted in it being identified as a constant jammer. Therefore, was it only possible to identify and classify WiFi interference to a certain degree.

6.2 Reactive jammer

For the reactive SFD jammer was 312 microseconds used for the jammer to switch between receiving mode and transmission mode. 312 microseconds has been proven to be the minimum effective jam burst length to corrupt a transmitted payload according to the paper.[11] The 312 microseconds were determined from the aspect of using CC2420 radio drivers. 312 microseconds did also seem to be enough time for the nRF52840s radio driver to be able to sniff an SFD, switch to TX-mode and corrupt a packet.

6.3 SpeckSense++ cluster

If it would only be to detect the proactive constant jamming attack and the reactive SFD jamming attacks would it have been enough to compare the clusters from the K-means algorithm against the pre-determined thresholds. But because the proactive *random* jammer is in play is it necessary to compare multiple interferences toward each other. The K-means clustering algorithm relies on consistency to identify jamming attacks. Most jamming attacks are consistent, but the proactive random jammer is not. In the worst-case could it have the same features as the constant jammer or the SFD jammer because its duration of jamming is random. In that case, would it be a true prediction that it was a jamming attack but identify it as the wrong jammer.

6.4 Anomaly detection

One limitation with this SpeckSense++ is that it is limited to the attacks it chooses to profile. As previously stated are there more or less unlimited numbers of jamming attacks that can be assembled. A good argument would be; why not use Anomaly detection? Anomaly detection is a branch in data mining which purpose is to detect anomalies. It does that by identifying observations and check if it deviates from the normal behaviour of the data set. Using anomaly detection, in this case, would be profiling a channel that the network is going

to use, and compare incoming RSSI-values towards that data set. If any RSSI-values deviated towards the normal behaviour of the data set could it be a potential attack and some type of countermeasure could be taken in place. This would indeed be a more efficient way if the main purpose was to avoid jamming attacks. It wouldn't however identify the attacks that were happening, it would only know that *something* is deviating, not what it is. SpeckSense++ main purpose is to be able to identify and distinguish the attacks. And to be able to do that is it necessary to have the acknowledgement of the attacks. To be able to identify other attacks than the attacks that are carried out in this thesis, would it be possible to profile those other attacks and compare them to identify them.

6.5 Avoidance strategy

It would be possible to implement an avoidance strategy on top of SpeckSense++. When a jamming attack has been identified, apply a specific countermeasure. One simple effective countermeasure would be to scan all channels on the spectrum when a jamming attack occurs, apply a time synchronization algorithm to inform all the other nodes in the network to swap to the channel that has the least interference on its channel, blacklist the interfered channel for n amount of time and swap. Figure *31* shows how it could look to avoid interference. This would however not be anything innovative, so the main focus was on how to identify and distinguish different attacks instead.

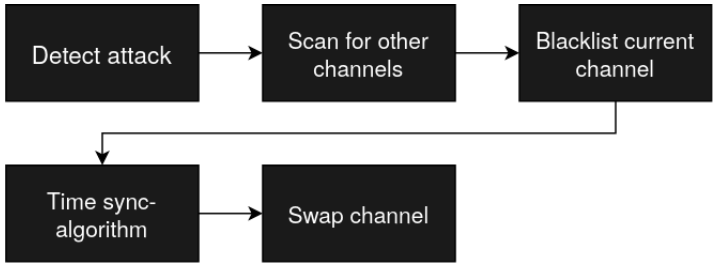

Figure 31: Theoretical avoidance strategy

6.6 Limitations

Some limitations are the few features that are used to identify different jamming attacks. Currently, are two different features used, which are power level and duration. These two have been proven to be enough to identify the different jammers. As long as two different interference sources don't share the same threshold area will they be identified separately. But the more interference sources that are going to be profiled and identified does it increase the risk of some interference sources sharing the same threshold. By increasing the number of features that are used to identify the jamming attacks, will also lower the risk

of multiple sources being identified as the same. With more features does also the computation time increase, so it is important to have a balance.

The experiments were carried out in an apartment, which leads to some limitations. One limitation is the scaling of the experiments. The distance the nodes could be tested from each other was limited. To simulate a longer range from each other did the jammer test different strengths in its power level.

6.7 Ethical and societal aspects

While conducting the experiments did jamming interference occur which can have affected the neighbours slightly. The jamming was however conducted on channels that are not usually used, and the signal strength wasn't that high, which means it was a low probability neighbour were affected.
Monitoring of network traffic was used to get more information, which also showed neighbours activity. It was however only signal strength that was monitored, and it wasn't possible to see from whom the information was taken. Figure *32* shows how it looked when RSSI values were monitored for debugging purposes.

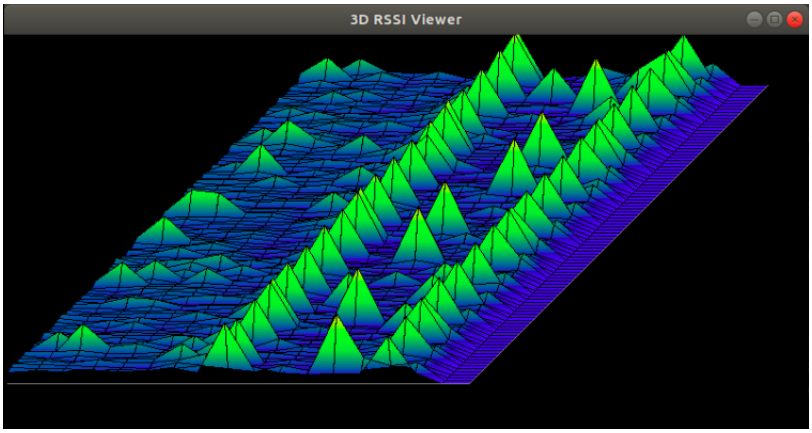

Figure 32: RSSI-scanner debug tool

The importance of counteracting interference jamming attacks is at its peak. The world becomes more modern every day, and more technology is becoming wireless. The damage a simple interference jamming attack can cause can be everything from denying human access to its garage to something more critical, that can be found inside a critical infrastructure. This work is a step towards a more secure society.

7 Conclusion

The main focus of this thesis was to find and implement a way to detect, identify and distinguish between different radio jamming interference attacks on low power hardware. The combination of an RSSI-sampler, K-means clustering, a weighted euclidean distance and a smart way to distinguish between different jamming interference gave those results. This thesis has shown the possibility to detect and identify the four most popular jamming attacks with high accuracy. An accuracy of *96* % to identify the constant jammer and the deceptive jammer, an accuracy of *89*% for the SFD jammer and an accuracy of *90*% for the proactive random jammer. SpeckSense++ does also maintain its possibility to identify unintentional interference on the 2.4gHz spectrum such as Bluetooth with an accuracy of *92*% and WiFi beacons to a certain degree with an accuracy of *85*%.
It is a possibility to identify other interference jamming attacks as well. If their behaviour can be simulated, they will also be able to be profiled and identified. This thesis does also implement interference jamming tools that are highly customizable that can be used for future work.

7.1 Future work

It would be possible to improve the identification step. As discussed in the *discussion* chapter, identifying more jamming attacks might be desirable. It would be a possibility to involve more features in the clustering process as it decreases the probability of jamming attacks overlapping. If more jamming attacks are profiled, it increases the risk of their unique threshold overlapping. By increasing the features does it lower the risk of different thresholds overlapping.
An additional function that would enhance the firmware would be to include that multiple sources that are jamming simultaneously could be identified and distinguished separately. Those characteristics would enhance the firmware in different situations. Being able to identify multiple sources simultaneously could make the system a possible tool inside the forensic area. When a company has been targeted by a cyberattack they will proceed to investigate what the attack was to make sure it doesn't happen again. Being able to extract more information from a situation would give more room for improvement. Detecting multiple sources would give more information in that sort of case.
Implement countermeasures could be the next step. Right now it is a possibility to identify and distinguish between radio jamming interference attacks, but it can't avoid the specific different jamming attacks. SpeckSense++ gives good information on what type of jamming attack it is. It makes it a possibility to apply specifically tailored countermeasures to specific attacks in order to defend against them.

References

[1] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[3] Purnima Bholowalia and Arvind Kumar. Ebk-means: A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105(9), 2014.

[4] Ismail Butun, Salvatore D Morgera, and Ravi Sankar. A survey of intrusion detection systems in wireless sensor networks. *IEEE communications surveys & tutorials*, 16(1):266–282, 2013.

[5] Kaushik R Chowdhury and Ian F Akyildiz. Interferer classification, channel selection and transmission adaptation for wireless sensor networks. In *2009 IEEE International Conference on Communications*, pages 1–5. IEEE, 2009.

[6] Adam Dunkels. Full tcp/ip for 8-bit architectures. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 85–98, 2003.

[7] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *29th annual IEEE international conference on local computer networks*, pages 455–462. IEEE, 2004.

[8] Simone Grimaldi, Aamir Mahmood, and Mikael Gidlund. An svm-based method for classification of external interference in industrial wireless sensor and actuator networks. *Journal of Sensor and Actuator Networks*, 6(2):9, 2017.

[9] Vipul Gupta, G Simmons David, et al. Building the web of things with sun spots. *JavaOneHands-on Lab*, 2010.

[10] T. Hamza, G. Kaddoum, A. Meddeb, and G. Matar. A survey on intelligent mac layer jamming attacks and countermeasures in wsns. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2016.

[11] Zhitao He and Thiemo Voigt. Precise packet loss pattern generation by intentional interference. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–6. IEEE, 2011.

[12] Frederik Hermans, Olof Rensfelt, Thiemo Voigt, Edith Ngai, Lars-Åke Nordén, and Per Gunningberg. Sonic: Classifying interference in 802.15. 4 sensor networks. In *Proceedings of the 12th international conference on Information processing in sensor networks*, pages 55–66, 2013.

[13] Anwar Hithnawi, Hossein Shafagh, and Simon Duquennoy. Tiim: Technology-independent interference mitigation for low-power wireless networks. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, pages 1–12, 2015.

[14] Venkatraman Iyer, Frederik Hermans, and Thiemo Voigt. Detecting and avoiding multiple sources of interference in the 2.4 ghz spectrum. In *European Conference on Wireless Sensor Networks*, pages 35–51. Springer, 2015.

[15] Oleg Kachirski and Ratan Guha. Effective intrusion detection using multiple sensors in wireless ad hoc networks. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 8–pp. IEEE, 2003.

[16] GS Kasturi, Ansh Jain, and Jagdeep Singh. Detection and classification of radio frequency jamming attacks using machine learning. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 11(4):49–62, 2020.

[17] Alex King and Utz Roedig. Differentiating clear channel assessment using transmit power variation. *ACM Transactions on Sensor Networks (TOSN)*, 14(2):1–28, 2018.

[18] Yuqing Mai, Radhika Upadrashta, and Xiao Su. J-honeypot: a java-based network deception tool with monitoring and intrusion detection. In *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, volume 1, pages 804–808. IEEE, 2004.

[19] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou. A survey on jamming attacks and countermeasures in wsns. *IEEE Communications Surveys Tutorials*, 11(4):42–56, 2009.

[20] Nima Namvar, Walid Saad, Niloofar Bahadori, and Brian Kelley. Jamming in the internet of things: A game-theoretic perspective. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[21] Opeyemi Osanaiye, Attahiru S Alfa, and Gerhard P Hancke. A statistical approach to detect jamming attacks in wireless sensor networks. *Sensors*, 18(6):1691, 2018.

[22] George D O'Mahony, Philip J Harris, and Colin C Murphy. Detecting interference in wireless sensor network received samples: A machine learning approach. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE, 2020.

[23] Raspberry Pi. Raspberry Pi 3 Model B. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, (visited on 2021-06-02).

[24] Brian Roberson. The colonel blotto game. *Economic Theory*, 29(1):1–24, 2006.

[25] John Romkey. Toast of the iot: the 1990 interop internet toaster. *IEEE Consumer Electronics Magazine*, 6(1):116–119, 2016.

[26] Markus Schuß, Carlo Alberto Boano, Manuel Weber, Matthias Schulz, Matthias Hollick, and Kay Römer. Jamlab-ng: Benchmarking low-power wireless protocols under controllable and repeatable wi-fi interference. In *EWSN*, pages 83–94, 2019.

[27] Nordic Semiconductor. About Nordic Semiconductor. https://www.nordicsemi.com/About-us/, (visited on 2021-06-02).

[28] Nordic Semiconductor. Direct test mode. https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/ble_sdk_app_dtm_serial.html, (visited on 2021-06-02).

[29] Nordic Semiconductor. nRF52840 DK. https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF52840-Dongle/, (visited on 2021-06-02).

[30] Nordic Semiconductor. nRF52840 Dongle. https://www.nordicsemi.com/Software-and-Tools/Development-Kits/nRF52840-DK/, (visited on 2021-06-02).

[31] Tarek S. Sobh. Wired and wireless intrusion detection system: Classifications, good characteristics and state-of-the-art. 2006.

[32] Radosveta Sokullu, Ilker Korkmaz, Orhan Dagdeviren, Anelia Mitseva, and Neeli R Prasad. An investigation on ieee 802.15. 4 mac layer attacks. In *Proc. of WPMC*, volume 41, pages 42–92, 2007.

[33] Xiao Tang, Pinyi Ren, and Zhu Han. Jamming mitigation via hierarchical security game for iot communications. *IEEE Access*, 6:5766–5779, 2018.

[34] Nicolas Tsiftes, Joakim Eriksson, and Adam Dunkels. Low-power wireless ipv6 routing with contikirpl. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 406–407, 2010.

[35] Satish Vadlamani, Burak Eksioglu, Hugh Medal, and Apurba Nandi. Jamming attacks on wireless networks: A taxonomic survey. *International Journal of Production Economics*, 172:76 – 94, 2016.

[36] Jane Webster and Richard T Watson. Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, pages xiii–xxiii, 2002.

[37] X. Wei, Q. Wang, T. Wang, and J. Fan. Jammer localization in multi-hop wireless network: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 19(2):765–799, 2017.

[38] Wenyuan Xu, Ke Ma, Wade Trappe, and Yanyong Zhang. Jamming sensor networks: attack and defense strategies. *IEEE network*, 20(3):41–47, 2006.

[39] Yun Yang and Jia Mi. Design and implementation of distributed intrusion detection system based on honeypot. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 6, pages V6–260. IEEE, 2010.

[40] Sven Zacharias, Thomas Newe, Sinead O'Keeffe, and Elfed Lewis. A lightweight classification algorithm for external sources of interference in ieee 802.15. 4-based wireless sensor networks operating at the 2.4 ghz. *International Journal of Distributed Sensor Networks*, 10(9):265286, 2014.

[41] Yongguang Zhang, Wenke Lee, and Yi-An Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, 9(5):545–556, 2003.