Detecting and Mitigating Jamming Attacks in IoT Networks Using Self-Adaptation

Maxim Reynvoet
Dept. of Computer Science
Kulak, KU Leuven, Belgium
maxim.reynvoet@student.kuleuven.be

Omid Gheibi
Dept. of Computer Science
KU Leuven, Belgium
omid.gheibi@kuleuven.be

Federico Quin
Dept. of Computer Science
KU Leuven, Belgium
federico.quin@kuleuven.be

Danny Weyns
KU Leuven, Belgium
Linnaeus University, Sweden
danny.weyns@kuleuven.be

***Abstract*—Internet of Things (IoT) networks consist of small devices that use a wireless communication to monitor and possibly control the physical world. A common threat to such networks are jamming attacks, a particular type of denial of service attack. Current research highlights the need for the design of more effective and efficient anti-jamming techniques that can handle different types of attacks in IoT networks. In this paper, we propose DeMiJA, short for Detection and Mitigation of Jamming Attacks in IoT, a novel approach to deal with different jamming attacks in IoT networks. DeMiJA leverages architecture-based adaptation and the MAPE-K reference model (Monitor-Analyze-Plan-Execute that share Knowledge). We present the general architecture of DeMiJA and instantiate the architecture to deal with jamming attacks in the DeltaIoT exemplar. The evaluation shows that DeMiJA can handle different types of jamming attacks effectively and efficiently, with neglectable overhead.**

***Index Terms*—IoT, jamming attacks, self-adaptation**

I. Introduction

An Internet of Things (IoT) network consists of a collection of small low-power devices that are connected through a wireless network to monitor and control the physical world using sensors and actuators [1]. IoT has been widely used in diverse domains such as smart cities [2], healthcare [3], manufacturing [4], and farming [5]. Yet, wireless networks are vulnerable to a variety of outside attacks [6]. One type of attack that has not been completely alleviated is a *jamming attack* [7]. In a jamming attack an adversary (the jammer) emits a radio signal that interferes with regular communication, corrupting it or making the channel appear busy [8].

Traditionally, dealing with jamming attacks consists of two stages: *detection* and *mitigation*. Detection techniques aim at determining the presence of a jammer automatically [9], [10], and optionally classifying the type of jamming attack [11]. Mitigation (or anti-jamming) techniques aim at alleviating the effect of the attack on the quality of services, such as packet loss and energy consumption of IoT devices. Note that the detection is a prerequisite for the mitigation.

The recent comprehensive survey of Pirayesh et al. [7] highlights the need for the design of more effective and efficient detection and mitigation techniques that can handle different types of attacks in IoT networks automatically. Hence, the research problem we tackle in this paper is:

> *How to protect wireless IoT systems against jamming attacks, automatically, effectively, and efficiently?*

With automatically, we mean that the system deals with jamming attacks without the need for human intervention. With effectiveness, we mean that jamming attacks are accurately detected and adequately mitigated. Finally, with efficiently, we mean that the jamming attacks are mitigated swiftly and the overhead caused by the solution is limited.

To tackle this problem, we present DeMiJA, a novel architecture-based adaptation approach [12]–[17] based on the MAPE-K reference model [18]–[20] that aims at providing a systematic and automatic approach to detect and mitigate jamming attacks in the network. This approach allows leveraging state-of-the-art techniques for detecting and mitigating jamming attacks, see e.g. [7].

II. Background

We start with introducing DeltaIoT that we use for evaluating DeMiJA. Next, we explain different types of jamming attacks, illustrated for DeltaIoT. Lastly, we discuss metrics for detection and two general types of mitigation techniques.

A. DeltaIoT

DeltaIoT [1] is an IoT network that can be used for research on self-adaptation. Figure 1 shows the topology of network.

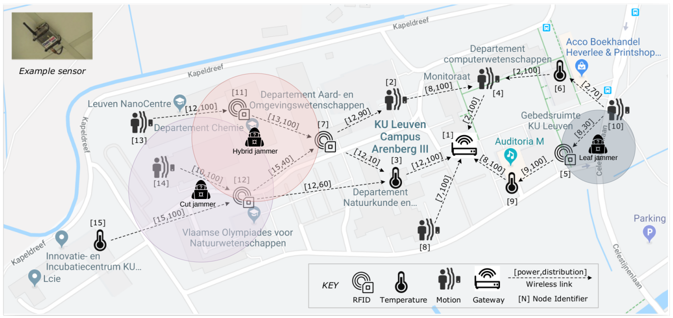


Fig. 1: DeltaIoT topology [20] with jammers (depicted by coloured jamming area) in the network.

1) Characteristics: DeltaIoT consists of 15 battery powered LoRa-based[1] motes. Each mote is equipped with a sensor, e.g. temperature, motion, or RFID. Central to the network is a gateway that is connected with a user application to monitor

[1]LoRa short for Long Range [21]

the environment. Motes send their sensor data to the gateway using multi-hop communication along the links of the network. This communication is time-synchronized, organized in cycles.

The network is subject to uncertainties that effect the proper functioning of the system. Examples are noise in the environment and load in the network due to e.g. the presence of people. Our focus in this paper is on uncertainty caused by malicious motes that jam the network (inject messages) to disrupt transmissions causing loss of data of users.

2) Adaptation: To deal with jamming attacks, we consider two types of adaptations: packet re-routing and channel hopping. These mitigation methods are supported by DeltaIoT.

Re-routing applies to motes with two (or more) children in the network. In particular, such motes can choose how to distribute the transmission of their packets in the direction of the gateway. Re-routing boils down to changing this distribution such that packets will be diverted from jammed links. Channel hopping on the other hand involves changing to another subset of frequencies on the frequency band that are used by a mote to transmit its data in the network.

B. Types of Jamming Attacks

A jamming attack is a type of Wireless Denial of Service (WDoS) attack, where an adversary (the jammer) uses a malicious node that emits a radio signal to interfere with the transmission and/or reception of legitimate communication [8]. According to Xu et al. [22] jammers can be classified into 4 different basic categories. In this paper, we will particularly focus on three common types of jammers: (i) a constant jammer, (ii) a reactive jammer, and (iii) a random jammer.[2]

a) Constant jammer: Continuously emits a radio signal to disrupt communications. It tries to block communication at all times on the channel, irrespective of the traffic pattern. In our evaluation we use a multi-channel constant jammer, jamming all network channels constantly.

b) Reactive jammer: Monitors a network channel and, if it detects a transmission, starts jamming the channel. We use a single-channel reactive jammer for our evaluation.

c) Random jammer: Alternates between sleeping and jamming, with t_j the time allocated to jamming and t_s the idle time. Values t_j and t_s can be randomly chosen or fixed. While the jammer is active, it can behave like any other type of jammer, i.e., constant or reactive. We use a single-channel random jammer in the evaluation of DeltaIoT with t_j=13 and t_s=5 cycles, and reactive behavior while being active.

C. Detection Metrics

The first step in dealing with jamming attacks is to accurately identify the presence of a jammer. In this section we present different metrics for intrusion detection of the physical layer, the primary focus of our work in this paper.

[2]Deceptive jammers that are situated at the MAC layer are out of scope.

1) RSSI: Received Signal Strength Indicator (RSSI) is an indicator for the actual value of the Received Signal Strength (RSS). RSS is defined as the power output (expressed in watts) of a transmitter as it was received by the antenna of a receiver that is placed at a distance from the transmitter [23]. The presence of a jammer can affect the statistical distribution of the measured RSSI values in the nodes that are being jammed. Xu et al. [22] have shown that RSSI can be used to detect potential jammers in the network.

2) PDR: The Packet Delivery Ratio (PDR) is defined as the number of packets that were received out of the total number of packets that were sent. Threshold values can be set to differentiate between a jammer and a congested network state when checking error correction codes. Unlike RSSI, PDR requires a specified timeframe [22].

D. Mitigation Techniques

We discuss two types of mitigation techniques that can be used to deal with jamming attacks in IoT networks.

1) Physical Layer techniques: These techniques aim at decreasing the jamming-to-signal ratio or avoiding collisions altogether. To that end, various techniques can be used such as increasing the power with which transmissions are sent, reducing the distance between the transmitter and the receiver, using carrier sense with collision avoidance, etc.

2) Spread Spectrum: This type of mitigation technique uses alterations to the spread spectrum technique used to transmit messages in the network. Examples are direct sequencing, frequency hopping, or a combination of these two [8]. These type of techniques spread out the signal over a wider frequency by making use of a noise-like sequential signal structure.

III. DeMiJA

We now propose DeMiJA, a novel self-adaptive approach for protecting IoT systems against jamming attacks. We start with outlining requirements, scope, and assumptions. Then we present the general architecture of DeMiJA. Finally, we instantiate and illustrate the architecture for DeltaIoT.

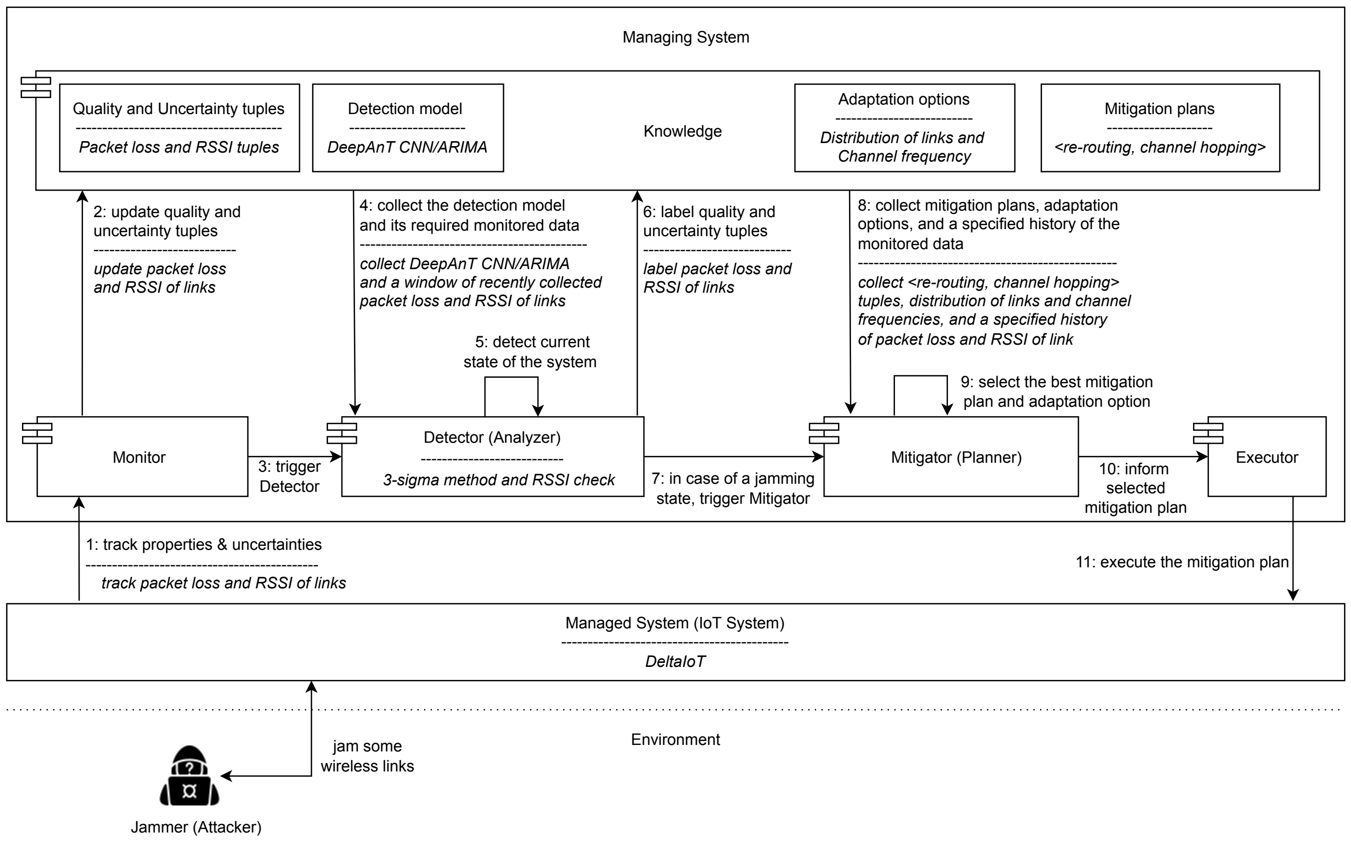
A. Requirements, Scope, and Assumptions

DeMiJA targets wireless networks, in particular IoT networks. The approach should support dealing with constant jammers, reactive jammers, and random jammers. We lay out three requirements for the architecture of DeMiJA:

- R1 DeMiJA should comply with architecture-based adaptation (1) and the MAPE-K reference model (2);
- R2 DeMiJA should provide the means for monitoring the managed system (1) and executing adaptation actions to deal with different types of jamming attacks (2);
- R3 DeMiJA should support the use of different techniques for the detection (1) and mitigation (2) of different types of jamming attacks.

We make the following three assumptions for DeMiJA:

- A1 A history of data that represents the regular behavior of the system is available.



Managing System
Quality and Uncertainty tuples
Packet loss and RSSI tuples
Detection model
DeepAnT CNN/ARIMA
Knowledge
Adaptation options
Distribution of links and Channel frequency
Mitigation plans
<re-routing, channel hopping>
2: update quality and uncertainty tuples
update packet loss and RSSI of links
4: collect the detection model and its required monitored data
collect DeepAnT CNN/ARIMA and a window of recently collected packet loss and RSSI of links
6: label quality and uncertainty tuples
label packet loss and RSSI of links
8: collect mitigation plans, adaptation options, and a specified history of the monitored data
collect <re-routing, channel hopping> tuples, distribution of links and channel frequencies, and a specified history of packet loss and RSSI of link
5: detect current state of the system
9: select the best mitigation plan and adaptation option
Monitor
3: trigger Detector
Detector (Analyzer)
3-sigma method and RSSI check
7: in case of a jamming state, trigger Mitigator
Mitigator (Planner)
10: inform selected mitigation plan
Executor
1: track properties & uncertainties
track packet loss and RSSI of links
11: execute the mitigation plan
Managed System (IoT System)
DeltaIoT
Environment
jam some wireless links
Jammer (Attacker)

Fig. 2: The MAPE-based architecture of self-adaptation for protecting against jamming attacks, with instantiation for DeltaIoT below dashed lines.

A2 The detection method that is used has been initialized (for example based on historical data of the system).

A3 The location of the gateway is not known to the jammer; we exclude jamming attacks that target all communication links to/from the gateway.

B. General architecture

Figure 2 shows the architecture of DeMiJA to defeat jamming attacks. The architecture complies with architecture-based adaptation (R1 first part), separating the managed system (IoT system) that deals with the domain concerns from the managing system that deals with the jamming attacks and is deployed at the gateway of the IoT network. Internally, the managing system complies with the MAPE-K model (R1 second part). We explain briefly the elements and workflow.

The knowledge keeps track of all the necessary runtime data that needs to be shared between the MAPE elements to realize their functions. This includes:

- A set of recent quality & uncertainty tuples $\langle t_i, k_i \rangle$, each tuple representing the relevant quality values of the system and uncertainties k_i, at a specific time t_i;
- A detection model: a pre-trained model that uses recently collected data of the system to predict whether the managed system is subject to jamming attacks;
- The adaptation options: possible configurations that can be used to adapt the managed system;
- The mitigation plans: an (ordered) set of plans that are available to adapt the managed system. A subset of them can be selected to apply an actual adaptation.

The monitor tracks properties and uncertainties relevant to jamming attacks and updates the knowledge with this data (1-2) (R2 first part). The detector (3) collects the data (4) for detection, and uses this data to identify the status of the system as ”jammed” or ”regular” (5) and label quality and uncertainty tuples (6) (e.g. jamming detected in a specific part of the system). Different detection methods can be plugged in (R3 first part). In case a ”jammed” status is identified, the mitigator (7) collects mitigation plans, adaptation options and a history of monitored data (8) to determine the best mitigation plan and corresponding adaptation option(s) (9). Different mitigation strategies can be used (R3 second part). Finally the executor (10) enacts the chosen mitigation plan (11) on the IoT system (R2 second part).

C. Instantiation of DeMiJA Architecture to DeltaIoT

We illustrate the DeMiJA architecture now for DeltaIoT. Figure 2 shows an instantiation of the architecture, specified under the dashed line in each component. We explain the different elements and how they interact to deal with jamming attacks, illustrated in the managed system (DeltaIoT).

a) DeltaIoT (Managed System): In Section II we outlined the details of the DeltaIoT network. We investigate now the three types of jammers based on their location that allow us to evaluate the proposed architecture, see Figure 1:

- Leaf jammer: jams the links of a mote located in the leaf of the network, e.g., motes 10, 15, 14, 13, and 8.
- Cut jammer: jams all (send and/or receive) links of a cut mote[3] in the mesh network, e.g., motes 12, 7, and 4.
- Hybrid jammer: jams both leaf and cut motes.

Note that each of these jammers can be a random, reactive, or a constant jammer, based on their jamming behavior.

b) Knowledge: For DeltaIoT, the following runtime data is maintained in the knowledge to handle jamming attacks:

- A set of recent quality & uncertainty tuples composed of packet loss and RSSI respectively (in the evaluation we use the tuples of the 30 last cycles);
- Two types of detection models: DeepAnT CNN and ARIMA statistical auto-regression model. For the evaluation, we trained these models using historical data of 500 cycles. We explain DeepAnT and ARIMA below.
- Two types of adaptation options: the distribution of messages sent over links and the channel frequency.
- Two types of mitigation plans: channel hopping and re-routing packets (in that order).

We briefly summarise DeepAnT and ARIMA. For a detailed explanation we refer to [25] and [26] respectively.

c) DeepAnT: a state-of-the-art semi-supervised anomaly detection method for time-series [25] that comprises two stages: training and decision-making. In the training stage, DeepAnT trains a convolutional neural network [27] (CNN) based on a window of historical data of the provided time-series to predict the next value in the series. After training the CNN, DeepAnT fits a normal distribution to the validation errors encountered during the training of the CNN. This results in a mean value μ and a standard deviation value σ that will be used for decision-making. In the decision-making stage, DeepAnt detects anomalies by first predicting the succeeding value in the time-series using the trained CNN, and then normalizing the prediction error made by the CNN. If the normalized prediction error exceeds a threshold[4], it will be reported as an anomalous event. In the evaluation, we use DeepAnT on the packet loss time-series.

d) Auto-Regressive Integrated Moving Average (ARIMA): another state-of-the-art semi-supervised anomaly detection method for time-series [26]. ARIMA follows the same workflow as DeepAnT. Contrary to DeepAnT that uses a CNN model to make predictions, ARIMA uses a statistical auto-regression model trained on historical data.

[3]In graph theory, a vertex in a graph is called a cut vertex, if removing it (and its directly connected edges), the corresponding graph will be disconnected, i.e., increasing the number of connected components [24]. Similarly, if we remove a cut mote and its send and/or receive links, there will be no path to reach the gateway from at least one mote in the network.

[4]We refer to [28] for the interested reader

IV. Evaluation

We evaluated DeMiJA using the DeltaIoT exemplar. For practical reasons, we used the DeltaIoT simulator [1] that provides a realistic simulation of the physical network (e.g., basic network interference along the different links is modeled based on field experiments). We enhanced the basic setup of DeltaIoT with support for multi-channel communication and RSSI modeling. Each jammer was modeled as a regular LoRa-based mote with its specific location and behavior that affected the communication with neighboring links. The tests were executed on a Lenovo ideapad L340-15IRH Gaming with an Intel(R) Core(TM) i7-9750H 2.60GHz CPU and 16 GB RAM. Further information and a reproduction package is available. [5]

A. Evaluation Scenarios

Table I lists the combinations of settings we used to evaluate DeMiJA. For the sake of brevity, we focus our evaluation on DeepAnT (similar results were obtained with ARIMA).

TABLE I: Evaluation scenarios

Method	Options	Extra features
Detection	DeepAnT	RSSI check on/off
Mitigation	Channel hopping & re-routing on/off	NA
Jammer	Hybrid, cut, leaf	NA

Combining the different options results in six scenarios per type of jammer (hence 18 scenarios in total). Three of these scenarios focus on detection (mitigation off); the other three focus on mitigation. For each scenario, we run 500 communication cycles of DeltaIoT, representing a total of about 70 hours wall clock time. The jammers can jam the network in the period from cycle 30 until cycle 120.

B. Evaluation Goals

We defined the following evaluation questions:

Q1 How effective is DeMiJA in detecting different types of jamming attacks?

Q2 How effective is DeMiJA in mitigating jamming attacks?

Q3 How efficient is DeMiJA in detecting and mitigating jamming attacks?

To evaluate Q1, we measure the percentage of cycles DeMiJA effectively detects jamming attacks for the different scenarios. In addition, we measure the false positives on jammed links and unjammed links[6].

To evaluate Q2, we measure the percentage of cycles DeMiJA effectively mitigates jamming attacks. Here we consider only scenarios with RSSI consistency check active.

To evaluate Q3, we measure the time it takes for DeMiJA to detect jamming attacks. We also measure the number of cycles it takes to mitigate different types of jamming attacks.

C. Evaluation Results

We report the results for the three evaluation questions.

[5]https://people.cs.kuleuven.be/danny.weyns/software/DeltaIoT/

[6]Jammed here refers to links that are located within the range of the jammer, and unjammed links are located outside the range of the jammer.

1) Detection Effectiveness: For the evaluation of detection effectiveness we consider the 6 scenarios. Table II shows the results for a reactive, random, and constant jammer respectively. We observe that DeepAnT is particularly effective in detecting the three types of jamming attacks. With RSSI consistency check activated, all jamming attacks were identified.

2) Mitigation Effectiveness: For the evaluation of mitigation effectiveness we consider the 3 scenarios per type of jammer, with RSSI consistency check and mitigation active. We measured effectiveness as the fraction of the number of cycles jamming is mitigated over the total number of cycles the jammer is active; mitigated means that the effect of the jammer on the packet loss is neutralized. The right column in Table II shows the results for a reactive, a random, and a constant jammer respectively. We observe that DeMiJA is very effective in mitigating reactive and random jammers, as well as constant jammers located near leaf nodes. On the other hand, the approach fails to mitigate the jamming scenarios Hybrid and Cut with constant jammers. The reason for this is that the two mitigation techniques we consider in this paper cannot be applied: a constant jammer jams all channels so channel hopping is not a solution, and the jammers jam all outgoing lines of the motes hence re-routing is not feasible. We elaborate on this issue further in the discussion section.

3) Efficiency of DeMiJA: For the evaluation of the efficiency of DeMiJA we measured the time for detecting jamming attacks and the time for mitigating the attacks. We measured a median detection time of 0.418 s (mean 0.425 s with std 0.043) for DeepAnT over all the different runs of the three types of jammers of Table I. We can conclude that the time to detect anomalies is negligible because the time it takes per cycle in the DeltaIoT network is in the order of minutes. We measured the mitigation time for different types jammers with DeepAnT, expressed as the number of cycles it takes from the moment the detection starts until the adaptation plan is enacted. The mitigation of reactive and random jammers are mitigated within one cycle (± 0). The mitigation of a constant jammer on the other hand takes 8 cycles (± 0), essentially since the effect of channel hopping is investigated over the different channels (8 total) before re-routing is applied.

D. Discussion

Regarding detection, the results show that DeepAnT effectively detects different types of jammers when combined with additional RSSI-checks. All jamming attacks are detected, while the number of false positives can be neglected. Mitigation with channel hopping works very well for reactive and random jammers (with random/reactive behavior) that jam one channel. These jamming attacks are mitigated using channel hopping after only a couple of cycles. Contrary, a constant jammer that jams all channels only performs well if it has re-routing options available (i.e., the Leaf jamming scenario in the evaluation). To figure out the adaptation options, this mitigation is less efficient in the time it takes to mitigate the jamming attack (since channel hopping is tested before re-routing). However, if the jammer is located in the network such that re-routing is not possible (as for the Hybrid and Cut jamming scenarios), the jamming attack cannot be mitigated. To deal with this issue, DeMiJA could be extended with additional mitigation strategies, such as learning-based techniques [29] and multi-channel ratio decoding technique [29]. We leave this extension as future work.

The efficiency of mitigation can also be enhanced by determining the type of jamming attacks that affect the network links. One approach for determining the type of jammer is through the analysis of the quality and uncertainty tuples and their detection results. Another approach presented by Kasturi et al. [11] uses a supervised machine learner to classify the jamming attack exploiting a history of the *packet delivery rate* and the RSSI values. Yet, this technique requires a large amount of labeled data of jamming attacks. Both these approaches allow tailoring the mitigating plans based on the type of jammer. We also leave this extension as future work.

V. Related Work

Multiple studies have tackled self-protection as testified in the survey by Yuan et al. [30]. We highlight a selection of relevant approaches related to DeMiJA. Schmerl et al. [31] used intrusion detection systems inside the Rainbow self-adaptive framework, utilizing a utility-based selection of mitigation tactics to deal with DoS attacks. Whilst their approach relies on a specific threshold on some measure quality properties for detection, our approach uses more advanced detection mechanisms. Chen et al. [32] proposed a model-based method for protecting a networked computing system against variant security attacks, opposite to our approach that deals with a particular type of attacks. However, in the detection stage, they focused on signature-based intrusion detection systems that require supervised data, opposite to our automated anomaly-based approach. He and Lacoste [33] applied a component-based approach, installing a proper software component in a malicious node when a security attack is detected. In contrast to DeMiJA, there is no evidence that it will work effectively and efficiently for IoT networks since no concrete examples of detection and mitigation are evaluated. Egendi et al. [34] proposed a MAPE-based feedback loop to protect cyber-physical systems (CPS) against specific types of DoS attacks, similar to DeMiJA. Their method used a support vector machine (SVM) trained over labeled data to detect anomalies in the behavior of the system in resource usage in contrast to DeMiJA.

VI. Conclusion

We introduced DeMiJA, a novel self-adaptive approach to deal with jamming attacks in wireless networks. We presented an architecture for DeMiJA and instantiated this architecture for DeltaIoT. The evaluation demonstrates that DeMiJA effectively and efficiently deals with different types of jammers. In the short term, we plan to investigate new adaptation strategies and mechanisms for identifying types of jammers. In the long term, our aim is to study self-adaptation to deal with other types of denial of service risks in wireless networks.

TABLE II: Jammer experiments with DeepAnT as detection method.

Jammer type	**RSSI consistency check**	**Mitigation active**	**Jamming scenario**	**False positives** ***jammed links*** **(% of cycles)**	**False positives** ***unjammed links*** **(% of cycles)**	**Jamming detection (% of cycles)**	**Mitigation effectiveness (% of cycles)**
Reactive	False	False	Hybrid	43.6	14.6	89.16	▨
			Cut	21.6	4.4	99.72	▨
			Leaf	25.61	2.9	81.11	▨
	True	True	Hybrid	0.0	0.0	100.0	99.98
			Cut	0.0	0.0	100.0	99.98
			Leaf	0.0	0.0	100.0	99.98
Random	False	False	Hybrid	43.4	5.0	90.0	▨
			Cut	19.2	1.4	99.61	▨
			Leaf	58.6	2.4	69.0	▨
	True	True	Hybrid	0.0	0.0	100.0	98.57
			Cut	1.2	0.0	100.0	98.57
			Leaf	0.0	0.0	100.0	98.57
Constant	False	False	Hybrid	47.2	8.6	85.83	▨
			Cut	46.8	4.4	100.0	▨
			Leaf	52.7	3.9	67.77	▨
	True	True	Hybrid	0.0	0.0	100.0	0.00
			Cut	0.0	0.0	100.0	0.00
			Leaf	0.0	0.0	100.0	91.11

REFERENCES

[1] M. U. Iftikhar, G. S. Ramachandran, P. Bollansee, D. Weyns, and D. Hughes, "Deltaiot: A self-adaptive internet of things exemplar," in *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 76–82, IEEE, 2017.

[2] R. Lea and M. Blackstock, "Smart cities: An iot-centric approach," in *Wweb intelligence and smart sensing*, pp. 1–2, 2014.

[3] K. Darshan and K. Anandakumar, "A comprehensive review on usage of internet of things (iot) in healthcare system," in *Emerging Research in Electronics, Computer Science and Technology*, IEEE, 2015.

[4] C. Yang, W. Shen, and X. Wang, "Applications of internet of things in manufacturing," in *20th International conference on computer supported cooperative work in design*, pp. 670–675, IEEE, 2016.

[5] R. Dagar, S. Som, and S. K. Khatri, "Smart farming–iot in agriculture," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 1052–1056, IEEE, 2018.

[6] M. Ingham, J. Marchang, and D. Bhowmik, "Iot security vulnerabilities and predictive signal jamming attack analysis in lorawan," *IET information security*, vol. 14, no. 4, pp. 368–379, 2020.

[7] H. Pirayesh and H. Zeng, "Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2022.

[8] K. Pelechrinis, M. Iliofotou, and S. V. Krishnamurthy, "Denial of service attacks in wireless networks: The case of jammers," *IEEE Communications Surveys Tutorials*, vol. 13, no. 2, pp. 245–257, 2011.

[9] 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 *International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2014.

[10] K. Davaslioglu, S. Soltani, T. Erpek, and Y. E. Sagduyu, "Deepwifi: Cognitive wifi with deep learning," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 429–444, 2019.

[11] G. Kasturi, A. Jain, and J. Singh, "Machine learning-based rf jamming classification techniques in wireless ad hoc networks," in *International Conference on Wireless Intelligent and Distributed Environment for Communication*, pp. 99–111, Springer, 2020.

[12] D. Garlan, S. Cheng, A. Huang, *et al.*, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer 37(10)*, 2004.

[13] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *Future of Software Engineering*, pp. 259–268, 06 2007.

[14] D. Weyns, S. Malek, and J. Andersson, "FORMS: Unifying Reference Model for Formal Specification of Distributed Self-adaptive Systems," *ACM Trans. on Autonomous and Adaptive Systems*, vol. 7, no. 1, 2012.

[15] B. Cheng *et al.*, "Software engineering for self-adaptive systems: A research roadmap," pp. 1–26, LNCS 5525, Springer, 2009.

[16] R. de Lemos et al., *Software Engineering for Self-Adaptive Systems: A Second Research Roadmap*, pp. 1–32. LNCS 7475, Springer, 2013.

[17] D. Weyns, *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective*. John Wiley & Sons, 2020.

[18] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[19] D. Weyns, U. Iftikhar, and J. Soderland, "Do external feedback loops improve the design of self-adaptive systems? a controlled experiment," in *Software Engineering for Adaptive and Self-Managing Systems*, 2013.

[20] M. U. Iftikhar and D. Weyns, "Activforms: Active formal models for self-adaptation," in *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, ACM, 2014.

[21] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[22] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Mobile Ad Hoc Networking and Computing*, ACM, 2005.

[23] H. Mohsin, K. Abdulameer, and Z. N. Khudhair, "Study and performance analysis of received signal strength indicator (rssi) in wireless communication systems," *International Journal of Engineering and Technology*, vol. 6, pp. 195–200, 01 2017.

[24] S. Xiong and J. Li, "An efficient algorithm for cut vertex detection in wireless sensor networks," in *2010 IEEE 30th International Conference on Distributed Computing Systems*, pp. 368–377, IEEE, 2010.

[25] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *Ieee Access*, vol. 7, pp. 1991–2005, 2018.

[26] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: A survey on the state-of-the-art," *arXiv preprint arXiv:2004.00433*, 2020.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT, 2016.

[28] V. Czitrom and P. D. Spagon, *Statistical case studies for industrial process improvement*. SIAM, 1997.

[29] W. Shen, P. Ning, X. He, H. Dai, and Y. Liu, "Mcr decoding: A mimo approach for defending against wireless jamming attacks," in *IEEE Conference on Communications and Network Security*, 2014.

[30] E. Yuan, N. Esfahani, and S. Malek, "A systematic survey of self-protecting software systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 8, no. 4, pp. 1–41, 2014.

[31] B. Schmerl *et al.*, "Architecture-based self-protection: composing and reasoning about denial-of-service mitigations," in *Symposium and Bootcamp on the Science of Security*, 2014.

[32] Q. Chen, S. Abdelwahed, and A. Erradi, "A model-based approach to self-protection in computing system," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, pp. 1–10, 2013.

[33] R. He and M. Lacoste, "Applying component-based design to self-protection of ubiquitous systems," in *3rd ACM workshop on Software engineering for pervasive services*, pp. 9–14, 2008.

[34] I. Elgendi, M. F. Hossain, A. Jamalipour, and K. S. Munasinghe, "Protecting cyber physical systems using a learned mape-k model," *IEEE Access*, vol. 7, pp. 90954–90963, 2019.