DiVA
http://www.diva-portal.org

Postprint

This is the accepted version of a paper presented at *21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT).*

Citation for the original published paper:

Zhu, S., Rahimian, F., Voigt, T., Ko, J. (2025)
AJDet: Lightweight Self-Adaptive JammingDetection for IoT Networks
In:

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-574600

AJDet: Lightweight Self-Adaptive Jamming Detection for IoT Networks

Shuai Zhu*†, Fatemeh Rahimian*, Thiemo Voigt*†, JeongGil Ko‡§
*RISE Research Institutes of Sweden, Kista, Sweden
†Uppsala University, Uppsala, Sweden
‡Yonsei University, Seodaemun-gu, Korea (the Republic of)
§POSTECH, Pohang, Korea (the Republic of)
{shuai.zhu, fatemeh.rahimian}@ri.se, thiemo.voigt@angstrom.uu.se, jeonggil.ko@yonsei.ac.kr

***Abstract*—Wireless IoT networks are becoming increasingly important as they support a diverse range of applications. However, due to their resource constraints and limited security capabilities, these networks are particularly susceptible to attacks. Among these, jamming attacks pose a significant threat by degrading packet delivery, disrupting communications, and depleting the limited networking and energy resources of IoT systems. Hence, effective jamming detection is essential for safeguarding wireless IoT networks. However, dynamic operating environments and diverse application scenarios make it challenging to design a robust jamming detection system. To address the challenge, this paper introduces AJDet, a lightweight, online, Adaptive Jamming attack Detection system. AJDet leverages an online adaptation approach that enables the detection of both proactive and reactive jamming attacks without relying on a deployment-specific configuration. Our experimental results demonstrate that AJDet achieves over 96.5% detection accuracy without the need for manual intervention after deployment. Moreover, the system exhibits an efficient memory footprint, making it well-suited for resource-constrained devices.**

***Index Terms*—Wireless IoT Networks, Jamming Attacks Detection, Online Self-adaptive System**

I. Introduction

Wireless Internet of Things (IoT) networks wirelessly connect resource-limited sensing platforms to enable smart applications across various domains, such as smart cities, industrial automation, and healthcare [1]. As these technologies gain popularity, their security and trustworthiness become significantly important and have attracted considerable interest. Secure wireless communications should satisfy the requirements of authenticity, confidentiality, integrity, and availability [2]. However, the inherent open nature of wireless communication, however, makes their availability extremely vulnerable to disruptions that prevent legitimate users from accessing services.

One serious threat are jamming attacks, where attackers transmit interfering radio signals to severely disrupt legitimate transmissions of IoT networks [3]. These threads are exacerbated by the IoT nodes' low transmit power. Therefore, jamming detection plays a vital role in wireless IoT network security. Wireless IoT networks are inherently dynamic due to the wide range of devices, protocols, and deployment contexts they encompass. The frequent network topology and resource availability changes underscore the dynamic nature of IoT environments [4]. Hence, jamming detection systems capable of adapting to this characteristic are indispensable for securing wireless IoT networks.

Motivation. The dynamic nature of IoT networks presents a fundamental challenge in designing jamming detection systems. Systems that rely on deployment-specific parameters—such as training data from a particular testbed and empirically determined thresholds—may fail to perform adequately when environmental conditions change. In contrast, a machine learning (ML)-based jamming detection approach, as proposed by Puñ et al. [5], demonstrates that utilizing training data from diverse deployment environments is essential for achieving high detection accuracy. Training solely on data from a single environment can lead to suboptimal performance, particularly in scenarios that deviate from the original training conditions. Kanwar et al. [6] introduce JamSense, a system that employs predetermined thresholds for jamming detection, with thresholds empirically determined for their specific testbed. Although JamSense achieves high accuracy within a specific environment, our experiments in a small-scale testbed (see Figure 6) reveal significant performance degradation when the system is applied to detect both proactive and reactive jamming attacks. We conducted 1,000 trials for each attack type, and the results, as illustrated in Figure 1, indicate that the system barely detects either type of jammer. Furthermore, with dynamically changing channel conditions and environmental factors, this limitation becomes even more prominent. These results emphasize the necessity for a jamming detection system that is self-adaptive to the deployment environment.

Challenges. As discussed above, network dynamics pose a critical challenge. In addition to these dynamics, there are two further challenges. First, IoT devices vary widely in their hardware capabilities, ranging from severely constrained microcontrollers, such as the Nordic Semiconductor nRF52840 [7], to more capable single-board computers, such as the Raspberry Pi [8]. Introducing additional hardware to support jamming detection is often impractical due to increased costs, complexity, and deployment constraints. Therefore, a jamming detection system must be lightweight and easily integrated into applications running on hardware-constrained devices without necessitating extra hardware. Second, the diversity of network protocols, particularly at the Medium Access Control (MAC) layer, presents another significant challenge [9]. Different pro-

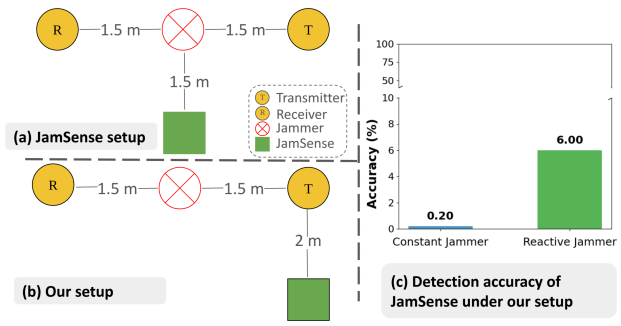


Fig. 1: Preliminary Study: Figures (a) and (b) show the original experimental setup of JamSense [6] and the setup we used to evaluate it, respectively. Figure (c) shows JamSense's detection accuracy in our setup. The accuracy is notably low because its threshold was predetermined for a static setup that slightly differs from ours.

tocols manage the shared communication medium in various ways, e.g., carrier sense multiple access (CSMA) vs. time-slotted channel hopping (TSCH) [10]). Consequently, jamming detection systems must employ methods that are generally compatible with the underlying communication protocol, such as capturing Signal Strength Indicator (RSSI) values, while accommodating the nuances of each protocol.

Approach. We present a lightweight, online, Adaptive Jamming attack Detection system (AJDet), which utilizes an online iterative adaptation approach that continuously adjusts its detection thresholds in real-time to accommodate dynamic network conditions. AJDet uses distinct detection thresholds tailored to networks employing different MAC protocols. Specifically, in CSMA-based networks, AJDet utilizes a signal *power-level* threshold for jammer detection and a duration threshold for classification; the *power-level* threshold is rendered adaptive to ensure robust detection. For networks employing TSCH, AJDet applies a pair of adaptive thresholds: a signal *power-level* threshold and a *power-level* gap threshold, i.e., the difference between the highest and lowest *power-level*.

In parallel with the detection process, AJDet implements an online iterative adaptation approach that continuously adjusts its detection thresholds in real-time to accommodate dynamic network conditions. The objective of this adaptation is to maximize and maintain detection accuracy. As illustrated in Figure 4, the online iterative adaptation mechanism consists of two components: a monitor and an adapter. The monitor evaluates the occurrence of detection errors by analyzing real-time packet delivery performance alongside detection results. It reports false positives when the system erroneously detects a jammer in the absence of reception disruptions and false negatives when significant disruptions or abnormalities in packet reception occur while no jamming is detected. Based on these monitoring results, the adapter adjusts the detection thresholds accordingly.

Contributions. We make the following contributions:

- Through initial experiments on jamming attack detection, we demonstrate the limitations of existing jamming detection schemes with fixed predetermined thresholds that tailored for specific network setups (e.g., network size/topology). We show that such schemes are susceptible to extremely low accuracy towards changes in the network environment configurations.
- We design an online iterative threshold adaptation approach for jamming detection. The optimization objective is to maximize the detection accuracy by dynamically adjusting the threshold. The process begins with empirically determined thresholds and repeats iteratively, to adapt the threshold in run time.
- We present AJDet, the first online self-adaptive jamming detection and classification system for low-power IoT networks. We provide two lightweight implementations of the online iterative optimization approach for networks with different MAC layer protocols, i.e., CSMA and TSCH. Evaluations show that AJDet achieves over 96.5% detection accuracy for proactive and reactive jammers without post-deployment configurations.

The rest of the paper is organized as follows: Section II reviews background and related work. Section III describes our system design. Section IV presents the evaluation results. Section V discusses the findings, and Section VI concludes the paper.

II. Background and Related Work

We start by providing a brief overview of jamming attacks and related works in jamming detection.

A. Jamming Attacks

Jamming attacks are considered among the most severe security threats for wireless IoT networks [3]. Their objectives vary considerably with the application; one common goal is to degrade the signal-to-noise ratio (SNR) and node-dependent metrics such as the packet loss rate (PLR) at the target node. Jammers with higher transmission power can more easily and rapidly interfere with communications over the same distance [11]. Although the capture effect in wireless communication may allow a receiver to successfully decode a stronger legitimate signal while rejecting a weaker jamming signal, a sufficiently powerful jammer can override the intended transmission and cause complete communication failure [12]. While several types of jamming attacks exist, the most prevalent are proactive and reactive jammers [13], [14].

Proactive Jammers. Proactive jammers aim to prevent legitimate communications by transmitting interference signals to occupy the channel regardless of actual packet transmissions. They can be broadly categorized into three types: constant, deceptive, and random jammers. Constant jammers continuously transmit random bits on the channel without adhering to MAC layer protocols. In contrast, deceptive jammers transmit regular packets without any pauses. Random jammers alternate between jamming and sleeping phases, rather than continuously transmitting interference signals. By varying the durations of these phases, random jammers can balance energy efficiency with jamming effectiveness based on application requirements.

Reactive Jammers. Reactive jammers remain silent until they detect a packet in the air. Once a packet is sensed, they immediately transmit interference signals to corrupt the legitimate transmission. A typical implementation is to transmit an interference signal upon detecting a preamble and a Start-of-Frame Delimiter (SFD) on the target channel [15]. This design exploits the fact that the synchronization header in both IEEE 802.11b and IEEE 802.15.4 standards consists of a preamble sequence followed by a fixed SFD.

B. Related Work

Kanwar et al. present JamSense, an interference and jamming classification system designed for low-power wireless networks [6] based on SpeckSense, an interference detector that distinguishes between different types of interference. JamSense consists of three phases: RSSI sampling, RSSI burst clustering, and jamming detection using fixed thresholds. Our work adopts a similar detection strategy for networks with contention-based MAC protocols, enhanced by online adaptation, while also proposing a novel strategy for schedule-based MAC protocols.

A canonical jamming detection method notes that no single feature is sufficient for reliable jammer detection [16]. The authors have proposed two major strategies: one that combines Packet Delivery Ratio (PDR) with signal strength features, and another that integrates PDR with location information, with both relying on predetermined detection thresholds. Our system leverages two complementary features for jamming detection: signal strength for initial detection and packet-based features as feedback for threshold adaptation.

Several prior studies employ machine learning algorithms for jamming detection given their robustness in autonomous parameter tuning [5], [17], [18]. However, these works rely on supervised learning approaches that require labeled datasets to train detection models. Unfortunately, supervised algorithms are highly dependent on the quality and comprehensiveness of the training dataset [19], [20]. In dynamic environments, it is impractical to collect such a dataset encompassing all possible scenarios, presenting a significant challenge for achieving high detection accuracy with supervised learning approaches.

III. System Design

In this section, we first provide a brief overview of AJDet. Next, we describe the jamming detection and online adaptation mechanisms employed by AJDet. Finally, we offer a concise description of our system's implementation.

A. System Overview

AJDet is an online, self-adaptive jamming detection system that comprises two primary functions: detection and adaptation, as illustrated in Figure 2. It is designed to identify jamming attacks that significantly impact packet delivery. Since jammers with higher transmission power are more likely to interfere with communications [11], we see signal strength as a valuable feature for jamming detection. Moreover, because jamming attacks intend to disrupt communication, packet delivery performance serves as a useful metric for assessing their existence. Specifically, AJDet utilizes signal strength, i.e., RSSI, for jamming detection, and leverages real-time packet delivery performance as a guiding metric to dynamically adjust detection thresholds. This adaptive approach ensures the accurate detection of jamming attacks. Table I summarizes typical jamming detection scenarios along with the corresponding actions taken by the system.

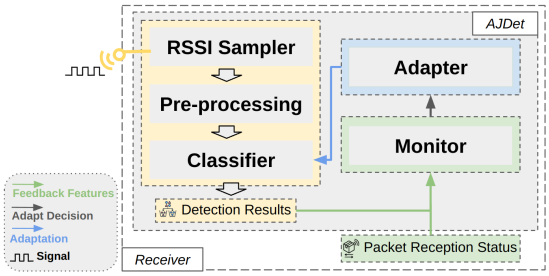


Fig. 2: Architecture of AJDet, consisting of two primary functions: detection and adaptation.

TABLE I: Jamming Detection Scenario and Adaptation Action

PDP*	Detection Result+	Typical Scenarios and Actions
True	No Jammer	Normal operation; no action needed
True	Jammer	False positive; system adjustment required
False	No Jammer	False negative; system adjustment required-
False	Jammer	Detected jamming that disrupts PDP

* **PDP** refers to Packet Delivery Performance: True indicates no issues in packet reception, False indicates otherwise.
\+ **Jammer**: detected jamming; **No Jammer**: no jamming detected.
\- Report sender failure (e.g., battery depletion) when no jamming is detected after adapting the threshold to the background noise level.

B. Jamming Detection Approach

AJDet is designed for integration into receivers. The system detects jamming attacks by acquiring signal strength measurement values measured by the receiver. Noteworthy, proactive jammers can also prevent senders from transmitting by causing the Clear Channel Assessment (CCA) check to fail. In this scenario, AJDet reports sender failure if it cannot detect the jammer. The jamming detection of AJDet relies on three main components: the RSSI sampler, pre-processing, and classifier. During detection, the RSSI sampler first acquires signal strength measurement values from the surrounding signals. The pre-processing component then processes these acquired RSSI values, and the classifier subsequently employs adaptive thresholds to determine the jamming type.

RSSI Sampler. The RSSI sampler captures the RSSI values measured by the receiver. These values measure the channel energy, reflecting both the intended signal strength of the receiving packets and external interference. RSSI values exhibit minor jitters over time due to multipath fading, interference, environmental changes, hardware variability, and dynamic power adjustments, all of which affect signal propagation and reception. Signals differing by less than approximately 3 dB generally have a similar effect on packet reception [12]. Therefore, AJDet quantizes the RSSI values into discrete *power-levels* using a 4 dB interval; for instance, *power-level 11*

corresponds to an RSSI range of -46 dBm to -43 dBm. This quantization helps to filter out minor fluctuations and save memory. Furthermore, the sampler accesses the communication medium via different methods depending on the MAC layer protocols, due to the various mechanisms used to manage the shared communication medium.

For CSMA networks, the RSSI sampler collects RSSI measurement over a period of time to capture the jamming signal. In addition to quantization, the sampler further compresses the samples using run-length encoding (RLE), a lossless data compression technique that replaces consecutive identical *power-levels* with a single *power-level* and a count, thereby reducing redundancy in repetitive data. The captured RSSI sequence is then represented as a sequence of two-dimensional (2D) vectors in the format *(power-level, duration)*. For example, an RSSI sequence such as *-93, -92, -93, -92, -29, -28, -70, -70, -70* would be mapped to *(1, 4), (17, 2), (7, 3)*. We empirically determine the sampling window, either collecting 500 samples or sampling for 1 second, to balance capture effectiveness and memory-computation efficiency.

In networks employing the TSCH protocol, the RSSI sampler uses a different strategy. Here, packet transmission follows a strict schedule, and the sampler can capture RSSI values only during reception (RX) timeslots, when the receiver node turns on its radio. During an RX timeslot, once a packet arrives at the MAC layer, the receiver waits until the entire packet is received before turning the radio off to complete subsequent operations, such as reading the packet. The waiting time depends on the packet size; for instance, a packet with a 64-byte payload has a time on air of approximately 2.6 ms, with the actual waiting time being slightly longer due to additional overhead. Given that the RX timeslot typically has a default duration of 10 ms, we set the sampling time window to 3 ms to avoid interfering with packet reception. The sampling process terminates after collecting 50 RSSI values or exceeding 3 ms.

The sampler operates under two scenarios: (1) when the receiver senses the arrival of a packet, and (2) when packet reception is disrupted and the radio is turned on. Samples collected in the first scenario are used to detect reactive jammers, while those from the second scenario are employed to detect proactive jammers. In the second scenario, the sampler is executed five times during each packet interval to reduce computational overhead, which is sufficient for detecting proactive jammers.

Pre-processing. The pre-processing component also applies different methods for different networks. For networks employing the CSMA protocol, the pre-processing component applies k-means clustering to group 2D vectors [21]. It designates the centroid of each cluster as the representative of its respective class. The intuition behind this is that the vector with similar power levels and duration are likely from the same jammer [22]. For networks utilizing the TSCH protocol, this component computes the mean *power-level* from samples for proactive jammer detection. For reactive jammer detection, AJDet determines the gap between the highest and lowest *power-level* during packet reception. Figure 3 shows the results

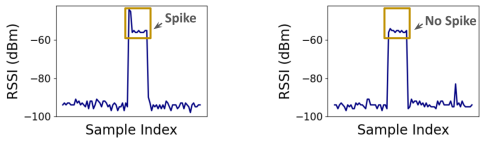


Fig. 3: The spike in the RSSI trace reveals the presence of a reactive jammer. The left trace shows the trace when a reactive jammer successfully disrupts packet reception, while the right trace shows an unsuccessful attempt.

of an experiment to motivate this design: Here, the sender persistently sends packets to the receiver at 5-second intervals, regardless of jamming. We use reactive jammers with different transmission strengths (+ 8dBM, +4 dBm, and 0dBm) and deploy them at different distances (1.5 and 2.5 meters) from the receiver. For each jamming scenario, we perform 500 packet transmissions. Our experiments show that there is always an apparent spike, when the reactive jammer disrupts packet transmission successfully. Hence, the detection of the spike reveals the presence of the reactive jammer.

Classifier. The classifier categorizes jamming attacks or other network issues (e.g., a sender battery failure). For networks employing the CSMA protocol, the classifier utilizes an adaptive *power-level* threshold for detection and a fixed duration threshold for classification. In contrast, for networks employing the TSCH protocol, the classifier applies both an adaptive *power-level* threshold and an adaptive *power-level* gap threshold for classification. Specifically, the classifier utilizes the *power-level* gap threshold for reactive jammer detection and the *power-level* threshold for proactive jammer detection. Proactive jammer detection occurs only when packet transmission is disrupted. Note that the system not only relies on the captured RSSI values but also considers packet delivery performance. It jointly analyzes real-time packet delivery performance and detection results, using them as feedback to enable adaptive thresholds through our online iterative adaptation approach, which is introduced in the next section.

C. Online Iterative Adaptation Approach

The online iterative adaptation approach enables the system to dynamically self-adapt in response to changing conditions, thereby ensuring optimal performance in real time. This approach consists of two key components: the monitor and the adapter, which operate in a continuous feedback loop, as shown in Figure 4.

Objective definition and system initialization. The objective is to achieve and maintain high jamming detection accuracy by dynamically adapting the detection thresholds. Thresholds are initialized using empirical values, potentially through preliminary runs in the target environment. The system then iteratively adjusts the threshold towards values that guarantee the elimination of false positives.

Monitor. The monitor generates feedback by jointly analyzing real-time packet delivery performance and detection results. Specifically, the monitor reports a false positive when a jammer is detected despite normal packet reception, and a

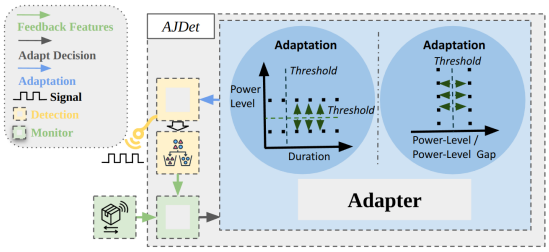


Fig. 4: AJDet Adaptation Implementations. For CSMA networks, the system utilizes an adaptive *power-level* threshold; for TSCH networks, the system employs two adaptive thresholds: *power-level* and *power-level* gap thresholds.

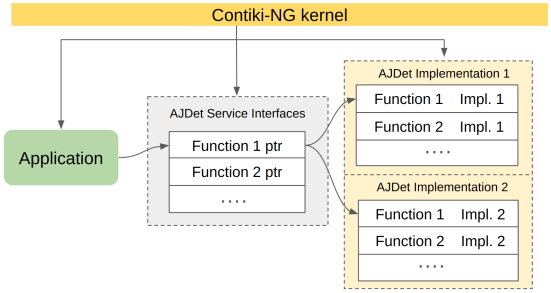


Fig. 5: Interface Specification of AJDet

false negative when a jammer is not detected despite abnormal packet reception, as summarized in Table I.

Adapter. The adapter adjusts detection thresholds based on feedback received from the monitor. A higher threshold increases the risk of false negatives, whereas the opposite introduces false positives. The reason is that a high threshold is more likely to fail to detect jamming signals that are not very strong but still sufficiently powerful to disrupt packet reception. In contrast, a low threshold is more likely to incorrectly detect environmental noise as jamming, despite normal packet reception. Accordingly, the adapter increases the thresholds when false positives are detected and decreases the thresholds when false negatives occur.

D. Implementation

We design and implement AJDet as a service within the Contiki-NG operating system [23]. Applications interact with AJDet via standardized interfaces that conform to an online self-adaptive interface specification. This specification facilitates an interface-oriented design that enhances modularity and extensibility, enabling seamless component upgrades and ensuring compatibility across various implementations, as illustrated in Figure 5. We provide distinct implementations tailored to networks employing CSMA and TSCH protocols.

IV. Evaluation

In this section we evaluate AJDet with threshold adaptation, detection accuracy, and memory overhead as performance metrics. Experiments are conducted on both a small scale network using CSMA and TSCH MAC baselines and also on a multi-hop TSCH-RPL network to demonstrate its effectiveness in real networking scenarios.

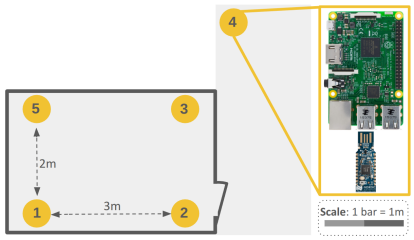


Fig. 6: Testbed layout and setup: Six ceiling-mounted Raspberry Pi nodes are deployed in an office and corridor, each connected to an nRF52840 dongle.

A. Setup

Testbed. We first utilize a small-scale indoor IoT testbed composed of five Nordic Semiconductor nRF52840 platforms [7] running Contiki-NG [23], as shown in Figure 6. Node 4 is located on the ceiling of a corridor, just outside an office where the remaining nodes are installed on the ceiling. We evaluate two MAC protocols: CSMA and TSCH. For testing purposes, TSCH is configured to operate on a single channel. All devices employ UDP and the RPL Routing Protocol [24]. Note that the packet transmission interval is 4 s, and experiments are conducted during regular working hours on weekdays in an office with consistent, moderate background noise.

Jammers. We evaluate the performance of AJDet under two jamming scenarios: a constant jammer and a reactive jammer. The reactive jammer features a very short jamming duration of 312 μs—just enough to corrupt an in-flight packet, and is activated when a packet transmission is detected. In contrast, the constant jammer continuously occupies the channel, representing a worst-case scenario, providing a baseline for assessing network resilience. The reactive jammer is particularly challenging due to its ability to selectively interfere with ongoing transmissions, and its brief jamming duration further complicates detection [25]. We implement both jammers using the nRF52840 platform, and both of them utilize a modulated carrier signal by transmitting a pseudo-random bit sequence, and they are categorized based on two key factors: the interfering distance and transmission power [11].

B. Threshold Adaptation

We begin by demonstrating the threshold adaptation capability of AJDet. In Section I, our initial experiments reveal the limitations of traditional methods that rely on fixed, predetermined jamming detection thresholds. Our goal here is to show that the adaptation approach autonomously converges to the optimal threshold value for different network setups in real time. As described in Section III-B, AJDet captures and quantizes RSSI values into *power-levels* to filter out minor fluctuations. The system employs both an adaptive *power-level* threshold and a *power-level* gap threshold for jamming detection, as introduced in Section III-C.

Setup. We conduct experiments in our indoor testbed as described in Section IV-A. Additionally, an extra jammer node is used to vary the jammer-receiver distance. We evaluate different network conditions by altering both the sender-receiver

and jammer-receiver distances. Note that the sender-receiver distance influences signal strength, transmission power, link quality, and overall network reliability, whereas the jammer-receiver distance affects interference strength and jamming effectiveness.

In the CSMA network, we evaluate the sequential adaptation of the power-level threshold under constant jamming attacks. In contrast, when the reactive jammer successfully corrupts packet transmissions, the number of retransmissions exceeds the expected transmission count (ETX), and the network marks the receiver as unreachable. At that point, no further transmissions can be detected.

1) Threshold Adaptation for Minimizing False Positives: The goal of this experiment is to demonstrate that the adaptation approach can adjust the threshold to minimize false positives. As discussed in Section III-C, a low threshold raises the likelihood of false positives. Thus, the system should increase the threshold when false positives occur. In addition, this experiment also demonstrates how different network conditions affect the threshold values.

Setup. In this experiment, we do not deploy a jammer. We deploy one sender on Node 2 and two receivers, respectively, on Node 1 and Node 3 in our testbed. Two receivers simultaneously operate at different distances from the sender: 2 m and 3 m. To fully showcase the adaptation process for minimizing false positives, we initialize the threshold to zero for both receivers. We perform experiments for the *power-level* threshold for CSMA Networks and the *power-level* gap threshold for TSCH Networks. We perform 1,000 detection runs for each. In TSCH networks, we do not conduct this experiment for the *power-level* threshold. The reason is that this threshold does not cause false positives, since the system utilizes it for proactive jammer detection only when packet reception is disrupted.

Results. Figure 7a plots the adaption process of the *power-level* threshold in networks employing CSMA protocols. The results show that the threshold adapts to different values depending on different sender-receiver distances. The changes in the threshold in the earlier parts of the experiment show that AJDet increases its threshold given that packets are successfully received (no existing jammers). Therefore, a shorter distance link stabilizes at a higher power-level threshold under similar network conditions due to stronger packet receptions. Figure 7b shows that the *power-level* gap threshold increases to the same value, because the RSSI values during a single packet reception are generally stable, as they reflect the signal strength at the moment of reception. Again, the threshold in the plots increases due to signal fluctuations which makes the power-level gap exceeding the threshold but without any packet loss. Environmental noise may lead to false positives and differ at different locations. This is why the two receivers increase their threshold at different times.

2) Threshold Adaptation For Minimizing False Negatives: The goal of this experiment is to show that our threshold adaptation can adjust the threshold to minimize false negatives. As discussed in Section III-C, a high threshold raises the likelihood of false negatives. Thus, the system should decrease the threshold when false negatives occur. In addition, this experiment also evaluates how different jammer-receiver distances affect the threshold values.

Setup. In this experiment, we deploy one sender and two receivers. The sender is located on Node 2. To demonstrate the impact of jammer-receiver distance, the sender is positioned equidistant from the two receivers, at a distance of 2 meters. Specifically, one receiver is located on Node 3, while the other is positioned between Node 1 and Node 2, exactly 2 meters away from Node 2 via a USB breakout connected by a 2-meter wire. We deploy the jammer in two distinct positions, resulting in two scenarios. In the first scenario, the jammer is equidistant from the sender and both receivers (2 meters away from each). In the second scenario, the jammer is placed 2 meters from one receiver and 1 meter from the other. This setup is used to examine the impact of jammer-receiver distance on threshold initialization in both CSMA and TSCH networks. When the jammer-receiver distance is uniform, we initialize the *power-level* threshold with two different values (15 and 20) to assess the effect of threshold variation. Conversely, when the jammer-receiver distance varies, we initialize the *power-level* threshold with a value of 20.

Results. We perform experiments on both CSMA and TSCH networks. As Figures 7c and 7e show, varying the jammer-receiver distance results in different threshold values, whereas Figures 7d and 7f illustrate that a consistent jammer-receiver distance yields the same threshold value. This is expected since the jammer-receiver distance determines the strength of the jamming signals observed at the receiver. A shorter distance results in stronger signal strength, while a consistent distance yields the same strength. AJDet adjusts the threshold to an appropriate value to detect jamming signals. The same jamming strength requires the same threshold, while a weaker signal needs a lower threshold than a stronger signal.

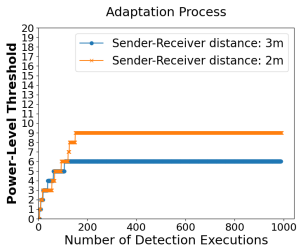
In TSCH networks, as Figures 7e and 7f show, AJDet adapts the threshold and detects the jammer before the node is excluded from the TSCH schedule due to jamming. Persistent packet loss or link failures may cause a node to be removed from the schedule, and once removed, it no longer participates in reception timeslots. Consequently, the system would cease functioning because it relies on sampling RSSI values during these timeslots. Our experimental results validate the system's effectiveness under these conditions.

Overall our experiments here show that AJDet can adapt its jamming detection threshold according to network dynamics, including variations in sender-receiver and jammer-receiver distances.

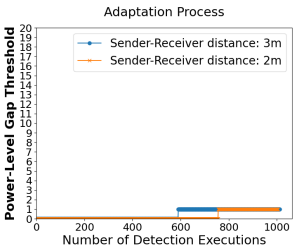
C. Detection Accuracy of AJDet

In this section, we evaluate the detection accuracy of AJDet for jammers with different configurations.

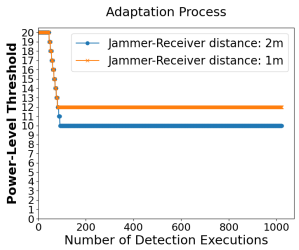
Setup. We use one sender-receiver pair, positioned 2 meters apart, deployed on Node 2 and Node 3, respectively, and evaluate the detection accuracy in both TSCH and CSMA networks. We use constant and reative jammers with two



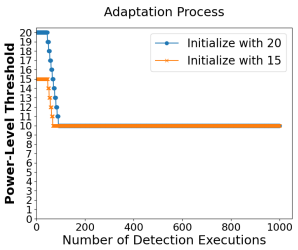
(a) Varying receiver-sender distances result in different threshold values in CSMA networks.



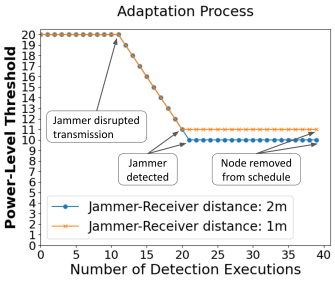
(b) Different receiver-sender distances yield similar *power-level* gap thresholds in TSCH networks.



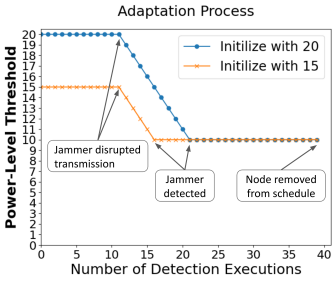
(c) The same initial threshold results in varying values with different jammers in CSMA networks.



(d) Different initial thresholds result in the same value with the same jammer in CSMA networks.



(e) The same initial threshold results in varying values with different jammers in TSCH networks.

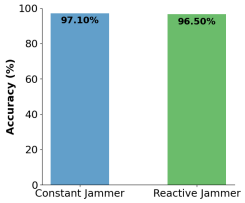


(f) Different initial thresholds result in the same values with the same jammer in TSCH networks.

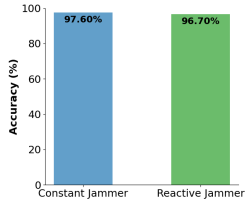
Fig. 7: Threshold adaptation across scenarios.

different transmission powers: 0 dBm and 8 dBm. During the evaluation, we move the jammers within a designated area, allowing them to be at varying distances from the receiver. For jammers with 8 dBm transmission power, the movement range covers jammer-receiver distances between 0.5 and 2 m. In contrast, for jammers with 0 dBm, the movement range is limited to a jammer-receiver distance between 0.5 and 1.5 m, as they cannot disrupt packet transmission beyond 1.5 m. Each experiment is performed for 30 minutes.

Results. As Figure 8 shows, AJDet achieves 96+% detection accuracy for both CSMA and TSCH networks. This demonstrates that our self-adaptation scheme functions as expected. Note that the detection accuracy is slightly higher with constant jammers, given the limited duration of reactive jamming. Most missed detections occur when the jammer is moved farther from the receiver. When the jammer-receiver distance increases, the signal strength will be lower, necessitating a lower threshold for detection. However, during the adaptation, some missed detections may occur. Additionally, missed detections can also result from the sampler recording a lower signal strength than usual, possibly due to sampler

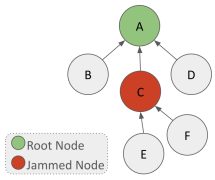


(a) CSMA Detection Accuracy.

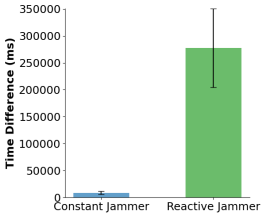


(b) TSCH Detection Accuracy.

Fig. 8: Detection accuracy for CSMA and TSCH networks



(a) The network topology.



(b) Time difference.

Fig. 9: Jamming Detection–Rerouting Time Difference.

limitations or a failure in the jammer's radio transmission.

D. Performance in a Multi-hop RPL-TSCH Network

Motivation. We design this experiment to demonstrate AJDet's performance in a larger network. We examine how fast jammed nodes can be detected compared to RPL's rerouting mechanism as a way to exploit AJDet for faster rerouting.
Setup. As shown in Figure 9a, we exploit a multi-hop TSCH-RPL network with six nodes. One as the root node, and the rest as children. We integrate AJDet into all child nodes and record when intermediate nodes detect jamming and when a child node selects a new parent. The time is based on the Absolute Slot Number (ASN) and TSCH's timeslot duration. We use constant and reactive jammers with low signal strength to interrupt only the intermediate node, Node C. For testing purposes, TSCH is configured to operate on a single channel.
Result. The result shows that AJDet can always report jamming attacks before the rerouting starts. As illustrated in Figure 9b, AJDet has a shorter mean time difference between detecting jammer and rerouting for the constant jammer, which immediately blocks reception by continuously occupying the channel (8805 ms vs. 277572 ms). In contrast, the low-power reactive jammer partially disrupts reception, allowing some packets through and delaying rerouting decisions. The results indicate that RPL could achieve faster rerouting under jamming attacks if it utilizes the detection results.

E. Memory Overhead of AJDet

We measure the memory footprint of AJDet's implementation as a service in Conitki-NG using the GNU Arm Embedded Toolchain [26]. We use nRF52840 platforms that feature 1024 KB of ROM and 256 KB of RAM. We plot the results in Figure 10. For CSMA networks, AJDet consumes 9.3 KB of RAM and 7.4 KB of ROM on the receiver side. AJDet introduces no memory overhead on the sender side since it is designed for receiver integration and does not compile into the sender firmware. The receiver firmware, including OS and

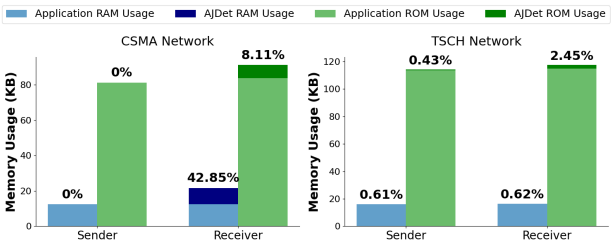


Fig. 10: AJDet adds very little memory overhead, making it applicable for resource-constrained devices.

application logic, requires 21.7 KB of RAM and 91.2 KB of ROM. For TSCH, the system adds memory overhead on the sender and receiver side. The reason is that AJDet adds logic in TSCH to notify the system when the RX time slot starts. The application firmware always includes TSCH as part of the network stack. Specifically, AJDet uses 0.1 KB RAM on both the sender and receiver, as well as 0.5 KB and 2.8 KB of ROM usage on the sender and receiver. The sender firmware uses 16.3 KB RAM and 117.5 KB ROM, while the receiver firmware uses 16.2 KB RAM and 114.1 KB ROM. Compared to TSCH, in CSMA, AJDet consumes more memory. This is because CSMA requires more RSSI sampling, as it lacks a transmission schedule, unlike TSCH. In summary, the result shows that AJDet is a lightweight system that is applicable to highly constrained IoT devices.

V. Discussion

Network Scalability. AJDet integrates seamlessly with the receiver without requiring information from neighboring nodes. Its lightweight design and use of intuitive metrics render it testbed-agnostic, thereby enabling it to scale efficiently to larger network sizes.

Initial Threshold Selection. AJDet requires an initial configuration of the jamming detection threshold. While determining a fixed threshold under dynamic network conditions is challenging, its adaptive mechanism continuously updates the threshold in response to changing channel conditions. This flexibility assures its practicality in diverse IoT network settings.

Attack Scalability. While we focus on two representative types of jamming attacks, AJDet is designed with a simple architecture and independence from proprietary metrics, facilitating its applicability to other forms of jamming attacks.

VI. Conclusion

In this work, we present AJDet, a lightweight online self-adaptive jamming attack detection system. The work provides an online iterative adaptation approach and two implementations for IoT networks with different MAC layer protocols. Our experiments with real hardware show that the adaptation works as expected. AJDet achieves over 96.5% accuracy in proactive and reactive jamming detection.

Acknowledgment

This work is partially supported by the Swedish Science Foundation (SSF) and the Korean Ministry of Science and ICT (MSIT) through IITP (RS-2024-00434743).

References

[1] M. Kocakulak and I. Butun, "An overview of wireless sensor networks towards internet of things," in *IEEECCWC*. Ieee, 2017.

[2] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1727–1765, 2016.

[3] H. Pirayesh and H. Zeng, "Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey," *IEEE communications surveys & tutorials*, vol. 24, no. 2, pp. 767–809, 2022.

[4] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[5] O. Puñal, I. Aktaş, C.-J. Schnelke, G. Abidin, K. Wehrle, and J. Gross, "Machine learning-based jamming detection for ieee 802.11: Design and experimental evaluation," in *IEEE WoWMoM*. IEEE, 2014, pp. 1–10.

[6] J. Kanwar, N. Finne, N. Tsiftes, J. Eriksson, T. Voigt, Z. He, C. Åhlund, and S. Saguna, "Jamsense: Interference and jamming classification for low-power wireless networks," in *IFIP WMNC*. IEEE, 2021.

[7] N. Semiconductor, "nrf52840," 2025. [Online]. Available: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840

[8] R. P. Foundation, "Raspberry pi," https://www.raspberrypi.org, 2025, accessed: January 22, 2025.

[9] I. Demirkol, C. Ersoy, and F. Alagoz, "Mac protocols for wireless sensor networks: a survey," *IEEE Communications magazine*, vol. 44, no. 4, pp. 115–121, 2006.

[10] IEEE, "Ieee standard for low-rate wireless networks," *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, pp. 1–800, 2020.

[11] C. A. Boano, Z. He, Y. Li, T. Voigt, M. Zúnniga, and A. Willig, "Controllable radio interference for experimental and testing purposes in wireless sensor networks," in *IEEE LCN*, 2009.

[12] M. Takai, J. Martin, and R. Bagrodia, "Effects of wireless physical layer modeling in mobile ad hoc networks," in *ACM international symposium on Mobile ad hoc networking & computing*, 2001.

[13] S. Vadlamani, B. Eksioglu, H. Medal, and A. Nandi, "Jamming attacks on wireless networks: A taxonomic survey," *International Journal of Production Economics*, vol. 172, pp. 76–94, 2016.

[14] T. Hamza, G. Kaddoum, A. Meddeb, and G. Matar, "A survey on intelligent mac layer jamming attacks and countermeasures in wsns," in *IEEE Vehicular Technology Conference (VTC-Fall)*. IEEE, 2016.

[15] Z. He and T. Voigt, "Precise packet loss pattern generation by intentional interference," in *IEEE DCOSS*, 2011.

[16] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *ACM international symposium on Mobile ad hoc networking and computing*, 2005.

[17] A. Hussain, N. Abughanam, J. Qadir, and A. Mohamed, "Jamming detection in iot wireless networks: An edge-ai based approach," in *International conference on the internet of things*, 2022.

[18] F. tu Zahra, Y. S. Bostanci, and M. Soyturk, "Lstm-based jamming detection and forecasting model using transport and application layer parameters in wi-fi based iot systems," *IEEE Access*, 2024.

[19] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE intelligent systems*, vol. 24, no. 2, pp. 8–12, 2009.

[20] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[21] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*, 1967.

[22] V. Iyer, F. Hermans, and T. Voigt, "Detecting and avoiding multiple sources of interference in the 2.4 ghz spectrum," in *European Conference on Wireless Sensor Networks (EWSN)*, 2015.

[23] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, "The Contiki-NG open source operating system for next generation IoT devices," *SoftwareX*, vol. 18, p. 101089, 2022.

[24] T. Winter *et al.*, "Rpl: Ipv6 routing protocol for low-power and lossy networks," Tech. Rep., 2012.

[25] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders, "Short paper: Reactive jamming in wireless networks: How realistic is the threat?" in *ACM conference on Wireless network security*, 2011, pp. 47–52.

[26] "GNU Arm Embedded Toolchain," https://developer.arm.com/downloads/-/gnu-rm, accessed: Feb. 12, 2025.