Physical-layer Jammer Detection in Multi-hop IoT Networks

Mostafa Abdollahi, *Student Member, IEEE,* Kousar Malekinasab, Wanqing Tu, *Senior Member, IEEE,* Mozafar Bag-Mohammadi

***Abstract*—The presence of a jammer in an IoT network severely degrades all communication efforts between adjacent wireless devices. The situation is getting worse due to retransmission attempts made by affected devices. Therefore, jammers must be detected or localized quickly to activate a series of corrective countermeasures so as to ensure the robust operation of the IoT network. This paper proposes a novel metric called the number of jammed slots (NJS). It can detect and localize both reactive and proactive jammers that follow arbitrary jamming attack patterns. NJS is applicable to all communication paradigms such as unicast, broadcast, and multicast. In NJS, the wireless medium status is monitored by IoT devices and summarized reports are sent to a central node. Then, the central node determines the jamming duration, the affected nodes, and the approximate location of the jammer(s). Also, the specificity, precision, and accuracy of NJS are at least 48%, 19%, and 20% better than the other state-of-the-art statistical methods, respectively. In addition, in terms of the detection time, NJS is four times faster when detecting an active jammer in the network. It can also localize the jammer with less jammer localization errors.**

***Index Terms*—jammer detection, reactive jammer, proactive jammer, IoT networks.**

I. Introduction

The Internet of Things (IoT) refers to a massive number of devices connected to the Internet so as to develop various smart services (e.g., smart buildings, smart transportation, and intelligent environments). IoT devices are often unsupervised and distributed in a large scale via multiple wireless hops, making IoT networks vulnerable to physical-layer jammers. By physical-layer jammers, we mean non-networking transmitters (e.g., a microwave oven, an outdoor working machine) sending disturbing signals with sufficient strength to cause interference or jams to valid data transmissions in wireless channels. These non-networking transmitters do not participate in networking communications, i.e., do not implement standards or protocols designed for networking devices, and hence the generated jams are mostly due to physical-layer noise. The generation of such jamming signals will potentially cause a set of communication issues, including hindering legitimate transmissions, prolonging channel busy time, disrupting packet receptions at receivers, draining the batteries of IoT devices, and degrading the overall performance of a IoT network. Therefore, it is crucial to quickly and accurately detect physical-layer jammers in IoT networks.

Current solutions for detecting jamming attacks are either employing a specialized hardware (e.g. measuring the noise power level) to analyze signal strengths [2] [3] or conducting statistical indices such as packet delivery ratios (PDR) [4], channel busy time [5]. The hardware-based solutions are faster and more accurate than statistical indices. However, they may not be applicable in an IoT network consisting of devices ranging from disposable devices to flagship gadgets. For example, it is impossible to equip out-dated IoT devices with required hardware. The statistical indices are intrinsically prone to false alarms due to background traffic variations, causing inherent inaccuracy and slow reaction to detect jammers. Jammer detection data which are acquired by either hardware assisted methods or statistical studies could be used to train neural networks [6], [7]. This technique may help to predict jammers. But it requires considerable time and computational complexity to be accurate in such predictions.

As for studies on localizing jammers, they usually start after detecting a jammer in the network [8] [9]. Having the position information of a jammer allows us to eliminate the jammer from the network and provide useful information about the jammer when designing a new MAC or routing layer protocol. Existing solutions are mainly range-based or range-free methods [10]. In the range-based schemes, the distance or angle is estimated as range metrics. Techniques employed include using received signal strength indication (RSSI), signal arriving angles, signal arriving times, and the time differences of arriving signals to determine the ranges of jammer locations. The cost issues and extra hardware requirements preclude the broad usage of range-based schemes. With regard to range-free methods, they infer the location of IoT devices using radio connectivity to communicate between them and carry their topological information. The inferior accuracy of range free methods is sufficient for many jammer detection applications [11].

This paper presents a new jammer detection and localization algorithm, called the number of jammed slots (NJS)-based algorithm, to enable that a multi-hop IoT network not only self-detects physical-layer jammers (without relying on any external hardware devices) but also accurately determines the locations of such jammers in a real-time fashion. With NJS, each IoT device records its channel states and reports its recorded states to a central node (e.g., a wireless gateway) periodically. Four different channel states: idle, receiving, transmitting, and corrupted are proposed to support our ac-

M. Abdollahi, K. Malekinasab, and M. Bag-Mohammadi was with the Wireless Network Labs, Engineering Faculty, Ilam University, Ilam, Iran. e-mail: { m.abdollahi, mozafar}@ilam.ac.ir

W. Tu was with the Department of Computer Science, Durham University, United Kingdom. e-mail: { wanqing.tu}@durham.ac.uk

This paper was presented in part at the 2021 IEEE 46th Conference on Local Computer Networks (LCN), Edmonton, AB, Canada, October 4-7 [1]

curate yet real-time jammer detections and localizations. The central node calculates the NJS metrics for all nodes that report their states. Based on which, our NJS-based algorithm is able to accurately determine whether there is a jamming attach or not and if so, where the jammer is in a real-time fashion. In general, the non-zero NJS value of an IoT device is a clear indication of the jammer presence in the network. By comparing the NJS values of neighboring nodes, the number of jammers and their approximate locations are revealed. More specifically, our contributions are concluded as below.

- **Comprehensive jammer detection.** Physical-layer jammers can be reactive and proactive[1]. As compared to our previous work [1] that uses witness nodes to detect reactive jammers, our new NJS algorithm can efficiently and accurately detect both reactive and proactive jammers. This is achieved by enabling the jammer monitor center (JMC) to identify a void region in the network where there have been no valid communication activities for a certain time period. Furthermore, the new NJS algorithm is able to detect coexistent reactive and proactive jammers as the algorithm can form understanding of witness nodes and void regions simultaneously.
- **Jammer detection in a multi-hop IoT network.** Our study in [1] detects jammers in a single-hop/local IoT network. The new NJS algorithm in this paper can determine jammers in a large-scale multi-hop IoT network. The multi-hop scenario may introduce overlaps between jammed areas, which cause our study in [1] not always to accurately detect or localize jammers. We solve this issue by integrating the NJS with the weighted centroid localization method [12], allowing the jammer localization errors to be reduced by at least 2.5 times as compared to existing methods.
- **Jammer detection for different transmission methods.** Our new NJS algorithm supports jammer detections in unicast, multicast, and broadcast, an enhancement to the unicast-based jammer detection in [1]. To achieve this new capability, a superset of MAC layer states is defined by integrating multiple MAC states into a single state, providing a coarse-grained view of the network topology. Such coarse-grained views help to generate more useful NJS reports for the JMC to determine and localize jammed nodes and jammers in multicast or broadcast transmissions.

Our simulation results show that our algorithm is superior to current statistical metrics such as PDR [5], carrier sense time (CST) [13], packet send ratio (PSR) [14], and time-series based method [15]. Also, the specificity and precision of NJS are 100% and at least 48% and 19% better than the other methods in all scenarios, broadcast, unicast, and multicast. In addition, accuracy of NJS is at least 20% better than the state-of-the-art statistical methods. Such performance is nearly as accurate as hardware-based methods. It is worth mention that hardware-based methods are hard to be implemented in heterogeneous environments such as IoT networks. Also, the NJS-based method is four times faster than statistical methods in detecting a reactive or proactive jammer in the IoT networks. With even such a fast detection process, NJS can still generate fewer false alarms than statistical methods in our simulations.

[1]Reactive jammers and proactive jammers are defined in Section III.

II. RELATED WORKS

Pirayesh etc. [16] surveyed jammer detection problems and anti-jamming techniques in nearly all types of wireless networks. As mentioned, studies on physical-layer jammers can be classified as jammer detections and jammer localization. Physical-layer jammer detections employ by hardware-assisted approaches [2], [17]–[19] or statistical indices methods [5], [13], [14], [20]–[24].

In hardware-assisted approaches [17], [19], jammers are detected by sensing the strength of jamming signals. The measurements are reported to a central node which runs a proprietary algorithm to locate a jammer based on the received jamming signal strength. In [2], jamming signal strengths are used to train a decision tree algorithm so that the jammer presence can be predicted. In [6], jamming signals are used to harvest energy by ambient backscatter techniques, instead of only transmitting data. First, jammers are detected by a detector circuit by comparing jamming signal strengths with benign signal strength. Then, IoT devices generate fake transmissions to stimulate a jammer to attack the channel. In [7], the jamming signal strength and localization information are used to train a neural network to detect jammers. Then, they use another neural network to schedule links based on the data produced by the first neural network. In [18], the location and transmission range of a directional jammer is found by analyzing jamming signal strengths from boundary nodes and the geographical location of jammed nodes.

Statistical indices methods detect jammers by conducting statistical analyses because jammer activities degrade wireless medium characteristics (e.g., channel busy time, PDR). In [5], the gradient descent method [5] is used to detect jammers by tracing PDR reduction in the observed network. The authors in [20] found that jammer activities increase packet transmission time. If the packet transmission time exceeds a predefined threshold, IoT devices determine that a jammer is detected. If so, IoT devices switch to a safe channel and retransmit the packets. Similarly, in [21], the packet-invalidity ratio is defined as the probability that packet transmission delays are greater than a threshold value. This ratio is used to detect jammers in the context of cognitive radio networks for IoT applications.

In [13], [14], a combination of CST, PDR, and PSR are examined to detect jammers. They found that jammer activities decrease PSR and PDR but increase CST. In [22], a jammer is identified by comparing the number of idle and busy slots before and after jammer activities. Each IoT devices maintains a medium profile which is updated periodically. These profiles are reported to and processed at a central node, in order to detect possible active jammers. In [23], PDR and bad packet ratio (BPR) are used for identifying an active jammer. The activation of a jammer degrades PDR and increases BPR. This approach is used by [8] [24] in a similar way.

In [15], a new detection method is proposed using a time series analysis. It modeled the network measurements, such as

busy time, taken over time as a time series and proposed an algorithm for detecting sequential change point. It can quickly detect anomalies using a proper indicator that varies with the jammer activities over time. In [25], the packet drop ratio and inverse PDR are used alongside a machine-learning algorithm to detect the presence of a jammer in vehicular networks. In [26], a random forest classifier is proposed in the context of vehicular networks using signal to noise ratio (SNR) and PDR as training data. Similarly, in [4], a machine learning-based method is employed to detect the presence of jammers in the network using metrics such as PDR, BPR, SNR, and RSSI. In [27], statistical process control (SCP) is used to detect jammers. It employs packet drop ratios as the center line with an upper control limit (UCL) and a lower control limit (LCL). The presence of a jammer is detected when the center line crosses the upper control limit. A very similar idea is designed in [28] for IoT networks. In this work, the center line is recalculated when two nodes encounter. Nodes only exchange summary vectors or new messages (if any) when the center line falls below LCL. A jammer presence is reported when the center line is above UCL. Nodes enter the suspicious zone when the center line is between LCL and UCL. Finally, the nodes that are detected to be within the range of the jammer, known as the jamming zone, will not participate in the subsequent packet forwarding processes. Contrary to the statistical-based method in [28], our method aims to identify an actual footprint of a jammer among the receptions of the affected nodes.

In general, with hardware-assisted solutions, all IoT devices need to be equipped with a specialized hardware, which not only increases deployment cost but also incurs the backward compatibility problem with IoT devices. Also, most hardware solutions work only in single-hop jamming detections. As for statistical indices, they need to be measured in a periodical manner, increasing the detection delays. Furthermore, the false alarm problem resulted from background traffic and/or channel issues decreases their accuracy.

In [29], multiple active jammers are localized in wireless networks via two steps. First, they use the received jamming signal strength (RJSS) value to determine the number of overlapped jammers. Then, they use Region Growth Algorithm (RGA) in an iterative manner to locate the jammers. In [30], a jammer in an Internet of vehicles is localised using 1-Time of Delay (TOD), 2-Time of Arrival (TOA), and 3-RSSI from jammed vehicles. To find the location of a single jammer in a wireless sensor network [31], an improved version of beetle antennae search algorithm is used.

III. SYSTEM MODELS AND PROBLEM DEFINITION

A. System Model

Suppose that there are N IoT nodes in the system with each denoted as n_i where $0 \leq i < N$. Also, there is a jammer monitor center (JMC) in our system, responsible for collecting information from IoT nodes and implementing our NJS algorithm to detect and localize a jammer. In practice, a JMC can be a wireless gateway in an IoT network. As shown in Fig. 1, in our system, IoT nodes may connect with each other for distributing IoT application data. A physical-layer jammer interferes with the data transmissions of those IoT nodes that are within the coverage of this jammer. Also, each IoT node periodically sends channel state packets to the JMC every m seconds, allowing the JMC to utilize the information for jammer detection and localization. The description of our NJS algorithm defines the following key terms.

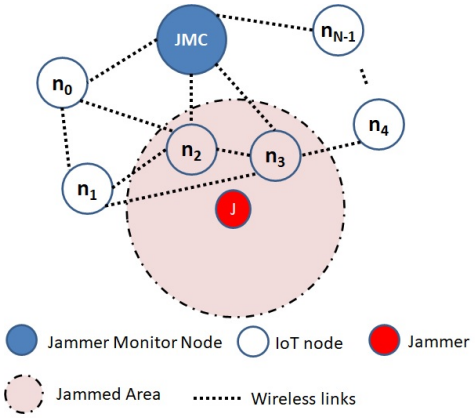

Fig. 1. An example of the system model in a single-hop IoT network.

1. Channel States: The JMC must have a broad view of all wireless media events in order to deduce the presence and location of a jammer correctly. These types of information are available at the MAC layer in the form of MAC layer states such as IDLE, waiting for CTS (WFCTS), and waiting for data (WFDATA) [32]. Similar states are also defined for broadcast and multicast scenarios [33]. The MAC-layer states of a wireless IoT node are defined as doing an action (such as frame transmission) or waiting for an action accomplishment (such as a data frame reception). The latter is done by adjusting a proper timer. The MAC layer switches between the defined states when it receives/transmits a frame or a timer expires. This will solves several media challenges such as hidden and exposed terminals.

In our NJS algorithm, multiple MAC states are combined into a superset to provide a coarse-grained view of the MAC-layer states of IoT devices for the JMC. We summarise various medium events as the following four channel states observed by an IoT node using the channel (see Fig. 1) :

- *Idle*: n_i is not sending/receiving a signal via the channel;
- *Receiving*: n_i is receiving a valid data or control frame correctly from the channel;
- *Transmitting*: n_i is sending a data or control frame to the channel;
- *Corrupted*: n_i is receiving a corrupted frame from the channel. The frame corruption might be due to collision, fading, jamming, etc.

2. Channel state updating period: The channel status is recorded every τ seconds by n_i ($0 \leq i < N$). Then, the node aggregates channel states and sends the aggregated states via a report every $m * \tau$. The period τ is proportional to the time required to send/receive a data frame. More specifically, its value must be slightly greater than the largest frame size divided by the channel bit rate. For example, τ could be 7ms,

if bandwidth was 2Mbps and the data frame length was 1518B.

Nodes in our system may employ some network timing protocol (say NTP [34]) to synchronize their periods, allowing a synchronized snapshot of the channel states from different viewpoints to be achieved in each time slot. Each wireless IoT device reports the collected MAC-layer states from the last m time period to the JMC. Similar to other well-known routing metrics in wireless networks [32], the collected reports could be delivered to the JMC by flooding or unicasting methods or via a direct channel between IoT nodes and JMC.

3. Jammer Types: We classify physical-layer jammers into proactive jammers and reactive jammers. A proactive jammer is a jammer that constantly jams the wireless medium regardless of channel states. Proactive jammers remain active for multiple continuous periods. In contrast, a reactive jammer is normally silent but becomes activated whenever it detects an ongoing transmission between other nodes. Therefore, when there is a reactive jammer in the network, even though IoT nodes sense the medium as idle, their transmissions may suffer from interference.

B. Problem Definition

The open access property of wireless transmission medium makes it challenging to detect physical-layer jammers since it is hard to differentiate between normal channel events and a jammer activity. Fortunately, in many cases, there are always some IoT nodes that receive *only* jamming signals when a jammer is active. Since physical-layer jamming signals are non-decodable, such a node will report a *Corrupted* state to JMC. The JMC will know the existance of a jammer based on the fact that there is no valid sender around the reporting node. To implement this basic idea to develop our NJS-based algorithm, we need to address the following major challenges:

Signal degradation vs. jamming attacks: The first challenge is to differentiate signal degradation caused by path loss, fading, shadowing, and interference from the deliberate collisions caused by the jammer. However, the jammer activity not only explicitly affects some nodes, but there are also some nodes that are simultaneously exposed to the jammer signal and a transmission of a valid sender. These nodes are receiving a corrupted signal from the channel. Receiving reports of these nodes by the JMC could be analyzed as follows. First, the JMC finds the explicitly jammed nodes. Having the location of these nodes, it can approximately predict the jammer location. Now, it can find an *implicitly* jammed node when the jammed node is in the vicinity of both the jammer and a valid sender at the same time. Otherwise, if the reporting IoT device is *only* located in the vicinity of one or multiple valid senders, the erroneous reception could be blamed on normal channel events.

Communication Primitives: Unicast communication primitive uses multiple control frames (i.e. CTS, RTS, and ACK) for handshaking between a sender and a receiver. The receiver must send CTS and ACK frames in response to RTS and DATA frames. Other wireless nodes around the receiver which receive a CTS or a ACK frame, report their NJS state as *Receiving* or *Corrupted* (in the case that they could not decode the received frame due to normal channel events). These reports confuse the JMC since there is no valid data sender at that time period around reporting nodes. In order to avoid possible confusions at the JMC, any attempt by a node to transmit a frame, including RTS, CTS, DATA, and ACK, is reported as *"Transmitting"* by NJS-enabled nodes. Also, reception of a frame on the channel could only be reported as *"Receiving"* or *"Corrupted"*. It means that NJS does not take into account the actual role of the node in the unicast communication. Instead, it pays attention to a node's status with regard to the transmission media.

During handshaking phase, an IoT device switches between sending and receiving modes. For example, the sender sends a RTS frame and receives a CTS frame in response before sending the DATA frame. Therefore, the device may experience both *Transmitting* and *Receiving* states at a single time slot. But, the IoT device is not allowed to report multiple states at a single time slot. The device must report its state as *Transmitting* because the JMC is sensitive to orphanage signals (i.e. a transmitted signal without owner).

In both broadcast and multicast modes, the number of reporting receivers is enough to deduce the jammer's presence. In fact, some receivers could be jammed implicitly or explicitly. In unicast mode, where RTS and CTS must be exchanged before receiving and acknowledging data frames, any node that occupies the media to send DATA, RTS, CTS, or ACK frames is considered as the sender. On the other hand, any node that receives such a frame from the channel is considered as a receiver. This strategy increases the number of collected reports and their diversity which enables the JMC to differentiate between channel interferences and jamming signals more accurately.

Frame Loss: Both control and data frames could be lost on the wireless media. It leads to an incomplete handshake or transmission. When a RTS or its subsequence CTS is lost, the handshake is incomplete. When a DATA frame or its corresponding ACK frame is lost, the transmission is incomplete. Since in all cases, the sender of a control or a data frame is forced to report its state to the JMC, the JMC is able to discriminate between an incomplete communication and an active jammer case.

Multiple Simultaneous Jammers: Suppose that there are multiple jammers that were activated simultaneously. Our mission is to detect the number of jammers and their location. However, in a multi-hop IoT network, jamming areas are not distinct. They are often overlapped. Such overlapping makes it more difficult to determine the number of jammers and localize these jammers. When the corresponding jamming areas are distinct, we can detect a jammer in each jamming area by our method in [1]. The location of the jammer could be estimated using weighted centroid algorithm [12].

C. An Example

NJS uses a similar set of states for unicast, broadcast and multicast transmission modes. We use a unicast scenario in the presence of a reactive jammer to discuss the effectiveness of the defined states. In Fig. 2, node A sends a RTS frame in

time slot t_0. Node B, after correctly receiving the RTS frame, transmits a CTS frame. Their NJS state is *"Transmitting"* since both nodes transmit a frame on the channel. Upon reception of the CTS frame by the reactive jammer, it starts to jam the network by sending a nonsense frame on the channel. Both C and D receive a corrupted frame from the channel. They record their NJS states as *"Corrupted"* in t_0 and t_1. The JMC finds the jammer's footprint because there is not a valid sender around D and it has reported a *"Corrupted"* state in time slots t_0 and t_1.

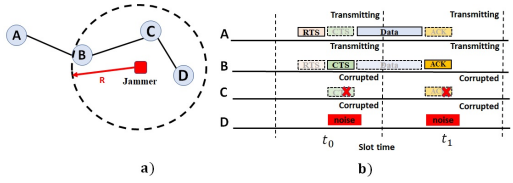

Fig. 2. A unicast scenario in the presence of a reactive jammer.

IV. NUMBE OF JAMMED SLOTS (NJS)

Our basic idea is to track the jammer's footprint among the reported states at JMC to detect and/or locate jammers. First, the JMC prunes the correct data frame's reception and transmissions based on the network topology and received timelines. A normal or correct reception has a valid sender in its transmission range. Also, the JMC assumes that all erroneous receptions in the neighborhood of a valid sender are due to normal channel errors. Hence, NJS is not sensitive to traffic variation, link quality, fading, etc., which enables it to accurately detect the jammer.

To find the actual effects of the jammer, we present a general and comprehensive assertion that helps us to prune normal frames from the jammer-influenced frames.

Assertion: *The frame or an erroneous signal received without a valid owner (sender) is meant to be sent by a jammer. This frame is called a* **jammed frame**.

Based on that assertion, two useful propositions are derived which help us to predict the jammer's type and its location. A proactive jammer continuously broadcasts the noise signal regardless of the existence of an active transmitter on the channel. As a result, the receivers will continuously receive frames/signals on the channel while there is no active sender around them. Therefore, we can use the following proposition to identify a proactive jammer. Suppose that the collision area of a wireless device is bounded by a circle whose radius is the device's transmission range R_S. The device is in the center of this circle (see Fig. 4).

Proposition 1: *"If we can find at least one time slot in which almost all nodes that reside in a collision area reported "Corrupted" receptions or did not report at all, and there is not a valid sender in their transmission ranges, a proactive jammer exists in the center of the collision area."*

A proactive jammer will black out all IoT devices in its radio range. Therefore, if a proactive jammer remains active for a time period longer than m time slots, all affected IoT devices cannot obtain the channel to send the recorded observations. However, prior to the jammer activation, the JMC was able to receive their reports. Therefore, during the jammer activity, the JMC notices that there is a permanent hole in the topology and recognize the jammer location easily using preposition 1. Also, the existence of another channel is helpful. This channel could be used to send NJS reports when a proactive jammer is active.

In contrast to a proactive jammer which continuously occupies the channel, a reactive jammer only sends a jammed frame whenever a valid sender is activated in its vicinity. The necessary and sufficient condition to detect the existence of a reactive jammer in the network is that the report message of the sender must be available at JMC. The existence of reactive jammers will be confirmed using the following proposition:

Proposition 2: *"If multiple jammed frames alongside a valid sender is found in a collision area in two or more consecutive time slots, there is a reactive jammer in the network."*

For detecting a reactive jammer, we introduce the concept of witness nodes. As depicted by Fig. 3, the reactive jammer should be in the transmission range of the sender (R_S) in order to detect the ongoing transmission. It then tries to jam the network by transmitting a meaningless signal. The grey nodes receive intact frame sent by S. the node in bold gray area only receive the jammer signal. But, we cannot find a valid sender around them. However, the nodes in hatched area have a very nice property. Although they are jammed by the jammer, we can find a valid sender in their transmission range. We call those nodes as witness nodes and the hatched area as the witness area. According to preposition 2, the presence of at least one witness node is a valid sign of presence of an active reactive jammer. Then, we can use the witness nodes and jammed nodes together to locate the jammer using well-known techniques such as the weighted centroid algorithm [12].

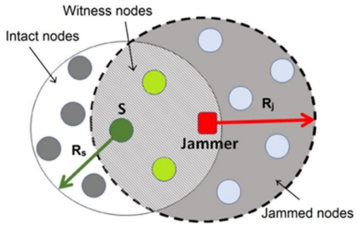

Fig. 3. Detection of a reactive jammer using witness nodes.

Lemma 1: *The density of witness nodes in witness area is more than one for a connected topology provided that the node density is more than one node per* R_s^2.

Proof: We show that the witness area is greater than R_s^2.This will complete the proof. We assume that the transmission ranges of the sender (R_S) and the jammer (R_j) are equal. It is worth noticing that in reality R_j is more than R_S. The distance between S and the jammer could not be less than R_S. Clearly, the witness area is slightly larger than the shaded area as shown in Fig. 4a. The area of shaded area is $\frac{\pi R_s^2}{3}$.

In Fig. 4b, we have depicted the most convenient arrangement of IoT devices to preserve our assumption about node

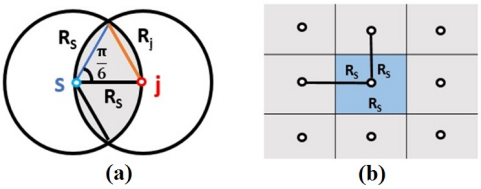

Fig. 4. a) the witness area, and b) minimum node density.

density (i.e. one node per R_s^2). The lemma 1 has a very important consequence. It guarantees that NJS could detect a reactive jammer even in the sparsely connected topologies.

A. The NJS algorithm

We need a carefully crafted algorithm to determine jammed nodes and the number of jammed slots in the presence of a jammer using the defined assertion and propositions. First, we notice that the jammer activity impacts the IoT devices in two different ways. Some wireless devices are only exposed to the jammer, while others are exposed to both a legitimate signal and the jammer's signal. In the first case, the jammer's footprint could be revealed *explicitly*. But in the second case, the jammer has *implicitly* affected the IoT devices.

The algorithm calculates the number of jammed slots from the node's perspective in the most recent m time slots. The NJS metric has a simple, robust way to find jammed nodes, jammed slots, and the number of jammers. It is composed of two distinct parts: finding explicitly jammed slots (NJS_{exp}) and inferring implicitly jammed slots (NJS_{imp}). First, to calculate the explicit part of NJS, i.e. NJS_{exp}, we seek for a time slot with corrupted receptions and without a valid sender. In other words, we ignore the corrupted receptions in the range of a valid sender. To infer NJS_{imp}, we look at the IoT nodes in the intersection area of a sender and the jammer. For those IoT nodes, the corrupted receptions are blamed on the presence of the jammer and their NJS_{imp} is increased per corrupted time slots. In the following, we have devised a simple algorithm to calculate NJS.

In lines 1-4 of the algorithm, the explicit counterpart of NJS is calculated in most recent m time slots. For each receiver node such as j in time slot i, if it could not find a valid sender in its neighborhood, $NJS_{exp}[j]$ is increased. In lines 5-8, NJS_{imp} is increased for those IoT nodes that their $NJS_{exp} > 0$. Finally, the value of the NJS metric for node is calculated as follow:

$$NJS[j] = NJS_{exp}[j] + NJS_{imp}[j] \quad (1)$$

Some interesting scenarios are depicted in Fig. 5. In Fig. 5a, a proactive jammer is active in the network and all transmitters are inactive due to channel business. Here, the IoT nodes have received a signal/frame on the channel from an invalid sender. Therefore, the $NJS_{exp} > 0$ is for all IoT devices located in the collision area of the proactive jammer. The JMC will easily locate the proactive jammer since the NJS_{exp} for all jammed nodes is the same. In other words, according to proposition 1, NJS_{exp} of all jammed nodes will be *1* and their NJS_{imp} will be 0 for time slot j.

Algorithm 1: NJS Calculation

Input: Nodes' state timeline
Output: NJS metric for all jammed nodes

1 **for** *slot* $i \leftarrow 0$ *to* m **do**
2 **for** *node* $j \leftarrow 0$ *to* N **do**
3 **if** *CheckJammedNode(i,j)==true* **then**
4 $NJS_{Exp}[j]++$
5 **for** *slot* $i \leftarrow 0$ *to* m **do**
6 **for** *node* $j \leftarrow 0$ *to* N **do**
7 **if** *IsThisAJammedSlot(i,j)==true* $\&\&NJS_{Exp}[j] > 0\&\&$ *CheckJammedNode(i,j)==false* $\&\&Timeline_{ji} != Idle$ **then**
8 $NJS_{Imp}[j]++$
9 **for** *node* $j \leftarrow 0$ *to* N **do**
10 $NJS[j] = NJS_{Exp}[j] + NJS_{Imp}[j]$
11 **Function** `CheckJammedNode` (*slot i,node j*) **:**
12 ValidSender=false
13 **if** $Timeline_{ji} == RX$ **then**
14 **for** *node* $k \leftarrow 0$ *to* N **do**
15 **if** $k != j \&\& distance_{kj} < R \&\& Timeline_{ki} == TX$ **then**
16 ValidSender=true
17 **if** $Timeline_{ji} == IdleorTX$ **then**
18 ValidSender=true
19 return !ValidSender
20 **Function** `IsThisAJammedSlot` (*slot i,node j*) **:**
21 Result=false
22 **if** $Timeline_{ji} != Idle$ **then**
23 **for** *node* $k \leftarrow 0$ *to* N **do**
24 **if** $k != j \&\& distance_{kj} < 2R \&\& CheckJammedNode(i,k) == true$ **then**
25 Result=true
26 return Result
27 **comments:**
28 $Timeline_{ji}$ == the state of node i in slot j.
29 $distance_{kj}$ == the distance between nodes k and j
30 R = Tx range

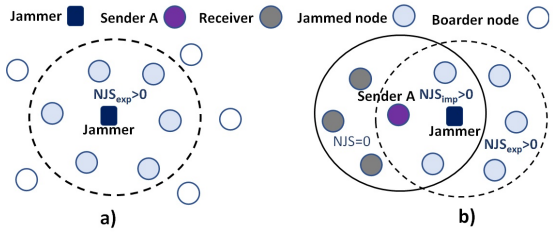

Fig. 5. a) a proactive jammer, and b) a reactive jammer.

In Fig. 5b, a legitimate sender (i.e., A) starts sending a data frame; the jammer is notified and jams the network simultaneously. The nodes in the intersection between the sender and jammer have $NJS_{imp} > 0$, while the nodes that are only in the range of jammer have $NJS_{exp} > 0$. The JMC could detect the reactive jammer by comparing explicit and implicit NJS values for the jammed nodes using proposition 2 (i.e. for any jammed slot j, NJS_{imp} of at least one jammed node will be 1). We note that the computational complexity of the algorithm is $O(N^{2m})$, where N and m are the number of nodes and slots in each timeline, respectively.

In Fig. 6, we have shown that how NJS detects a reactive jammer in a complex network. In the beginning, the jammer is silent. When the jammer detects some active senders in the network, it jams the network in consecutive time slots.

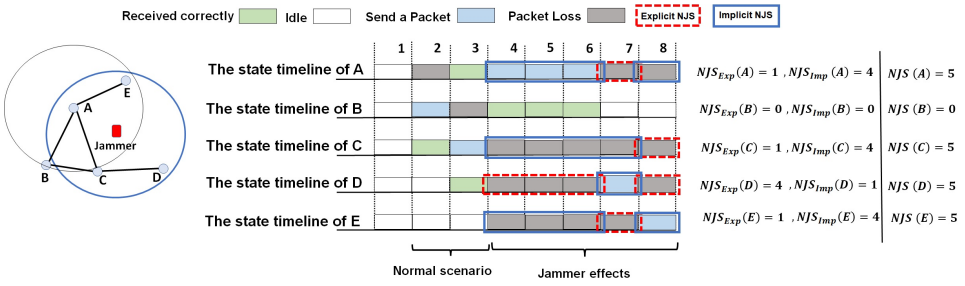

Fig. 6. The calculation of NJS_{imp} is illustrated.

Since there is no legitimate sender around D in time slots 4-6, NJS deduces that (in the first loop of the algorithm) the corrupted receptions at node D are due to the existence of an active jammer. Therefore, it increases NJS_{exp} for D. More interestingly, although it finds a valid sender in these time slots, it increases NJS_{imp} for nodes A, C, and E since there is a node (i.e. D) with $NJS_{exp} \geq 0$ in their vicinity.

B. The NJS Properties

NJS metric has the following properties:

1) A non-zero NJS metric is a clear indicator of the presence of an active jammer.
2) Similar NJS values in a neighborhood means that only a single jammer is active.
3) Non-similar NJS values in a neighborhood is resulted from the presence of multiple jammers. For example, the NJS values for IoT nodes that exist in the intersection area of two active jammers are expected to be more.
4) When NJS values are composed of both implicit and explicit values for some nodes, the jammer type is proactive.

C. Discussion

- **Inconsistency**: Since an IoT device arbitrarily claims the channel, it is impossible to always acquire the channel in the start of a time slot. Therefore, a transmission may span two consecutive slots. In that case, the transmission is started in slot i and finished at slot $i+1$. In NJS, all IoT nodes are obliged to consider the finish time of an action when reporting to the JMC.
- **None-NJS aware devices**: In general, some IoT devices may not wholly participate in NJS algorithm due to one of the following reasons:
  1) The devices are old and it is not possible to update their protocol.
  2) Some of the device's reports are lost due to collision, interference, etc.
  3) Some nodes are involved with internal issues such as insufficient energy or periodical software updates.

  For a persistent proactive jammer, the non-cooperation of some nodes has no harmful effect on the NJS accuracy. In that case, the presence of the jammer could be detected through the missing reports of the jammed node. More interestingly, the silent device works in the favor of NJS in this situation. If the jammer is not persistent, the accuracy of the NJS protocol in detecting the jammer will be degraded. If the status information of some nodes is not available, the JMC only checks the nodes whose adjacent nodes' information is available. For example, in Fig. 7, node C is a non-NJS aware device and does not report its status information to JMC.
  When node C transmits a frame, the reactive jammer is activated and its jamming signal affects B, D and F. In this case, there is an active transmitter in B's range (i.e. C) that does not send its report to JMC. So, the JMC receives multiple Corrupted reports without a valid sender around reporting nodes and records the issue as a suspicious case. If this phenomenon is repeated, the JMC concludes that there is a jammer in this area. The transmission of other nodes does not differ with normal case if we assume that C does not exist.

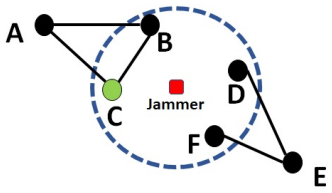

Fig. 7. The impact of non-NJS aware nodes on the algorithm accuracy.

D. Protocol Details

One of main IoT networks challenges is the inherent limitation in power, processing, and memory. Therefore, the processing power of IoT devices must be taken into account in design of the NJS algorithm. In general, the key design choices are:

- First, all processing tasks are carried out in a gateway device (i.e. the JMC) where power and processing constraints are not violated.
- The NJS algorithm is executed in two modes: *periodic* and *on-demand*. In the periodic mode, each IoT device must send a REPORT message to the JMC at a specified time interval (i.e. τ). In the on-demand mode, the JMC floods a SOLICITE message in the network. The IoT devices who received this message send their periodic reports to the JMC. The JMC could cancel the reports later by flooding a PAUSE message in the network.

- The REPORT messages are sent using TCP protocol. This message contains the link status of the most recent m slots, node Id and its location information if available.
- The SOLICIT and PAUSE messages are broadcast over the network. The SOLICIT message contains the values of τ and m fields.

V. SIMULATION RESULTS

In order to evaluate the NJS metric, several extensive simulations are carried out using an improved version of MIXIM 2.3 framework and OMNET++ simulator. The NJS is compared with the following major relevant studies. Conventional PDR (labeled as PDR-c) [35] detects the presence of the jammer in the network whenever PDR of links between IoT devices reduces dramatically. Conventional busy-time (labeled as BT-c) [13] utilizes an increase in the channel busy time as an indication of the jammer presence. The increase of channel busy time is used by conventional busy-time (labeled as BT-c) [13] to detect the presence of the jammer. In PDR-c and (BT-c), the presence of a jammer is proclaimed when the average PDR (BT-c) is decreased (increased) by 10% in comparison to the previous time interval. Also, we have implemented the time series analysis as proposed in [15] with z=0.2. It considered network measurements taken over time as a time series and proposed an algorithm for detecting the state alterations in the time series. We have applied their algorithm to PDR and busy-time measurements. The resulting methods are called PDR-t and BT-t for convenience. In PDR-t and BT-t, the measurement is updated every 20 time slots (i.e. 140ms). Finally, we have simulated [28] (labeled as DR). This work is using packet drop ratio to detect the jammer. The sample size is set to 50 and updated every 10 time slots.

A. SIMULATION SETTINGS

Our simulations are implemented in different scenarios: broadcast, multicast, and unicast. Table I shows the common simulation settings of these different scenarios. The channel is exposed to the normal frame loss due to fading, propagation, etc. Thus, to calculate the probability of receiving a frame successfully as a function of the distance between two nodes, we use the model in [36]. The model employs path loss exponent or power attenuation factor $\beta = 2$ and a transmission range with the radius up to R. The network topology is a simple grid, as shown in Fig. 8. For each square and simulation, a wireless node is placed in a random location inside the square. Also, the simulation time is 10s, equal to 1428 slot times, that is enough for study jammer detection under different scenarios. Finally, each simulation experiment is repeated 50 times. Also, the confidence interval of 95% shows in the diagrams.

B. PERFORMANCE METRICS

The following performance metrics are used to evaluate the effectiveness of the evaluated methods:

Accuracy: It is defined as the percentage of the number of correctly predicted nodes (jammed or non-jammed) over the total number of predicted nodes, i.e.,

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad , \tag{2}$$

Where true positive (*tp*) and true negatives (*tn*) record the number of truly predicted jammed and non-jammed IoT nodes, respectively. Also, the false positive (*fp*) and false negative (*fn*) outcomes occur when the system incorrectly predicts jammed and non-jammed IoT nodes, respectively. The accuracy demonstrates how close a jammer detection method can identify jammed and non-jammed nodes in the network. The accuracy is of a vital importance in routing applications that reroute the traffic around the jammed area. In such applications, detection of both jammed and non-jammed nodes are of equal importance.

Precision: It is defined as the ratio of the number of correctly predicted jammed nodes to the number of correctly and incorrectly predicted jammed nodes, i.e.,

$$Precision = \frac{tp}{tp + fp} \quad , \tag{3}$$

The precision metric demonstrates how a jammer detection method can guarantee the detection of all jammed nodes. In jammer localization methods, the location of a jammer is predicted based on the locations of the relevant jammed nodes. Hence, the precision of the jammer detection method plays an important role is such applications.

Recall: It is defined as the ratio of the number of correctly predicted jammed nodes to the total number of nodes that are correctly predicted as jammed nodes and incorrectly predicted as non-jammed nodes (i.e., the total number of real jammed nodes). The recall is defined as:

$$Recall = \frac{tp}{tp + fn} \quad , \tag{4}$$

The recall demonstrates how a jammer detection method can guarantee the identification of all jammed nodes.

Specificity: It is defined as the ratio of correctly predicted non-jammed nodes (*tn*) over the total number of correctly predicted non-jammed nodes (*tn*) and incorrectly predicted jammed nodes (*fp*). The formula of specificity is defined by:

$$Specificity = \frac{tn}{tn + fp} \quad . \tag{5}$$

The specificity demonstrates how a jammer detection method can guarantee the identification of all non-jammed nodes.

Jammer localization error (JLE): JLE indicates the relative error of the predicted jammer location to its actual location. First, a centroid localization algorithm [19] is used to predict the jammer location as follows:

$$(X_{avr}, Y_{avr}) = (\frac{\sum_{i=1}^{N} x_i}{N}, \frac{\sum_{i=1}^{N} y_i}{N}) \quad , \tag{6}$$

Where N is the total number of the predicted jammed nodes, and (X_{avr}, Y_{avr}) is the average of their Cartesian location. Then, the distance between the actual and the predicted positions of the jammer is calculated. Then, the error is obtained

by dividing the difference by the transmission range of the jammer (R_J) as follow:

$$JLE = \frac{\sqrt{(X_{avr} - X_i)^2 + (Y_{avr} - Y_j)^2}}{R_J} \quad (7)$$

Where (X_{avr}, Y_{avr}) is Cartesian location of the jammer. The JLE demonstrates how a jammer detection method can guarantee to localize the actual position of the jammer in the network.

TABLE I
THE SIMULATION PARAMETERS.

Name	Value
MAC Protocol	802.11
Tx radius of nodes (R)	220m
Tx radius of jammer	220m, 330m
Bandwidth	2Mb
Tx rate of nodes	Random(0,100), (0,200), and (0,1000)ms
Packet size of nodes	1500 B
Packet (noise)size of jammer	500, 1000 B
Simulation time	(1428 slots) = 10s

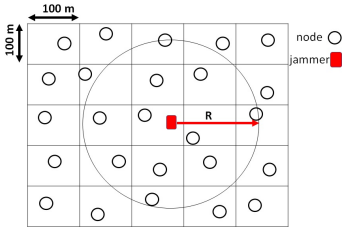

Fig. 8. A generated network topology with a random distribution of the nodes.

C. JAMMING STRATEGIES

In the ON-OFF jamming attack, the jammer follows a cyclical pattern switching between OFF and ON states. In our simulations, the jammer is active for one second and inactive in the next second. A proactive jammer always jams the network during ON cycle. But the reactive jammer only performs the jamming task when it senses an ongoing transmission over the channel in ON state. As a result, the jammer detection methods cannot find enough evidence to detect the jammer, and the jammer remains hidden in the network for a longer time. Also, IoT devices follow a random pattern to generate data packets in the network layer. They firstly wait for a random time between (0,1)s and then generate a data packet, and continue this pattern to generate another data packets.

The simulations for ON-OFF attack have been shown for unicast, broadcast, and multicast scenarios in Fig. 9. As shown in Fig. 9a, the NJS accuracy for both reactive and proactive cases in broadcast scenario is at least 20% better than other jammer detection methods. Also, the NJS precision is always 100% and it is at least 23% better than other jammer detection methods. It implies that NJS does not generate any false positive. Similarly, NJS specificity is equal to 1. It means that NJS also correctly identifies all non-jammed nodes (i.e. *fp* is zero). However, other jammer detection methods have lower specificity because due to false positive errors. For example, PDR_T and BT_T have specificity equal to zero because these

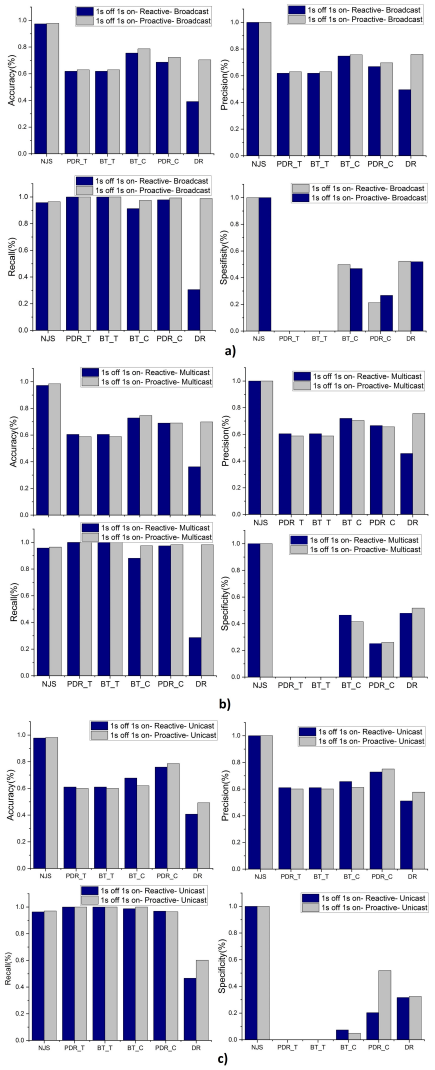

Fig. 9. The performance of evaluated methods during the ON-OFF attacks for a) Broadcast scenario, b) Multicast, and c) Unicast.

methods labels all nodes as jammed nodes. In terms of recall, NJS is 1-2% lower than PDR_T and BT_T. The recall of DR [28] is less than other methods since the number of undetected jammers (fn) is comparable to the number of detected ones in this methods. This phenomenon is accentuated when the network traffic is increased as it is evident in fig. 12. By considering all performance metrics, including accuracy, as the most important metric, recall, precision, and specificity, the NJS has better performance among all jammer detection methods.

Fig. 9b reveals that NJS accuracy is at least 23% better than other jammer detection methods in the presence of the reactive and proactive jammers for the multicast scenario. In addition, the precision and specificity of the NJS metric are always one and at least 27% and 54% better than the other methods, respectively. However, NJS recall is 2-3% lower than PDR_T and BT_T because these methods predict almost all nodes as jammed nodes. The overall NJS performance is much better than other jammer detection methods. Figure 9c shows the results for the unicast scenario. The NJS accuracy is at

least 19% better than other jammer detection methods. The precision and specificity of NJS are always one and, at least 25% and 48% better than other methods, respectively. In the unicast scenario, more frames are exchanged due to data frame retransmissions and transmission of RTS, CTS, as well as ACK frames. Thus, BT_C predicts most nodes as jammed nodes, and its specificity is lower than NJS in broadcast and multicast scenarios.

According to [8], in DIFS jamming attack, a jammer will generate jamming signals over the channel after DCF Interframe Space (DIFS) time period. Therefore, jamming signals of the jammer will corrupt data packets in broadcast and multicast scenarios and RTS/CTS frames in unicast scenario. As shown in Fig. 11, the NJS accuracy is almost one and at least 21% better than other jammer detection methods for the broadcast scenario. The precision and specificity of NJS are always one and, 37% and 51% better than other jammer detection methods, respectively. However, in terms of recall, NJS is 1-2% lower than PDR_T and BT_T. The overall performance of NJS is much better than other jammer detection methods because NJS correctly detect jammed and non-jammed nodes in the network and does not have false positive. But, other jammer detection methods suffer from high false positive rate since they predict most nodes as jammed node. Similarly, the NJS performance for multicast and unicast are better than other jammer detection methods. For the brevity, we have just presented the result of the broadcast scenario.

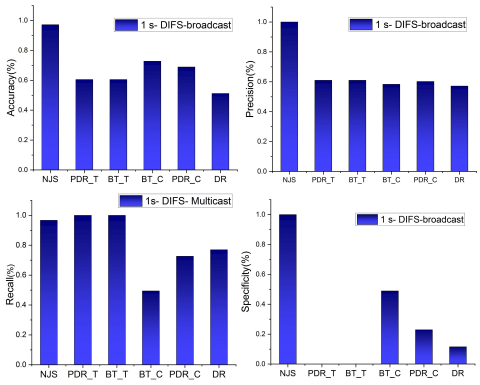

Fig. 10. The performance of NJS and the other jammer detecion methods during DIFS attacks in the broadcast scenario.

D. EFFECTS OF JAMMER DETECTION ON JAMMER LOCALIZATION

In the next set of experiments, we evaluate the performance of jammer detection methods on the jammer localization accuracy using centroid localization [12]. We measure JLE in the presence of one reactive or proactive jammer, two simultaneous reactive or proactive jammers, and a mix a proactive and a reactive jammer in the network. As shown in Fig. 11, NJS JLE values are at least 2.5X better than other jammer detection methods in all scenarios. In fact, other jammer detection methods cannot accurately determine the coverage area of the jammer due to high false positive and false negative rate.

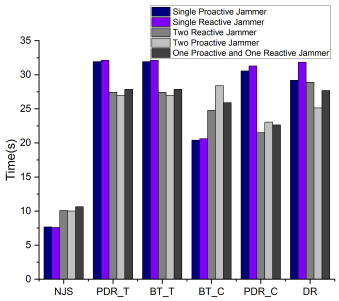

Fig. 11. The performance evaluation of the jammed detection methods on jammer localization.

E. EFFECTS OF BACKGROUND TRAFFIC

To evaluate the performance of NJS and other jammer detection methods under high traffic rates, we use (0,100ms) and (0,200ms) cases. In the random (0,100ms) case, each node generates a random number r between 0 and 100ms and waits for r to generate a data packet in the network layer. After generating a packet, the node will generate another random number r between 0 and 100ms and follow this pattern until the end of the simulation. Clearly, the frame generation rate of (0,200ms) scenario is two times larger than (0,100ms) scenario on average.

the simulation results for the broadcast scenario are provided in Fig. 12a. The NJS accuracy is 27% and 15% better than other methods in the presence of a proactive jammer for a transmission rate of (0,100ms) and (0,200ms), respectively. Also, the NJS accuracy in the presence of a reactive jammer for a transmission rate of (0,100ms) and (0,200ms) are 26% and 20% better than other jammer detection methods. In the proactive case, NJS finds more footprints of the jammer at the higher rate than the lower rate, so its accuracy for the higher rate is better than the lower transmission rate. However, the NJS accuracy for the reactive case for (0,200ms) is lower than (0,100ms) because NJS cannot find enough evidence in the higher rate. In another word, in most cases, there is a valid sender in the vicinity of the node that received a corrupted frame. Moreover, the precision for NJS metric is equal to 1 and it is at least 34%, 21% better than other jammer detection methods for the both transmission rates in the presence of a proactive jammer. Additionally, the NJS precision in detecting reactive jammer is 37% and 22% better than other methods for transmission rate of (0,100ms) and (0,200ms), respectively. However, the recall of NJS in the proactive (reactive) scenario is 4% and 3% (16% and 5%) lower than the current best among other methods for (0,100ms) and (0,200ms) cases. The overall performance of the NJS is much better than other methods because NJS correctly detect jammed and non-jammed nodes due to its zero false positive rate.

Fig. 12b shows the performance evaluation for the multicast scenario. Similar to the broadcast case, the accuracy, precision, specificity, and overall performance of NJS is better than other jammer detection methods. Figure 12c shows the performance evaluation for the unicast scenario. The accuracy of NJS in the presence of a proactive jammer is 16% and 29% better than other jammer detection methods for the transmission rate

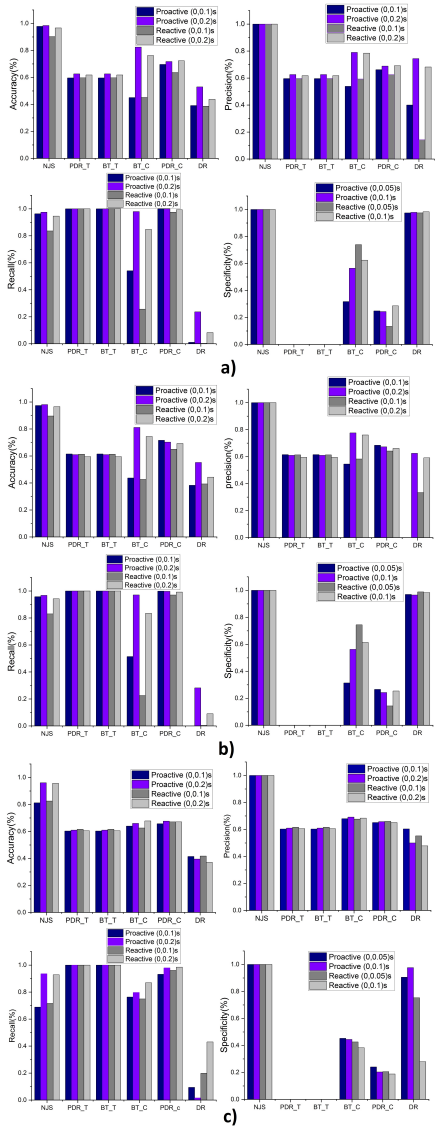

Fig. 12. High traffic rate in a)broadcast, b)multicast, and c)unicast.

of (0,100ms) and (0,200ms). Also, for the reactive jammer case, the accuracy of NJS is 15% and 28% better than other methods for two transmission rates. Since in the unicast scenario, many legal transmissions, including RTS, CTS, and ACK and data frame, are occurred around each wireless nodes. Therefore, it becomes harder for NJS to filter these events and detects the jammer activities. However, NJS is still better than other methods in terms of precision, specificity, and overall performance

F. NJS PROPERTIES

The first detection time metric is depicted in Fig. 13. Clearly, NJS detects reactive and proactive jammers sooner than other methods. It is four times faster than the closest rival (i.e. BT_T).

Next, we evaluate the ability of the NJS metric in detecting two adjacent reactive jammers, as depicted in Fig. 14. NJS1 and NJS2 are calculated for the nodes in the vicinity of jammer 1 and jammer 2. Evidently, the NJS value for the nodes in the

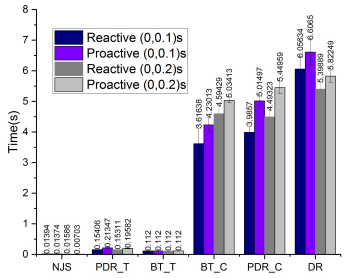

Fig. 13. First time detection of a jammer.

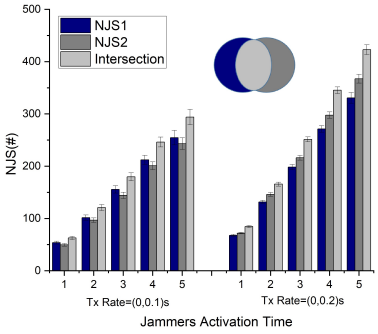

Fig. 14. The NJS values when two active jammers simultaneously jam the network.

intersection area is higher than NJS1 and NJS2. It means that the proposed algorithm is able to deal with this situation and detect both jammers.

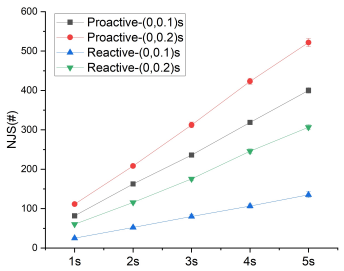

Fig. 15. The NJS value during the activation time of a jammer.

Figure 15 shows the average of the NJS values for all jammed nodes vs. the jammer activation time. In the proactive scenarios, the jammer always occupies the same number of slots proportional to its ON/OFF rate. Hence, the number of frame collisions in (0,100)ms 2 configuration is higher than (0,200)ms ones. As a result, NJS is lower in (0,100)ms case. Similarly, one could conclude that the NJS value should be lower in (0,100)ms configuration for reactive scenarios as depicted in the figure.

VI. CONCLUSIONS

The jammers exploit the open access nature of the shared wireless media to execute sophisticated jamming patterns and disrupt the communication for wireless nodes. The current proposals for jamming detection and localization either need a proprietary hardware or suffer from subpar performance.

In this study, a local, straightforward, and numerical metric, called NJS, is proposed by which reactive and/or proactive jammers are detected and localized quickly and precisely. NJS is superior to current jamming detection proposals in terms of accuracy, recall, and specificity. Also, it determines affected IoT devices accurately. Then, we can use the location of affected device to locate the jammer. NJS's accuracy in locating the jammer is also superior to current localization methods. NJS's operation and performance is independent from jammers type and numbers.

REFERENCES

[1] M. Abdollahi, K. Malekinasab, W. Tu, and M. Bag-Mohammadi, "An efficient metric for physical-layer jammer detection in internet of things networks," in *Proc. of IEEE LCN*, 2021.

[2] B. Upadhyaya, S. Sun, and B. Sikdar, "Machine learning-based jamming detection in wireless iot networks," in *Proc. of IEEE APWCS*, 2019.

[3] A. Hussain, N. Abughanam, J. Qadir, and A. Mohamed, "Jamming Detection in IoT Wireless Networks: An Edge-AI Based Approach," in *Proc. of ACM IoT*, 2022.

[4] S. Hong, K. Kim, and S.-H. Lee, "A Hybrid Jamming Detection Algorithm for Wireless Communications: Simultaneous Classification of Known Attacks and Detection of Unknown Attacks," *IEEE Communications Letters*, 2023.

[5] K. Pelechrinis, I. Koutsopoulos, I. Broustis, and S. V. Krishnamurthy, "Jammer localization in wireless networks: An experimentation-driven approach," *Computer Communications*, vol. 86, pp. 75–85, 2016.

[6] D. T. Hoang, D. N. Nguyen, M. A. Alsheikh, S. Gong, E. Dutkiewicz, D. Niyato, and Z. Han, "Borrowing arrows with thatched boats: The art of defeating reactive jammers in iot networks," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 79–87, 2020.

[7] Y. Ju, M. Lei, M.-M. Zhao, M. Li, and M. Zhao, "A joint jamming detection and link scheduling method based on deep neural networks in dense wireless networks," in *Proc. of IEEE VTC2019-Fall*.

[8] X. Wei, Q. Wang, T. Wang, and J. Fan, "Jammer localization in multi-hop wireless network: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 765–799, 2016.

[9] A. Hussain, P. Tedeschi, G. Oligeri, A. Mohamed, and M. Guizani, "Energy-Harvesting Based Jammer Localization: A Battery-Free Approach in Wireless Sensor Networks," in *Proc. of IEEE GLOBECOM*, 2022.

[10] T. Wang, X. Wei, J. Fan, and T. Liang, "Jammer localization in multihop wireless networks based on gravitational search," *Security and Communication Networks*, vol. 2018, 2018.

[11] S. Singh and C. Sharma, "Range free localization techniques in wireless sensor networks: A review," *Procedia Computer Science*, vol. 57, pp. 7–16, 2015.

[12] H. Liu, X. Wenyuan, Y. Chen, and Z. Liu, "Localizing jammers in wireless networks," in *Proc. of IEEE PerCom*, 2009.

[13] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proc. of ACM MOBIHOC*, 2005.

[14] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: attack and defense strategies," *IEEE network*, vol. 20, no. 3, pp. 41–47, 2006.

[15] M. Cheng, Y. Ling, and B. Wu, "Time series analysis for jamming attack detection in wireless networks," in *Proc. of IEEE GLOBECOM*, 2017.

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

[17] Z. Liu, H. Liu, W. Xu, and Y. Chen, "An error-minimizing framework for localizing jammers in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 508–517, 2013.

[18] J. Fan, T. Liang, T. Wang, and J. Liu, "Identification and localization of the jammer in wireless sensor networks," *The Computer Journal*, vol. 62, no. 10, pp. 1515–1527, 2019.

[19] Z. Liu, H. Liu, W. Xu, and Y. Chen, "Error minimizing jammer localization through smart estimation of ambient noise," in *Proc. of IEEE MASS*, 2012.

[20] R. D. Halloush, "Transmission early-stopping scheme for anti-jamming over delay-sensitive iot applications," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7891–7906, 2019.

[21] H. A. B. Salameh, S. Almajali, M. Ayyash, and H. Elgala, "Spectrum assignment in cognitive radio networks for internet-of-things delay-sensitive applications under jamming attacks," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1904–1913, 2018.

[22] Y. Cai, K. Pelechrinis, X. Wang, P. Krishnamurthy, and Y. Mo, "Joint reactive jammer detection and localization in an enterprise wifi network," *Computer Networks*, vol. 57, no. 18, pp. 3799–3811, 2013.

[23] M. Çakiroğlu and A. T. Özcerit, "Jamming detection mechanisms for wireless sensor networks," in *Proceedings of the 3rd international conference on Scalable information systems*, 2008.

[24] 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, 2010.

[25] N. V. Abhishek and M. Gurusamy, "Jade: Low power jamming detection using machine learning in vehicular networks," *IEEE Wireless Communications Letters*, vol. 10, no. 10, pp. 2210–2214, 2021.

[26] B. Turan, A. Uyrus, O. N. Koc, E. Kar, and S. Coleri, "Machine learning aided path loss estimator and jammer detector for heterogeneous vehicular networks," in *Proc. of IEEE GLOBECOM*, 2021.

[27] M.-A. El Houssaini, A. Aaroud, A. El Hore, and J. Ben-Othman, "Detection of jamming attacks in mobile ad hoc networks using statistical process control," *Procedia Computer Science*, vol. 83, pp. 26–33, 2016.

[28] J. Singh, I. Woungang, S. K. Dhurandher, and K. Khalid, "A jamming attack detection technique for opportunistic networks," *Internet of Things*, vol. 17, p. 100464, 2022.

[29] X. Wei and T. Wang, "Aigsa-based multi-jammer localization in wireless networks," *Applied Soft Computing*, vol. 103, p. 107131, 2021.

[30] S. Kumar, K. Singh, S. Kumar, O. Kaiwartya, Y. Cao, and H. Zhou, "Delimitated anti jammer scheme for internet of vehicle: Machine learning based security approach," *IEEE Access*, 2019.

[31] Y. Lyu, Y. Mo, S. Yue, and W. Liu, "Improved beetle antennae algorithm based on localization for jamming attack in wireless sensor networks," *IEEE Access*, vol. 10, pp. 13 071–13 088, 2022.

[32] H.-H. Ng, W.-S. Soh, and M. Motani, "Maca-u: A media access protocol for underwater acoustic networks," in *Proc. of IEEE GLOBECOM*, 2008.

[33] Y. Zhao, Y. Xiang, L. Xu, and M. Shi, "A multi-channel medium access control protocol for multicast in mobile ad-hoc network," in *Proc. of IEEE PIMRC*, 2003.

[34] P. Ferrari, P. Bellagente, A. Depari, A. Flammini, M. Pasetti, S. Rinaldi, and E. Sisinni, "Evaluation of the impact on industrial applications of ntp used by iot devices," in *Proc. of IEEE International Workshop on Metrology for Industry 4.0 & IoT*, 2020.

[35] L. Mokdad, J. Ben-Othman, and A. T. Nguyen, "Djavan: Detecting jamming attacks in vehicle ad hoc networks," *Performance Evaluation*, vol. 87, pp. 47–59, 2015.

[36] J. Kuruvila, A. Nayak, and I. Stojmenovic, "Hop count optimal position-based packet routing algorithms for ad hoc wireless networks with a realistic physical layer," *IEEE Journal on selected areas in communications*, vol. 23, no. 6, pp. 1267–1275, 2005.