An Efficient Metric for Physical-layer Jammer Detection in Internet of Things Networks

Mostafa Abdollahi *, Kousar Malekinasab†, Wanqing Tu‡, Mozafar Bag-Mohammadi*
*Engineering Faculty, Ilam University, Iran
†Iran University of Science and Technology, Department of Computer Engineering, Iran
‡Department of Computer Science, Durham University, United Kingdom

***Abstract*—An active jammer could severely degrade the communication quality for wireless networks. Since all wireless nodes openly access the shared media, the harsh effects are exaggerated by retransmission attempts of affected devices. Fast and precise detection of the jammer is of vital importance for heterogeneous wireless environments such as the Internet of things (IoT). It could activate a series of corrective countermeasures to ensure the robust operation of the network. In this paper, we propose a local, straightforward, and numerical metric called the number of jammed slots (NJS), by which we can quickly detect the presence of a jammer and identify the jammed nodes at the software level in broadcast networks. NJS calculation is carried out by a central node which collects the MAC-layer statuses of all wireless nodes in a periodical fashion. Our simulation results indicate that NJS outperforms current detection methods in terms of accuracy and precision.**

***Index Terms*—Jamming, jammer detection, jammer localization, IoT networks.**

I. Introduction

The Internet of Things (IoT) is an emerging paradigm that intends to connect a massive number of devices to the Internet and develop smart services for users. The IoT networks are vulnerable to a variety of security threats because of their large scale and heterogeneity. Jamming attacks are among the most important issues of IoT networks. The IoT devices might be exposed to undesirable transmissions of wireless devices of adjacent networks (i.e., unexpected jammers) and intelligent jammer (intentional jammer). An intentional jammer deliberately generates jamming signals over the transmission channel to hinder legitimate transmissions, increase the channel busy time, disrupt the packet receptions in the receivers, and drain the battery of IoT devices degrading the overall performance of the IoT network. The proactive and reactive jammers, as the time domain physical layer jammers [1], are two common vulnerabilities of IoT networks. A reactive jammer sends radio signals over the channel after sensing an ongoing wireless transmission. In contrast to reactive jammers, a proactive jammer continuously jams the channel without sensing the channel for the existence of any ongoing transmission before sending the jamming signals.

The hardware approach and statistical indices are two common methods to detect and localize proactive and reactive jammers in IoT networks. The former detects jammers at the hardware level by detecting their physical footprint, measuring signal strength in receiving nodes [2], [3]. The latter uses some predefined statistical indices, such as packet delivery ratio (PDR) and busy time, [4] to estimate the probable presence of a jammer in the network based on its effects on transmission and reception. Clearly, the hardware-based methods require specialized hardware to detect the signal strength of the jammer. The majority of the deployed IoT nodes are low-cost devices with limited processing power and storage capacity which makes the special hardware solution infeasible in today's heterogeneous IoT environments [5]. On the other hand, the statistical approach suffers from the false jammer detection (false positive), inherent inaccuracy of the statistical measures, and sensitivity to the traffic variation.

We noted that the cooperation between MAC layers of individual IoT devices enables us to detect jammers in a simple and reliable way and overcome the above challenges. As an intuitive example (see Fig. 1), suppose that the node *S* sends a data packet over the channel in the presence of a reactive jammer. The radio ranges of *S* and the jammer are shown with two circles around them. The jammer tries to disturb this transmission by generating noise signals during the transmission. Naturally, the IoT devices in the hatched area cannot decode the reception due to the collision between the sender and the jammer. Also, the IoT devices in the gray area cannot infer the jammer presence even though they receive an intact signal from the jammer. If we force all nodes to report their radio states in the MAC-layer alongside their locations to a central node, the central node has enough information to deduce that there is no legitimate sender in the gray area. It then came to the conclusion that the meaningless receptions in that region are caused by a jammer. For the node in the hatched area, the central node could not find another legitimate sender other than *S*. Therefore, it assumes that the collisions in this area have resulted from the presence of a jammer. It could even detect the jammer location by using well-known techniques such as the weighted centroid algorithm [6]. *Therefore, corrupted receptions without any legitimate sender are considered to be from the presence of a jammer, and corrupted receptions with a legitimate sender are blamed on normal wireless environments' events such as fading, collision, propagation, etc.*

In this paper, we proposed a reliable method to detect the physical layer jammers in wireless IoT networks. In our method, the wireless devices record their MAC-layer states in each timeslot. The MAC-layer state can be *idle*, *receiving*,

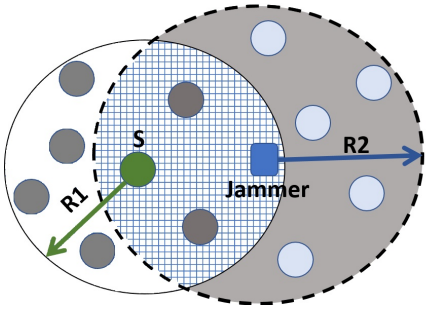


Fig. 1: A legitimate sender, *S*, (with Tx range *R1*) sends a data packet, and a reactive jammer (with Tx range $R2 \geq R1$) transmits radio signals over the channel simultaneously.

transmitting, or *corrupted*. The corrupted state occurs when a node receives a meaningless signal which could be resulted from the 1) collision, 2) presence of a jammer, or 3) bit-level errors. A batch (i.e., timeline) of recorded state within time interval $[t1, t2]$ is sent to a central node in a periodic manner. The central node calculates the number of jammed slots (NJS) for each node. For each specific time slot, it increases the NJS of a node if the node is in *receiving* or *corrupted* states, but there was not a legitimate wireless device around it. The non-zero NJS value for a node indicates that a jammer exists in the IoT network. By comparing the NJS values of neighboring nodes, we can estimate the number of jammers and their approximate locations. The simulation results show that the precision and accuracy of an NJS-based method are 6% and 33% better than the state of art methods. Also, the NJS metric is four times faster than other methods in determining when a jammer is activated in the IoT networks.

In section 2, we briefly review the related works. The next section is devoted to the system model and the main body of the proposed method. Simulation results are presented in section 4. Finally, section 5 concludes the paper.

II. Related Works

The detection of physical layer jammers is done via hardware-assisted approaches and statistical indices methods. In [3], [7], the affected nodes detect the constant jammer by sensing the strength of the jamming signal (JSS). They also report their measurements to a designated node in wireless networks. The designated node looks for the jammer location based on JSS data from the nodes near the jammer using a proprietary algorithm. In [2], a machine-learning-based scheme is proposed which requires anchor nodes to collect information such as jammers' signal strength from the transmission medium. The information is then used to train a decision tree algorithm in order to accurately detect or predict a jammer. To detect a reactive jammer in IoT networks, [8] proposed a hardware-based method that compares the receiving signals with the signal of conventional IoT devices using a detector circuit. After detecting a jammer, the IoT devices stimulate the jammer to attack the channel by generating fake transmissions. They use the jamming signal to harvest energy or as a communication means to transmit their data through it using RF energy harvesting and ambient backscatter techniques.

In [9], two connected neural networks are studied to detect jammers and schedule links. The first neural network detects jammers based on the location of ordinary wireless nodes and their received signal strength. The second neural network is for scheduling links by using the results from the first neural network as its input. The localization of a directional jammer has been studied in [10]. Based on the jamming signal strengths from the boundary (i.e. in the border of the coverage area of the jammer) and jammed nodes, and the geographical locations of jammed nodes, an algorithm is presented to find out affected nodes in a directional coverage of the jammer as well as determine the transmission ranges of such jammers.

In statistical indices methods, the jammers are detected by measuring their footprints in the transmission medium and wireless nodes. The jammer activity alters the statistical indices of the network. These changes are reflected through some discrete time measurements. When a meaningful difference is found in two consecutive measurements, an active jammer is detected. The gradient descent method [4] is used to detect jammers by tracing PDR reduction in the network. In [11], the packet transmission time is used as a statistical index to detect the presence of jammer in time-critical IoT applications. If the packet transmission time exceeds a threshold, the node will conclude the delay is resulted from a simultaneous jammer activity. Hence, it switches to a safe channel and retransmits the packet. Similarly, in the context of cognitive radio networks for IoT applications, [12] used a packet-invalidity ratio that represents the probability that the packet transmission delay is greater than a threshold. When the ratio exceeds the threshold, the IoT device implicitly assumes the presence of a jammer in the network and switches to a secure channel to transmit the packet.

In [13] and [14], a variation of carrier sense time (CST), PDR, and packet send ratio (PSR) are examined when a jammer is active. They found that PSR and PDR are decreased, and CST is increased dramatically in the presence of an active jammer. When the jammer acquires the channel completely, the aforementioned indices (i.e., PDR and PSR) became zero. In [15], the jammer is identified by comparing the number of idle and busy slots before and after the activation of the jammer. Each node maintains a profile containing past medium status, which is updated periodically. A central node collects all profiles and processes them to detect the jammer. Reference [16] uses two metrics, i.e., PDR and bad packet ratio (BPR), for identifying an active jammer. They found that in the presence of an active jammer, PDR is deceased and BPR is increased. There are other proposals that use similar methods for detecting the jammer [1], [17].

In [18], a new detection method is proposed using a time series analysis. They 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. We have implemented their method for the comparison purpose. In order to detect jammers based on statistical indices methods, existing studies employed metrics such as PDR and BPR to evaluate the receiving performance. However, in the lack of an additional helping method, these metrics may not tell whether the performance degradation along these metrics is caused by jamming or other reasons (e.g., wireless overload links). In addition, the existing metrics are sensitive to fading, link quality, etc.

III. The Number of Jammed Slots Metric

A. System Model

In our system, there are N wireless nodes and a central control node (e.g., a wireless gateway). The MAC-layer of a wireless node n_i ($i \in [0, N-1]$) could have one of the following four statuses:

- *Idle*: n_i is not sending signals or occupying the channel;
- *Receiving*: n_i is receiving a frame correctly from the channel;
- *Transmitting*: n_i is sending a frame to the channel;
- *Corrupted*: n_i is receiving a corrupted frame from the channel due to collision, fading, jamming, etc.

The wireless node records the channel status from its viewpoint periodically. The timeline is divided into slots (periods) of a fixed length. The slot length τ is proportional to the time required to send/receive a data frame. Clearly, τ value must be slightly greater than the normal packet length divided by the channel bit rate. For example, τ could be 7ms when bandwidth is 2Mbps and the data frame length is 1518B. All wireless nodes are synchronized through NTP [19]. This would allow us to have a clear snapshot of the MAC-layer status from different viewpoints in each time slot. It is worth mentioning that each node could be in only one state in each time slot. Each wireless node n_i reports the collected MAC-layer statuses from the last m time-slots to a central control node. The central control node, after receiving wireless nodes' reports, runs the NJS algorithm (subsection III-C) and compares the timeline of individual nodes using network topology information, such as transmission range and location of nodes, to infer jamming attacks. Similar to other well-known routing metrics in wireless networks [20], the reports could be delivered to the central node by flooding or unicasting methods in arbitrary time intervals (e.g. 0.5-1 minute).

B. The NJS Background

Our basic idea is to track the jammer's footprint among the reported MAC statuses at central node to detect jamming signals. First, it 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, it treats all erroneous receptions in the neighborhood of a valid sender as normal channel errors. Therefore 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 is defined as a circle with a radius greater than that of a wireless node's transmission range.

Proposition 1: *If we can find at least one time slot in which almost all nodes that reside in a collision area report multiple jammed frames, there is a proactive jammer in the center of the collision area.*

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. Therefore, 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 timeslots, there is a reactive jammer in the network.*

C. The NJS Algorithm

We need a carefully crafted algorithm to determine jammed nodes and the number of jammed slots for those nodes in the presence of a jammer using the defined assertion and propositions. The algorithm calculates the number of jammed slots from the node's perspective in the most recent m timeslots. 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 timeslot 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 nodes in the intersection area of a sender and the jammer. For those nodes, the corrupted receptions are blamed on the presence of the jammer and their NJS_{imp} is increased per corrupted timeslots. 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 timeslots. For each receiver

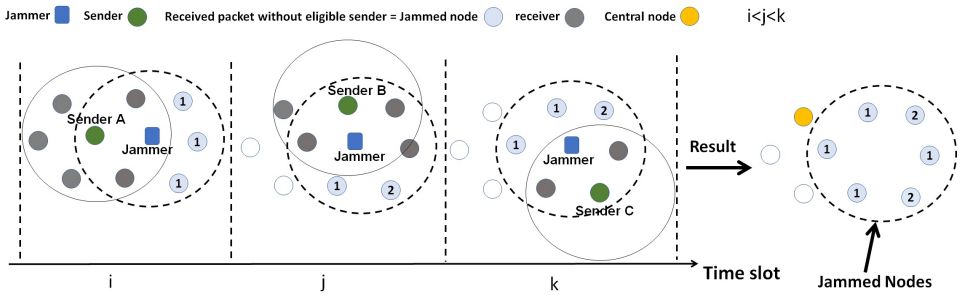


Fig. 2: The calculation of explicit NJS.

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 return !ValidSender
18 **Function** `IsThisAJammedSlot` (*slot i,node j*) **:**
19 Result=false
20 **if** $Timeline_{ji} != Idle$ **then**
21 **for** *node* $k \leftarrow 0$ *to* N **do**
22 **if** $k != j \&\& distance_{kj} < 2R$ $\&\&CheckJammedNode(i,k) == true$ **then**
23 Result=true
24 return Result
25 **comments:**
26 $Timeline_{ji}$ == the state of node i in slot j.
27 $distance_{kj}$ == the distance between nodes k and j
28 R = Tx range

node such as j in timeslot i, if it could not find a valid sender in its neighborhood, $NJS_{exp}[j]$ is increased. As shown in Fig. 2, the NJS_{exp} is increased for some nodes in time slot i in the presence of a reactive jammer. The number of jammed slots is written in the related circles. The same would happen for other nodes in timeslots j and k. Thus, all jammed nodes are detected after some slots. In lines 5-8, NJS_{imp} is increased for those nodes that their $NJS_{exp} > 0$. In Fig. 3, NJS_{imp} will be non-zero for jammed nodes that are in the intersection between the jammer and a sender. NJS_{imp} is useful for the cases when both a jammer and a valid sender are active at the same timeslot. The increment value is dependent on the number of timeslots in which the jammer is active. Finally, the value of the NJS metric for node j is calculated as follow:

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

It is worth mentioning that although the nodes in the intersection between the sender (e.g. sender A) and the jammer have a valid sender, the random access to the channel performed by all wireless nodes gives this opportunity for jammed nodes to have non-zero NJS_{exp} when they are not in the range of a valid sender (e.g. jammed nodes in slots j and k). Also, when a proactive jammer is active in the network, all jammed nodes almost have the same NJS_{exp} because the jammed nodes will record the same number of $jammed$ slots.

In Fig. 3, we have shown that how NJS detects a reactive jammer. In the beginning, the jammer is silent. When the jammer detects some active senders in the network, it jams the network in consecutive timeslots. Since there is no legitimate sender around D in timeslots 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 timeslots, 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. We note that the computational complexity of the algorithm is $\mathcal{O}(N^2 m)$, where N and m are the number of nodes and slots in each timeline, respectively.

D. 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.

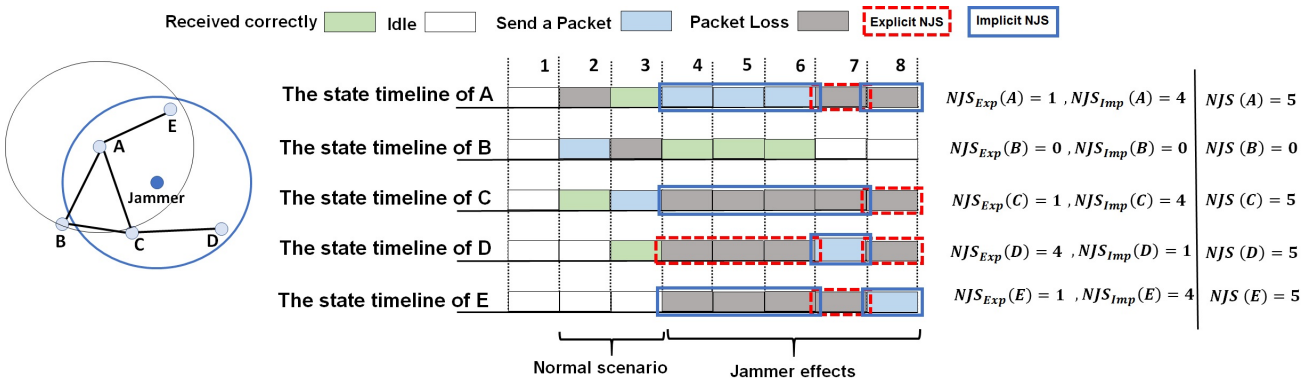


Fig. 3: The calculation of NJS_{imp} is illustrated.

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 nodes that exist in the intersection area of two active jammers are expected to be more (see Fig. 4).

Some interesting scenarios are depicted in Fig. 5. In Fig. 5-a, a proactive jammer is active in the network and all transmitters are inactive due to channel business. Here, the nodes have received a signal/packet on the channel from an invalid sender. Therefore, the central node will easily locate the proactive jammer using proposition 1 since the NJS_{exp} for all jammed nodes is almost the same. In other words, according to proposition 1, for a timeslot j, NJS_{exp} of all the jammed nodes for that timeslot will be 1 and NJS_{imp} will be 0. In Fig. 5-b, a legitimate sender (i.e., A) starts sending a data packet; the jammer is notified and jams the network simultaneously. The nodes that located at the intersection of the jammer and the sender will receive erroneous packets. At the same time, some nodes will receive the data packet correctly, while there are some nodes that only receive the jammed packets. The central node 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 for the timeslot).

A more challenging scenario is depicted in Fig. 5-c where a reactive jammer exists in the network. The jammer is completely in the transmission range of node A. The transmission range of node B has some overlaps with the transmission range of the jammer. Although the jammer is not detectable when A is active, it could be easily detected when B is active since the NJS value is more than one for the node marked by the bold arrow. It is worth noticing that NJS could detect a jammer that copies legitimate packets from other sources. In that case, the central node will find that some nodes have received a packet from an inactive sender.

IV. Performance Evaluation

A. Performance Metrics

- *The Precision and accuracy metric*: The true positive (tp) (true negative (tn)) outcome happens when a jammed (normal node) is correctly detected by the system. Also, the false positive (fp) and false negative (fn) outcomes occur when the system incorrectly predicts jammed and normal nodes, respectively. With these terms, precision and accuracy, as important measures in the field of information retrieval that determine the degree of suitability of documents retrieved by the system, are defined as follows:

$$Precision = \frac{tp}{tp + fp} \tag{2}$$

and

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad . \tag{3}$$

In the jammer localization problem, the jammed nodes should locate the jammer in the network. In such applications, the behavior of normal nodes is irrelevant (i.e. tn and fn) since it does not affect the jammer's location. Hence, the scheme's precision is more important than its accuracy. In contrast, all nodes are participating in rerouting the traffic to avoid the jammed area. Here, the behaviors of both jammed and non-jammed nodes are important. For such applications, accuracy is a better metric.
- *The First detection time*: To calculate the speed of detecting jammed nodes in the network, we defined the first detection time metric. This metric reflects how fast the jammer is detected by the scheme. It is defined as the time difference between the jammer activation time and the earliest jammer detection time reported or detected by any normal node.

B. Simulations

We used an improved version of MIXIM 2.3 framework and omnet++ simulator to implement all the simulations. All nodes are based on 802.11 MAC protocol that opportunistically broadcast the data packets and RTS/CTS is disabled.

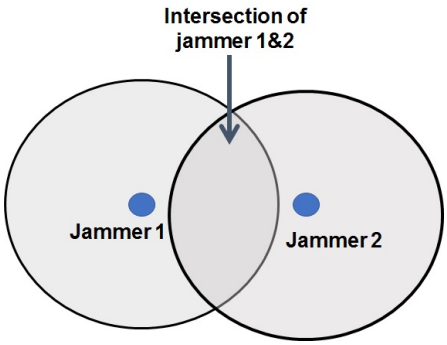


Fig. 4: The intersection of two active jammers.

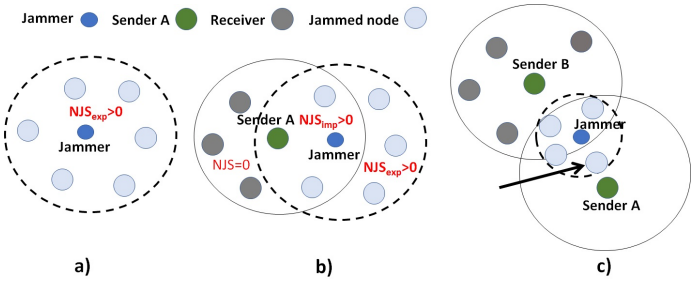


Fig. 5: The noise generation scenarios.

TABLE I: The simulation parameters.

Name	Value
Tx radius of nodes (R)	220m
Tx radius of jammer	R and 1.5R
CTS/RTS	Disabled
Bandwidth	2Mb
Tx rate of nodes	Random(0,50)ms/Random(0,100)ms
Packet size of nodes	1500 B
Packet (noise)size of jammer	500, 1000 B

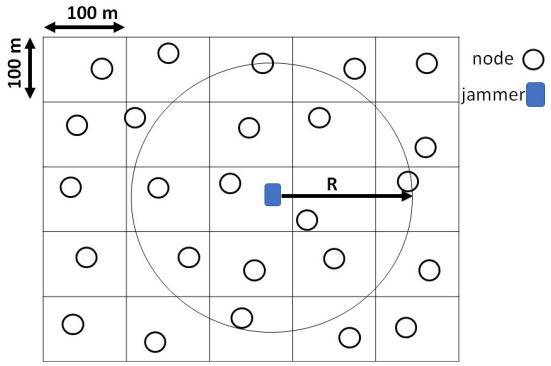
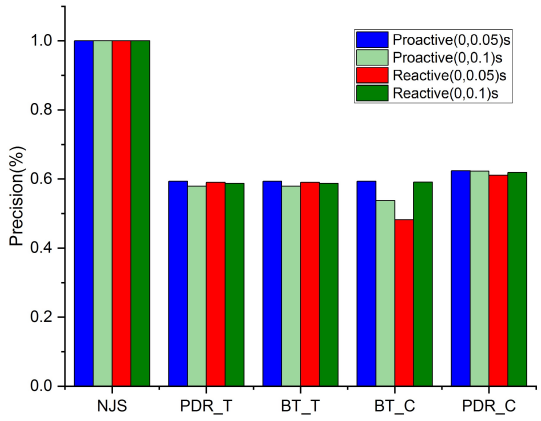


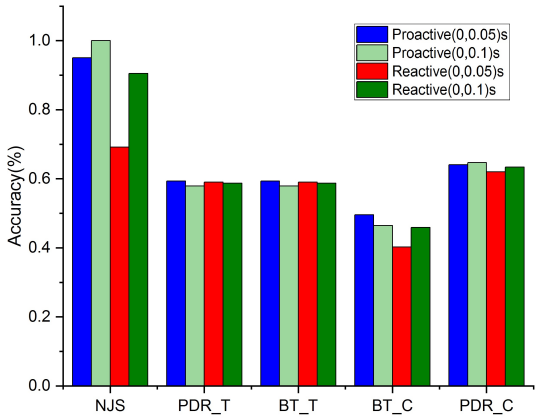
Fig. 6: A generated network topology with a random distribution of the nodes.

The channel is exposed to the normal packet loss due to fading, propagation, etc. Thus, to calculate the probability of receiving a packet successfully as a function of the distance between two nodes, we used the model in [21]. They employ path loss exponent or power attenuation factor $\beta = 2$ and maximum transmission range of R. The simulation parameters are depicted in the table I.

In order to evaluate the efficiency of the proposed metric, we have implemented NJS metric alongside conventional PDR (labeled as PDR-c) [22] and busy-time (labeled as BT-c) metrics [13]. Also, we have implemented the time series analysis as proposed in [18] with z=0.2. It considered network measurements taken over time as a time series and proposed an algorithm for detecting the change of state in the time series. We have applied their algorithm to PDR and busy-time



(a)



(b)

Fig. 7: a) The precision of detected jammed nodes and b) accuracy of detected jammed and non-jammed nodes for evaluated methods.

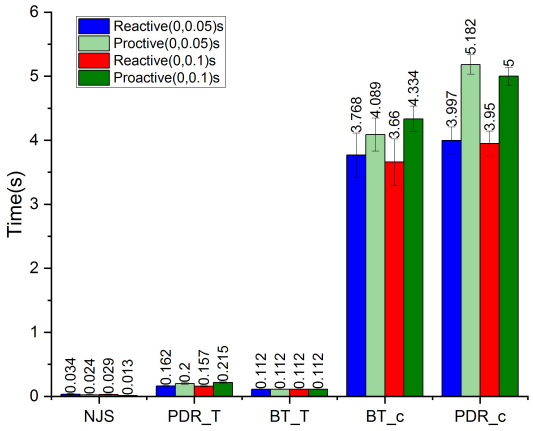


Fig. 8: First time detection of a jammer.

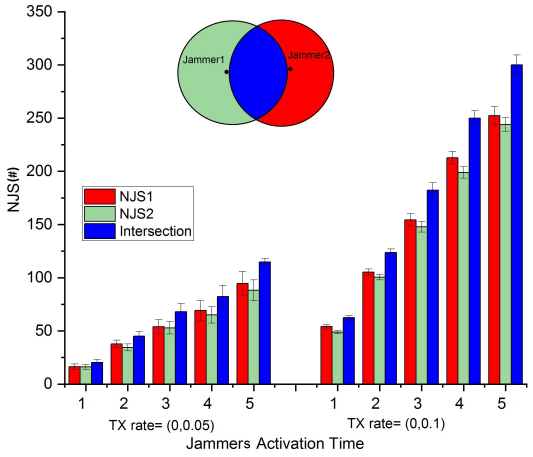


Fig. 9: The NJS values when two active jammers simultaneously jam the network.

measurements. The resulted methods are called PDR-t and BT-t for convenience. In PDR-c (BT-c), the presence of jammer is announced when the average PDR (BT-c) is decreased (increased) by 10% in comparison to the past second. In PDR-t and BT-t, the measurement is updated every 20 timeslots (i.e. 140ms). The overall simulation time is 10 seconds. The network topology is a simple grid as shown in Fig. 6. For each square, a wireless node is placed in a random location inside the square. Finally, each simulation experiment is repeated 50 times. Also, the confidence interval of 95% shows for the diagrams.

Fig. 7-a depicts the precision of evaluated methods for reactive and proactive jammers for different traffic ratios. For example, in a random (0,0.05) scenario, each node generates a random number r between 0 and 50 ms. It then waits for r seconds and transmits a data frame. It then generates the next random numbers. Clearly, the frame generation rate of (0, 0.1) scenario is two times larger than (0,0.05) scenario on average. In both scenarios, NJS precisely detects the jammer. In contrast, other methods could not detect near 40% of jammer activities. In Fig. 7-b, the accuracy of evaluated methods is depicted. Clearly, NJS is the most accurate method. Its accuracy for low-rate traffic is 100%. The accuracy decreases slightly for high-rate traffic. In that case, the wireless media is nearly occupied by the jammer and NJS could not find enough evidence to detect the jammer. This phenomenon is accentuated for high-rate traffic in reactive jammer scenarios.

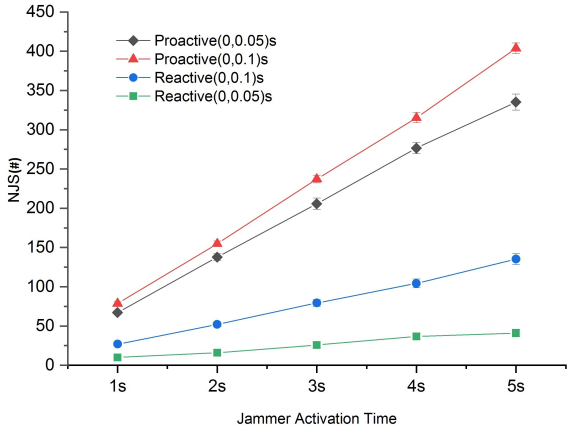


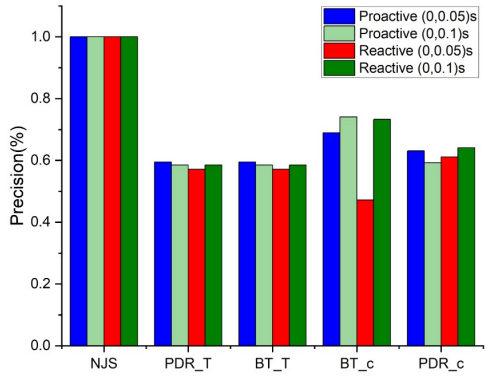
Fig. 10: The NJS value during the activation time of a jammer.

The first detection time metric is depicted in Fig .8. Clearly, NJS detects reactive and proactive jammers sooner than other methods. It is four times faster than 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. 9. 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 intersection area is higher than NJS1 and NJS2. It means that the proposed algorithm is able to deal with this situation.

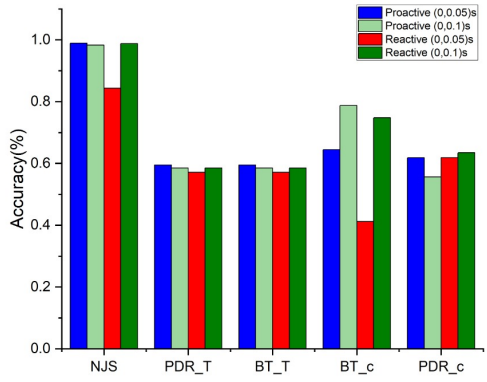
Fig. 10 shows the average of the NJS values for all jammed nodes vs. the jammer activation time. In the proactive scenarios, the jammer always occupies the same number of slots proportional to its rate. Hence, the number of collisions in (0,50) ms configuration is higher than (0,100) ms ones. As a result, NJS is lower. Similarly, NJS is lower in (0,50) ms configuration for reactive scenarios. Finally, we have evaluated NJS efficiency in the presence of a powerful jammer (Fig. 11). In this scenario, the jammer's transmission range is 50% more than normal nodes. Again, NJS precision is 100%. Also, its accuracy is more than 95% while other methods struggle to reach 60% accuracy in most cases.

V. Conclusion

The open access of the shared media allows active jammers to have severe effects on the communication quality of wireless nodes. Several direct and indirect methods are proposed to detect active jammers in wireless networks. However, the methods are not very reliable in terms of accuracy and precision. In this paper, we propose a local, straightforward, and numerical metric called the number of jammed slots (NJS), by which we can detect the proactive and/or reactive jammer quickly and



(a)



(b)

Fig. 11: a) The precision of detected jammed nodes and b) accuracy of detected jammed and non-jammed nodes for evaluated methods in the presence of powerful jammer (with Tx range of 1.5R).

precisely. The calculation of the metric is based on collecting the MAC-layer state of all wireless nodes in a periodical manner by a central node. Our simulation results indicate that the proposed method outperforms current detection methods in terms of accuracy and precision. As future work, we are working on a more general metric that utilizes the same idea to detect and locate the jammer in traditional wireless networks that use the unicast communication paradigm.

REFERENCES

[1] T. W. X. Wei, Q. 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.

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

[3] W. X. Z. Liu, H. Liu 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.

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

[5] A. Gouissem, K. Abualsaud, E. Yaacoub, T. Khattab, and M. Guizani, "Game Theory for Anti-Jamming Strategy in Multi-Channel Slow fading IoT Networks," *IEEE Internet of Things Journal*, 2021.

[6] H. Liu, X. Wenyuan, Y. Chen, and Z. Liu, "Localizing jammers in wireless networks," in *2009 IEEE International Conference on Pervasive Computing and Communications*. IEEE, 2009, pp. 1–6.

[7] W. X. Z. Liu, H. Liu and Y. Chen, "Error minimizing jammer localization through smart estimation of ambient noise," *MASS*, pp. 308–316, 2012.

[8] 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.

[9] M. Z. M. L. Y. Ju, M. Lei and M. Zhao, "A joint jamming detection and link scheduling method based on deep neural networks in dense wireless networks," *IEEE VTC*, 2019.

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

[11] 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.

[12] 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.

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

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

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

[16] M. Çakirolu and A. Özcerit, "Jamming detection mechanisms for wireless sensor networks," *Infoscale*, 2008.

[17] M. I. K. Pelechrinis and S. Krishnamurthy, "Denial of service attacks in wireless networks: The case of jammers," *IEEE Communications surveys & tutorials*, vol. 13, no. 2, pp. 245–257, 2010.

[18] Y. M. Cheng, Y. Ling and W. Wu, "Time series analysis for jamming attack detection in wireless networks," *GLOBECOM*, 2017.

[19] 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," pp. 223–228, 2020.

[20] M. Abdollahi, F. Eshghi, M. Kelarestaghi, and M. Bag-Mohammadi, "Opportunistic routing metrics: A timely one-stop tutorial survey," *Journal of Network and Computer Applications*, p. 102802, 2020.

[21] A. N. J. Kuruvila and I. Stojmenovic, "Hop count optimal position-based packet routing algorithms for ad hoc wireless networks with a realistic physical layer," *IEEE JSAC*, vol. 23, no. 6, pp. 1267–1275, 2005.

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