S. Frisbie and M. Younis, "AI-Enabled Jammer Deception Using Decoy Packets," *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, Rio de Janeiro, Brazil, 2022, pp. 5013-5018, doi: 10.1109/GLOBECOM48099.2022.10001651.

https://doi.org/10.1109/GLOBECOM48099.2022.10001651

AI-Enabled Jammer Deception Using Decoy Packets

Stephan Frisbie[1,2]
[1]*The Johns Hopkins University Applied Physics Laboratory*
Laurel, Maryland, 20723
Stephan.Frisbie@jhuapl.edu

Mohamed Younis[2]
[2]*Department of Computer Science and Electrical Engineering*
University of Maryland, Baltimore County
Baltimore, Maryland, United States
younis@umbc.edu

***Abstract*—In this work, we present a learning algorithm for a wireless communications network to transmit decoy packets to counter an adversarial sensing-reactive jammer. As the jammer is required to search across channels for data transmissions, decoy packets can have the effect of stalling the jammer on a particular channel, preventing it from continuing its search and leaving legitimate packets unimpeded. A reinforcement learning algorithm trains a deep neural network with an exploration-exploitation algorithm and experience replay. The state- and action-space and reward function are presented as components of the reinforcement learning framework. Our algorithm is tested with software simulations, modeling ZigBee communications nodes using time-division multiple access for medium access control. A reactive jammer is modeled in the simulation, with the goal of disrupting any detected ZigBee transmissions. A means to measure and distribute the reward function and system state to enable edge-learning in this context is presented as part of the implementation. The results demonstrate the effectiveness of our algorithm in mitigating the jamming attack, outperforming a random decoy strategy by a factor of two.**

***Index Terms*—anti-jamming, deep reinforcement learning, reactive jammer, wireless networks, internet of things**

I. Introduction

Wireless communications-enabled devices have become an integral part of society, including such application spaces as home and industrial automation, health care, transportation systems, first responders, etc. [1]. The proliferation of these devices regrettably is matched with increased exploitation by malicious actors. Denial-of-service (DoS) attacks, such as jamming, have raised security and even reliability concerns, where industrial equipment risks damage or destruction if the command and control backhaul is impeded, causing monetary loss to corporations [2]. Vehicle-to-vehicle wireless systems risk causing accidents, resulting in injury or worse, if interfered with by actors seeking to manipulate traffic in their favor or to achieve criminal objectives [3].

Defending against jammer systems has been of interest essentially since the advent of wireless communication technology. The increased availability of software defined radio (SDR) technology, as well as the feasibility of wireless attacks being demonstrated on cheap COTS equipment such as hacked WiFi dongles [2], have elevated the importance of defense strategies to an all-time high. Unless the frequency channel is known, the attacker generally targets a wide frequency band and either senses the spectrum to find active channels or just targets the entire band at once. The latter requires very large power and specialized jamming equipment. Classic approaches to jamming resilience include leveraging strong forward error correcting (FEC) schemes to maximize robustness to interference. Frequency diversity offered by spread spectrum techniques like frequency hopping (FHSS) or direct-sequence spread spectrum (DSSS) can further suppress the effects of a jamming signal.

Reactive jammers are a class of intelligent jammers that aim to cause DoS to a network using minimal power. These jammers sense the spectrum for activity by scanning the available channels and wait until transmissions are detected to launch their attack. Such an attack strategy provides two key benefits: (i) the jamming signal is enabled at a lower duty cycle, saving power, and (ii) it makes the jammer more difficult to detect. It is important to consider typical values for sensing and jamming times of a reactive jammer. Commercial radio frequency integrated circuits (RFIC) cannot make changes to their configuration instantaneously; typically, the following parameters will be specified by the manufacturer: the time required for the device to change frequency, change amplifier gain, and switch between transmit and receive mode. RFICs can take as long as hundreds of milliseconds to perform some changes, depending on the part and re-calibrations that must take place; although, in the context of a competent reactive jammer, these parameters are often cited in the tens to hundreds of microseconds [2]–[4].

We hypothesize that the reliance on time-segmented scanning leaves the jammer vulnerable to having its measurements polluted by decoy packets transmitted on intelligently-selected channels. These decoys can be transmitted with the intent of stalling the jammer or polluting its prediction of the data channel pattern. In this work, we aim to provide a framework to achieve this function as well as construct a software implementation to assess its performance. Our contributions are as follows: (i) we formulate a Q-learning framework to enable a wireless network to intelligently transmit deceptive transmissions, (ii) in the form of an algorithm, we present a practical means to measure the state of the environment such that online learning can take place, and (iii) we present simulation results using the NS-3 discrete event simulator.

The rest of the paper is outlined as follows. Section II covers related anti-jamming methods. Section III discusses the system and jammer models. Section IV presents our reinforcement learning-based approach. Section V presents the simulation setup and results. Lastly, Section VI concludes the paper.

II. Related Work

A review of contemporary anti-jamming literature can be found in [2]. Given the contribution, we focus here on on existing techniques for countering reactive jammers, which can generally be categorized as: game theoretic, deep learning approaches, and decoy transmissions.

Game Theoretic– This jamming defense strategy has shown success in deriving equilibrium solutions, maximizing network utility in spite of the jammer making its own optimal decisions. The problem is modeled as a Stackelberg game in [5] in an environment with multiple communication networks and jammers. A pure strategy Nash equilibrium solution for channel selection is learned for each network and jammer, mutually maximizing the utility of every actor in the environment. This work is extended in [6], where a hypergraph model is applied to consider cumulative weak co-channel interference.

Deep Learning– Deep reinforcement learning has been gaining attention in anti-jamming literature. Xiao et al. [7] trained a relay to make mobility and frequency pattern decisions in order to establish reliable communications between nodes in an indoor environment in the presence of a sweeping jammer. Wang et al. [8] applied a poisoning algorithm to degrade a jammer's ability to predict and jam acknowledgement messages. In [9], the authors trained a deep reinforcement learning algorithm to steer an unmanned aerial vehicle to spatially avoid detected jammers. A deep-dueling neural network is pursued in [10] to quickly learn a policy to launch deceptive packets, tricking a jammer into responding with an attack. If a jamming signal is detected, it is used to the advantage of the system either for energy harvesting or backscatter communications; if no jamming signal is detected, the transmitter proceeds to transmit data. Li et al. [11] devised a utility function for a channel-aware jammer and conducted white- and black-box adversarial attacks on such a utility function, acting on channel selection and transmit power. The jammer then observes a noise-like spectrum, degrading the sensing accuracy.

Decoy Transmission Algorithm– In [12], the authors configured a network of 512 nodes to randomly transmit decoy packets in the hope of misleading reactive jammers. At any time slot, a node will either listen for data, transmit data, or transmit a decoy, all on a random channel. It is intuitive that using uncoordinated, random channels to transmit data would degrade network performance. Moreover, such an approach places additional requirements on edge devices. However, the large size of the network ensures a high probability that at least one node chose to listen to a channel containing a data packet. This allows for data packets to propagate through the network with resilience.

III. System and Attack Models

A. Communications Model

Throughout this paper, the word "channel" refers to a spectral frequency at which a system can operate. Communicating nodes are assumed to be time-synchronized and all transmissions are bursty and slotted in time. The communication system can operate on one of N_c channels at a time in half-duplex. The operating channel of the system at some time slot k is denoted as $f_c^{(k)}$. At any time instance, nodes can assume one of the following four roles: (1) broadcast a data packet on channel $f_c^{(k)}$, (2) listen for a data packet on channel $f_c^{(k)}$, (3) transmit a decoy packet on some channel $f_d^{(k)} \neq f_c^{(k)}$, or (4) perform an energy scan of the environment. An energy scan involves rapidly hopping across the available channels with some small dwell time and recording the maximum energy reported on each channel. Transmitting a decoy or performing an energy scan have an opportunity cost as any data packets transmitted will be missed during that time.

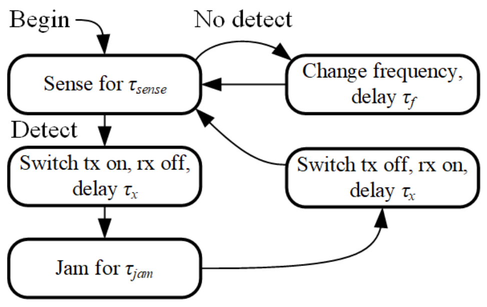

Begin
No detect
Sense for τ_{sense}
Change frequency, delay τ_f
Detect
Switch tx on, rx off, delay τ_x
Switch tx off, rx on, delay τ_x
Jam for τ_{jam}

Fig. 1. Jammer behavioral state diagram

B. Jammer Model

The jammer behavior considered in this work is one of a sensing-reactive nature. Reactive jammers emit a signal if and only if activity on a particular channel is detected. These jammers often operate in a half-duplex mode; in other words, they cannot sense and jam at the same time. Therefore, the jammer is parameterized by a sensing dwell time τ_{sense} and a jam time τ_{jam}. If the jammer has uncertainty in the channel being used by the target system, it must distribute its resources temporally across each channel. The time required to switch frequencies, and the time required to switch between transmit and receive mode, respectively, are denoted as τ_f and τ_x. The jammer's bandwidth and transmit power are denoted as B_j and P_j, respectively.

To monitor activity, the jammer employs an energy detector, as described in Section III-C. Only the null hypothesis $\mathcal{H}_0$ needs to be evaluated. The number of samples processed for energy detection M is equal to the jammer's time-bandwidth product $\lfloor \tau_{sense} \times B_j \rfloor$. For any scan of a given jammer channel, if a transmission is detected, the jamming channel will switch from the sensing state to the jamming state, with a latency of τ_x. After jamming for τ_{jam}, the state returns to the sensing state, again with a latency of τ_x. If no activity is detected, the jammer channel will change frequencies, with a latency of τ_f, to the next channel in the scan list $\boldsymbol{f}_j$. The frequency pattern that the jammer follows cycles through a fixed schedule, though not necessarily a linear sweep through the spectrum. A state diagram of the jammer's behavior is shown in Figure 1. At time k, the jammer state vector $\boldsymbol{s}_j^{(k)}$

contains the frequency at the start of time slot k as well as static parameters $\boldsymbol{p}_j = [\boldsymbol{f}_j, \tau_x, \tau_f, \tau_{jam}, \tau_{sense}, B_j]$.

C. Energy Detection

This section describes the statistical model of energy detection, which is used by both the jammer and the communicating nodes. Energy is a fundamental property of wireless transmissions, which can be used to characterize the content of a sample vector. The jammer uses this in order to detect target transmissions, while the communication nodes use this to detect jamming and decoy actions of neighboring nodes.

Energy is measured as $E_{\boldsymbol{x}} = ||\boldsymbol{x}||_2^2$, where $\boldsymbol{x} \in \mathbb{C}^M$ is the sample vector and M is the number of complex samples processed. The null hypothesis asserts that the sample vector is composed of only thermal noise $\boldsymbol{n} \in \mathbb{C}^M$, where $n_i \sim \mathcal{CN}(0,1)$, with noise power P_n. Under this hypothesis, the energy measurement $E_{\boldsymbol{x}}$ will be chi-squared distributed with $2M$ degrees of freedom [13]. The null hypothesis is rejected if the measured metric exceeds some threshold. This threshold can be determined by setting a constant false alarm rate P_{FA} and evaluating the inverse CDF of $E_{\boldsymbol{x}}$ at $(1-P_{FA})$.

$$\mathcal{H}_0 : \boldsymbol{x} = \sqrt{\frac{P_n}{2}}\boldsymbol{n} \tag{1}$$

An alternate hypothesis $\mathcal{H}_1$ can be evaluated if a transmission from a particular source is expected. This hypothesis asserts that the sample vector is composed of both the expected signal $\boldsymbol{s} \in \mathbb{C}^M$ plus thermal noise. The received power of $\boldsymbol{s}$ is expected to be P_s. Under this hypothesis, the energy measurement $E_{\boldsymbol{x}}$ will be non-central chi-squared distributed with $2M$ degrees of freedom and a non-centrality parameter of $M \times P_s$. Again, $\mathcal{H}_1$ is rejected if the measured metric exceeds some threshold, calculated by evaluating the inverse CDF of $E_{\boldsymbol{x}}$ at $(1-P_{FA})$.

$$\mathcal{H}_1 : \boldsymbol{x} = \boldsymbol{s} + \sqrt{\frac{P_n}{2}}\boldsymbol{n} \tag{2}$$

If both $\mathcal{H}_0$ and $\mathcal{H}_1$ are rejected, $\mathcal{H}_2$ is pursued. $\mathcal{H}_2$ states that there is an anomalous signal $\boldsymbol{a} \in \mathbb{C}^M$ in the received sample vector, possibly along with the signal $\boldsymbol{s}$ expected by $\mathcal{H}_1$, with the indicator function $I \in \{0,1\}$.

$$\mathcal{H}_2 : \boldsymbol{x} = \boldsymbol{a} + I\boldsymbol{s} + \sqrt{\frac{P_n}{2}}\boldsymbol{n} \tag{3}$$

IV. Decoy Transmission Algorithm

This section describes the proposed approach. An underlying time-division multiple access (TDMA) MAC scheme is discussed, followed by the data-driven algorithm, and finishing with our method to distribute or measure requisite information.

A. Time-slotted Coordination

A known pseudo-random sequence indicates which channel a data transmission will be on in each time slot. The time slot duration should be greater than or equal to the time required to both change frequencies and change between transmit and receive modes plus the duration of a packet. TDMA is used for medium access control. Each node is assigned an identifier $ID \in \{0, 1, ...N_n - 1\}$, where N_n is the number of nodes in the network. At any time slot k, the node in the network satisfying $\text{mod}(k, N_n) = ID$ transmits a data packet. During the time slot preceding a node sending data, when $\text{mod}(k+1, N_n) = ID$, the node will have the opportunity to transmit a decoy, indicated by the decoy strategy $Q()$. Our strategy should instruct a deceptive packet to be sent on a decoy channel that is separate from the data channel $f_c^{(k)}$ as to not directly impose a collision. Two time slots preceding a node sending data, when $\text{mod}(k+2, N_n) = ID$, the node will perform an energy scan across each of the available data channels in order to measure the decoy action f_d component of the system state $\boldsymbol{s}^{(k)}$. The energy scan results will be denoted as $\boldsymbol{e} \in \mathbb{R}^{N_c}$, where $e_f \in \mathbb{R}$ is the energy measured on channel f. In order for each node to be able to estimate the reward of a particular action, each node's data transmission will include an N_n-bit vector indicating that node's vector of past data packet successes. Details on usage of the energy scan and reward distribution are provided in the next section. The TDMA schedule of the network is illustrated in Figure 2.

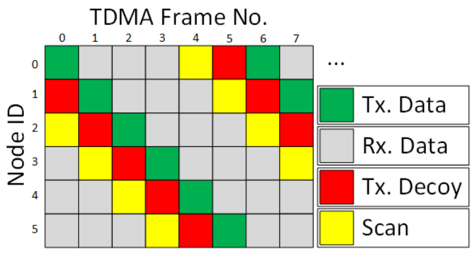


Fig. 2. TDMA schedule example for 6 nodes

The functions outlined in this section, while expected to benefit the network overall, come at a cost. If a node is performing an energy scan or sending a decoy, it cannot listen for a data packet. This will reduce the total throughput on the network by up to two packets in any time slot, or only one packet if a node does not send a decoy. This also has implications on energy expenditure of each node, which is captured in the reward function, Eq. (7). Further, the communication overhead required is N_n bits per packet, because each node must communicate a history of data reception successes in the last TDMA round, which lasts N_n slots.

B. Data-Driven Model

The jamming mitigation strategy involves transmission of decoy packets in an intelligent manner. At any time slot k when a wireless network is transmitting data, there is a chance that a reactive jammer's scanning algorithm will coincide with the transmission and deploy its jamming attack. This probability is a function of (i) the channel of the transmitted data, which is known to the network, and (ii) the jammer parameters $\boldsymbol{p}_j$ and state $\boldsymbol{s}_j^{(k)}$, which are not explicitly known to the network. Given an estimate of the jammer state $\hat{\boldsymbol{s}}_j^{(k)}$, an optimal channel to deploy a decoy transmission for stalling the jammer and protect the legitimate data transmission may exist, denoted as $f_{od}^{(k)}$.

$$\hat{f}_{od}^{(k)} = \arg\max_{f_d} Q(f_d, f_c^{(k)}, \hat{\boldsymbol{s}}_j^{(k)} | \boldsymbol{\theta}) \quad (4)$$

In Eq. (4), $Q()$ is a model with parameters $\boldsymbol{\theta}$ mapping information on the jammer and the communication system to some quality metric of sending a decoy packet on channel indicated by f_d. Both f_d and f_{od} are contained in the set $\{1, ...N_c, \varnothing\}$. This set contains an integer for each channel between 1 and the number of channels, N_c, and $\varnothing$ indicating the non-existence of an optimal decoy channel. In other words, $\varnothing$ indicates that a decoy should not be transmitted.

It is difficult or impossible to know at any point in time on what channel the jammer is sensing, as there is no feedback when the jammer is in receive mode. One source of feedback available is the performance of the network, which is a function of the jammer state and parameters. A finite history of N past data channels $\boldsymbol{f}_{c,p}^{(k)} = [f_c^{(k-i)}]_{1<i\leq N}$, the success of those data packets $\boldsymbol{j}_{c,p}^{(k)} = [j_c^{(k-i)}]_{1<i\leq N}$, and the decoy actions at those time slots $\boldsymbol{f}_{d,p}^{(k)} = [f_d^{(k-i)}]_{1<i\leq N}$ can provide insight into this unknown.

$$\hat{\boldsymbol{s}}_j^{(k)} = h(\boldsymbol{f}_{c,p}^{(k)}, \boldsymbol{j}_{c,p}^{(k)} | \hat{\boldsymbol{p}}_j) \quad (5)$$

In Eq. (5), $h()$ is a mapping between the channel and the success of past data transmissions and an estimate of the jammer state at time k. Eq. (5) replaces the jammer state estimate in Eq. (4) to provide a more generalized function relating the quality of a decoy transmission on any channel to the feedback available to the network, in Eq. (6). For brevity, the concatenation of all of the state information at time k is denoted as $\boldsymbol{s}^{(k)} = [\boldsymbol{f}_{c,p}^{(k)}, \boldsymbol{j}_{c,p}^{(k)}, \boldsymbol{f}_{d,p}^{(k)}, f_c^{(k)}]$.

$$\hat{f}_{od}^{(k)} = \arg\max_{f_d} Q(f_d, \boldsymbol{s}^{(k)} | \boldsymbol{\theta}) \quad (6)$$

If $Q()$ is implemented as a data-driven model, then the quality estimate function $Q()$, the estimate of $h()$, and the jammer parameters $\hat{\boldsymbol{p}}_j$ are implicitly learned as parameters of the model $\boldsymbol{\theta}$. Q-learning will be used to train the model parameters. A reward $r^{(k)}$ is determined for each time slot, as an indicator of the performance of the model, shown below.

$$r^{(k)} = w_s N_s^{(k)} - w_d N_d^{(k)} \quad (7)$$

In Eq. (7), $N_s^{(k)}$ and $N_d^{(k)}$ are the number of successfully received data packets and the number of decoy packets transmitted, respectively, in time slot k; w_s and w_d are constant weighting factors; and $r^{(k)}$ is the reward of time slot k. In other words, successfully transmitted packets are positively rewarded, whereas decoy transmissions are negatively rewarded. While it is of interest to transmit decoy packets to manipulate the jammer, the inclusion of them as negative feedback is to discourage the model from learning to excessively transmit decoys, which may waste energy. In case the transmit power is not of concern, this constraint can be lifted by fixing w_d to 0. $Q()$ will be trained to iteratively improve its reward estimate at time instance k using the Bellman Equation below.

$$Q(f_d^{(k)}, \boldsymbol{s}^{(k)} | \boldsymbol{\theta}) \leftarrow Q(f_d^{(k)}, \boldsymbol{s}^{(k)} | \boldsymbol{\theta}) + \alpha \cdot (r^{(k)} + \gamma \cdot \max_{f_d} Q(f_d, \boldsymbol{s}^{(k+1)} | \boldsymbol{\theta}) - Q(\boldsymbol{f}_d^{(k)}, \boldsymbol{s}^{(k)} | \boldsymbol{\theta}))) \quad (8)$$

In Eq. (8), α is the learning rate and γ is the discount factor. The purpose of γ is to consider future reward of state $\boldsymbol{s}^{(k+1)}$ when taking action in state $\boldsymbol{s}^k$. An exploration-exploitation policy is implemented, where a random action is selected with probability ϵ and an action indicated by $Q()$ is selected otherwise. The value of ϵ assumes initial value ϵ_0, selected as 0.75. ϵ decreases by a factor of $d\epsilon$ each TDMA frame, which is selected as 0.05. These values were selected through experimentation, providing a balance between exploration and exploitation at the start of a simulation and exhibiting performance convergence at a reasonable rate. Experience replay is implemented, where a memory of state-action-reward tuples are recorded to the node's memory M for each decoy action performed. Every TDMA frame, each node performs a learning step using minibatch of 64 examples from M. It is critical to note that the reward function is dependent on the performance of the network as a whole and not any individual node, posing two notable challenges. First, as nodes should make decoy transmission decisions locally, state information of the decoy actions in the previous time slot is needed. Decoy actions are uncoordinated and a node sending a decoy doesn't have sufficient time between slots to communicate its decoy action to the network. Second, in order for learning to take place, the reward function of any time slot must be estimated by the node that sent a decoy at that time. This information can be communicated on the network; however, jammed payloads can impact the distribution of this requisite information.

C. State and Reward Estimation

This section describes how each node individually determines state-action-reward tuples in order to both evaluate $Q()$ as well as conduct learning. In order for node i to evaluate $Q()$, the system state $\boldsymbol{s}^{(k)}$ must be estimated. Of the components that make up $\boldsymbol{s}^{(k)}$, the vector of past data channels $\boldsymbol{f}_{c,p}^{(k)}$ and the current data channel $f_c^{(k)}$ are deterministic and therefore do not need to be estimated. The vector of past data packet successes $\boldsymbol{j}_{c,p}^{(k)}$ can be populated based on what messages node i successfully decoded for time slots where node i was listening for data packets. In time slots where node i transmitted a decoy, it will rely on the other nodes broadcasting their history of successful receptions on the network. Node i will assume that if more than half of the nodes indicate a successful packet reception in a time slot where node i transmitted a decoy, then the data transmission did not get jammed. In time slots where node i executed an energy scan, $\mathcal{H}_1$ can be evaluated on the energy scan results for the data channel, given that node j was expected to transmit a data packet and the pathloss between nodes i and j is known. The vector of past decoy channels is populated based on a node's energy scan results from the time slot prior to it deploying a decoy transmission. If $\mathcal{H}_0$ is accepted for all channels other than $f_c^{(k)}$, then node i assigns $\varnothing$ to $f_d^{(k)}$. Given that node l was expected to transmit a decoy packet and the pathloss between nodes i and l is known, the maximum likelihood channel to contain a decoy transmission from node l, based on the energy measured in each channel, $\text{argmax}_{f_d} P(e_{f_d}|\mathcal{H}_1)$ is assigned as the decoy action.

Lastly, node i measures its reward function based on the other nodes broadcasting their history of successful packets in their data transmissions. Since a node may not receive all $N_c - 2$ reports of packet success in a TDMA frame due to jammed packets, the estimated reward is scaled up based on the number of reports received in a TDMA frame to produce the reward. Our algorithm is summarized in Algorithm 1.

V. Validation Results

Validation Environment and Setup– Our anti-jamming approach is validated through simulation using NS-3. The deep learning model is implemented in PyTorch. A network of 32 ZigBee devices, all within range of one another, is implemented using the "lr-wpan" model in NS-3. The log-distance path loss model with a path loss exponent of 3 models the power loss between nodes. Stationary ZigBee devices are placed randomly on a 50×50 meter grid. ZigBee signals can occupy one of the sixteen 2 MHz wide channels with a transmit power as high as 100 mW. ZigBee does not implement FEC; hence a single channel error will result in a packet error. Error detection is achieved with a 16-bit cyclic redundancy check (CRC) following the payload. It is assumed that any packet error is caught by the CRC and that CRC errors are infrequent enough such that they do not affect performance. Data packets contain 64 bytes, requiring a packet duration of 2.7 ms, and the TDMA frame duration is 3.8 ms.

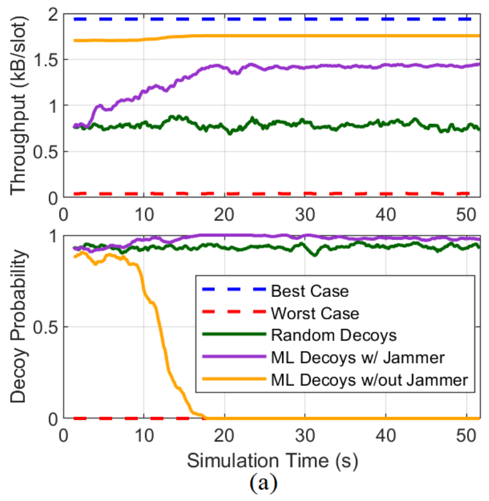
The function $Q()$ from Eq. (6) is implemented as a dense neural network with a single hidden layer. Channels f_c are one-hot encoded vectors with N_c elements– one for each possible channel. Similarly, decoy actions f_d are one-hot encoded with $N_c + 1$ elements, having an additional element indicating no decoy transmission. The input size of the neural network model is equal to the size of the state vector $\boldsymbol{s}^{(k)} = [\boldsymbol{f}_{c,p}^{(k)}, \boldsymbol{j}_{c,p}^{(k)}, \boldsymbol{f}_{d,p}^{(k)}, f_c^{(k)}]$. Given that $N_c = 16$, f_c is a size-16 vector and f_d is a size-17 vector and using a history of 1 time slot for vectors $\boldsymbol{f}_{c,p}^{(k)}$ and $\boldsymbol{j}_{c,p}^{(k)}$; this brings the size of $\boldsymbol{s}^{(k)}$ to 50. The input layer is followed by a single hidden layer with size 256 and a rectified linear unit (ReLU) activation function. The output size of the model is equal to 17, the size of f_d, and no activation function is used at the output.

A reactive jammer is placed in the 50×50 meter grid and tailors its attacks to specifically target a ZigBee network. A brief jamming pulse can cause significant damage to a ZigBee packet due to the lack of FEC in the ZigBee protocol, enabling highly efficient jamming attacks. We assume the jammer channel scan pattern is independent of its detections and is not performing hop-prediction. The jammer is parameterized as follows: $B_j = 10$MHz, $\tau_{jam} = 1$ms, $\tau_{sense} = 1$ms, $\tau_x = 10\mu$s, $\tau_f = 10\mu$s, and $P_{FA} = 0.1\%$. The frequency sense schedule follows a linear sweep between 2.4 and 2.5 GHz in increments of B_j, so $\boldsymbol{f}_j = [2.405, 2.415, ...2.485, 2.495]$GHz.

Algorithm 1: Anti-jamming policy used by each node

```
$k, k_d, N_{rx}, N_{d,rx} \leftarrow 0$
$M \leftarrow \{\}$
while ALWAYS do
    $f_c^{(k)} \leftarrow$ Next channel in data channel list
    if $mod(k, N_c) = ID$ then
        Change channel to $f_c^{(k)}$
        Build packet $P$ with $P.\boldsymbol{j}_c \leftarrow [j_c^{(k-i)}]_{i \le 1 \le N_c}$
        Transmit data packet $P$
    else if $mod(k+1, N_c) = ID$ then
        $r^{(k_d)} \leftarrow N_{d,rx} \times N_c / N_{rx}$
        $j_c^{(k_d)} \leftarrow round(N_{d,rx}/N_{rx})$
        $\boldsymbol{s}^{(k_d)} \leftarrow [\boldsymbol{f}_{c,p}^{(k_d)}, \boldsymbol{j}_{c,p}^{(k_d)}, \boldsymbol{f}_{d,p}^{(k_d)}, f_c^{(k_d)}]$
        $\boldsymbol{s}^{(k_d+1)} \leftarrow [\boldsymbol{f}_{c,p}^{(k_d+1)}, \boldsymbol{j}_{c,p}^{(k_d+1)}, \boldsymbol{f}_{d,p}^{(k_d+1)}, f_c^{(k_d+1)}]$
        $M \leftarrow \{M, (f_d^{(k_d)}, \boldsymbol{s}^{(k_d)}, \boldsymbol{s}^{(k_d+1)}, r^{(k_d)})\}$
        Train $\boldsymbol{\theta}$ on a minibatch from $M$
        $f_d^{(k)} \leftarrow \text{argmax}_{f_d} Q(f_d, \boldsymbol{s}^{(k)}|\boldsymbol{\theta})$
        Change channel to $f_d^{(k)}$
        Transmit decoy packet $P_d$
        $N_{d,rx} \leftarrow 0$
        $N_{rx} \leftarrow 0$
        $k_d \leftarrow k$
    else if $mod(k+2, N_c) = ID$ then
        $\boldsymbol{e} \leftarrow$ Energy scan
        $j^{(k)} \leftarrow \left(e_{f_c^{(k)}} \overset{?}{\in} T\right)$
        $f_d^{(k)} \leftarrow \text{argmax}_{f_d} P(e_{f_d}|\mathcal{H}_1)$
    else
        Change channel to $f_c^{(k)}$
        $P \leftarrow$ Listen for data packet
        if $P.ValidCRC$ then
            $j_c^{(k)} \leftarrow 1$
            $N_{rx} \leftarrow N_{rx} + 1$
            $N_{d,rx} \leftarrow N_{d,rx} + P.j_c^{(k_d)}$
        else if $(!P.ValidCRC)||(P = \varnothing)$ then
            $j_c^{(k)} \leftarrow 0$
    end
    $k \leftarrow k + 1$
    Wait for next time slot
end
```

Simulation Results– Our data-driven decoy transmission scheme is evaluated in two scenarios: one with the jammer present and one without, with results shown in Figure 3 (a). Upper and lower bounds on the system throughput are evaluated with simulations of the network not employing jamming mitigation. The upper bound has no jammer present, whereas the lower bound does have the jammer. When the jammer is not present, the throughput of the network approaches 1.75 kB/slot as the probability of sending a decoy falls to



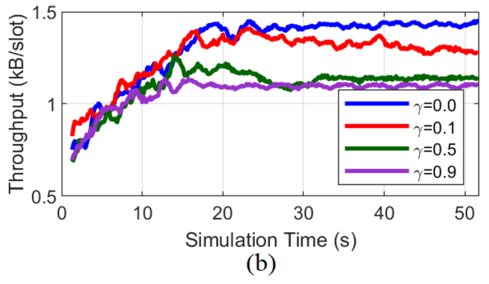


Fig. 3. Results comparing (a) best- and worst-case performance, random decoy actions, and our solution (b) the effect of γ on throughput

0. The throughput difference between this simulation and the upper bound indicates the cost of implementing our mitigation. When the jammer is present, the throughput of the network starts at 0.75 kB/slot and approaches 1.5 kB/slot as time progresses. The probability of a decoy being transmitted in a slot is ≈ 1 for the duration of the simulation. As a point of reference, a random decoy scheme, with decoy transmission logic similar to [12], is simulated and presented. This scheme has a lower cost to implement– no sensing requirement and no communications overhead, which is accounted for in the figure. In this scheme, the throughput is approximately 0.75 kB/slot and the probability of sending a decoy is ≈ 1 for the duration of the simulation. Our data-driven mitigation method is outperforming this method by almost a factor of two, indicating that our algorithm is indeed effectively countering the jammer, and not making random decisions.

Figure 3 (b) presents this comparison of the effect of changing γ. The discount factor is intended to account for the best possible reward of the state that an action moves the environment to. We see, however, that a greedy strategy is performing best with respect to throughput, as increasing this hyperparameter above 0 degrades the performance.

VI. Conclusion

We have outlined an anti-jamming technique using a data-driven model to instruct a network how to send deceptive transmissions in order to mislead an adversarial jammer. We presented a practical means to measure all of the information necessary to enable wireless nodes to learn online how to combat a jammer with a range of parameters. Simulation results using NS-3 are used to validate this technique, showing that (1) decoy transmissions can improve the throughput when the network is under attack and (2) using reinforcement learning to measure the effect of a decoy action and improve future actions can provide additional throughput improvements.

A vulnerability to this technique lies in the dependency on energy scanning. Energy, while a simple and useful feature for evaluating the hypothesis tests outlined in Section III-C, does not say much about the content of the received signal. As such, using this to measure part of the system state leaves sensing nodes open to having their own sensing measurements polluted by the adversary. A more intelligent detector could fill this vulnerability– possibly RF fingerprinting.

Lastly, a more intelligent jammer could employ the following: (1) hop prediction, (2) covert jamming, or (3) poisoning attacks, any of which could impact the performance of this technique. To address (1), using a learning model to drive the data channel pattern is of interest, though poses a substantial technical challenge. To address (2) and (3), specialized detectors can be designed to improve state measurements. The aforementioned issues will be addressed by our future work.

References

[1] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," IEEE Internet of Things Journal, vol. 7, no. 1, pp. 16-32, 2020.
[2] H. Pirayesh, and H. Zeng. "Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey." IEEE Comm. Surveys & Tutorials, 2022.
[3] Ó. Puñal, C. Pereira, A. Aguiar and J. Gross, "Experimental Characterization and Modeling of RF Jamming Attacks on VANETs," IEEE Trans. on Vehicular Tech., vol. 64, no. 2, pp. 524-540, 2015.
[4] K. Grover, A. Lim, and Q. Yang. "Jamming and anti–jamming techniques in wireless networks: a survey." Int. J. of Ad Hoc and Ubiquitous Computing vol 17, no. 4, pp. 197-215, 2014.
[5] F. Yao, et al. "A hierarchical learning approach to anti-jamming channel selection strategies." Wireless Networks vol. 25, no. 1, pp. 201-213, 2019.
[6] L. Jia, et al. "A game-theoretic learning approach for anti-jamming dynamic spectrum access in dense wireless networks." IEEE Trans. on Vehicular Technology vol 68, no.2, pp. 1646-1656, 2018.
[7] L. Xiao, D. Jiang, D. Xu, H. Zhu, Y. Zhang and H. V. Poor, "Two-Dimensional Antijamming Mobile Communication Based on Reinforcement Learning," IEEE Trans. on Vehicular Tech., vol. 67, no. 10, pp. 9499-9512, 2018.
[8] X. Wang, M. Cenk Gursoy, T. Erpek and Y. E. Sagduyu, "Jamming-Resilient Path Planning for Multiple UAVs via Deep Reinforcement Learning," Proc. IEEE Int. Conf. on Comm. Workshops, 2021.
[9] N. I. Mowla, N. H. Tran, I. Doh and K. Chae, "AFRL: Adaptive federated reinforcement learning for intelligent jamming defense in FANET," J. of Comm. and Networks, vol. 22, no. 3, pp. 244-258, 2020.
[10] N. Van Huynh, D. N. Nguyen, D. T. Hoang and E. Dutkiewicz, "Defeating Reactive Jammers with Deep Dueling-based Deception Mechanism," Proc. IEEE Int. Conf. on Comm., 2021.
[11] W. Li, J. Wang, L. Li, X. Chen, W. Huang and S. Li, "Countermeasure for Smart Jamming Threat: A Deceptively Adversarial Attack Approach," Proc. IEEE Int. Conf. on Comm., 2021.
[12] S. Sciancalepore, et al. "Strength of Crowd (SOC)-Defeating a Reactive Jammer in IoT with Decoy Messages." Sensors, vol. 18, no. 10, pp. 3492, 2018.
[13] H. Urkowitz, "Energy detection of unknown deterministic signals," Proc. of the IEEE, vol. 55, no. 4, pp. 523-531, 1967.